Schemata, Distributions and Graphical Models in Evolutionary Optimization

Published in: *Journal of Heuristics*, 5, pp. 215-247, 1999

Heinz Mühlenbein, Thilo Mahnig and Alberto Ochoa Rodriguez[1]
Real World Computing Partnership Theoretical Foundation GMD Laboratory
GMD - Forschungszentrum Informationstechnik
53754 St. Augustin.

Abstract
In this paper the optimization of additively decomposed discrete functions is investigated. For these functions genetic algorithms have exhibited a poor performance. First the schema theory of genetic algorithms is reformulated in probability theory terms. A schema defines the structure of a marginal distribution. Then the conceptual algorithm *BEDA* is introduced. BEDA uses a Boltzmann distribution to generate search points. From BEDA a new algorithm, *FDA*, is derived. FDA uses a factorization of the distribution. The factorization captures the structure of the given function. The factorization problem is closely connected to the theory of conditional independence graphs. For the test functions considered, the performance of FDA - in number of generations till convergence - is similar to that of a genetic algorithm for the $OneMax$ function. This result is theoretically explained.

Keywords
graphical models, conditional independence graphs, additively decomposed functions, estimation of distributions, population based search, genetic algorithm

1. Introduction

A genetic algorithm is a population based search method. A set of points is generated, promising points are selected, and new points are generated using the genetic operators recombination/crossover and mutation. The simple genetic algorithm (Goldberg 1989) selects promising points according to proportionate selection

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

Here $x = (x_1, x_2, \ldots, x_n)$ denotes a vector of discrete variables (genotype), $p(x,t)$ is the distribution of x at generation t and $\bar{f}(t) = \sum p(x,t)f(\boldsymbol{x})$ is the average fitness of the population. For simplicity we assume binary variables $x_i \in \{0,1\}$. In principle one would like to generate new points according to

$$p(x,t+1) \approx p^s(x,t). \tag{2}$$

In general a discrete probability is defined by 2^n parameters. Therefore a straightforward implementation is computationally too expensive. Nevertheless, an investigation of the following question is promising: Can Equation 2 be approximated

with substantially less than exponential computational effort? Instead of extending the genetic algorithm, we take a new approach based on probability theory. We try to approximate the equation by explicit aggregation.

A possible structure for aggregation is a schema (Holland 1992). We just give an example for a schema. Extending the usual notation,

$$H(x_i, x_k) = (*, \ldots, *, x_i, *, \ldots, *, x_k, *, \ldots *) \tag{3}$$

defines a schema where the values of the variables x_i and x_k are held fixed, the other variables are free. If we sum Equation 1 over all x which are members of schema $H(x_i, x_k)$ we obtain the probability of $H(x_i, x_k)$ after selection

$$p^s(H(x_i, x_k), t) = \sum_{x \in H} \frac{p(x,t) f(x)}{\bar{f}(t)}. \tag{4}$$

In probability terms $p^s(H(x_i, x_k), t)$ is a marginal distribution which we abbreviate by $p^s(x_i, x_k, t)$. We can now state the problem of aggregation in terms of probability theory: *Does a set of marginal distributions exist the combination of which gives a good approximation to Equation 2 and which can be computed in polynomial time.*

The simplest approximation uses first order schemata. These schemata define univariate marginal distributions. This approximation is used by the Univariate Marginal Distribution Algorithm UMDA (Mühlenbein 1998). New points are generated according to

$$p(x, t+1) = \prod_{i=1}^{n} p^s(x_i, t). \tag{5}$$

$p^s(x_i, t)$ denotes the probability (after selection) of a first order schema defined at loci i. If the distribution $p^s(x, t)$ in Equation 1 is complex, first order schemata give a bad approximation. In order to get a better approximation for complex distributions higher order schemata have to be used. But which higher order schemata should be used? In this paper this problem is solved for a class of functions called *additively decomposed functions* (ADF).

The outline of the paper is as follows. In Section 2 we introduce ADFs. Then we define evolutionary algorithms based on probability distributions. For the theoretical analysis the *Boltzmann distribution* is used. For this distribution convergence of a conceptual algorithm is shown. The Boltzmann distribution can be efficiently computed if it can be factorized. In Section 4 our main factorization theorem is proven. Then a new algorithm, the *Factorized Distribution Algorithm FDA* is described. The relation between our factorization method and the factorization used for *graphical models* is shown next.

FDA is extended to constraint optimization problems in Section 7. The problem of decomposing an unknown function defined by a function value table into an ADF is briefly investigated in Section 8. Then the schema theorem of genetic algorithms is discussed in the context of FDA. In Section 10 approximate convergence results for FDA are derived. These are used to discuss the numerical results of a suite of test functions.

2. Additively Decomposed Functions

Numerically the deficiencies of genetic algorithms using Mendelian string based recombination methods have been first demonstrated with a simple class of fitness functions, called deceptive functions of order k. They are defined as a sum of more elementary deceptive functions f_k of k variables (Goldberg et al. 1993).

$$f(x) = \sum_{j=1}^{l} f_k(s_j), \tag{6}$$

where $\boldsymbol{s}_j$ are non-overlapping substrings of x containing k elements. We take as example a deceptive function of order three. Let u denote the number of bits turned on in the sub-string. Then f^3_{dec} is defined as follows:

$$f^3_{dec} = \begin{cases} 0.9 & \text{for } u = 0 \\ 0.8 & \text{for } u = 1 \\ 0.0 & \text{for } u = 2 \\ 1.0 & \text{for } u = 3 \end{cases}$$

Deceptive functions have been shown to be very difficult to optimize for a genetic algorithm. Because it is deceived by the fitness values, a genetic algorithm will converge to $x = (0, \ldots, 0)$. The genetic algorithm implicitly rates $x_i = 0$ higher than $x_i = 1$ because it evaluates the variables x_i individually and not in combination (Mühlenbein, 1998). But deceptive functions are mathematically trivial to optimize. Because the sets $\boldsymbol{s}_j$ are disjoint, the fitness function is separable. It can be optimized by separately optimizing the function f_k on each set.

