Evolutionary Algorithms: From Recombination to Search Distributions

H. Mühlenbein and Th. Mahnig

RWCP*** Theoretical Foundation GMD† Laboratory
D-53754 Sankt Augustin
E-mail: *muehlenbein@gmd.de*

Abstract. First we come to the conclusion that all genetic algorithms can be approximated by an algorithm which keeps the population in linkage equilibrium. Here the genetic population is given as a product of univariate marginal distributions. We describe a simple algorithm which keeps the population in linkage equilibrium. It is called the Univariate Marginal Distribution Algorithm (UMDA). Our main result is that UMDA transforms the discrete optimization problem into a continuous one defined by the average fitness $\tilde{W}(p_1, \ldots, p_n)$ as a function of the univariate marginal distributions p_i. For proportionate selection UMDA performs gradient ascent in the landscape defined by $W(\boldsymbol{p})$. We derive a difference equation for p_i which has already been proposed by Wright in population genetics. We show that UMDA solves difficult multimodal optimization problems. But for functions with highly correlated variables it has to be extended to marginal and conditional distributions. The Factorized Distribution Algorithm (FDA) uses a general factorization of the distribution. For decomposable functions the optimal factorization can be explicitly computed. In general it has to be computed from the data. Each distribution can be represented as a Bayesian network. Computing the structure from the data is called learning in Bayesian network theory. The problem of finding a minimal structure which explains the data is discussed in detail. It is shown that the Bayesian Information Criterion is a good score for this problem. This is used by the algorithm LFDA.

Keywords
genetic algorithms, linkage equilibrium, factorization of distribution, replicator equation, Bayesian networks, Boltzmann distribution

1 Introduction

Simulating evolution as seen in nature has been identified as one of the key computing paradigms for the next 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

*** Real World Computing Partnership
† GMD - Forschungszentrum Informationstechnik

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 ends: 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 evolutionary algorithms for optimization. Here results from classical population genetics and statistics can be used. The outline of the chapter is as follows. In section 2 we summarize research which indicates that all simple genetic algorithms can be approximated by an algorithm keeping the population in linkage equilibrium. The simplest algorithm keeping the population in linkage equilibrium is the *Univariate Marginal Distribution Algorithm* UMDA. UMDA uses search distributions instead of recombination. Its mathematical analysis is done in section 3.

In section 4 we shortly survey algorithms using univariate marginal distributions. The mathematical behavior of UMDA with proportionate selection is described by a differential equation called the *replicator equation*. This equation is discussed in section 5. Then numerical results for UMDA are presented. They confirm that UMDA is able to solve difficult multimodal optimization problems. Certain optimization problems need marginal distributions of higher order. The *Factorized Distribution Algorithm* FDA uses a general factorization of the distribution. It is introduced in section 7.

The mathematical theory of UMDA and FDA is based on infinite populations. The problem of finite samples is discussed in section 8. In the final section LFDA is introduced. This algorithm computes a plausible factorization of the search distribution from a finite sample of data.

2 The Simple Genetic Algorithm

In this section we discuss the standard genetic algorithm, also called the Simple Genetic Algorithm (SGA). The algorithm is described by Holland [12] and Goldberg [9]. It consists of

- fitness proportionate selection
- recombination/crossover
- mutation

In this section we will analyze selection and recombination only. Mutation is considered to be a background operator. It can be analyzed by known techniques from stochastics [21,18]. We will investigate two widely used recombination/crossover schemes.

Definition 1. Let two strings $\boldsymbol{x}$ and $\boldsymbol{y}$ be given. In *one-point crossover* the string $\boldsymbol{z}$ 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 $\{x_i, y_i\}$.

Let $\boldsymbol{x} = (x_1, \ldots, x_n)$ 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. Let a function $f : \boldsymbol{X} \to R^{\geq 0}$ be given. We consider the optimization problem $\boldsymbol{x}_{\text{opt}} = \operatorname{argmax} f(\boldsymbol{x})$.

Definition 2. Let $p(\boldsymbol{x}, t)$ denote the probability of $\boldsymbol{x}$ in the population at generation t. Then $p(x_i, t) = \sum_{\boldsymbol{x}, X_i = x_i} p(\boldsymbol{x}, t)$ defines a univariate marginal distributions.

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

$$\bar{f}(t) = \sum_x p(x, t) f(x),$$

$$V(t) = \sum_x p(x, t) \left(f(x) - \bar{f}(t)\right)^2.$$

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

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

2.1 Proportionate Selection

Proportionate selection changes the probabilities according to

$$p(x, t + 1) = p(x, t) \frac{f(x)}{\bar{f}(t)}. \tag{2}$$

Lemma 1. *For proportionate selection the response is given by*

$$R(t) = \frac{V(t)}{\bar{f}(t)}. \tag{3}$$

Proof. We have

$$\begin{aligned} R(t) &= \sum_x p(x, t) \frac{f^2(x)}{\bar{f}(t)} - \bar{f}(t) \\ &= \sum_x p(x, t) \frac{f^2(x) - \bar{f}^2(t)}{\bar{f}(t)} \\ &= \frac{V(t)}{\bar{f}(t)}. \end{aligned}$$

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

2.2 Recombination

For the analysis we introduce a special distribution, called Robbins' proportions.

Definition 3. Robbins' proportions are given by the distribution π,

$$\pi(x,t) := \prod_{i=1}^{n} p(x_i,t). \tag{4}$$

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

Geiringer [8] has shown that all reasonable recombination schemes lead to the same limit distribution.

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

Complete recombination means that for each subset S of $\{1, \ldots, n\}$, 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 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*

$$D(t) = (-1)^{|x|^2}\big(p(\boldsymbol{x},t) - p_1(x_1,0)p_2(x_2,0)\big). \tag{5}$$

Furthermore the factor $D(t)$ is halved each generation,

$$D(t+1) = \frac{1}{2}D(t). \tag{6}$$

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

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

we obtain

$$D(t) = p(1,1,t) - p_1(1,0)p_2(1,0).$$

This gives (5) for $\boldsymbol{x} = (1, 1)$. 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

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

By computing $D(t + 1)$, (6) is obtained.

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

$$DSQ(t) = \sum_{\boldsymbol{x}} \left(p(\boldsymbol{x}, t) - p_1(x_1) p_2(x_2)\right)^2. \tag{7}$$

From the above theorem we obtain:

Corollary 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.

Table 1 gives numerical results for $n = 8$ loci. For the initial probabilities $q_0 = q_7 = 0.5$ linkage equilibrium is given by $q_i = 1/8$. One-point crossover converges slowly to linkage equilibrium, uniform crossover converges very fast.

We have to mention an important fact. In a finite population linkage equilibrium cannot be exactly achieved. We take the uniform distribution as example. Here linkage equilibrium is given by $p(\boldsymbol{x}) = 2^{-n}$. This value can only be obtained if the size of the population N is substantially larger than 2^n ! The finite size effect is demonstrated in table 2. There the numerical value for $DSQ(t)$ is shown for different population sizes. In addition $c = DSQ(t + 1)/DSQ(t)$ is displayed. From theorem 2 a factor of $c = 0.25$ is expected.

For a population of $N = 1000$ the minimum deviation $DSQ_{\min}$ from Robbins' proportions is already achieved after four generations, 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 & Mühlenbein [1]. It will not be considered here.

t	q_0	q_1	q_2	$q_0 - 1/2^8$	$q_1 - 1/2^8$	$q_2 - 1/2^8$
0	0.5	0.0	0.0	0.4961	-3.906E-3	-3.906E-3
1	0.3774	0.0177	0.0	0.3735	1.369E-2	-3.906E-3
2	0.2879	0.0262	0.0012	0.2840	2.229E-2	-2.706E-3
3	0.2225	0.0303	0.0028	0.2186	2.639E-2	-1.106E-3
4	0.1768	0.0314	0.0042	0.1729	2.749E-2	+0.294E-3
5	0.1421	0.0298	0.0050	0.1382	2.589E-2	+1.094E-3
0	0.5	0.0	0.0	0.4961	-3.906E-3	-3.906E-3
1	0.2533	0.0020	0.0023	0.2494	-1.927E-3	-1.646E-3
2	0.0895	0.0097	0.0101	0.0856	+5.834E-3	+6.174E-3
3	0.0323	0.0093	0.0102	0.0283	+5.434E-3	+6.244E-3
4	0.0148	0.0074	0.0072	0.0108	+3.574E-3	+3.254E-3
5	0.0090	0.0057	0.0056	0.0051	-1.794E-3	+1.794E-3

Table 1. Comparison of convergence to Robbins' proportions for $n = 8$ loci, one-point (upper half) and uniform crossover, $q_0 = p(0, \ldots, 0)$, $q_1 = p(0, \ldots, 1)$, $q_2 = p(0, \ldots, 1, 0)$, population size $N = 1000$, averages over 100 runs; $q_0 = q_7 = 0.5$.

