# Evolutionary Algorithms and the Boltzmann Distribution

**Heinz Mühlenbein**
FhG AiS
53754 Sankt Augustin

**Thilo Mahnig**
FhG AiS
53754 Sankt Augustin

**Abstract**
We perform a stochastic analysis of evolutionary algorithms. The analysis centers on the question how to efficiently compute probabilities of promising alleles derived from evolving populations under selection and how to use these probabilities to generate new points. We shortly discuss the Univariate Marginal Distribution Algorithm (*UMDA*). It uses univariate marginals to generate new search points. We extend *UMDA* to the Factorized Distribution Algorithm (*FDA*) which uses a factorization of the *Boltzmann distribution*. We describe a well known algorithm to compute a factorization based on junction trees. We explain the sampling method of *FDA* and discuss the difference to Simulated Annealing. We introduce mutation into the algorithm with the help of a *Bayesian hyper parameter*. We show that *FDA* using Boltzmann selection fulfills an equation which Holland claimed to be necessary for an almost "optimal" algorithm. We formulate *FDA* as a *population dynamics* algorithm. We conclude with a short discussion about the interdisciplinary research to approximate distributions, especially the Boltzmann distribution.

*Keywords* linkage equilibrium, marginal distributions, conditional probabilities, Wright's equation, Boltzmann distribution, factorization of distribution, Holland's schema analysis, population dynamics

## 1 Introduction

Any evolutionary algorithm using a population of points can be seen as a stochastic process. There are at least three approaches to understand and analyze stochastic processes– the *microscopic view*, the *mesoscopic view*, and the *macroscopic view*.

In the microscopic view the dynamic behavior of a population of objects is simulated. In *genetic algorithms,* for instance, a set of points is generated. From this set promising

points (points with high fitness) are selected. These points are used as the "parents" of the next set. Each run is unique, therefore a mathematical analysis is almost impossible.

Let $x$ be a vector (genotype) of size $n$, defining the configuration space. In the mesoscopic view a probability distribution $p(\mathbf{x}, t)$ is introduced. From an initial distribution $p(\mathbf{x}, 0)$ a population (ensemble) is generated. Promising points are selected. The corresponding distribution of the selected points $p^s(\mathbf{x}, t)$ is estimated and then used to generate new points. For binary variables the distribution $p(\mathbf{x}, t)$ has $2^n - 1$ free parameters. The standard stochastic analysis is based on transition matrices $M = (p(\mathbf{x}|\mathbf{x}'))$ describing the probability of a transition from configuration $\mathbf{x}$ to configuration $\mathbf{x}'$. Even for small $n$ the size of the matrix is very large. Therefore we will approximate the distribution $p(\mathbf{x}, t)$ by a small number of conditional distributions. With the *schema theory* (Holland 1992) also tried a mesoscopic analysis of genetic algorithms. We will show that schema probabilities define nothing else than marginal distributions and their corresponding subspaces. Using *conditional marginal distributions* will be the key to the mesoscopic analysis. In the macroscopic view one is interested in macroscopic variables, like the average fitness $E[f(\mathbf{x})](t) = \sum_x p(\mathbf{x}, t) f(\mathbf{x})$.

In this paper we concentrate on the mesoscopic view. First we analyze an evolutionary algorithm using univariate marginal distributions (*UMDA*) to generate a population instead of recombination and mutation of strings as it is done by genetic algorithms. *UMDA* might fail to optimize functions with strongly correlated variables. Therefore we extend *UMDA* to a population based search algorithm using a general search distribution. In our case we will use the *Boltzmann distribution*. This algorithm, called *BEDA*, converges to the set of global optima. Convergence is here defined in the strong sense of numerical analysis as approaching a stable equilibrium distribution $p_{eq}(\mathbf{x})$ which concentrates around the optima. Without mutation the equilibrium distribution is a point distribution, where the points are given by the optima. From *BEDA* we derive our main algorithm, called the Factorized Distribution Algorithm *FDA*. It is a mathematical extension of *UMDA*, and is based on a factorization of the distribution. We show that *BEDA* fulfills a differential equation Holland (Holland 1992) postulated for a good optimization algorithm. Then we describe the *junction tree algorithm* which can be used to compute a factorization. We discuss mutation and the adaptive annealing schedule *SDS*.

*FDA* can be formulated as a *population dynamics* model. Different species try to solve a common problem (the optimization of a common fitness function) by *cooperation*. We conclude the paper by surveying the ongoing interdisciplinary research to approximate distributions. It combines statistics (graphical models), artificial intelligence (belief propagation), statistical physics (advanced mean-field methods) and probabilistic logic (maximum entropy).

## 2 The Univariate Marginal Distribution Algorithm UMDA

Let $\mathbf{x} = (x_1, \ldots, x_n)$ denote a vector, $x_i \in \Lambda_i = \{0, 1, 2, \ldots, m_i\}$. We use the following conventions. Capital letters $X_i$ denote the names of variables, lower case letters $x_i$ assignments. The *distinction between the name of a variable and an assignment* is essential for the definition of marginal distributions. When there cannot be a confusion between name or assignment, we will use lower case letters and abbreviations. For notational

simplicity we will assume *binary variables* $x_i \in \{0,1\}$. Important definitions will be given for the general case.

Let a function $f : \boldsymbol{X} \to \mathbb{R}_{\geq 0}$ be given. We consider the optimization problem

$$\mathbf{x}_{opt} = \operatorname{argmax} f(\mathbf{x}) \tag{1}$$

**Definition 1** *Let* $0 \leq p(\mathbf{x},t) \leq 1$ *denote the probability of* $\mathbf{x}$ *in the population at generation* $t$. *Then* $p_i(x_k,t) = \sum_{\mathbf{x},X_i=x_k} p(\mathbf{x},t)$ *defines the univariate marginal distributions of variable* $X_i$. *Let* $\mathbf{x}_\xi$ *be a sub-vector of* $\mathbf{x}$. *Then the* **marginal distribution** *is defined as* $p(\mathbf{x}_\xi,t) = \sum_{\mathbf{x},X_\xi=x_\xi} p(\mathbf{x},t)$ *Let* $\mathbf{y},\mathbf{z}$ *be disjoint sub-vectors of* $\mathbf{x}$. *Then* **conditional probabilities** *are defined as* $p(\mathbf{y}|\mathbf{z}) = p(\mathbf{y},\mathbf{z})/p(\mathbf{z})$ *for* $p(\mathbf{z}) > 0$.