The optimization of a broader set of functions has been investigated by Tanese (1989). She used Walsh polynomials to generate the fitness functions. The analysis has been extended by Forrest and Mitchell (1993). They allowed functions with overlapping sets, i.e. $\boldsymbol{s}_i \cap \boldsymbol{s}_j \neq \emptyset$. This class of functions is called *additively decomposed functions* (ADF). ADFs have been also proposed as test functions by Whitley et al. (1995) for continuous variables. ADFs are important in many areas of science and economics.

A number of new evolutionary algorithms have been proposed which optimize ADFs better than genetic algorithms. These algorithms try to detect and exploit the structure of an ADF. The methods used can be described as follows:

- Adaptive recombination
- Explicit detection of relations (Kargupta & Goldberg 1997)
- Bivariate marginal distributions (Pelikan & Mühlenbein 1999)
- Dependency trees (Baluja & Davies 1997)
- Estimation of distributions (Mühlenbein & Paaß(1996), De Bonet et al., (1997))

Adaptive recombination uses a number of heuristics to modify two-parent recombination. Kargupta's (1997) Gene Expression Messy Genetic Algorithm (GEMGA) tries to detect dependency relations by manipulating individual substrings.

The last three methods are based on probability theory and statistics. They use all the statistical information contained in the population of selected points to detect dependencies. These algorithms will be investigated in this paper.

3. Evolutionary Algorithms Using Distributions

First we have to define the notation. In the following X_i denotes a *variable*, $X = (X_1, \ldots, X_n)$ a vector of variables. Small letters denote an assignment, i.e. $X_i = x_i$. $x = (x_1, \ldots, x_n)$ denotes an assignment vector. Π_{s_i} are subsets of variables, π_{s_i} is a corresponding assignment.

Definition: An *additively decomposed function* (ADF) is defined by

$$f(X) = \sum_{s_i \in S} f_i(\Pi_{s_i}) \qquad S = \{s_1, \ldots, s_l\} \quad s_i \subseteq \{X_1, ldots, X_n\} \tag{7}$$

For simplicity we assume $x_i \in \{0, 1\}$. We will often use shorthand $f_i(x)$ instead of $f_i(\Pi_{s_i})$.

UMDA is one of the simplest algorithms using a probability distribution instead of string recombination and mutation. It can be generalized to an algorithm which estimates a probability distribution and uses this information to generate new points. We call this conceptual algorithm EDA.

EDA

- **STEP 0:** Set $t \Leftarrow 1$. Generate $N \gg 0$ points randomly.
- **STEP 1:** Select $M \leq N$ points according to a selection method. Estimate the distribution $p^s(x, t)$ of the selected set.
- **STEP 2:** Generate N new points according to the distribution $p(x, t+1) = p^s(x, t)$. Set $t \Leftarrow t + 1$.
- **STEP 3:** If termination criteria are not met, go to STEP 1.

The next proposition shows the convergence of EDA to the optimum.

Proposition 1: *Let the maximum $f(x_m) = max f(x)$ be unique. If the selection method fulfills*

$$1 - p^s(x_m, t) \leq c(1 - p(x_m, t)) \quad 0 \leq c < 1 \tag{8}$$

then EDA converges to the maximum

$$\lim_{t \to \infty} p(x_m, t) = 1 \tag{9}$$

The assumption used for Proposition 1 is difficult to prove for a general distribution. For a specific probability model, the *Boltzmann distribution,* convergence can be proven fairly easily. The Boltzmann distribution is often used in statistical physics and plays a major role in the analysis of *simulated annealing* (Aarts et al. 1997).

Definition: *The Gibbs or Boltzmann distribution of a function f is defined for $u \geq 1$ by*

$$p(x) := \frac{\mathrm{Exp}_u f(x)}{\sum_y \mathrm{Exp}_u f(y)} \tag{10}$$

where for notational convenience

$$\mathrm{Exp}_u f(x) := u^{f(x)} \quad F_u := \sum_y \mathrm{Exp}_u f(y)$$

Remark: The Boltzmann distribution is usually defined as $e^{-\frac{g(x)}{T}}/Z$. The term $g(x)$ is called the energy. Setting $g(x) = 1/f(x)$ gives Equation 10.

We now define a conceptual EDA with Boltzmann selection.

BEDA

- **STEP 0:** Set $t \Leftarrow 1$. Generate $N \gg 0$ points according to $p(x,0) = \mathrm{Exp}_u f(x)/F_u$ with $u \geq 1$.
- **STEP 1:** [Boltzmann selection] Compute for given $v > 1$

$$p^s(x,t) = p(x,t)\frac{\mathrm{Exp}_v f(x)}{\sum_y p(y,t)\,\mathrm{Exp}_v f(y)} \tag{11}$$

- **STEP 2:** Generate N new points according to the distribution $p(x,t+1) = p^s(x,t)$. Set $t \Leftarrow t+1$.
- **STEP 3:** If termination criteria are not met, go to STEP 1.

Boltzmann selection has been investigated by de la Maza and Tidor (1993). It was not very successful for genetic algorithms. BEDA is more similar to a *simulated annealing algorithm.* But it generates a population of points instead of a single point in each step using the exact Boltzmann distribution. Simulated annealing only approximates the Boltzmann distribution (Aarts 1997).

For BEDA the probability distribution can be explicitly computed. We start with a lemma.