	$N = 1000$		$N = 10000$		$N = 20000$	
t	*DSQ*	c	*DSQ*	c	*DSQ*	c
0	9.00E-2		9.00E-2		9.00E-2	
1	2.28E-2	0.25	2.25E-2	0.25	2.25E-3	0.25
2	6.37E-3	0.28	5.77E-3	0.25	5.59E-3	0.25
3	2.34E-3	0.37	1.55E-3	0.27	1.45E-3	0.26
4	1.62E-3	0.70	4.96E-4	0.32	4.24E-4	0.29
5	1.91E-3	1.03	2.43E-4	0.49	1.67E-4	0.39
8	2.62E-3	1.13	2.25E-4	1.10	1.41E-4	1.10

Table 2. Convergence to linkage equilibrium for $n = 2$ loci.

2.3 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 researchers [23].

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

$$p(x, t+1) = \sum_{y,z} R_{x,yz} p^s(y) p^s(z). \tag{8}$$

$p^s(x)$ denotes the probability of string x after selection. For n loci the recombination distribution R consists of $2^n \times 2^n$ parameters. Recently Christiansen and Feldman [4] have written a survey about the mathematics of selection and recombination from the viewpoint of population genetics. A new technique to obtain the equations has been developed by Vose [31]. 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 $n = 3$ loci. For notational convenience we use the integer representation i_x for x. Furthermore we set $q_{i_x} := p(x)$. In the next theorem we only give the equations for q_7 and q_3. From these expressions the reader can extrapolate the remaining five equations.

Theorem 3. *For proportionate selection and uniform crossover the probabilities are given by*

$$\begin{aligned} q_7(t+1) = q_7(t)\frac{f_7}{\bar{f}(t)} + \frac{1}{\bar{f}(t)^2}\Big(&-\frac{1}{2}f_7q_7(f_1q_1 + f_2q_2 + f_4q_4) - \frac{3}{4}f_7q_7f_0q_0 \\ &+ \frac{1}{2}(f_3q_3f_5q_5 + f_3q_3f_6q_6 + f_5q_5f_6q_6) \\ &+ \frac{1}{4}(f_1q_1f_6q_6 + f_2q_2f_5q_5 + f_3q_3f_4q_4)\Big) \end{aligned}$$

and

$$\begin{aligned} q_3(t+1) = q_3(t)\frac{f_3}{\bar{f}(t)} + \frac{1}{\bar{f}(t)^2}\Big(&-\frac{1}{4}f_3q_3(f_0q_0 + f_5q_5 + f_6q_6) - \frac{3}{4}f_3q_3f_4q_4 \\ &+ \frac{1}{2}(f_1q_1f_7q_7 + f_2q_2f_7q_7 + f_1q_1f_2q_2) \\ &+ \frac{1}{4}(f_0q_0f_7q_7 + f_1q_1f_6q_6 + f_2q_2f_5q_5)\Big). \end{aligned}$$

Proof. We outline the proof. After proportionate selection and recombination we obtain

$$\begin{aligned} q_7(t+1) &= q_7(t)\frac{f_7}{\bar{f}(t)^2}\left(\sum_{j=1}^{7} 2R_{7,7j}q_j(t)f_j - q_7f_7\right) + \mathit{rest}, \\ \mathit{rest} &= \sum_{i\neq 7, j\neq 7} R_{7,ij}q_i(t)\frac{f_i}{\bar{f}(t)}q_j(t)\frac{f_j}{\bar{f}(t)}. \end{aligned}$$

For uniform crossover we compute $R_{7,70} = 1/8, R_{7,71} = R_{7,72} = R_{7,74} = 1/4$, $R_{7,73} = R_{7,75} = R_{7,76} = 1/2$. Inserting these numbers and rearranging the terms we obtain

$$\begin{aligned} q_7(t+1) = q_7(t)\frac{f_7}{\bar{f}(t)} + \frac{1}{\bar{f}(t)^2}\Big(&-\frac{1}{2}f_7q_7(f_1q_1 + f_2q_2 + f_4q_4) - \frac{3}{4}f_7q_7f_0q_0\Big) \\ &+ \mathit{rest}. \end{aligned}$$

The term *rest* is computed in the same way.

We have split the difference equation into the selection term and a recombination term. The recombination term consists of two terms: the probabilities that x will be reduced by recombination with other genotypes and the probabilities that recombination of two strings different from x will produce string x. A mathematical analysis of the mathematical properties of 3 loci systems is difficult. But we easily obtain an interesting result for a special case.

Definition 4. A fitness function is called *multiplicative* if

$$f(\boldsymbol{x}) = C \cdot \prod_i e^{a_i x_i}. \tag{9}$$

We easily obtain from theorem 3 the following corollary. The proof is left to the reader.

Corollary 2. *If the fitness function is multiplicative, if the population is in linkage equilibrium, and if the assumptions of theorem 3 are valid, then the population remains in linkage equilibrium and the probabilities are given by (2).*

We *conjecture that the corollary is true for arbitrary* n. It seems to be known in population genetics, but we have not found a proof. Unfortunately for all other fitness functions selection and recombination leads to *linkage disequilibrium.*

The question is whether linkage disequilibrium is important for evolutionary optimization. The answer is no. We provide evidence for this statement by citing a theorem from [18]. It describes the difference equations for the univariate marginal frequencies.

Theorem 4. *For* any *complete recombination/crossover scheme used after proportionate selection, the univariate marginal frequencies are determined by*

$$p(x_i, t+1) = \sum_{\boldsymbol{x}|X_i=x_i} \frac{p(\boldsymbol{x}, t) f(\boldsymbol{x})}{\bar{f}(t)}. \tag{10}$$

Proof. After selection the univariate marginal frequencies are given by

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

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

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

This theorem can be formulated in the terms of Holland's schema theory[12]. Let $H(x_i) = (*, \dots, *, x_i, *, \dots, *)$ be a first-order schema at locus i. This schema includes all strings where the gene at locus i is fixed to x_i. The univariate marginal frequency $p(x_i, t)$ is obviously identical to the frequency of schema $H(x_i)$. The fitness of the schema at generation t is given by

$$f(H(x_i), t) = \frac{1}{p(x_i, t)} \sum_{\boldsymbol{x} | X_i = x_i} p(\boldsymbol{x}, t) f(\boldsymbol{x}). \tag{11}$$

From theorem 4 we obtain:

Corollary 3 (First-order schema theorem). *For a genetic algorithm with proportionate selection using any complete recombination, the frequency of first-order schemata changes according to*

$$p_i(x_i, t+1) = p_i(x_i, t) \frac{f(H(x_i), t)}{\bar{f}(t)}. \tag{12}$$

Note that the above corollary is valid for an infinite population with proportionate selection and recombination. Holland's famous schema theorem implies for first order schemata only an inequality [12]:

$$p_i(x_i, t+1) \geq p_i(x_i, t) \frac{f(H(x_i), t)}{\bar{f}(t)}.$$

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 characterize the fundamental algorithm.

3 UMDA – The Univariate Marginal Distribution Algorithm

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* [22]. 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 $p^s(x_i)$ in the next population, where $p^s(x_i)$ is the distribution of x_i in the selected parents. Thus new strings $\boldsymbol{x}$ are generated according to the distribution

$$p(\boldsymbol{x}, t+1) = \prod_{i=1}^{n} p^s(x_i, t). \tag{13}$$

One can simplify the algorithm still more by directly computing the univariate marginal frequencies from the data. Then (13) can be used to generate new strings. This method is used by the *Univariate Marginal Distribution Algorithm* (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 $p^s(x_i, t)$ of the selected set.

STEP 2: Generate N new points according to the distribution $p(\boldsymbol{x}, t+1) = \prod_{i=1}^n p^s(x_i, t)$. Set $t \Leftarrow t+1$.

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

UMDA formally needs $2n$ parameters, the marginal distributions $p(x_i)$. We consider $\bar{f}(t)$ as a function which depends on $p(x_i)$. To emphasize this dependency we write

$$W(p(x_1 = 0), p(x_1 = 1), \ldots, p(x_n = 1)) := \bar{f}(t). \tag{14}$$

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

Theorem 5. *For infinite populations and proportionate selection the difference equations for the gene frequencies used by UMDA are given by*

$$p(x_i, t+1) = p(x_i, t)\frac{\bar{f}_i(x_i, t)}{W(t)}, \tag{15}$$

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

$$p(x_i, t+1) = p(x_i, t) + p(x_i, t)\frac{\frac{\partial W}{\partial p(x_i)} - W(t)}{W(t)}, \tag{16}$$

$$p_i(t+1) = p_i(t) + p_i(t)(1 - p_i(t))\frac{\frac{\partial \tilde{W}}{\partial p_i}}{\tilde{W}(t)}. \tag{17}$$

Proof. Equation (15) has been proven in [18]. Equation (16) directly follows. We only have to prove (17). 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 (17).

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

Corollary 4. *UMDA solves the continuous optimization problem* $\operatorname{argmax} \tilde{W}(\boldsymbol{p})$ *on the unit cube* $[0, 1]^n$. *The continuous problem is an extension of the discrete optimization problem* $\operatorname{argmax} f(\boldsymbol{x})$.

Equation (17) is especially suited for theoretical analysis. It has first been proposed by Wright [32]. Wright's 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. This is discussed next.