**Remark:** *Marginal distributions and the schema theory*
Marginal distributions are equivalent to *schema* probabilities introduced in (Holland 1992). We just give an example for $n = 5$. Let $\xi = (1,0,*,*,*)$ define a schema. Then the probability of the instances of schema $\xi$ in the population $P(t)$ is by definition equal to the marginal distribution $p(X_1 = 1, X_2 = 0, t)$. Thus Holland's schema analysis is nothing else than a mesoscopic analysis in the space of marginal distributions. We prefer to use the notation common in probability theory. In fact, one of the main reasons that schema theory did not come very far is the imprecise terminology. In our mesoscopic analysis conditional probabilities play an essential role. But the concept of conditional schema probabilities has not yet entered the traditional schema theory.

Note that $\sum_{x_k \in \Lambda_i} p_i(x_k,t) = 1$. This means that the univariate marginal distributions are not independent. For notational simplicity we will consider $p_i(0)$ to be the dependent parameter, which can be eliminated, if appropriate. We write $p_i(x_k)$ if just one generation is discussed. For the binary case ($x_i \in \{0,1\}$) $p(\mathbf{x},t)$ is defined by $2^n$ parameters.

Genetic algorithms are defined on a microscopic level. Given two strings, a new point is generated by recombination/crossover. A stochastic analysis of a genetic algorithm requires the computation of a recurrence equation

$$p(\mathbf{x},t+1) = \sum_{\mathbf{x}'} p(\mathbf{x}|\mathbf{x}',t)p(\mathbf{x}',t) \tag{2}$$

Here $p(\mathbf{x}|\mathbf{x}',t)$ denotes the probability for a transition from $\mathbf{x}'$ to $\mathbf{x}$ at generation $t$. Vose (Vose 1999) has derived such an equation for the Simple Genetic Algorithm with proportionate selection, crossover, and mutation. The computation of the crossover probabilities are especially difficult. Because crossover operates on two arbitrary strings $\mathbf{x}$ and $\mathbf{y}$ of the selected population, one has to use the joint distribution $p(\mathbf{x};\mathbf{y})$ in equation (2). But even for the binary case with mutation and selection only, the transfer matrix $p(\mathbf{x}|\mathbf{x}')$ is of size $2^n \times 2^n$. It is extremely difficult to analyze the distribution using this general equation.

We proceed further. Equation (2) is not the end result of a mesoscopic analysis, but just the beginning. We will concentrate on distributions which are defined by a small number of parameters or can be approximated by distributions by a small set of parameters. Because we treat the marginal distributions as *deterministic* variables, the mesoscopic analysis is valid for *infinite* populations. Fluctuations arising because of finite populations can be

investigated in principle, but it is extremely difficult. Because of the sampling theory in statistics our analysis can be seen as the limit case of large finite populations when the size goes to infinity.

We first consider the simplest approximation. This approximation has been used for a long time in population genetics.

**Definition 2** *Robbins' proportions are defined by the distribution*

$$\pi_p(\mathbf{x}, t) := \prod_{i=1}^{n} p(X_i = x_i, t) \tag{3}$$

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

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 and Voigt 1996). In gene pool recombination a new string is computed by randomly taking a gene for each locus from the distribution of the selected parents. This means that gene $x_i$ occurs with probability $p^s(x_i)$ in the next population. $p^s(x_i)$ is the distribution of $x_i$ in the selected parents. Thus new strings $\mathbf{x}$ are generated according to the distribution

$$p(\mathbf{x}, t+1) = \prod_{i=1}^{n} p_i^s(x_i, t) \tag{4}$$

One can simplify the algorithm further by directly computing the univariate marginal frequencies from the data. Then equation (4) can be used to generate new strings. Equation (4) has been often used as an approximation. In physics it is called the *mean field approach* (Opper and Saad 2001). We call our algorithm the *Univariate Marginal Distribution Algorithm* (*UMDA*).

**Algorithm 1: *UMDA***

1. $t \Leftarrow 1$. Generate $N \gg 0$ individuals randomly.
2. **do** {
3. Select $M \leq N$ individuals according to a selection method. Compute the sample marginal frequencies $p_i^s(x_i, t)$ of the selected set.
4. Generate $N$ new points according to the distribution $p(\mathbf{x}, t+1) = \prod_{i=1}^{n} p_i^s(x_i, t)$.
5. $t \Leftarrow t + 1$ .
6. } **until** Termination criterion fulfilled.

**Remark:** *UMDA* uses a finite population. Thus an exact theoretical analysis of *UMDA* has to consider the stochastic fluctuations introduced by finite populations. This is very difficult. We therefore assume an infinite population for the analysis. This assumption can be justified, because the probabilities of the infinite population can be seen as the *expected* value of the probabilities for finite population. This was also observed in (Vose 1999)

Let $v = \sum_{i=1}^{n}(m_i + 1)$. *UMDA* formally depends on $v$ parameters, the marginal distributions $p_i(x_k)$. We now consider the average $E[f(\mathbf{x})] = \sum_x p(\mathbf{x},t)f(\mathbf{x})$ as a function which depends on $p_i(x_k)$. To emphasize this dependency we write (in accordance with Wright)

$$W(p_1(x_1), p_1(x_2), \ldots, p_n(x_{m_n})) := E[f(\mathbf{x})] \tag{5}$$

**Definition 3** *Fitness proportionate selection changes the frequencies according to*

$$p^s(\mathbf{x},t) = p(\mathbf{x},t)\frac{f(\mathbf{x})}{W(t)} \tag{6}$$

A detailed mathematical analysis of *UMDA* can be found in (Mühlenbein 1997; Mühlenbein and Mahnig 2000). For notational simplicity we assume $x_i \in \{0,1\}$. Let $\tilde{W}(t)$ denote the average fitness as a function of the *independent* variables $p_i = p(X_i = 1)$ only. Then the following theorem is valid:

**Theorem 4 (Wright's Equation (Wright 1932))** *For infinite populations and proportionate selection UMDAchanges the gene frequencies as follows:*

$$p_i(t+1) = p_i(t) + p_i(t)(1 - p_i(t))\frac{\frac{\partial \tilde{W}}{\partial p_i}}{\tilde{W}(t)} \tag{7}$$

*The stable attractors of equation* (7) *are at the corners of the unit cube, 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* $W(\mathbf{p})) = 0$. *The attractors are local maxima of* $f(x)$ *according to one bit changes. Thus UMDA tries to solve the continuous optimization problem* $\text{argmax}\{\tilde{W}(\mathbf{p})\}$ *in the unit cube by gradient ascent. The average fitness never decreases*

**Remark:** Equation (7) can easily be extended to include *mutation* (Mühlenbein and Mahnig 2002a; Mühlenbein and Mahnig 2002b).