Lemma: *Let $p(x)$ be given by Equation 10. If Boltzmann selection is used with basis v then the distribution of the selected points is given by*

$$p^s(x) = \frac{\mathrm{Exp}_{u\cdot v}\, f(x)}{\sum_y \mathrm{Exp}_{u\cdot v}\, f(y).} \tag{12}$$

Proof: We compute

$$\begin{aligned} p^s(x) &= \frac{\mathrm{Exp}_u\, f(x)}{F_u} \frac{\mathrm{Exp}_v\, f(x)}{\sum_y p(y)\, \mathrm{Exp}_v\, f(y)} \\ &= \frac{\mathrm{Exp}_{u\cdot v}\, f(x)}{F_u \sum_y \frac{\mathrm{Exp}_u f(y)}{F_u} \cdot \mathrm{Exp}_v\, f(y)} \\ &= \frac{\mathrm{Exp}_{u\cdot v}\, f(x)}{\sum_y \mathrm{Exp}_{u\cdot v}\, f(y)}. \end{aligned}$$

□

The next theorem follows by applying the lemma t times.

THEOREM 1 *For BEDA the probability distribution at generation t is given by*

$$p(x,t) = \frac{\mathrm{Exp}_w\, f(x)}{\sum_y \mathrm{Exp}_w\, f(y)} \tag{13}$$

with $w = u \cdot v^t$.

Equation 13 is a system of nonlinear equations for the variables $p(x,t)$. It completely describes the dynamics of the system. Note that $p(x,t)$ remains a Boltzmann distribution, only the basis changes with t. The next theorem is identical to the convergence theorem of simulated annealing (Aarts, 1997).

THEOREM 2 (CONVERGENCE) *Let $X_{opt} = \{x_{1opt}, x_{2opt}, ..\}$ be the set of optima. Then under the assumptions of Theorem 1*

$$\lim_{t\to\infty} p(x,t) = \begin{cases} \frac{1}{|X_{opt}|} & x \in X_{opt} \\ 0 & else \end{cases} \tag{14}$$

Because in general the computation of the distribution $p(x,t)$ requires the computation of 2^n parameters, EDA and BEDA are not efficient algorithms. Mühlenbein and Paaß (1996) have implemented an algorithm which approximates the distribution $p^s(x,t)$. This implementation, however, was not successful because the algorithm tried both - to estimate the probability distribution and to generate new points according to this distribution.

In this paper we investigate the case that the probability $p^s(x,t)$ can be generated by a small number of factors. This method is called *factorization of the probability* according to a *probability model*. This problem is investigated next.

4. Factorization of the Probability

In this section we describe a method for computing a factorization of the probability, given an ADF. The algorithm uses the following sequence of sets as input.

Definition: *Given* $S = \{s_1, \ldots, s_l\}$*, we define for* $i = 1, 2, \ldots, l$ *sets* d_i,b_i *and* c_i

$$d_i := \bigcup_{j=1}^{i} s_j \tag{15}$$
$$b_i := s_i \setminus d_{i-1} \tag{16}$$
$$c_i := s_i \cap d_{i-1} \tag{17}$$

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).

THEOREM 3 (FACTORIZATION THEOREM) *Let* $p(x)$ *be a Boltzmann distribution on* X *with*

$$p(x) = \frac{\mathrm{Exp}_u f(x)}{F_u} \quad \textit{with } u > 1 \textit{ arbitrarily.} \tag{18}$$

If

$$b_i \neq \emptyset \quad \forall i = 1, \ldots, l; \quad d_l = \tilde{X}, \tag{19}$$
$$\forall i \geq 2 \; \exists j < i \;\; \textit{such that } c_i \subseteq s_j \tag{20}$$

then

$$p(x) = \prod_{i=1}^{l} p(\pi_{b_i} | \pi_{c_i}) \tag{21}$$

Proof: The proof is based on a lemma.

Lemma: *Let* $p(x_\alpha, x_\beta, x_\gamma)$ *be a marginal distribution with*

$$p(x_\alpha, x_\beta, x_\gamma) = g(x_\alpha, x_\gamma) h(x_\beta, x_\gamma),$$

then

$$p(x_\alpha, x_\beta, x_\gamma) = p(x_\alpha, x_\gamma) p(x_\beta | x_\gamma) \tag{22}$$
$$p(x_\alpha, x_\gamma) = g(x_\alpha, x_\gamma) \cdot \tilde{h}_\gamma(x_\gamma) \tag{23}$$
$$\text{and} \qquad p(x_\alpha, x_\beta | x_\gamma) = p(x_\alpha | x_\gamma) p(x_\beta | x_\gamma), \tag{24}$$

that is, x_α and x_β are conditionally independent given x_γ.

Proof: We calculate the marginal frequencies:

$$p(x_\alpha, x_\gamma) = g(x_\alpha, x_\gamma) \sum_{y_\beta} h(y_\beta, x_\gamma)$$
$$p(x_\beta, x_\gamma) = h(x_\beta, x_\gamma) \sum_{y_\alpha} g(y_\alpha, x_\gamma)$$
$$p(x_\beta | x_\gamma) = \frac{h(x_\beta, x_\gamma)}{\sum_{y_\beta} h(y_\beta, x_\gamma)}$$

This gives Equation (22), Equation (24) follows directly from the definitions. By setting $\tilde{h}_\gamma(x_\gamma) = \sum_{y_\beta} h(y_\beta, x_\gamma)$, the proof of the lemma is complete. □

The factorization will be proven by inverse induction on i. The induction step is stated as follows. If

$$p(x) = p(\pi_{d_i}) p(\pi_{b_{i+1}} | \pi_{c_{i+1}}) \cdots p(\pi_{b_l} | \pi_{c_l}), \tag{25}$$
$$p(\pi_{d_i} x) = F_{s_1}^{(i)}(x) \cdots F_{s_i}^{(i)}(x) \tag{26}$$

then there exist $F_{s_j}^{(i-1)}(x)$ such that

$$p(x) = p(\pi_{d_{i-1}}) p(\pi_{b_i} | \pi_{c_i}) \cdots p(\pi_{b_l} | \pi_{c_l})$$
$$p(\pi_{d_{i-1}}) = F_{s_1}^{(i-1)}(x) \cdots F_{s_{i-1}}^{(i-1)}(x)$$