3.1 Computing the Average Fitness

In mathematical terms the discrete optimization problem with variables $\boldsymbol{x}$ is transformed into a continuous optimization problem with variables $\boldsymbol{p}$. We will take a closer look on this transformation. For notational convenience, we introduce a multi-index $\alpha = (\alpha_1, \ldots, \alpha_n)$, and define

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

Definition 5. The representation of a binary discrete function using the ordering according to function values is given by

$$f(\boldsymbol{x}) = f(0, \ldots, 0)(1 - x_1) \cdots (1 - x_n) + \cdots + f(1, \ldots, 1) x_1 \cdots x_n. \tag{18}$$

The representation using the ordering according to variables is

$$f(\boldsymbol{x}) = \sum_\alpha a_\alpha x^\alpha. \tag{19}$$

$\max\{|\alpha|_1 : \ a_\alpha \neq 0\}$ is called the order of the function.

In both representations the function is linear in each variable x_i. The following two lemmas are obvious.

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

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

Lemma 3. $\tilde{W}(p) := \bar{f}(t)$ *is an extension of $f(x)$ to the unit cube* $[0, 1]^n$. *There exist two representations for $\tilde{W}(p)$, given by*

$$\tilde{W}(p) = f(0, \ldots, 0)(1 - p_1) \cdots (1 - p_n) + \cdots + f(1, \ldots, 1) p_1 \cdots p_n \tag{20}$$

$$\tilde{W}(p) = \sum_\alpha a_\alpha p^\alpha. \tag{21}$$

Equation (21) can also be used to compute the derivative. It is given by

$$\frac{\partial \tilde{W}(p)}{\partial p_i} = \sum_{\alpha | \alpha_i = 1} a_\alpha p^{\alpha'}, \tag{22}$$

with $\alpha'_i = 0, \alpha'_j = \alpha_j$. If the function is of a low order, the partial derivatives can be easily evaluated. This allows to compute the difference equations for p_i. In special cases the difference equation can even be solved analytically. We will discuss examples later. First we have to show that proportionate selection has a serious drawback, both for breeding of livestock as well for evolutionary algorithms. It selects too weakly for optimization purposes.

3.2 The Selection Problem

Fitness proportionate selection is the undisputed selection method in population genetics. It is considered to be a model for *natural selection*. But the selection strongly depends on the fitness values. 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. 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.

We use mainly truncation selection in our algorithms. Another popular scheme is *tournament selection of size k*. Here k individuals are randomly chosen. The best individual is taken as parent. Unfortunately the mathematics for both selection methods is more difficult. Analytical results for tournament selection have been obtained by Mühlenbein [18].

3.3 Tournament Selection

We model binary tournament selection as a game. Two individuals with genotype $\boldsymbol{x}$ and $\boldsymbol{y}$ "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

$$a_{xy} = \begin{cases} 2 & \text{when} \quad f(\boldsymbol{x}) > f(\boldsymbol{y}), \\ 1 & \text{when} \quad f(\boldsymbol{x}) = f(\boldsymbol{y}), \\ 0 & \text{when} \quad f(\boldsymbol{x}) < f(\boldsymbol{y}). \end{cases}$$

With some effort one can show that

$$\sum_{\boldsymbol{x}} \sum_{\boldsymbol{y}} p(\boldsymbol{x}, t) a_{xy} p(\boldsymbol{y}, t) = 1. \tag{23}$$

After a round of tournaments the genotype frequencies are given by

$$p^s(\boldsymbol{x}, t+1) = p(\boldsymbol{x}, t) \sum_{\boldsymbol{y}} a_{xy} p(\boldsymbol{y}, t). \tag{24}$$

If we set

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

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

$$p(x_i, t+1) = p(x_i, t) \cdot \bar{B}_i(t), \tag{25}$$

$$\bar{B}_i(t) = \sum_{\boldsymbol{x}, X_i = x_i} b(\boldsymbol{x}, t) \prod_{\substack{j=1 \\ j \neq i}}^{n} p(x_j, t). \tag{26}$$

The difference equation for binary tournament selection is more difficult than for proportionate selection: $\bar{B}_i$ is quadratic in $p(x_j)$. 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 analysis is still more difficult for truncation selection. Therefore breeders have developed a macroscopic theory using average fitness and variance of the population.

3.4 The Science of Breeding

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,

$$S(t) = W(\boldsymbol{p}^s(t+1)) - W(\boldsymbol{p}(t)). \tag{27}$$

Similarly the response $R(t)$ is defined:

$$R(t) = W(\boldsymbol{p}(t+1)) - W(\boldsymbol{p}(t)). \tag{28}$$

Next a linear regression is done,

$$R(t) = b(t) S(t), \tag{29}$$

where $b(t)$ is called *realized heritability*. The selection differential can often be approximated by

$$S(t) \approx I_\tau V^{\frac{1}{2}}(t), \tag{30}$$

where I_τ is called the *selection intensity*. V is the variance of the fitness distribution. Combining the two equations we obtain the *famous equation for the response to selection*:

$$R(t) = b(t) I_\tau V^{\frac{1}{2}}(t). \tag{31}$$

The most difficult problem is to estimate $b(t)$. Breeders use the estimate

$$b(t) \approx \frac{V_A(t)}{V(t)}, \tag{32}$$

where $V_A(t)$ denotes the additive genetic variance. For UMDA it is defined as

$$V_A(t) = \sum_{i=1}^{n} \sum_{x_i} p(x_i, t) \left(\frac{\partial W}{\partial p(x_i)} - W(t) \right)^2 . \tag{33}$$

These equations are discussed in depth in [18]. For the special case that all univariate marginal distribution are equal, i.e., $p_i := p$, the response to selection equation gives a difference equation for p. Thus it might be possible to obtain an analytical solution for $p(t)$.

We cite from [18] the analytical solutions for the linear function $OneMax(n) = \sum_i x_i$. For completeness we give the difference equation and its solution.

Theorem 6. *If in the initial population all univariate marginal frequencies are identical to $p_0 > 0$, then we obtain for UMDA and OneMax*

$$R(t) = 1 - p(t), \tag{34}$$

$$p(t) = 1 - (1 - p_0)(1 - \frac{1}{n})^t, \tag{35}$$

for proportionate selection, and

$$R(t) \approx I_\tau \sqrt{np(t)(1 - p(t))}, \tag{36}$$

$$p(t) \approx 0.5 \left(1 + \sin\left(\frac{I_\tau}{\sqrt{n}} t + \arcsin(2p_0 - 1) \right) \right), \tag{37}$$

for truncation selection.

These solutions perfectly match the figures obtained from actual UMDA runs. In figure 1, the analytical results for proportionate selection and truncation selection with $\tau = 0.3$ are almost identical to the simulation results.

Using proportionate selection, it takes the population 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. Truncation selection with $\tau = 0.6$ behaves very similarly to tournament selection. This is known from [18].

The theory of breeding uses macroscopic variables, the average and the variance of the population. But there exists 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-Bennett and Shapiro [26] 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.

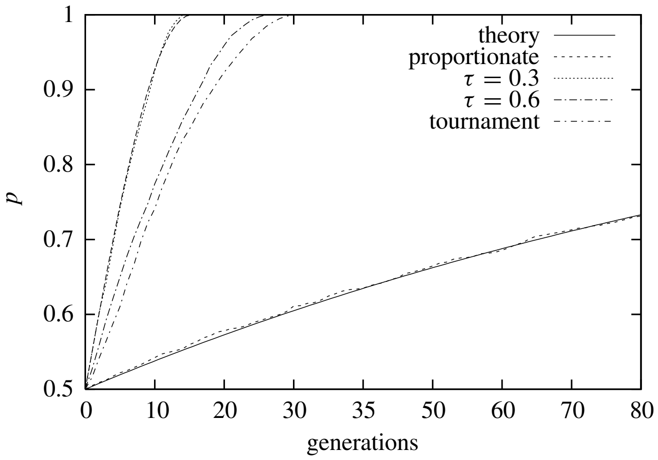

Fig. 1. Comparison of selection methods for *OneMax*(128).

Before we numerically show the optimization power of UMDA, we shortly discuss other methods also using univariate marginal distributions.

4 Optimization Methods Using Univariate Distributions

The importance of using univariate marginal distributions has been independently discovered by several researchers. We just discuss PBIL of Baluja and Caruana [2] and *ant colony optimization* by Dorigo & Di Caro [5]. PBIL does not use strict Darwinian selection in populations, but the the probabilities are updated according to

$$p(x_i, t+1) = p(x_i, t) + \lambda\big(p^s(x_i, t) - p(x_i, t)\big). \tag{38}$$

The string $\boldsymbol{x}$ is generated as before:

$$p(\boldsymbol{x}, t+1) = \prod_{i=1}^{n} p(x_i, t+1). \tag{39}$$

The convergence speed of this algorithm critically depends on λ. For $\lambda = 0$ we have a random search, for $\lambda = 1$ we obtain UMDA. The smaller λ, the slower the convergence speed. This problem is discussed in [18]. Our numerical experiments indicate that the UMDA method is to be preferred, because it is very difficult to choose λ for a given problem.