Wright's equation consists of $n$ parameters. Any application needs the the average fitness $\tilde{W}$. This requires the summation over all $2^n$ possible configurations!

But if $f(\mathbf{x})$ is given in a normalized form, the computation of $\tilde{W}$ is very easy, one has only to exchange $x_i$ with $p_i$ (Mühlenbein and Mahnig 2000). We just give an example

$$f(\mathbf{x}) = a_0 + \sum_i x_i + \sum_{i,j} a_{ij}x_i x_j + \sum_{i,j,k} a_{ijk}x_i x_j x_k \tag{8}$$

$$\tilde{W}(\mathbf{p}) = a_0 + \sum_i p_i + \sum_{i,j} a_{ij}p_i p_j + \sum_{i,j,k} a_{ijk}p_i p_j p_k \tag{9}$$

We will now apply Wright's equation for the analysis of a very specific optimization problem.

**Example:** Needle-Trap function

$$f(\mathbf{x}) = \sum_{i=1}^{n} x_i + a\prod_{i=1}^{n}(1 - x_i) \tag{10}$$

This function is identical to $OneMax = \sum x_i$ with the exception of the string $x = (0, \ldots, 0)$. For this string the function values is $a$. For $a > n$ this string is the global maximum. But this maximum is surrounded by strings with the lowest fitness. In contrast, the second maximum is surrounded by strings of high fitness. One expects that evolution is driven to the second maximum.

For this function we compute

$$\tilde{W}(p_1, \ldots, p_n) = \sum_{i=1}^{n} p_i + a \prod_{i=1}^{n} (1 - p_i) \tag{11}$$

In order to determine the attractor of Wright's equation starting with an unbiased random configuration ($p_j = 0.5, j = 1, .., n$), we compute the gradient of $\tilde{W}$ at this point. We have

$$\begin{aligned} \frac{\partial \tilde{W}(p_1, .., p_n)}{\partial p_i} &= 1 - a \prod_{j \neq i}^{n} (1 - p_j) \\ \frac{\partial \tilde{W}(1/2, .., 1/2)}{\partial p_i} &= 1 - a \left(\frac{1}{2}\right)^{n-1} \end{aligned}$$

The sign of the derivative of $\tilde{W}$ gives the direction $p_i$ moves. We obtain the result $p_i \to 1$ for $a < 2^{n-1}$ and $p_i \to 0$ for $a > 2^{n-1}$. For $a = 2^{n-1}$ we have an unstable fix-point at $p_i = 0.5$. Thus even the infinite population model converges to the trap, if the fitness advantage of the needle is not large enough. For *finite populations* we will observe with high probability convergence to 1, if the population does not include the string $\mathbf{x} = (0, \ldots, 0)$ in a certain fraction. If the size of the population is exponential ($N = C \cdot 2^n$)) with $C > 1$ then the finite population might also converge to the optimum string.

In this artificial example the difference between the result for infinite population and for finite populations is extremely large.

## 3 The Factorized Distribution Algorithm FDA

The simple product distribution of *UMDA* cannot capture dependencies between variables. When the detection of the dependencies is necessary to find the global optimum, *UMDA* and simple genetic algorithms fail. We need a more complex distribution to reach this goal. A good candidate for optimization using a search distribution is the Boltzmann distribution.

**Definition 5** *For $\beta \geq 0$ define the **Boltzmann distribution** of a function $f(x)$ as*

$$p_\beta(x) := \frac{e^{\beta f(x)}}{\sum_y e^{\beta f(y)}} =: \frac{e^{\beta f(x)}}{Z_f(\beta)} \tag{12}$$

*where $Z_f(\beta)$ is the partition function. To simplify the notation $\beta$ and/or $f$ can be omitted.*

The Boltzmann distribution is usually defined as $e^{-\frac{g(x)}{T}}/Z$. The term $g(x)$ is called the energy and $T = 1/\beta$ the temperature. The Boltzmann distribution is suited for

optimization because it concentrates with increasing $\beta$ around the global optima of the function. In theory, if it were possible to sample efficiently from this distribution for arbitrary $\beta$, optimization would be an easy task.

### 3.1 Boltzmann selection

Our proposed algorithm incrementally computes the Boltzmann distribution by using Boltzmann selection.

**Definition 6** *Given a distribution $p$ and a selection parameter $\Delta\beta$, **Boltzmann selection** calculates the distribution of the selected points according to*

$$p^s(x) = \frac{p(x)e^{\Delta\beta f(x)}}{\sum_y p(y)e^{\Delta\beta f(y)}} \tag{13}$$

**Algorithm 2: *BEDA* – Boltzmann Estimated Distribution Algorithm**

1. $t \Leftarrow 1$. Generate $N$ points according to the uniform distribution $p(x,0)$ with $\beta(0) = 0$.
2. **do** {
3. With a given $\Delta\beta(t) > 0$, let
$$p^s(x,t) = \frac{p(x,t)e^{\Delta\beta(t)f(x)}}{\sum_y p(y,t)e^{\Delta\beta(t)f(y)}}.$$
4. Generate $N$ new points according to the distribution $p(x,t+1) = p^s(x,t)$.
5. $t \Leftarrow t+1$.
6. } **until** (stopping criterion reached)

We can now define the *BEDA* (Boltzmann Estimated Distribution Algorithm). *BEDA* is a conceptional algorithm, because the calculation of the distribution requires a sum over exponentially many terms. It can easily be proven that *BEDA* converges to the set of all global optima (Mühlenbein and Mahnig 2002c).