For $i = l$ we set $F_{s_j}^{(l)}(x) := \mathrm{Exp}_u f_{s_j}(x)$ with $j > 1$ and $F_{s_1}^{(l)}(x) := \frac{1}{F_u} \mathrm{Exp}_u f_{s_1}(x)$. Then the assumptions of the proposition are fulfilled, because $p(\pi_{d_l}) = p(x)$. Now let $1 < i \leq l$ be given. By (26),

$$p(\pi_{d_i}) = \underbrace{F_{s_1}^{(i)}(x) \cdots F_{s_{i-1}}^{(i)}(x)}_{g(x)} \cdot \underbrace{F_{s_i}^{(i)}(x)}_{h(x)}$$

Setting $\alpha := d_{i-1} \setminus c_i$, $\beta = b_i$ and $\gamma = c_i$, as $s_i = b_i \cup c_i$ and $b_i \cap d_{i-1} = \emptyset$, we get by applying the lemma

$$p(\pi_{d_i}) = p(\pi_{d_{i-1}}) p(\pi_{b_i} | \pi_{c_i}) \tag{27}$$
$$p(\pi_{d_{i-1}}) = F_{s_1}^{(i)}(x) \cdots F_{s_{i-1}}^{(i)}(x) \tilde{h}_{c_i}(x) \tag{28}$$

Because of the assumption $c_l \subseteq s_j$ for some $j < i$, we can set

$$F_{s_k}^{(i-1)}(x) := \begin{cases} F_{s_k}^{(i)}(x) & , k \neq j \\ F_{s_k}^{(i)}(x) \cdot \tilde{h}_{c_i}(x) & , k = j \end{cases}$$

and thus

$$p(\pi_{d_{i-1}}) = F_{s_1}^{(i-1)}(x) \cdots F_{s_{i-1}}^{(i-1)}(x)$$

By inserting (27) into (25), we get

$$\begin{aligned} p(x) &= p(\pi_{d_i}) p(\pi_{b_{i+1}} | \pi_{c_{i+1}}) \cdots p(\pi_{b_l} | \pi_{c_l}) \\ &= p(\pi_{d_{i-1}}) p(\pi_{b_i} | \pi_{c_i}) p(\pi_{b_{i+1}} | \pi_{c_{i+1}}) \cdots p(\pi_{b_l} | \pi_{c_l}) \end{aligned}$$

This proves the induction step. □

4.1. The Running Intersection Property

Assumption 20 is called the *running intersection property* (Lauritzen 1996). If this assumption is violated the factorization might not be exact. This problem and its importance for optimization is discussed with the next example.

Example: Let the function be defined as

$$f(x) = f_1(x_1, x_2, x_3) + f_2(x_1, x_4, x_5) + f_3(x_2, x_4, x_6)$$

The numerical values of the functions are contained in Table 1.

Table 1. Function values

abc	$f_1(abc)$	$f_2(abc)$	$f_3(abc)$
000	0.8163	0.1629	0.6005
100	0.486	0.8815	0.5727
010	0.2703	0.4323	0.2719
110	0.5687	0.7498	0.7565
001	0.7921	0.0627	0.7949
101	0.4131	0.0911	0.3155
011	0.1524	0.8559	0.3235
111	0.616	0.7146	0.6533

The maximum is $x_m = (100001)$ and the second best value is $x^* = (111100)$. The factorizations

$$\tilde{p}_1(x) = p(x_1, x_2, x_3)p(x_4, x_5|x_1)p(x_6|x_2, x_4)$$
$$\tilde{p}_2(x) = p(x_2, x_4, x_6)p(x_1, x_5|x_4)p(x_3|x_1, x_2)$$

are only approximations. They violate the running intersection property.

Numerical results of a BEDA simulation are shown in Table 2. p^s denotes the distribution after selection, $\tilde{p}$ its factorized approximation, used for generating the new points. The factorization $\tilde{p}_1$ generates the optimum less frequently than it appears after selection ($\tilde{p}_1(x_m, t+1) < p^s(x, t)$). The difference is so big that ultimately the optimum will be lost. With factorization $\tilde{p}_2$ the effect is reversed. The probability of the maximum steadily increases compared to the selection probability.

An exact factorization according to the factorization theorem is

$$p(x) = p(x_1, x_2, x_3)p(x_4, x_5|x_1, x_2)p(x_6|x_2, x_4).$$

For an exact factorization we have $p(x, t+1) = p^s(x, t)$. The results from Table 3 confirm the theory. Note that FDA with a valid factorization converges more slowly to the optimum than with the approximate factorization $\tilde{p}_2$.

Remark: The counterexample only shows, that if the running intersection property is violated, BEDA might not find the optimum. It does **not** show that

Table 2. Numerical results for two approximate factorizations

Gen	$p_1^s(x_m)$	$\tilde{p}_1(x_m)$	$p_1^s(x^*)$	$\tilde{p}_1(x^*)$	$p_2^s(x_m)$	$\tilde{p}_2(x_m)$	$p_2^s(x^*)$	$\tilde{p}_2(x^*)$
1	-	0.0156	-	0.0156	-	0.0156	-	0.0156
2	0.0318	0.0248	0.0301	0.0256	0.0318	0.0367	0.0301	0.0251
3	0.0420	0.0319	0.0410	0.0359	0.0628	0.0670	0.0405	0.0352
4	0.0479	0.0362	0.0509	0.0455	0.1012	0.1041	0.0502	0.0452
5	0.0502	0.0379	0.0597	0.0542	0.1441	0.1459	0.0592	0.0549
6	0.0502	0.0381	0.0679	0.0625	0.1894	0.1904	0.0673	0.0641