In principle, univariate marginal distributions are also used in ant colony optimization (ACO, [5]). For each ant k, a probability p_{ij} is computed to move from state i

to state j. The equation is given by

$$p_{ij}^k := \begin{cases} \frac{\tau_{ij}(t)}{\sum_{j\in N(i)} \tau_{ij}(t)} & \text{for } j \in N(i), \\ 0 & \text{for } j \notin N(i). \end{cases} \tag{40}$$

The variable τ_{ij} is updated according to

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \Delta\tau_{ij}. \tag{41}$$

Here $p_{ij} = p_i(x_j)$ plays the role of our univariate marginal distributions. If all states j are contained in the neighbourhood $N(i)$ then we have an UMDA algorithm with integer variables. A solution $\boldsymbol{x}$ is generated with probability

$$p(\boldsymbol{x}) = \prod_i p_i(x_j).$$

Ant colony optimization is mainly applied to constrained combinatorial optimization problems like the travelling salesman problem (TSP) or the quadratic assignment problem. Here the neighbourhoods have to be dynamically changed. We take TSP as example. If city l is chosen to be at a certain place of the tour, it is not allowed to be chosen again. ACO solves this problem by setting all p_{il}^k with $i > l$ to 0. The other p_{ij}^k values are renormalized, so that the sum of the probabilities of all feasible moves is 1. Thus ACO constructs feasible solutions. But the UMDA theory does not apply because of the renormalization of the probabilities.

5 The Replicator Equation and Combinatorial Optimization

For mathematical analysis the equations for discrete generations are often approximated by equations with continuous time. That is, the *difference equations* are approximated by *differential equations*. The reason is that the mathematical analysis of differential equations is easier. In general, this approximation is a complicated issue. We consider only the proportionate selection equation (2).

Let $n_x(t)$ denote the number of occurences of string $\boldsymbol{x}$ at generation t. Let $N(t)$ denote the size of the population. Then we have $p(\boldsymbol{x}, t) = n_x(t)/N(t)$.

Lemma 4. *If the occurrences grow according to their fitness,*

$$n_x(t+1) = f(\boldsymbol{x}) n_x(t),$$

then

$$p(\boldsymbol{x}, t+1) = p(\boldsymbol{x}, t)\frac{f(\boldsymbol{x})}{\bar{f}(t)}.$$

This is just the equation describing proportionate selection. It is obtained from

$$N(t+1) = \sum_x f(x) \frac{n_x(t)}{N(t)} N(t) = \bar{f}(t) N(t).$$

The corresponding lemma for continuous t is as follows:

Lemma 5. *If the occurrences grow according to their fitness,*

$$\frac{dn_x(t)}{dt} = f(\boldsymbol{x}) n_x(t),$$

then

$$\frac{dp(\boldsymbol{x},t)}{dt} = p(\boldsymbol{x},t)\big(f(\boldsymbol{x}) - \bar{f}(t)\big). \tag{42}$$

Proof. We have

$$\begin{aligned} \frac{dp(\boldsymbol{x},t)}{dt} &= \frac{\frac{dn_x}{dt} N - \frac{dN}{dt} n_x}{N^2} \\ &= \frac{n_x(t)}{N(t)} \big(f(x) - \bar{f}(t)\big). \end{aligned}$$

Equation (42) is a special case of a differential equation defined by

$$\frac{dp_\alpha}{dt} = p_\alpha \left(f_\alpha(\boldsymbol{p}) - \sum_x p_\alpha f_\alpha(\boldsymbol{p}) \right), \tag{43}$$

where p_α now denotes $p(\boldsymbol{x})$. This equation is called the *replicator equation*. It is of great importance in many fields connected to biology. For a general investigation of these equations the reader is referred to [11]. Interesting discussions can also be found in [6] and [25].

We introduce an extension of the replicator equation, called the *diversified replicator equation* [30].

Definition 6. Let $p_{\alpha k} \geq 0$ be defined for $1 \leq k \leq m$ with $\sum_k^m p_{\alpha k} = 1$. Then a diversified replicator equation is defined for discrete time by

$$p_{\alpha k}(t+1) - p_{\alpha k}(t) = p_{\alpha k}(t) \frac{f_{\alpha k}(\boldsymbol{p}(t)) - \sum_{k=1}^m p_{\alpha k}(t) f_{\alpha k}(\boldsymbol{p}(t))}{\sum_{k=1}^m p_{\alpha k} f_{\alpha k}(\boldsymbol{p}(t))}. \tag{44}$$

The corresponding differential equation is given by

$$\frac{dp_{\alpha k}}{dt} = p_{\alpha k} \left(f_{\alpha k}(\boldsymbol{p}) - \sum_{k=1}^m p_{\alpha k} f_{\alpha k}(\boldsymbol{p}) \right). \tag{45}$$

The replicator and the diversified replicator equation differ in the constraints. We have $\sum_\alpha p_\alpha = 1$ for the replicator equation and $\sum_k p_{\alpha k} = 1$ for the diversified replicator equation. Our central UMDA equation (16) is a special case of a diversified replicator equation. This can be seen by setting $k \in \{0, 1\}$, $\alpha \in \{1, \ldots, n\}$, $p_{\alpha k} := p(x_\alpha = k)$ and

$$f_{\alpha k}(\boldsymbol{p}) = \frac{\partial W}{\partial p_{\alpha k}}. \tag{46}$$

Thus (16) defines a *gradient system* with the potential $W(\boldsymbol{p})$. Gradient systems have nice properties. We just give one example.

Theorem 7. *If the diversified replicator equation is a gradient system, the potential W never decreases, i.e.,*

$$\frac{dW}{dt} \geq 0. \tag{47}$$

Proof. We compute

$$\begin{aligned}
\frac{dW}{dt} &= \sum_\alpha \sum_k \frac{\partial W}{\partial p_{\alpha k}} \frac{dp_{\alpha k}}{dt} \\
&= \sum_\alpha \sum_k \frac{\partial W}{\partial p_{\alpha k}} p_{\alpha k} \left(f_{\alpha k}(\boldsymbol{p}) - \sum_{k=1}^{m} p_{\alpha k} f_{\alpha k}(\boldsymbol{p}) \right) \\
&= \sum_\alpha \left(\sum_k p_{\alpha k} f_{\alpha k}^2(\boldsymbol{p}) - (\sum_k p_{\alpha k} f_{\alpha k}(\boldsymbol{p}))^2 \right) \\
&\geq 0.
\end{aligned}$$

The diversified replicator equation has been used by Voigt [30] and Mühlenbein [19] to solve combinatorial problems. In their method the difference equations are iterated until all probabilities p_{ik} have converged. This method poses a major difficulty. It stops at local maxima in the interior of the unit cube. But these points are not feasible. Therefore Voigt [30] has developed adaptive techniques which drive the probabilities to the corner of the simplices.

Using the UMDA algorithm is a much simpler solution to the problem. We interpret p_{ik} as the probability that x_i is set to k. We generate a population of solutions, select the better ones and compute the probabilities $p^s(x_i)$ of the selected strings. The new population is generated by the usual UMDA method

$$p(\boldsymbol{x}, t+1) = \prod_i p^s(x_i). \tag{48}$$

The diversified replicator equation has been used in [30,19] to solve combinatorial problems like the *graph partitioning problem* (GPP) and the TSP. It is worthwhile to test UMDA on this problem.

We will numerically investigate UMDA in the next section. The application domain is the optimization of discrete functions.

6 Numerical Results for UMDA

This section solves the problem put forward by Mitchell et al. [14]: to understand the class of problems for which genetic algorithms are most suited, and in particular, for which they will outperform other 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.

6.1 Royal Road Function

We discuss the Royal Road function R_1, which was used by Mitchell et al. [14]. It is defined as follows:

$$R_1(l, x) = \sum_{i=0}^{l-1} \prod_{j=1}^{8} x_{8i+j}. \tag{49}$$

The function is of order 8. The Building Block Hypothesis (BBH, [12]) 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.

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

Table 3. Mean function evaluations for Royal Road(8).

Table 3 confirms and extends the results of Mitchell et al. [14]. The really bad performance of SGA is mainly a result of proportionate selection. UMDA with proportionate selction (UMDA 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 [17]. But it performs only good if the fitness function never decreases with increasing number of bits. Almost identical performance to

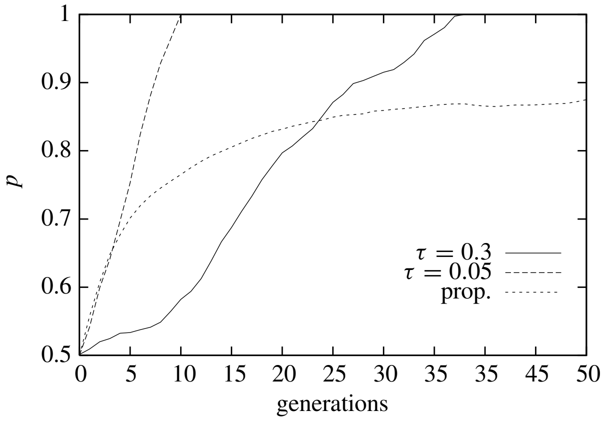

Fig. 2. Convergence of Royal Road.

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 7.

Figure 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 proportionate selection, if we added 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

$$\tilde{W}(\boldsymbol{p}) = \sum_{i=0}^{l-1} \prod_{j=1}^{8} p_{8i+j},$$

$$\frac{\partial \tilde{W}}{\partial p_k} = \prod_{\substack{j=1 \\ 8i+j \neq k}}^{7} p_{8i+j}, \qquad \text{for } 8i \leq k < 8i+8.$$

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 $f(1,\ldots,1)$. 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

$$V_8(t) = 8p(t)^8(1 - p(t)^8). \tag{50}$$

We can now formulate the theorem.