We next transform *BEDA* into a practical algorithm. This means to reduce the number of parameters of the distribution and to compute an adaptive schedule for $\beta$.

### 3.2 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 7** *Let $s_1, \ldots, s_m$ be index sets, $s_i \subseteq \{1, \ldots, n\}$. Let $f_i$ be functions depending only on the variables $x_j$ with $j \in s_i$. Then*

$$f(\mathbf{x}) = \sum_{i=1}^{m} f_i(\mathbf{x}_{s_i}) \tag{14}$$

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

We also need the following definitions

**Definition 8** *Given* $s_1, \ldots, s_m$, *we define for* $i = 1, \ldots, m$ *the sets* $d_i$, $b_i$ *and* $c_i$:

$$d_i := \bigcup_{j=1}^{i} s_j, \qquad b_i := s_i \setminus d_{i-1}, \qquad c_i := s_i \cap d_{i-1} \tag{15}$$

*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). In (Mühlenbein, Mahnig, and Ochoa 1999) we have proven the following theorem.

**Theorem 9 (Factorization Theorem)** *Let* $p_\beta(\mathbf{x})$ *be a Boltzmann distribution with*

$$p_\beta(\mathbf{x}) = \frac{e^{\beta f(\mathbf{x})}}{Z_f(\beta)} \tag{16}$$

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

$$b_i \neq \emptyset \quad \forall i = 1, \ldots, m; \quad d_m = \{x_1, \ldots, x_n\}, \tag{17}$$

$$\forall i \geq 2 \; \exists j < i \;\; \textit{such that } c_i \subseteq s_j \tag{18}$$

*then*

$$p_\beta(\mathbf{x}) = \prod_{i=1}^{m} p_\beta(\mathbf{x}_{b_i} | \mathbf{x}_{c_i}) = \frac{\prod_{i=1}^{m} p_\beta(\mathbf{x}_{b_i}, \mathbf{x}_{c_i})}{\prod_{i=2}^{m} p_\beta(\mathbf{x}_{c_i})} \tag{19}$$

The constraint defined as equation (18) is called the *running intersection property*. It is a severe assumption.

The factorization theorem can be seen as a mathematical complete *schema theorem*. It tells which schemata are necessary to generate the *whole* distribution. The usual schema theorems describe only the evolution of schemata, but not how the distribution can be generated.

With the help of the factorization theorem, we can turn the conceptional algorithm *BEDA* into *FDA*, the Factorized Distribution Algorithm. If the conditions of the factorization theorem are fulfilled, the convergence proof of *BEDA* applies to *FDA*. In principle *FDA* can be run with any selection scheme, but then the convergence proof is no longer valid. Therfore we believe that Boltzmann selection is an essential part in using the *FDA*.

Because *FDA* uses finite samples 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 and Mahnig 1999) the scaling behavior for various test functions has been studied.

**Algorithm 3: *FDA* – Factorized Distribution Algorithm**

1. Calculate $b_i$ and $c_i$ from the decomposition of the function.
2. $t \Leftarrow 1$. Generate an initial population with $N$ individuals from the uniform distribution.
3. **do** {
4. Select $M \leq N$ individuals using Boltzmann selection.
5. Estimate the conditional probabilities $p(x_{b_i}|x_{c_i}, t)$ from the selected points.
6. Generate new points according to $p(x, t+1) = \prod_{i=1}^{m} p(x_{b_i}|x_{c_i}, t)$.
7. $t \Leftarrow t + 1$.
8. } **until** (stopping criterion reached)

# 4 Holland's schema analysis and the Boltzmann distribution

Before discussing *FDA* in more detail, we will turn to the very first analysis of genetic algorithms made by Holland ((Holland 1992)). We will use here Holland´s terminology. (We remind the reader that from a schema $\xi$ in Holland's terminology its probability $P(\xi, t)$ is computed. This is just the marginal distribution $p(\mathbf{x}_\xi, t)$.) He derived the following conjecture about a good population based search algorithm.

((Holland 1992),p.88): *Each (schema) $\xi$ represented in (the current population) $B(t)$ should increase (or decrease) in a rate proportional to its "observed" "usefulness" $\hat{\mu_\xi}(t) - \hat{\mu}(t)$ (average fitness of schema $\xi$ minus average fitness of the population)*

$$\frac{dP(\xi, t)}{dt} = (\hat{\mu_\xi}(t) - \hat{\mu}(t))P(\xi, t) \tag{20}$$

Holland claimed that the simple genetic algorithm behaves according to the above equation. This is not true. Instead we have the surprising result:

**Theorem 10** *The Boltzmann distribution $p(\mathbf{x}, t) = e^{tf(\mathbf{x})}/Z_f(t)$ with $P(\xi, t) = \sum_{X|X_\xi = x_\xi} p(\mathbf{x}, t)$ fulfills Holland's equation (20).*

**Proof:** Taking the derivative of the Boltzmann distribution we easily obtain

$$\frac{p(\mathbf{x}, t)}{dt} = p(\mathbf{x}, t)(f(\mathbf{x}) - \bar{f}(t)) \tag{21}$$

Let $\xi$ define a schema, $\mathbf{x}_\xi$ the corresponding marginal distribution. Then

$$\begin{aligned} \frac{dP(\xi, t)}{dt} = \frac{dp(\mathbf{x}_\xi, t)}{dt} &= p(\mathbf{x}_\xi, t) \left( \frac{1}{p(\mathbf{x}_\xi, t)} \sum_{X|X_\xi = x_\xi} p(\mathbf{x}, t)(f(\mathbf{x}) - \bar{f}(t)) \right) \\ &= P(\xi, t)(\hat{\mu_\xi}(t) - \hat{\mu}(t)) \end{aligned}$$

Thus the Boltzmann distribution with the fixed *annealing schedule* $\beta(t) = t$ fulfills Holland's equation. *According to Holland's analysis FDA with this schedule should be an almost optimal algorithm!*

We now discuss the evolution of marginal distributions for a specific function. The probability distribution corresponding to this function cannot be factorized. *FDA* has to use the full distribution.