Table 3. Results for an exact factorization

GEN	$p^s(x_m)$	$p(x_m)$	$p^s(x^*)$	$p(x^*)$	$p_{123}(100)$	$p_{123}(111)$
1	-	0.0156	-	0.0156	0.1250	0.1250
2	0.0318	0.0318	0.0301	0.0301	0.1268	0.1732
3	0.0520	0.0520	0.0465	0.0465	0.1262	0.2220
4	0.0742	0.0742	0.0628	0.0628	0.1315	0.2636
5	0.0974	0.0974	0.0780	0.0780	0.1432	0.2958
6	0.1215	0.1215	0.0920	0.0920	0.1594	0.3196

the running intersection property is **necessary.**

The running intersection property is fulfilled if the interaction graph derived from the sets s_i is like a tree (Lauritzen 1996). Therefore the class of ADFs with a numerical efficient exact factorization is limited, as the following 2-D Ising spin systems shows.

$$F_{Ising}(y) = \sum_i \sum_{j \in N(i)} J_{ij} x_i x_j \quad x_i \in \{-1, 1\} \tag{29}$$

The sum is taken over the four spatial neighbors $N(i)$, but each J_{ij} is used only once. The objective function is purely quadratic. All factorizations fulfilling the running intersection property on 2-D grids need large sets. We state without proof.

Proposition 2: *All exact factorizations of ADFs defined on 2-D grids require the computation of conditional marginal distributions of size $O(\sqrt{n})$ where n is the size of the grid.*

We summarize the intricate relation between factorization and optimization of ADFs: *If the running intersection property is violated then BEDA may not converge to the optimum. This can happen if the approximate factorization $\tilde{p}(x)$ underestimates the exact probability to generate the optimum ($\tilde{p}(x_m) < p^s(x_m)$). But*

there exist also approximate factorizations, which overestimate the probability of the optimum. They converge faster than BEDA with an exact factorization. For exact factorizations the probability of generating the optimum depends only on the strength of selection defined by the basis v.

4.2. Tree-like Factorizations

The factorization Equation 21 can be put into a normal form, where b_i consists of one variable only. Each distribution can be factored into

$$p(x) = p(x_1)p(x_2|x_1)p(x_3|x_2, x_1)\dots p(x_n|x1, \dots, x_{n-1}) \tag{30}$$

For each variable X_i let $\Pi_i \subseteq \{X_1, \dots, X_{i-1}\}$ be a set of variables that renders X_i and $\{X_1, \dots, X_{i-1}\} \setminus \Pi_i$ conditionally independent. That is

$$p(x_i|x_1, \dots x_{i-1}) = p(x_i, \pi_i) \tag{31}$$

Then we obtain

$$p(x) = \prod_{i=1}^{n} p(x_i, \pi_i) \tag{32}$$

It is easy to show that both factorizations (Equations 21 and 32) define the same distribution. The tree factorization is used for Bayes networks (Frey, 1998).

5. The Factorized Distribution Algorithm

The theoretical results are used to define the *Factorized Distribution Algorithm* **FDA**. We assume that an ADF and a factorization of the probability distribution is given. The factorization can also be used at the initialization. For faster convergence, a proportion of $r*N$ individuals can be generated with a local approximation of the conditional marginal distributions. The method will be explained in Section 5.1.

FDA$_r$

- **STEP 0:** Set $t \Leftarrow 0$. Generate $(1-r)*N \gg 0$ points randomly and $r*N$ points according to Equation 33.
- **STEP 1:** Selection
- **STEP 2:** Compute the conditional probabilities $p^s(\pi_{b_i}|\pi_{c_i}, t)$ using the selected points.
- **STEP 3:** Generate a new population according to $p(x, t+1) = \prod_{i=1}^{l} p^s(\pi_{b_i}|\pi_{c_i}, t)$

- **STEP 4:** If termination criteria is met, FINISH.
- **STEP 5:** Add the best point of the previous generation to the generated points (elitist).
- **STEP 6:** Set $t \Leftarrow t+1$. Go to STEP 2.

FDA can be used with an exact or an approximate factorization. It uses *finite samples* of points. Convergence of FDA to the optimum will depend on the size of the samples. FDA can be run with any popular selection method. We usually apply truncation selection. A comparison between Boltzmann selection and truncation selection is made in Section 10.

FDA uses the probability distribution of the selected points for generating new search points. This leads to fast convergence, with a certain risk of missing the optimum. Sometimes we use the more conservative estimate

$$p(x, t+1) = \lambda p^s(x,t) + (1-\lambda)p(x,t)$$

Here the new probability is a weighted average of the probability $p^s(x,t)$ and $p(x,t)$. Normally λ is set to 1.

5.1. Generating the Initial Population

Traditionally the initial population is generated randomly. But if an ADF is given, initial points can be generated using this information. The idea is to generate substrings π_{s_i} with large local fitness values (i.e. $f_i(\pi_{s_i}$ is large) more often than substrings with small fitness values.

The following method has been implemented. The true Boltzmann distribution $p(x)$ is approximated by a distribution $\tilde{p}(x)$ which uses the same factorization as $p(x)$. In the approximation the conditional probabilities are computed using the local fitness functions f_i.

$$\tilde{p}(\pi_{b_i}|\pi_{c_i}) = \frac{\tilde{p}(\pi_{s_i})}{\tilde{p}(\pi_{c_i})} := \frac{\mathrm{Exp}_u\, f_i(\pi_{s_i})}{\sum_{\substack{y \in \Pi_{s_i} \\ \Pi_{c_i} = \pi_{c_i}}} \mathrm{Exp}_u\, f_i(y)} \tag{33}$$

with $u \geq 1$. The larger u becomes, the "steeper" the distribution becomes. $u = 1$ yields a uniform distribution. u can be chosen so that

$$\frac{1}{10} \leq \frac{\tilde{p}(\pi_{b_i}|\pi_{c_i})}{\tilde{p}(\pi_{b_i}|\pi_{c_i})} \leq 10 \qquad i = 1, 2, \ldots, l$$

by setting

$$\begin{aligned} \mathrm{span} &:= \max_i \{ \max_{x,y} |f_i(x) - f_i(y)| \} \\ u &:= 10^{1/\mathrm{span}} \end{aligned}$$

Let us just discuss one example, initialization for function $OneMax(n) = \sum_{i=1}^{n} x_i$. For linear functions we have the factorization

$$p(x) = \prod_{i=1}^{n} p(x_i)$$

For $OneMax$ we obtain span $= 1$ and thus $u = 10$. This leads to

$$\tilde{p}(x_i = 0) = \frac{1}{11} \qquad \tilde{p}(x_i = 1) = \frac{10}{11}$$

This means that $x_i = 1$ is ten times more generated than $x_i = 0$.