Theorem 8. *If all univariate marginal distributions are identical to $p(t)$ and $p(0) = p_0$ then we obtain for proportionate selection*

$$p(t+1) - p(t) = \frac{1 - p(t)}{8}, \tag{51}$$

$$p(t) = 1 - (1 - p_0)(\tfrac{7}{8})^t. \tag{52}$$

For truncation selection with threshold τ we approximately get

$$R(t) \approx b(t) I_\tau \sqrt{8p(t)^8(1 - p(t)^8)}, \tag{53}$$

$$p(t)^8 \approx 0.5 \left(1 + \sin\Big(\frac{b(t)I_\tau}{\sqrt{8}} t + \arcsin(2p_0^8 - 1)\Big)\right). \tag{54}$$

Proof. The conjectures for proportionate selection directly follow from (17). 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 (37).

In figure 3, a comparison between the theoretical solution and a simulation run is made. The simulation run gives slightly larger values of W. The reason is that in finite populations there are random fluctuations. UMDA is not able to keep $p_i = p_j := p$. We use the estimate $p(t) = 1/n \sum_i p_i(t)$ to compute W for the figure. But the similarity between the theory and the simulations is still impressive.

In order to apply (54) 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 $b(t) \propto t$. Figure 4 shows a comparison between (54) and a simulation with truncation threshold 0.05. 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 $b(t)$.

6.2 Multimodal Functions Suited for UMDA Optimization

Equation (16) 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 as first

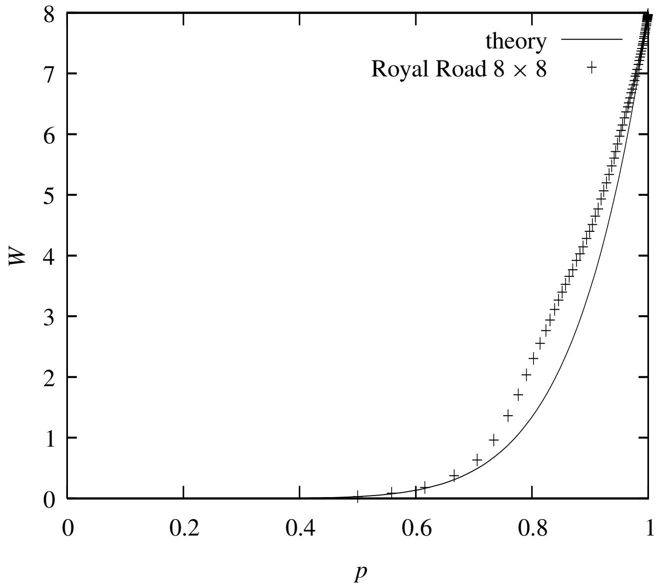

Fig. 3. Proportionate selection for Royal Road: theory and simulation.

example the function *BigJump*. It is defined as follows, with $|\boldsymbol{x}|_1 = \sum x_i$ equal to the number of 1-bits:

$$BigJump(n, m, k, \boldsymbol{x}) := \begin{cases} |\boldsymbol{x}|_1 & \text{for } 0 \le |\boldsymbol{x}|_1 \le n - m, \\ 0 & \text{for } n - m < |\boldsymbol{x}|_1 < n, \\ k \cdot n & \text{for } |\boldsymbol{x}|_1 = n. \end{cases} \tag{55}$$

The bigger m, the wider the valley. The parameter k can be increased to give bigger weight to the maximum. For $m = 1$ we obtain the popular *OneMax* function defined by $OneMax(n) = |x|_1$.

BigJump depends only on the number of bits on. We assume that all $p(x_i = 1)$ are identical to a single value denoted as $p(t)$. Then W depends only on one parameter, p. $W(p)$ is shown for $m = 30$ and $k = 20$ in figure 5. In contrast to the discrete function the average fitness $W(p)$ looks fairly smooth. The open circles are the values of $p(t)$ determined by an UMDA run, setting $p(t) := 1/n \sum_i p_i(t)$. Note how closely the simulation follows the theoretical curve. Because we use discrete generations in UMDA the population is able to pass the local minimum at about $p = 0.83$.

This simple example confirms in a nutshell the results of our theory. *Evolutionary algorithms transform the original fitness landscape given by $f(\boldsymbol{x})$ into a fitness landscape defined by $\tilde{W}(\boldsymbol{p})$. This transformation smoothes the rugged fitness land-*

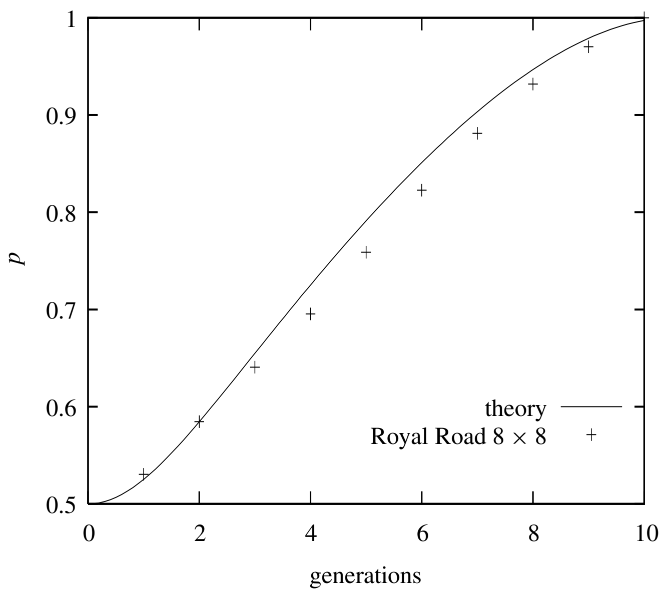

Fig. 4. Truncation selection for Royal Road: theory and simulation.

scape $f(\boldsymbol{x})$. In these landscapes simple evolutionary algorithms will find the global optimum.

A still more spectacular example is the *Saw* landscape. The definition of the function can be extrapolated from figure 6. 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 $W(p)$ is fairly smooth. An example is shown in figure 7. Whereas $f(x)$ has 5 isolated peaks, $W(p)$ has three plateaus, a local peak and the global peak. Therefore we expect that UMDA should be able to cross the plateaus and terminate at the local peak. This behavior can indeed be observed in figure 7. Furthermore, as predicted by (16), the progress of UMDA slows down on the plateaus.

Next we will investigate UMDA with truncation selection. We have not been able to derive precise analytical expressions. In figure 8 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 $\tilde{W}(p)$. 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!

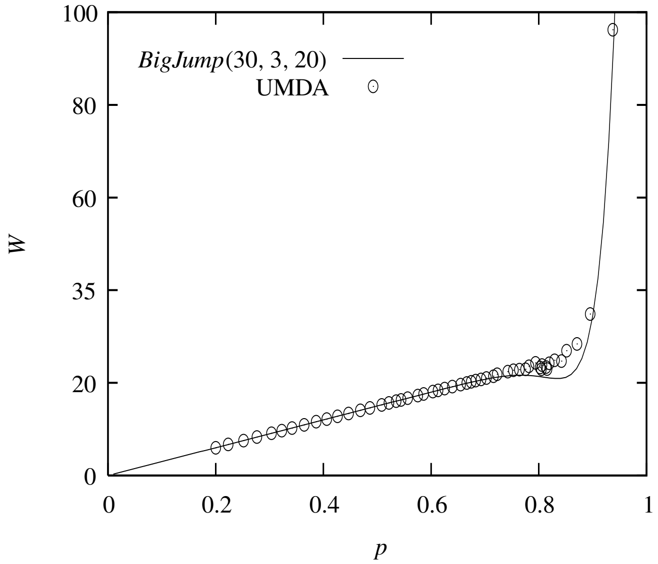

Fig. 5. *BigJump*(30,3,20), UMDA, p versus average fitness, population size 2000.

These two examples show that UMDA can solve difficult multimodal 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.

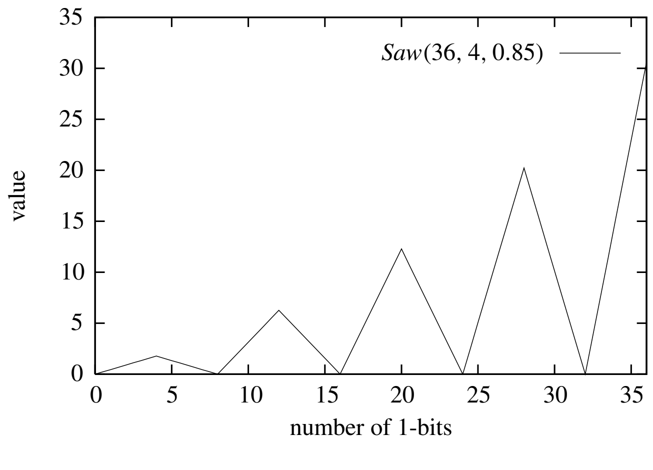

Fig. 6. Definition of *Saw*(36,4,0.85).