**Example:** Normalized Trap function of size 3

$$f(|\mathbf{x}|) = (1-\alpha, 1-2\alpha, 0, 1) \quad |x| = (0,1,2,3)$$

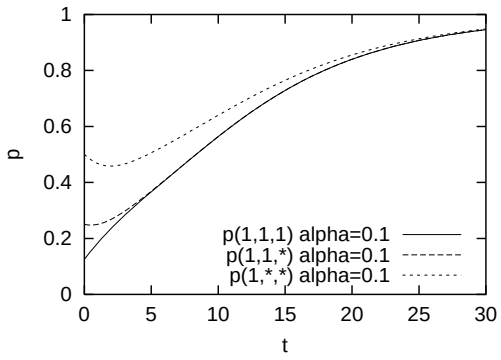


**Figure 1** Evolution of marginal distributions ($t = \beta$)

Figure 1 shows that $p(1,1,1)$ continuously increases, whereas $p(1,1,*)$ and $p(1,*)$ decrease for one or even two generations. This result demonstrates that the evolution of single schemata itself does not give much information about the optimization process. As the factorization theorem shows, one has to consider all schemata which generate the distribution.

## 5 Computing a factorization by junction trees

The approximation of the Boltzmann distribution and of distributions in general, constrained by a *graphical model*, is an important problem for different scientific disciplines. For our application the following definition is sufficient.

**Definition 11** *A graphical model is a graph G, where two variables are connected by an edge if they are correlated (appear together in one sub-function $f_i$).*

New methods to compute a factorization start from a *graphical model* $G$ of the distribution (Lauritzen 1996). In order to find the separators $c_i$ the algorithm computes *cliques* and generates a *junction tree* $J$. There is lots of literature available about this method, e. g. (Lauritzen 1996; Huang and Darwiche 1996; Meyer 1998; Jensen and Jensen 1994). A

junction tree is an undirected tree the nodes of which are clusters of variables. The clusters satisfy the *junction property*: For any two clusters $a$ and $b$ and any cluster $h$ on the unique path between $a$ and $b$ in the junction tree the relation

$$a \cap b \subseteq h \tag{22}$$

is true. The junction property is identical to the running intersection property defined by equation (18). The edges between the clusters are labeled with the intersection of the adjacent clusters; we call these labels *separating sets* or *separators*.

A junction tree is constructed from the graphical model by the following steps:

**Triangulating the graph:** A graph $G$ is triangulated if it contains no chordless circle with more than three vertices. An algorithm for adding the necessary edges is described in (Huang and Darwiche 1996).

**Finding the cliques:** A clique $C$ in a graph is a maximal totally connected subgraph. That means that in $C$ every node is connected to every other node in $C$, and there is no clique $C'$ which contains $C$.

**Generating the clusters:** For each clique generate a cluster containing its variables. This cluster will become a node of the junction tree $J$.

**Building the junction tree:** Find pairs of clusters with maximal intersection and connect them. Label the edge with the separating set. Repeat this until the tree is complete.

This results in a tree which fulfills the junction property. For example, a circle of bi-variate marginals results in a circular graph $G$, which can be triangulated by connecting one node with all other nodes. An example for $n = 8$ variables is shown in figure 2.

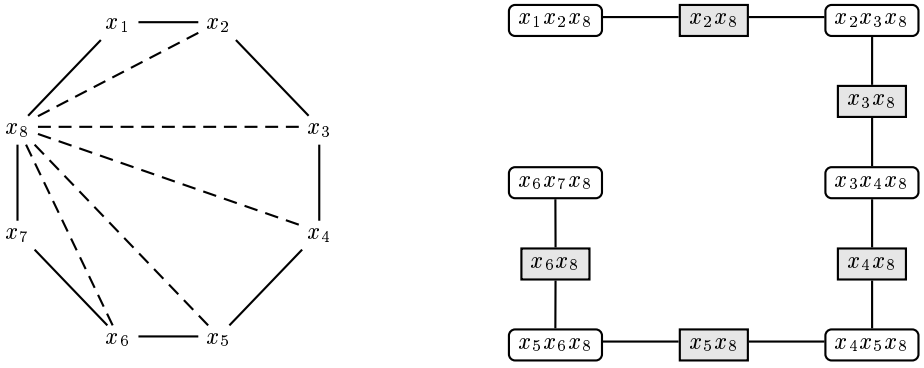


**Figure 2** Graphical model with triangulation and junction tree for a 1-D bi-variate circle. The left figure shows the graph $G$; the dashed lines are inserted for the triangulation. The cliques of the triangulated graph are the clusters of the junction tree $J$ (right figure, white boxes). The separators are the shaded boxes.

The distribution is now factorized into the cliques given by the clusters of the junction tree. However, the junction tree now contains non-local marginal distributions of order 3.

Figure 2 defines the following factorization

$$p(\mathbf{x}) = \frac{p(x_1, x_2, x_8)p(x_2, x_3, x_8)p(x_3, x_4, x_8)p(x_4, x_5, x_8)p(x_5, x_6, x_8)p(x_6, x_7, x_8)}{p(x_2, x_8)p(x_3, x_8)p(x_4, x_8)p(x_5, x_8)p(x_6, x_8)} \quad (23)$$

The approximation of the marginals of order 3 by the given marginals of order 2 will be described next.

### 5.1 Approximation of distributions by marginals

For the circle the factorization consists of non-local marginals of order three. These marginals can easily be estimated by $FDA$, but there exist also methods to compute the non-local marginals by the given marginals of order two. This method is essential if the factorization leads to large cliques. We just describe a method based on *iterative proportional fitting* and the *maximum entropy principle*.

**Maximum entropy principle:** *Find the maximal entropy distribution for $p(\mathbf{x})$ which satisfies the given marginals.*

The maximum entropy principle has a long history in physics and probabilistic logic. The interested reader is referred to (Jaynes 1957; Smith and Grand 1985). The following theorem holds ((Cover and Thomas 1989).

**Theorem 12** *If the given marginal distributions $p_k(\mathbf{x}_k)$ are consistent, then there exist a unique distribution $q(\mathbf{x})$ of maximum entropy.*

Consistent means that the marginal distributions fulfill the constraints defined by probability theory. The most popular algorithm to compute the maximum entropy distribution is called *iterative proportional fitting*.