Such a biased sample may not generate a Boltzmann distribution. Therefore we generate only half of the population by this method. The other half is generated randomly.

5.2. Analysis of the Factorization Problem

The computational complexity of FDA depends on the factorization and the population size N. The number of function evaluations to obtain a solution is given by

$$FE = GEN_e * N \tag{34}$$

GEN_e denotes the number of generations till convergence. The computational complexity of computing N new search points is given by

$$compl(Npoints) \approx l * N \tag{35}$$

$|s_i|$ denotes the number of elements in set s_i. The computational complexity of computing the probability $p^s(x,t)$ is given by

$$compl(p) \approx (\sum_{i=1}^{l} 2^{|s_i|}) * M \tag{36}$$

where M denotes the number of selected points.

Thus the computationally complexity of FDA depends on N and the size of the defining sets s_i. In order to exactly compute all the probabilities we need an infinite population. A numerical efficient FDA should use a minimal population size N^* still giving good numerical results. The determination of N^* is a difficult problem for any search method using a population of points. We have implemented a simple factorization algorithm which assumes that the defining sets are sorted into a sequence s_i. Then the sets b_i and c_i such that $b_i \neq \emptyset$ are computed according to the factorization theorem. Changing the sequence will change the factorization. We have implemented only an elementary graph manipulation method to obtain a

good factorization. This is called *Join functions*. Functions f_{s_i} which are not used in the factorization are joined with functions used in the factorization yielding new functions that depend on more variables. Joining obviously increases the numerical complexity of FDA. For the root set b_1 the sub function which is maximally non-linear (measured as deviance from a linear square predictor) is chosen.

To compute a factorization with minimal complexity for an arbitrary ADF is a very difficult task. We conjecture that this problem is in NP. This research needs deep results from graph theory. The problem of factorization of a probability distribution is also dealt with in the theory of *graphical models*. The relation of this concept to our method is explained in the next section. All progress in the theory of graphical models can also be used for FDA.

6. Graphical Models

We have made a connection between ADFs and probability theory by using a Boltzmann distribution. For the Boltzmann distribution we have explicitly computed a factorization. In this section we show how an ADF with a corresponding Boltzmann distribution can be mapped to a *graphical model* (Frey (1998), Lauritzen (1996)).

Graphical models are probability models for $p(x)$ where the independence structure of the random variables x_i is characterized by a graph. This graph is usually called a *conditional independence graph* (CIG).

Definition: *The random variables x_i and x_j are independent if and only if $p(x_i, x_j) = p(x_i)p(x_j)$. This is denoted by $x_i \perp x_j$, which is a non-reflexive, non-transitive and symmetric relation between x_i and x_j.*

Next we define conditional independence.

Definition: *The random variables x_i and x_j are conditionally independent given a set of random variables X_k with $p(X_k) > 0$ if and only if $p(x_i, x_j|X_k) = p(x_i|X_k)p(x_j|X_k)$. This is written $x_i \perp x_j|X_k$.*

We are now able to define a conditional independence graph.

Definition: *To the set of variables $\mathcal{X}$ we associate a conditional independence graph (CIG) $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The variables $x_i \in \mathcal{X}$ define the vertices $\mathcal{V}$, and the set of edges $\mathcal{E}$ are constructed as follows*

$$(x_i, x_j) \notin \mathcal{E} \Leftrightarrow x_i \perp x_j | \mathcal{V} \backslash \{x_i, x_j\}$$

This means, in a CIG the absence of an edge in $\mathcal{E}$ implies an independence relation between the two random variables with respect to the remaining $n-2$ vertices. The construction of a CIG starts with the complete graph. Then edges are eliminated according to their conditional independence to the rest.

We now relate our ADF structure to the concept of conditional independence. The following theorem defines a conditional independence relation for the variables of an ADF.

THEOREM 4 *Let* $S_i := \{s \in S | x_i \in s\}$, $S_j := \{s \in S | x_j \in s\}$ *and* $S_R := S \setminus (S_i \cup S_j)$. *This means* S_i *is the union of all sets containing* x_i. *If* $S_i \cap S_j = \emptyset$ *then with*

$$f(x) = \sum_{s \in S_i} f_s(\pi_s) + \sum_{s \in S_j} f_s(\pi_s) + \sum_{s \in S_R} f_s(\pi_s)$$
$$p(x) = \frac{\mathrm{Exp}_u f(x)}{F_u}$$

x_i *is conditionally independent of* x_j *given* $R := X \setminus \{x_i, x_j\}$, *i.e.*

$$p(x_i, x_j | R) = p(x_i | R) \cdot p(x_j | R) \tag{37}$$

Proof: According to the the lemma from Theorem 3, Equation (24), it is sufficient to show that

$$p(x_i, x_j, R) = g(x_i, R) \cdot h(x_j, R)$$

Remember that f_s is a function of $\tilde{X}$, but depends only on the variables from s. Then

$$p(x_i, x_j, R) = \left(\prod_{s \in S_i} \mathrm{Exp}_u f_s(\pi_s) \right) \cdot \left(\frac{\prod_{s \in S_j} \mathrm{Exp}_u f_s(\pi_s) \prod_{s \in S_R} \mathrm{Exp}_u f_s(\pi_s)}{F_u} \right)$$

As $x_j \notin s$ for $s \in S_i$, the left term is a function that does not depend on x_j, whereas the right term does not depend on x_i by the assumptions. Then, applying the lemma, the theorem follows. □

Theorem 4 shows how a CIG for an ADF can be constructed. But the construction is fairly complicated. It can be simplified as the following proposition shows.