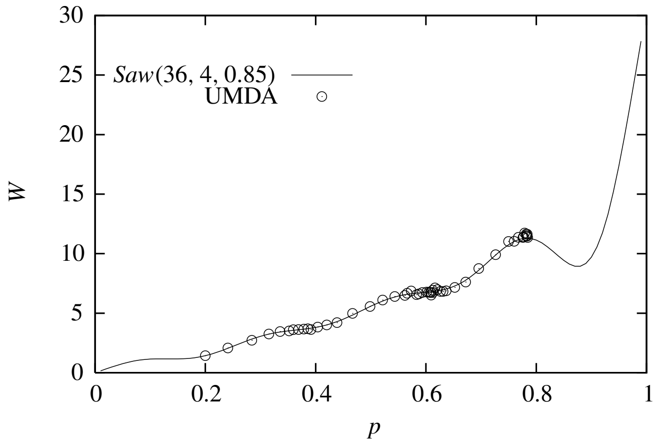

Fig. 7. *Saw*(36,4,0.85), UMDA, p versus average fitness, population size 2000.

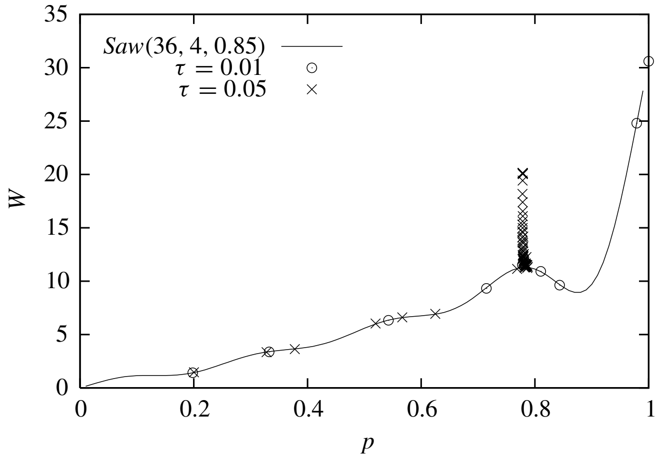

Fig. 8. Results with normal and strong selection.

6.3 Deceptive Functions

But there are 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

$$Deceptive(\boldsymbol{x}, k) := \begin{cases} k - 1 - |\boldsymbol{x}|_1 & \text{for } 0 \le |\boldsymbol{x}|_1 < k, \\ k & \text{for } |\boldsymbol{x}|_1 = k. \end{cases} \tag{56}$$

The global maximum is isolated at $x = (1, \dots, 1)$. A deceptive function of order n is a needle-in-a-haystack problem. This is far too difficult to optimize for any

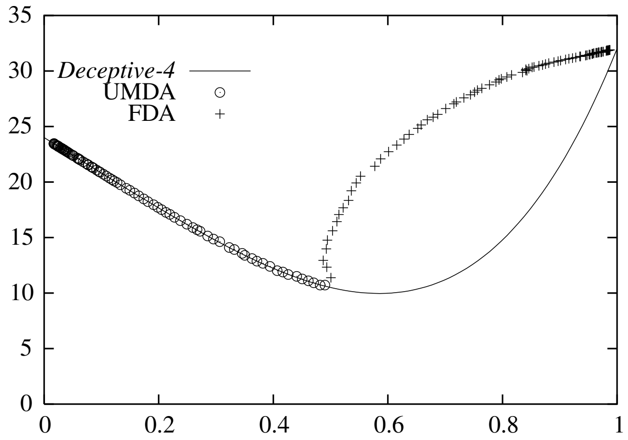

Fig. 9. Average fitness $W(p)$ for UMDA and FDA for *Deceptive*(36,4).

optimization method. We simplify the optimization problem by adding l distinct *Deceptive*(k)-functions to give a fitness function of size $n = l \times k$,

$$Deceptive(n,k) = \sum_{i=1,k+1,\ldots}^{n} Deceptive\big((x_i, x_{i+1}, \ldots, x_{i+k-1}), k\big). \tag{57}$$

This function is also deceptive. The local optimum $x = (0, \ldots, 0)$ is surrounded by good fitness values, whereas the global optimum is isolated.

In figure 9, we show the average fitness $W(p)$ and an actual UMDA run. Starting at $p(0) = 0.5$, UMDA converges to the local optimum $\boldsymbol{x} = (0, \ldots, 0)$. UMDA will converge to the global optimum if it starts near to the optimum e.g. $p(0) \geq 0.6$. Also shown is a curve derived from FDA. FDA uses fourth order marginal distributions. It converges to the global optimum, even if the initial population is generated randomly. But one can see in the figure that $p(t)$ also decreases first. FDA is discussed in section 7.

6.4 Numerical Investigations of the Science of Breeding

In this section we show that the science of breeding can be very usefully applied to evolutionary optimization. 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 $f(\boldsymbol{x}) = \prod_i (1-s)^{1-x_i}$.

Figure 10 confirms the theoretical results from section 2 (V_A and V are multiplied by 10 in this figure). Additive genetic variance is identical to the variance and the heritability is 1. The function is highly nonlinear of order n, but nevertheless it is

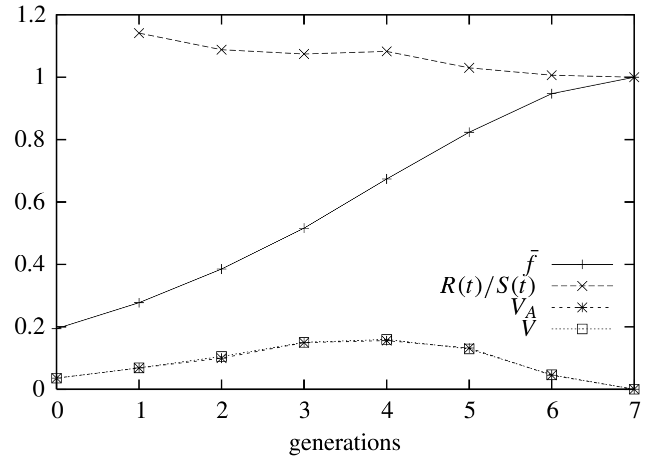

Fig. 10. Heritability and variance for a multiplicative function.

easy to optimize. The function has also been investigated by Rattray and Shapiro [27]. They have not observed that the population remains in linkage equilibrium, making their calculations very difficult.

The function *Saw* is difficult to optimize. We see in figure 11 that for a long time there is no progress. An increase of the average fitness occurs at generations 6 till 9. During this time the additive genetic variance V_A is higher. But the heritability is almost zero almost anywhere.

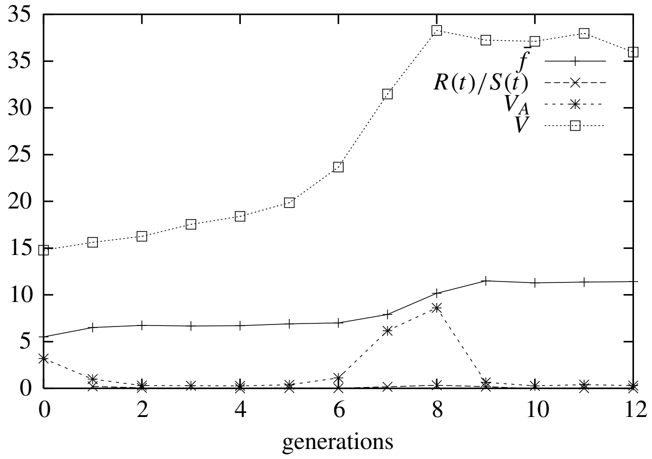

Fig. 11. Heritability and variance for function *Saw*.

An interesting case is the function *Deceptive-4*. In figure 12, the function is optimized for 32 bits. As predicted by the theory, UMDA converges to the local optimum $\boldsymbol{x} = (0, \ldots, 0)$. Heritability is almost zero at the beginning, indicating that the optimization problem is difficult. In the beginning the competition between setting the genes to 0 or to 1 is undecided. UMDA decides to go in 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.

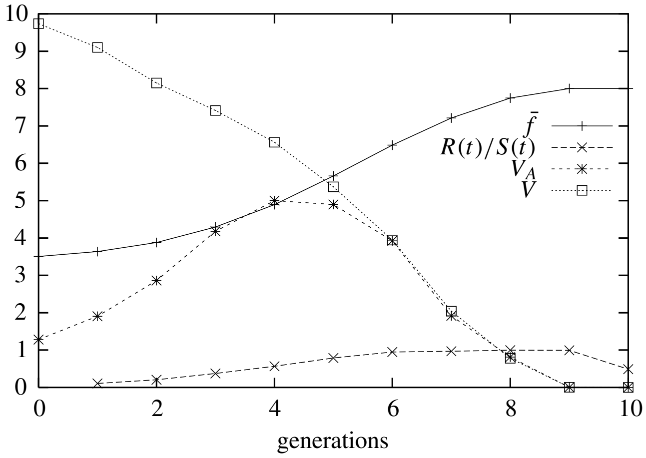

Fig. 12. Heritability and variance for *Deceptive-4*: $\tau = 0.3$.

The examples demonstrate that it is worthwhile to compute the quantities used for a scientific breeding programme. 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 multimodal 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 marginal distributions of higher order.