### 5.2 Iterative proportional fitting

Iterative proportional fitting computes iteratively a distribution $q_\tau(\mathbf{x})$ from given marginals $p_k(\mathbf{x}_k)$, $k = 1, \ldots, K$, where $\mathbf{x}_k$ is a sub-vector of $\mathbf{x}$ and $\tau = 0, 1, 2, \ldots$ is the iteration index. Let $n$ be the dimension of $\mathbf{x}$ and $d_k$ be the dimension of $\mathbf{x}_k$. Then the update formula is

$$q_{\tau+1}(\mathbf{x}) = q_\tau(\mathbf{x}) \frac{p_k(\mathbf{x}_k)}{\sum_{y \in \{0,1\}^{n-d_k}} q_\tau(\mathbf{x}_k, \mathbf{y})} \quad (24)$$

with $k = ((\tau - 1) \mod K) + 1$.

One can show that IPF converges to the distribution with *maximum entropy* (Cover and Thomas 1989). For the 1-D circle we have $\mathbf{x}_k = \{x_k, x_{k+1}\}$, $k = 1, \ldots, n$, $x_{n+1} = x_1$. Since the distribution $q$, which has to be stored and updated in every time step, has exponential size, the naive implementation takes exponential time and space.

The complexity can be greatly reduced by using a factorization of the distribution (Jiroušek and Přeučil 1995; Meyer 1998). The algorithm uses only the computed clusters of the factorization as marginals. The marginals can later be combined to compute the

whole distribution $q(\mathbf{x})$. This algorithm produces exactly the same result as the standard iterative proportional fitting. We have implemented a special version of this algorithm developed by (Meyer 1998) in probabilistic logic. Due to space limitations we cannot describe the algorithm here.

### 5.3 The factorization problem

The question, which additively decomposed functions can be factored into marginal distributions of small order, is difficult to answer. It is not difficult to prove that all one-dimensional graphical models can easily be factored by the junction tree method presented. Unfortunately higher dimensional spatial graphs like 2-D grids lead to factors with a large number of variables. In fact, the number of variables for some factors will be of order $O(n)$ (Mühlenbein, Mahnig, and Ochoa 1999). This is a substantial reduction compared to the total number of variables $n^2$, but the amount of computation needed for these marginals is still exponential in $n$.

Thus the junction tree algorithm cannot be used in these cases. To solve this problem, different methods have to be used. These methods iteratively approximate the unknown distribution by the given marginals using an approximate junction tree. Other methods do not use junction trees at all, but iteratively approximate $p(\mathbf{x})$ for a given $\mathbf{x}$. These methods are called *generalized belief propagation* and *advanced mean-field methods*. They are outside the scope of this paper. The interested reader is referred to (Opper and Saad 2001).

## 6 Sampling from a Boltzmann distribution

The Boltzmann distribution as a means to optimize functions has been proposed several times in different scientific fields. The most popular variant is *Simulated Annealing* SA (Kirkpatrick, Gelatt, and Vecchi 1983). The main problem of any implementation using a Boltzmann distribution is to efficiently *sample* from the Boltzmann distribution. The *discrete optimization* problem I

$$\mathbf{x}_{opt} = \text{argmax}\, f(\mathbf{x}) \tag{25}$$

is converted to the *continuous optimization* problem II

$$\mathbf{x}_{opt} = \text{argmax}\, p_\beta(\mathbf{x}) \tag{26}$$

Solving problem II instead of problem I makes only sense if problem II is easier than problem I. Thus it should be easier to generate vectors $\mathbf{x}$ with high $p_\beta(\mathbf{x})$ than with high values of $f(\mathbf{x})$. Moreover, vectors $\mathbf{x}$ with a high probability $p(\mathbf{x})$ should be generated in a reasonable small sample. In this respect $FDA$ and $SA$ differ substantially.

We will first explain the sampling method of $FDA$ in detail. Let the function to be optimized be

$$f(\mathbf{x}) = x_1 + x_2 + x_3 + x_4 - 2x_1x_2 + x_2x_3 + 2x_3x_4 \tag{27}$$

Using a factorization $FDA$ will generate a sample from

$$\hat{p}(\mathbf{x}) = p_\beta(x_1)p_\beta(x_2|x_1)p_\beta(x_3|x_2)p_\beta(x_4|x_3) \tag{28}$$

The samples are computed as follows. First $x_1$ is set according to $p_\beta(x_1, t)$. Fixing $x_1$, $x_2$ is computed from $p_\beta(x_2|x_1, t)$. Then $x_3$ is computed, and so on.

Sampling theory from statistics can be used to compute the expected frequency $p_\beta(x_{opt}$ of the optimal string in a sample of size $N$. The expected average sample average to generate one instance is given by $N_{av} = 1/p_\beta(x_{opt})$.

The above estimate assumes that the conditional distributions needed to compute $p(\mathbf{x}, t)$ are exact. Statistical sampling theory can be used to estimate a conditional distribution $p(x_i|\mathbf{z})$ up to a precision of $\epsilon$. But it seems difficult to apply statistical sampling theory to obtain lower bounds of the sample size $N$ needed to approximate the full distribution $p(\mathbf{x}, t)$ up to a precision $\epsilon$. We therefore leave it at this informal discussion.

The situation in simulated annealing is much worse. SA suffers from the *mixing problem*. If the current vector $\mathbf{x}(t)$ is at a local optimum, it needs an exponential time in the difference between the fitness of the local optimum and the nearest local minimum to leave the area of the local optimum. It can be shown that even $UMDA$ has no difficulties with medium rugged landscapes. This issue is discussed in (Mühlenbein and Zimmermann 2000).

# 7 The adaptive annealing schedule *SDS*

Boltzmann selection needs a good annealing schedule. If we cool down (anneal) too fast, the approximation error of the Boltzmann distribution due to the sampling error can be very large. To consider an extreme case, if the annealing parameter is very large, the second generation should only consist of the global maxima. But if we anneal too slowly, then it takes a long time to approach the optima.