Proposition 3: *Two variables* x_i,x_j *of an ADF are connected by an edge in the CIG if and only if there exists* s_k *with* $x_i, x_j \in s_k$.

The proof follows directly from Theorem 4 and the definition of conditional independence.

An ADF can be mapped to a corresponding CIG. But obviously information is lost. For instance, the two functions

$$f(x) = f(x_1, x_2, x_3)$$
$$f_a(x) = f(x_1, x_2) + f(x_2, x_3) + f(x_3, x_2)$$

give the same CIG, but the structure of the two functions is very different. f_a is obviously easier to optimize than f.

The main factorization theorems of CIGs are difficult and are based on graph theory. The interested reader is referred to Frey (1998) and Lauritzen (1996). They have shown that the running intersection property is only fulfilled for tree like CIGs.

7. Constraint Optimization Problems

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

Definition: *Let an ADF be given. We assume that the constraints are given by expressions $C_i(Y_{s_i})$. The constraints are compatible to the ADF structure if*

$$Y_{s_i} \subseteq \Pi_{s_i}. \tag{38}$$

Each C_i defines an assignment of values to variables in Π_{s_i} considered to be *illegal*.

The next theorem shows that FDA will not generate illegal assignments if the initial population consists of legal assignments only. $x \in X$ is said to *fulfill the constraints* ($x \in X^c$) if

$$\forall i : \pi_{s_i} \in X_i^c = \Pi_{s_i} \setminus Y_{s_i} \tag{39}$$

THEOREM 5 *If the initial search points fulfill the constraints then all points generated by FDA fulfill the constraints.*

Proof: We show by induction that

$$\forall t \geq 0, \pi_{s_i} \in Y_{s_i} \implies p(\pi_{s_i}, t) = 0 \tag{40}$$

- $t = 0$: If there exists i such that $\pi_{s_i} \in Y_{s_i}$, then by definition $x \notin X^c$ and by assumption $p(x, 0) = 0$.

- $t \to t + 1$: We assume that $\forall x$ with $\pi_{s_i} \in Y_{s_i}$: $p(\pi_{s_i}, t) = 0$. Now let x be arbitrary with $\pi_{s_i} \in Y_{s_i}$, then $p(\pi_{s_i}, t) = 0$ and thus $p(x, t) = 0$. For FDA we get $p^s(x, t) = 0$. As this is valid for all x with $\pi_{s_i} \in Y_{s_i}$, we have $p^s(\pi_{s_i}, t) = 0$. The conjecture follows from

$$p(\pi_{s_i}, t + 1) = p^s(\pi_{s_i}, t) = 0.$$

□

A different algorithm for constraint optimization based on graphical models was proposed by Dechter and Pearl (1988). In their model the constraints define the *legal* assignments. Therefore the ADF graph has to be contained in the constraint graph ($X_{s_i} \subseteq Y_{s_i}$).

8. Detection of a Decomposition

If the function is not explicitly given as an ADF, either the function has to be decomposed or the Boltzmann distribution has to be estimated without any knowledge about the structure of the function. FDA is based on a known decomposition. It does *model fitting*, not *model selection*. This separation of model fitting and model selection is used in many areas of statistics because model selection is more difficult than model fitting.

For discrete variables the model selection problem can be described as follows. The number of possible probability models is equal to the power set of the number of free parameters. Even if we assume that the number of free parameters of the probability models is restricted, e.g. n, the number of all *possible* probability models is 2^n. This demonstrates how difficult model selection is in general. Several methods have been proposed to detect a decomposition. We just summarize these here.

8.1. Detection of the ADF Structure

The following simple method is not intended to suggest a general solution to the selection problem. But it demonstrates that it is possible to detect a decomposition in $O(n^2)$ time if additional assumptions are made concerning the class of functions considered.
Each discrete function of n binary variables can be written in the following form.

$$f(x) = a_0 + \sum_{i=1}^{n} a_i x_i + \sum_{1 \le j_1 < j_2 \le n} a_{j_1 j_2} x_{j_1} x_{j_2} + \dots \tag{41}$$

Definition: *A function f is hierarchically decomposable of order k if $a_{j_1 \dots j_l} = 0$ for $l > k$. Furthermore $a_{j_1 \dots j_l} \neq 0$ for $l \le k$ if and only if all lower order coefficients containing a subset of the indices are not equal to zero.*

THEOREM 6 *If the function f is hierarchically decomposable of order k and if for each j_1 the number of coefficients $a_{j_1 j_2} \neq 0$ is less or equal to k then the coefficients of Equation 41 can be determined in at most $O(n^2 + k^{(k-1)} n)$ function evaluations.*

Proof: The proof is done by computing the representation (41). We have

$$\begin{aligned} a_0 &= f(0,\ldots,0) \\ a_i &= f(0,\ldots,0,1,0\ldots 0) - a_0 \quad i = 1,\ldots,n \\ a_{j_1,j_2} &= f(0,\ldots,1,0,\ldots,1,0,\ldots,0) - a_0 - a_{j_1} - a_{j_2} \quad 1 \le j_1 < j_2 \le n \end{aligned}$$

This computation needs $O(n^2)$ steps. In the next step we compute the coefficients of order 3 for $a_{j_1 j_2} \neq 0$ and $a_{j_2 j_3} \neq 0$. We obtain

$$a_{j_1 j_2 j_3} = f(x_{j_1 j_2 j_3}) - a_0 - a_{j_1} - a_{j_2} - a_{j_3} - a_{j_1 j_2} - a_{j_2 j_3} - a_{j_1 j_3}.$$

$x_{j_1 j_2 j_3}$ denotes the vector with bits on at loci j_1, j_2, j_3 and 0 everywhere else. The number of function evaluations is at most $n * k * k$. This can be shown as follows. Because of the hierarchical decomposition there are at most $k*n$ coefficients $a_{j_1 j_2} \neq 0$ and at most k coefficients $a_{j_2 j_k} \neq 0$ for given j_2. This means that only $k^2 n$ coefficients $a_{j_1 j_2 j_3}$ are not equal to zero.