7 FDA – The Factorized Distribution Algorithm

For the mathematical analysis we will use Boltzmann selection. Boltzmann selection can be seen as proportionate selection applied to the transformed function $F(\boldsymbol{x}) = \exp(\beta f(\boldsymbol{x}))$.

Definition 7. For Boltzmann selection the distribution after selection is given by

$$p^s(\boldsymbol{x}, t) = p(x, t)\frac{e^{\beta f(x)}}{W_\beta}, \tag{58}$$

where $\beta > 0$ is a parameter, also called the inverse temperature, and

$$W_\beta = \sum p(\boldsymbol{x}, t)e^{\beta f(x)}$$

is the weighted average of the population.

For Boltzmann distributions we have proven a factorization theorem for the distribution $p(\boldsymbol{x}, t)$ and convergence for an algorithm using this factorization [20]. The proof is simple, because if $p(x, t)$ is a Boltzmann distribution with factor β_1 and Boltzmann selection is done with factor β_2, then $p(\boldsymbol{x}, t + 1) = p(\boldsymbol{x}, t)$ is a Boltzmann distribution with factor $\beta = \beta_1 + \beta_2$.

Theorem 9. *Let $p(\boldsymbol{x}, 0)$ be randomly distributed. Let $\beta_1, \ldots, \beta_{t-1}$ be the schedule of the inverse temperature for Boltzmann selection. Then the distribution is given by*

$$p(\boldsymbol{x}, t) = \frac{e^{\beta f(\boldsymbol{x})}}{Z_\beta}, \tag{59}$$

where $\beta = \sum_{i=1}^{t-1} \beta_i$. Z_β is the partition function $Z_\beta = \sum_{\boldsymbol{x}} e^{\beta f(\boldsymbol{x})}$.

Equation (59) is a complete analytical solution of the dynamics. But it cannot be used for an algorithm. $p(\boldsymbol{x}, t)$ consists of $2^n - 1$ variables. Therefore the amount of computation is exponential. But there are many cases where the distribution can be factored into conditional marginal distributions each depending only on a small number of parameters. We recall the definition of conditional probability.

Definition 8. The conditional probability $p(\boldsymbol{x}|\boldsymbol{y})$ is defined as

$$p(\boldsymbol{x}|\boldsymbol{y}) = \frac{p(\boldsymbol{x}, \boldsymbol{y})}{p(\boldsymbol{y})}. \tag{60}$$

From this definition the following theorem easily follows.

Theorem 10 (Bayesian Factorization). *Each probability can be factored into*

$$p(\boldsymbol{x}) = p(x_1)\prod_{i=2}^{n} p(x_i|pa_i). \tag{61}$$

Proof. By definition of conditional probabilities we have

$$p(\boldsymbol{x}) = p(x_1) \prod_{i=2}^{n} p(x_i | x_1, \cdots, x_{i-1}). \tag{62}$$

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 $p(x_i | x_1, \cdots, x_{i-1}) = p(x_i | pa_i)$.

pa_i are called the parents of variable X_i. This factorization defines a directed graph. In the context of graphical models the graph and the conditional probabilities are called a Bayesian network [13,7]. The factorization is used by the Factorized Distribution Algorithm (FDA).

FDA

STEP 0: Set $t \Leftarrow 0$. Generate $N \gg 0$ points randomly.
STEP 1: Selection.
STEP 2: Compute the conditional probabilities $p^s(x_i | pa_i, t)$ using the selected points.
STEP 3: Generate a new population according to

$$p(x, t+1) = \prod_{i=1}^{n} p^s(x_i | pa_i, t).$$

STEP 4: If termination criteria is met, FINISH.
STEP 5: Set $t \Leftarrow t + 1$. Go to STEP 2.

FDA can be used with an exact or an approximate factorization. It is not restricted to Bayesian factorization. FDA uses *finite samples* of points to estimate the conditional distributions. Convergence of FDA to the optimum will depend on the size of the samples.

If the factorization does not contain conditional marginal distributions, but only marginal distributions, FDA can be theoretically analyzed. The difference equations of the marginal distributions are of the form given in (16) [15].

The amount of computation of FDA depends on the size of the population (N) and the number of variables used for the factors. There exist many problems where the size of the factors is bounded by k independent from n. In this case FDA is very efficient [16]. But for the function *BigJump* an exact factorization needs a factor of size n. Then the amount of computation of FDA is exponential in n. We have seen before that for *BigJump*, UMDA will already find the global optimum. Thus an exact factorization is not a necessary condition for convergence. But it is necessary if we want to be sure that the optimum is found.

8 Finite Populations

In finite populations, convergence of UMDA or FDA can only be probabilistic. Since UMDA is a specialized FDA algorithm, it is sufficient to discuss FDA. This section is extracted from [16].

Definition 9. 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

$$N^*(\epsilon) = \min_N P_{conv}(N) \geq 1 - \epsilon. \tag{63}$$

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

$$P_{conv}(N_1) \leq P_{conv}(N_2), \quad \text{for } N_1 \leq N_2.$$

The critical question is: how many sample points are necessary to reasonably approximate the distribution used by FDA? A general estimate from Vapnik [29] 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*. $Int(\boldsymbol{x})$ gives the integer value of the binary representation:

$$Int(n) = \sum_{i=1}^{n} 2^{i-1} x_i. \tag{64}$$

The fitness distribution of this function is not normally distributed. The function has 2^n different fitness values. We show the cumulative fixation probability in table 4 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.

Boltzmann selection with $\beta = 0.01$ 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 with $\beta = 0.01$ is very high ($N^* > 700$). For truncation selection with $\tau = 0.25$ we have $N^*(0.1) \leq 80$.

Because Boltzmann selection in finite populations critically depends on a good annealing schedule, we normally run FDA with truncation selection. This selection

t	$\tau = 0.25$ $N = 30$	$\tau = 0.5$ $N = 30$	$\tau = 0.25$ $N = 80$	$\tau = 0.5$ $N = 60$	Boltz. $N = 500$	Boltz. $N = 700$
1	0.0955	0.0035	0.0	0.0	0.2520	0.0885
2	0.4065	0.0255	0.0025	0.0095	0.2980	0.1110
3	0.5955	0.1040	0.0165	0.0205	0.3180	0.1275
4	0.6880	0.2220	0.0355	0.0325	0.3295	0.1375
5	0.7210	0.3270	0.0575	0.0490	0.3385	0.1455
6	0.7310	0.4030	0.0695	0.0630	0.3435	0.1510
7	0.7310	0.4470	0.0740	0.0715	0.3505	0.1555
8	0.7310	0.4705	0.0740	0.0780	0.3530	0.1565
9	0.7310	0.4840	0.0740	0.0806	0.3555	0.1575

Table 4. Cumulative fixation probability for *Int*(16). Truncation selection vs. Boltzmann selection with $\beta = 0.01$.

method is a good compromise. It has an important property, which we formulate as an empirical law. It has been confirmed by many numerical experiments.

Empirical law. *Let ϵ be reasonable small e.g. $\epsilon = 0.1$. Then the number of generations to converge to the optimum remains constant for $N \geq N^*(\epsilon)$,*

$$GEN_e(N^*(\epsilon)) = GEN_e(N) = GEN_e(N = \infty), \quad \text{for } N \geq N^*(\epsilon). \tag{65}$$

Truncation selection has a free parameter, the truncation threshold τ. It seems obvious that the smaller the threshold τ, the larger N^* has to be. But numerical experiments have shown that there exists a threshold $\tau_{\min}$ which leads to a minimal $N^*_{\min}$. This means that N^* also increases for very low selection. The reason for this phenomenon is genetic drift. Slow selection leads to a large number of generations which increases the probability of gene fixation. This problem has been first investigated by Mühlenbein and Schlierkamp-Voosen [21] for *OneMax* and genetic algorithms.

We denote the critical population size for given τ by $N^*(\epsilon, \tau)$. Because ϵ is fixed, we omit ϵ and write just $N^*(\tau)$. For *Int* we have approximately computed $N^*(\tau)$ by a a Markov chain analysis. The Markov model was simplified, therefore we formulate the result as a conjecture.

Conjecture 1. Let $\tau_k = 2^{-k}$. For FDA with fitness function *Int*, the critical population size $N^*(\tau)$ is approximately given by

$$N^*(\tau_k) \approx N^*(\tau_1) * 2^{\frac{k-1}{2}}, \quad \text{for } k \geq 1.$$

If $N^*(\tau)$ has been determined, then an optimal truncation threshold τ_{opt} can be computed. This threshold gives the minimum number of function evaluations *FE*.

Definition 10. The optimum truncation threshold τ_{opt} is defined by

$$\tau_{\text{opt}} = \min_{\tau} FE(\tau) = \min_{\tau} GEN_e(\tau) * N^*(\tau). \tag{66}$$

In general τ_{opt} is different from $\tau_{\min}$ which needs the minimal population size. The following result follows from $k > 1$ from the above conjecture.

Empirical Law. *For Int the optimal truncation threshold τ is contained in the interval* $[0.125, 0.4]$.