## 7.1 Taylor expansion of the average fitness

In order to determine an adaptive annealing schedule, we will make a Taylor expansion of the average fitness. The average fitness $E_\beta[f(x)]$ is now seen as a function of the inverse temperature. We have proven (Mahnig and Mühlenbein 2001):

**Theorem 13** *The average fitness $E_\beta[f(x)]$ using Boltzmann distributions has the following expansion in $\beta$:*

$$E_{\tilde{\beta}}[f(x)] = E_\beta[f(x)] + \sum_{i \geq 1} \frac{(\tilde{\beta} - \beta)^i}{i!} M^c_{i+1}(\beta) \tag{29}$$

*where $M^c_i$ are the centered moments*

$$M^c_i(\beta) := \sum_x \big[f(x) - E_\beta[f(x)]\big]^i p(x) \tag{30}$$

**Corollary 14** *We have approximatively*

$$E_{\tilde{\beta}}[f(x)] - E_\beta[f(x)] \approx (\tilde{\beta} - \beta) \cdot \sigma^2_f(\beta) \tag{31}$$

*where $\sigma^2_f(\beta)$ is the variance defined as $\sigma^2_f(\beta) := M^c_2(\beta)$. For any $\tilde{\beta} > \beta$ we have $E_{\tilde{\beta}}[f(x)] > E_\beta[f(x)]$ unless $f(x) = const$.*

The proof of the above theorem can be found in (Mühlenbein and Mahnig 2001). Equation (31) was already proposed in (Kirkpatrick, Gelatt, and Vecchi 1983). It is a macroscopic equation relating the average fitness and the variance. From (31) we can derive an adaptive annealing schedule. We recall that truncation selection has proven to be a robust and efficient selection scheme. For truncation selection the *response to selection* $R(t)$ (Mühlenbein and Mahnig 2000) is approximatively given by equation

$$R(t) := E_{t+1}[f(x)] - E_t[f(x)] \approx I_\tau b(t) \sigma_f(t) \tag{32}$$

$I_\tau$ is the selection intensity which depends on the truncation threshold $\tau$. We will make the Boltzmann schedule to mimic truncation selection by setting $\Delta\beta(t)$ accordingly.

**Definition 15** *The **standard deviation schedule** SDSis defined by* $\beta(t+1) = \beta(t) + c/\sigma_f(\beta(t))$.

Using *SDS* we obtain from equation (31)

$$R(t) = E_{\beta(t+1)}[f(x)] - E_{\beta(t)}[f(x)] \approx c \cdot \sigma_f(t) \tag{33}$$

Thus *SDS* with Boltzmann selection behaves similarly to truncation selection if $c = I_\tau b(t)$. We recently found that *SDS* has already been used for genetic algorithms in (Prügel-Bennet and Shapiro 1997). But there *SDS* has been derived from a different perspective.

## 8 Mutation and the Bayesian hyper parameter

Both $UMDA$ and $FDA$ estimate marginal distributions from frequencies. Let $X_i = 1$ be a fixed allele. Let $m$ be the number of instances of $X_i = 1$ in the population. Usually $p_i$ is estimated as $p_i = m/N$. But in the *Bayesian approach* the estimated probability becomes $p{=}(m+r)/(N+2r)$. The *hyper parameter* $r$ has to be chosen in advance (Jordan 1999). The hyper parameter is also called a *Bayesian prior*. It is related to *mutation* in genetic algorithms. Mutation works in the following way: When generating new individuals, with a probability of $\mu$ every bit is changed.

**Theorem 16** *For binary variables, the expectation value for the probability using a Bayesian prior with parameter $r$ is the same as mutation with mutation rate $\mu = r/(N+2r)$. If we set $r = N/(n-2)$ we obtain $\mu = 1/n$.*

The theorem can easily be proven by calculating the probability of generating a particular bit for both cases.

A hyper parameter moves the attractors from the corners of the unit cube into the interior. For $r \to \infty$ there is a unique attractor at $p_i = 0.5$. Using this prior *UMDA* becomes a purely random algorithm. This algorithm will generate the optimum after many generations, but it will not stop at the optimum. Nevertheless, some researcher call this convergence. But this convergence cannot be observed by any numerical algorithm. We require that $r$ should be small enough that the equilibrium distribution is concentrated nearby the optimum.

**Definition 17** *UMDA or FDA are strongly converging, if they convergence to an equilibrium distribution* $p_{eq}(\mathbf{x})$*, and if sampling from this distribution generates the optimum with probability at least* 1/3.

For $r = 0$ the equilibrium distribution is a point distribution. For $r > 0$ the equilibrium distribution is obtained from a *dynamic equilibrium* between *selection* (driving the population to the corner) and *mutation* (driving the population to the interior). The higher the selection pressure, the larger the hyper parameter can be. In (Mühlenbein and Mahnig 2002a) the surprising result is shown that the recommended mutation rate of $\mu = 1/n$ is too large for proportionate selection, even for the OneMax function $\sum x_i$. The above analysis has been extended to marginal distributions of any order (Mühlenbein and Mahnig 2002b).

We summarize: *A Bayesian hyper parameter increases the robustness of the algorithm with respect to premature convergence because of a too small population size. But the prior has to be chosen carefully in order that convergence can be numerically observed.*

## 9 A population dynamics implementation of *FDA*

The main problem of the application of *FDA* is the computation of the factorization. There are many optimization problems where the factorization leads to cliques, which are so large that a use of *FDA* is not feasible.

We have already mentioned several approximation techniques, which do not use a factorization. The most popular is Pearl's *generalized belief propagation* (Opper and Saad 2001). This algorithm can be easily implemented in a way which mimics *cooperation* and *coevolution*.

Given an additively decomposed function we can assign each sub-function to a different *species*. These species *cooperate* to optimize the global fitness function $f$. Each species may set the variables $\mathbf{x}_{s_i}$ which are contained in the sub-function they are responsible for. There is a big difference between *separable* and *non-separable* functions. For separable functions each species can separately optimize its sub-functions. If the fitness function is not separable, the species have to send their proposals for setting $\mathbf{x}_{s_i}$ to a central instance. This instance computes the fitness function and does selection.

There are several implementations possible, differing in the amount of centralism in the selection step. The central part computes the fitness functions and does the selection. If the graphical model corresponding to $f(\mathbf{x})$ can be factored, the species are connected by a tree. In Pearl's belief propagation species $i$ receives messages from its parent species and sends messages to its child species. The messages are used to update conditional probabilities. The update process is derived from the factorization theorem. In this case we connect the species on a line. Species $i-1$ sends the values of all variables set so far to species $i$, species $i$ computes the values of $\mathbf{x}_{b_i}$ given $\mathbf{x}_{c_i}$ and adds these values to the string. The string is send to species $i+1$. The iteration stops, if the leave is reached.

Belief propagation can formally be extended to arbitrary graphical models. which are not arranged on a tree. If the graphical model contains cycles, the messages have to be sent a

number of iterations until the probabilities do not change anymore. Whether the iterations converge is an open problem.

Many other schemes are possible. They differ in the assignment of variables as well in the amount of centralism, cooperation and competition. The simplest implementation allocates just one variable to each species. This algorithm has been discussed in (Potter and Jong 2000). This algorithm mimics our *UMDA*.

## 10 A kingdom for approximating the Boltzmann distribution

Our research has shown the importance of the Boltzmann distribution for evolutionary algorithms. *FDA* samples efficiently from a Boltzmann distribution. It does the sampling very different from Metropolis sampling used by Simulated Annealing.

If there is no structural information available about the function to be optimized, then it is possible to learn a probabilistic structure from the data. The theory is called learning in graphical models (Jordan 1999). There exist several optimization algorithms using this approach. Our algorithm, called *LFDA* is described in (Mühlenbein and Mahnig 1999). A theoretical analysis of this class of algorithms is very difficult. We believe that combinations of the *FDA* approach (using structural information) and the *LFDA* approach (computing a small graphical model from the data) will be successful heuristics. In (Mühlenbein and Mahnig 2002b) the graph bipartitioning problem has been solved by such an algorithm.

The advantage of our approach is that *FDA* can be seen as a special method to approximate a distribution. This problem arises in many scientific areas. Therefore a fascinating interdisciplinary research is now going on – bringing together such diverse fields as population based optimization, probabilistic reasoning, probabilistic logic, and statistical physics. The core of the theory is the same: the factorization of the distribution if the corresponding factor graph is singly connected. If an exact factorization leads to marginal distributions with too many parameters, then new techniques are currently explored.

## References


Cover, T. M. and J. Thomas (1989). *Elements of Information Theory*. New York: Wiley.

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

Huang, C. and A. Darwiche (1996). Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning 15*(3), 225–263.

Jaynes, E. (1957). Information theory and statistical mechanics. *Phys. Rev 6*, 620–643.

Jensen, F. V. and F. Jensen (1994). Optimal junction trees. In *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*, Seattle, Wash., pp. 360–366.

Jiroušek, R. and S. Přeučil (1995). On the effective implementation of the iterative proportional fitting procedure. *Computational Statistics & Data Analysis 19*, 177–189.

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

Kirkpatrick, S., C. Gelatt, and M. Vecchi (1983). Optimization by simulated annealing. *Science 220*, 671–680.

Lauritzen, S. L. (1996). *Graphical Models*. Oxford: Clarendon Press.

Mahnig, T. and H. Mühlenbein (2001). A new adaptive boltzmann selection schedule SDS. In *Proceedings CEC 2001*, Piscataway, pp. 183–190. IEEE Press.

Meyer, C.-H. (1998). *Korrektes Schließen bei unvollständiger Information*. Ph. D. thesis, Fernuniversität Hagen.

Mühlenbein, H. (1997). The equation for the response to selection and its use for prediction. *Evolutionary Computation 5(3)*, 303–346.

Mühlenbein, H. and T. Mahnig (1999). FDA – a scalable evolutionary algorithm for the optimization of additively decomposed functions. *Evolutionary Computation 7*(4), 353–376.

Mühlenbein, H. and T. Mahnig (2000). Evolutionary algorithms: From recombination to search distributions. In L. Kallel, B. Naudts, and A. Rogers (Eds.), *Theoretical Aspects of Evolutionary Computing*, Natural Computing, pp. 137–176. Berlin: Springer Verlag.

Mühlenbein, H. and T. Mahnig (2001). Evolutionary computation and beyond. In Y. Uesaka, P. Kanerva, and H. Asoh (Eds.), *Foundations of Real-World Intelligence*, pp. 123–188. Stanford, California: CSLI Publications.

Mühlenbein, H. and T. Mahnig (2002a). Evolutionary computation and wright's equation. *Theoretical Computer Science 287*, 145–165.

Mühlenbein, H. and T. Mahnig (2002b). Evolutionary optimization and the estimation of search distributions with applications to graph bipartitioning. *Journal of Approximate Reasoning 31*(3), 157–192.

Mühlenbein, H. and T. Mahnig (2002c). Mathematical analysis of evolutionary algorithms. In C. C. Ribeiro and P. Hansen (Eds.), *Essays and Surveys in Metaheuristics*, Operations Research/Computer Science Interface Series, pp. 525–556. Norwell: Kluwer Academic Publisher.

Mühlenbein, H., T. Mahnig, and A. R. Ochoa (1999). Schemata, distributions and graphical models in evolutionary optimization. *Journal of Heuristics 5*, 215–247.

Mühlenbein, H. and H.-M. Voigt (1996). Gene pool recombination in genetic algorithms. In J. Kelly and I. Osman (Eds.), *Metaheuristics: Theory and Applications*, Norwell, pp. 53–62. Kluwer Academic Publisher.

Mühlenbein, H. and J. Zimmermann (2000). Size of neighborhood more important than temperature for stochastic local search. In *Proceedings of the 2000 Congress on Evolutionary Computation*, New Jersey, pp. 1017–1024. IEEE Press.

Opper, M. and D. Saad (Eds.) (2001). *Advanced Mean Field Methods*. Cambridge: MIT Press.

Potter, M. A. and K. A. D. Jong (2000). Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation 8*(1), 1–29.

Prügel-Bennet, A. and J. Shapiro (1997). An analysis of a genetic algorithm for simple random ising systems. *Physica D 104*, 75–114.

Smith, C. R. and W. T. Grand (Eds.) (1985). *Maximum-Entropy Principle and Bayesian Methods in Inverse Problems*. D. Reidel Publishing.

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

Wright, S. (1932). The roles of mutation, inbreeding, crossbreeding and selection in evolution. In *Proc. 6th Int. Congr. on Genetics*, pp. 356–366.