Proof. Part of the result follows from the approximate formulas. For $\tau = 2^{-k}$ we obtain using the critical population size

$$FE \approx \frac{n}{k} \times N^*(\tau_1) \times 2^{\frac{k-1}{2}} \propto \frac{1}{\sqrt{\tau}\log(1/\tau)}, \quad \text{for } k \geq 1. \tag{67}$$

The minimum is at $k = 0.5(1 + \sqrt{17})$.

This short discussion indicates the difficulty of the critical population size problem. In principle UMDA depends only on one parameter, the critical population size. But this size depends on the function to be optimized. Numerical estimates are very difficult. The finite size problem is discussed in a different context in the next section. There we introduce an algorithm which computes a good factorization from search points.

9 LFDA – Learning a Bayesian Factorization

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 chapter to discuss this problem in more detail. The interested reader is referred to the two papers by Heckerman and Friedman et al., both in [13].

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.

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

$$\mathrm{MDL}(B, D) = -\,\mathrm{ld}(P(B)) + M \cdot H(B, D) + \frac{1}{2} PA \cdot \mathrm{ld}(M), \tag{68}$$

with $\mathrm{ld}(x) := \log_2(x)$. $P(B)$ 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

$$H(B, D) = -\sum_{i=1}^{n} \sum_{pa_i} \sum_{x_i} \frac{m(x_i, pa_i)}{M} \,\mathrm{ld}\, \frac{m(x_i, pa_i)}{m(pa_i)}, \tag{69}$$

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. $0.5PA \cdot \mathrm{ld}(M)$ is the length required to code the parameter of the model with precision $1/M$. Normally one would need $PA \cdot \mathrm{ld}(M)$ 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 $0.5\,\mathrm{ld}(M)$ 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 ($\mathrm{ld}(L(B|D))$). Thus we arrive at the following principle:

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

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 to be maximized is called *BIC*:

$$BIC(B, D, \alpha) = -M \cdot H(B, D) - \alpha PA \cdot \mathrm{ld}(M). \tag{70}$$

This measure with $\alpha = 0.5$ has been first derived by Schwarz [28] as the *Bayesian Information Criterion.*

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, $k_{\max}$)

STEP 0: Start with an arc-less network.
STEP 1: Add the arc (x_i, x_j) which gives the maximum increase of $BIC(\alpha)$ 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 [24] 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 [3]. The *BIC* measure with $\alpha = 1$ has also been proposed by Harik [10]. 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.

We have applied LFDA to many problems [16]. The results are encouraging. Here we only discuss the functions introduced in section 6. We recall that UMDA finds the optimum of *BigJump* and *Saw*. UMDA uses univariate marginal distributions only. Therefore its Bayesian network has no arcs.

Table 5 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 *BigJump* and *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

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
BigJump(30,3,1)	30	UMDA	200	0.3	100	0.0
	30	0.25	200	0.3	58	4.9
	30	0.5	200	0.3	96	2.0
	30	0.75	200	0.3	100	0.0
	30	0.25	400	0.3	100	0.0
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

Table 5. Numerical results for different algorithms, LFDA with BN(α, 8).

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 [33].

10 Conclusion

We have shown that evolutionary algorithms can be approximated by an algorithm which keeps the population in linkage equilibrium. This algorithm, called UMDA, transforms the discrete optimization problem $\max f(\boldsymbol{x})$ into a continuous one defined by $\max \tilde{W}(p_1, \ldots, p_n)$. $0 \leq p_i \leq 1$ is a univariate marginal distribution. With proportionate selection UMDA performs gradient ascent on $\tilde{W}$.

UMDA solves difficult multimodal optimization problems. But there are functions with highly correlated variables, where a search distribution using multivariate distributions and conditional marginal distributions has to be used. This is done by the algorithm FDA. Ultimately our theory leads to a synthesis problem: finding a good factorization for a search distribution defined by a finite sample. This problem lies in the centre of probability theory. One approach to this problem uses Bayesian networks. For Bayesian networks numerical 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 substantial higher than of UMDA. Thus UMDA should be the first algorithm to be tried in a practical algorithm. In a next step all three algorithms have to be extended to optimization problems with constraints. We believe that with distributions constraints can be easier handles than with recombination. A first step has already been made in [20].

References

1. H. Asoh and H. Mühlenbein. On the mean convergence time of evolutionary algorithms without selection and mutation. In Y. Davidor, H.-P. Schwefel, and R. Männer, editors, *Parallel Problem Solving from Nature*, Lecture Notes in Computer Science 866, pages 88–97. Springer-Verlag, 1994.
2. S. Baluja and R. Caruana. Removing the genetics from the standard genetic algorithm. In *Proc. of the 12th Intern. Conf. on Machine Learning*, Lake Tahoe, 1995.
3. R.R. Bouckaert. Properties of bayesian network learning algorithms. In R. Lopez de Mantaras and D. Poole, editors, *Proc. Tenth Conference on Uncertainty in Artificial Intelligence*, pages 102–109, Morgan Kaufmann, San Francisco, 1994.
4. F.B. Christiansen and M.W. Feldman. Algorithms, genetics and populations: The schemata theorem revisited. *Complexity*, 3:57–64, 1998.
5. M. Dorigo and G. Di Caro. The ant colony optimization meta-heuristic. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, MacGraw-Hill, New York, 1999.
6. R. Feistel and W. Ebeling. *Evolution of Complex Systems. Self-Organitation Entropy and Development*. Kluwer, Dordrecht, 1989.
7. B.J. Frey. *Graphical Models for Machine Learning and Digital Communication*. MIT Press, Cambrigde, 1998.
8. H. Geiringer. On the probability theory of linkage in mendelian heredity. *Annals of Math. Stat.*, 15:25–57, 1944.
9. D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, 1989.
10. G. Harik. Linkage learning via probabilistic modeling in the ecga. Technical Report IlliGal 99010, University of Illinois, Urbana-Champaign, 1999.
11. J. Hofbauer and K. Sigmund. *Evolutionary Games and Population Dynamics*. Cambridge University Press, Cambridge, 1998.
12. J.H. Holland. *Adaptation in Natural and Artificial Systems*. Univ. of Michigan Press, Ann Arbor, 1975/1992.
13. M.I. Jordan. *Learning in Graphical Models*. MIT Press, Cambrigde, 1999.

14. M. Mitchell, J.H. Holland, and St. Forrest. When will a genetic algorithm outperform hill climbing? *Advances in Neural Information Processing Systems*, 6:51–58, 1994.
15. H. Mühleinbein and Th. Mahnig. Convergence theory and applications of the factorized distribution algorithm. *Journal of Computing and Information Technology*, 7:19–32, 1999.
16. H. Mühleinbein and Th. Mahnig. FDA – A scalable evolutionary algorithm for the optimization of additively decomposed functions. *Evolutionary Computation*, to be published 1999.
17. H. Mühlenbein. Evolution in time and space - the parallel genetic algorithm. In G. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 316–337, Morgan-Kaufman, San Mateo, 1991.
18. H. Mühlenbein. The equation for the response to selection and its use for prediction. *Evolutionary Computation*, 5(3):303–346, 1997.
19. H. Mühlenbein, M. Gorges-Schleuter, and O. Krämer. Evolution algorithms in combinatorial optimization. *Parallel Computing*, 7:65–88, 1988.
20. H. Mühlenbein, Th. Mahnig, and A. Rodriguez Ochoa. Schemata, distributions and graphical models in evolutionary optimization. *Journal of Heuristics*, 5:215–247, 1999.
21. H. Mühlenbein and D. Schlierkamp-Voosen. The science of breeding and its application to the breeder genetic algorithm. *Evolutionary Computation*, 1:335–360, 1994.
22. H. Mühlenbein and H.-M. Voigt. Gene pool recombination in genetic algorithms. In J.P. Kelly and I.H Osman, editors, *Metaheuristics: Theory and Applications*, pages 53–62, Kluwer Academic Publisher, Norwell, 1996.
23. T. Nagylki. *Introduction to Theoretical Population Genetics*. Springer, Berlin, 1992.
24. M. Pelikan, D.E. Goldberg, and E. Cantu-Paz. Boa: The bayesian optimization algorithm. Technical Report IlliGal 99003, University of Illinois, Urbana-Champaign, 1999.
25. M. Peschel and W. Mende. *Predator-Prey-Model: Do we live in a Volterra world?* Akademie-Verlag, Berlin, 1986.
26. A. Prügel-Bennet and J.L. Shapiro. An analysis of a genetic algorithm for simple random ising systems. *Physica D*, 104:75–114, 1997.
27. L. M. Rattray and J.L. Shapiro. Cumulant dynamics in a finite population. *Theoretical Population Biology*, 1999. to be published.
28. G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 7:461–464, 1978.
29. V. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.
30. H.-M. Voigt. *Evolution and Optimization*. Akademie-Verlag, Berlin, 1989.
31. M. Vose. *The Simple Genetic Algorithm: Foundations and Theory*. MIT Press, Cambridge, 1999.
32. S. Wright. Random drift and the shifting balance theory of evolution. In K. Kojima, editor, *Mathematical Topics in Population Genetics*, Springer, Berlin, 1970.
33. Byoung-Tak Zhang, Peter Ohm, and Heinz Mühlenbein. Evolutionary induction of sparse neural trees. *Evolutionary Computation*, 5:213–236, 1997.