Now for $k \geq 4$ the fourth order coefficients are computed under the assumption $a_{j_1 j_2 j_3} \neq 0$ and $a_{j_3 j_4} \neq 0$. Here the number of function evaluations is $k^2 n * k$. This procedure is iterated until order k. The coefficients of k-th order are computed in $O(k^{k-1} n)$ steps. □

For hierarchically decomposable functions an additively decomposed function can be easily generated after the subsets s_k with $a_{j_1,..j_k} \neq 0$ have been determined. We just discuss a simple example. Let the representation be

$$f(x) = a_0 + \sum_{i=1}^{3} a_i x_i + a_{12} x_1 x_2 + a_{23} x_2 x_3.$$

Then $s_1 = \{x_1, x_2\}$ and $s_2 = \{x_2, x_3\}$. Now set

$$\begin{aligned} f(x) &= f_1(x_1, x_2) + f_2(x_2, x_3) \\ f_1(x_1, x_2) &= 1/2 a_0 + a_1 x_1 + 1/2 a_2 x_2 + a_{12} x_1 x_2 \\ f_2(x_2, x_3) &= 1/2 a_0 + a_3 x_3 + 1/2 a_2 x_2 + a_{23} x_2 x_3 \end{aligned}$$

8.2. Detection of Independent Variables

In statistics a discrete version of the mixed partial derivative is often used to detect that two variables are contained in different subsets s.

LEMMA 1 *Define for x and $\{\overline{i,j}\} = I_n \setminus \{i,j\}$*

$$\begin{aligned} \Delta_{i,j}(f, x_{\{\overline{i,j}\}}) := & [f(x_i{=}1, x_j{=}1, x_{\{\overline{i,j}\}}) - f(x_i{=}0, x_j{=}1, x_{\{\overline{i,j}\}})] \\ & - [f(x_i{=}1, x_j{=}0, x_{\{\overline{i,j}\}}) - f(x_i{=}0, x_j{=}0, x_{\{\overline{i,j}\}})] \end{aligned} \tag{42}$$

Then we have: $\Delta_{i,j}(f, x_{\overline{\{i,j\}}}) = 0$ *for all* $x \in X$ *if and only if there is no* k *such that* $X_i, X_j \in s_k$.

A general dependency measure for two variables has been proposed by Linfoot. It is now known as Kullback-Leibler cross entropy measure (Jordan, 1999). It is similar to the measure of linkage disequilibrium used in quantitative genetics (Mühlenbein, 1998).

$$Dep(X_i, Y_j) = \sum_{x_i, y_j} (p(x_i, x_j) \log p(x_i, x_j) - p(x_i)q(x_j) \log p(x_i)p(x_j)) \tag{43}$$

We have $Dep(X_i, Y_j) \geq 0$ and $Dep(X_i, Y_j) = 0$ if and only if X_i and Y_j are independent. Dep gives a quantitative measure of dependency, not just a binary variable with values dependent/independent.

8.3. Learning the Structure of the Probability Model

For graphical models a number of methods have been proposed to learn the structure of the probability model. The interested reader is referred to (Jordan, 1999). The proposed methods require between $O(n^2)$ and $O(n^2 * N)$ computations. In the next section we discuss FDA in the context of the schema theorem of genetic algorithms.

9. The Schema Theorem and Factorization

Our results provide also additional insight into the *schema theorem* of genetic algorithms. We state the schema theorem in its simplest form (Goldberg (1989), Holland (1992)).

THEOREM 7 (SCHEMA THEOREM) *For a genetic algorithm using proportionate selection, mutation rate* p_m*, and crossover rate* p_c*, the expected proportion* $p(H, t)$ *of a schema in population* $P(t)$ *changes according to*

$$p(H, t+1) \geq p^s(H, t) \, (1 - e(p_c, p_m)(1 - p(H, t))) \tag{44}$$

where

$$\begin{aligned} p^s(H, t) &= p(H, t) \frac{f(H, t)}{\bar{f}(t)} \\ p(H, t) &= \sum_{x \in H \cap P(t)} p(x, t) \\ f(H, t) &= \frac{1}{p(H, t)} \sum_{x \in H \cap P(t)} p(x, t) f(x) \\ \bar{f}(t) &= \sum_{x \in P(t)} p(x, t) f(x) \end{aligned}$$

We do not need the numerical expression of the factor e here. The schema theorem is used to argue as follows: *If a schema H is always a factor $1+c$ better than the average, it will increase exponentially in the population* (Goldberg, 1989). This follows from

$$p(H,t+1) \geq (1+c)p(H,t)(1-e(p_c,p_m)).$$

This qualitative argument cannot be used to explain the dynamic behaviour of schemata. First, the value of a schema $f(H,t)$ depends on the population $P(t)$. Second, we need precise estimates of schemata *containing the optimum* to prove convergence of the genetic algorithm to the optimum. Other schemata are irrelevant. But one can easily construct fitness functions where the frequencies of all schemata containing the optimum decrease. The most popular example is the "deceptive function" of order 3 defined in Section 2. It is known that a genetic algorithm with proportionate selection and crossover converges to $x = (0,\ldots,0)$ instead of $x = (1,\ldots,1)$. This result has been called "anomalous" because it violates the conclusion drawn from the schema theorem. But there is nothing anomalous. The schema theorem cannot be used to explain the evolution of schemata in a population.

We will show the complexity of the dynamics of the frequencies of schemata with BEDA and FDA. Let us recall from Equation 4 that the fixed variables of H define the corresponding marginal distribution. If s_H denote the fixed variables then $p(\pi_{s_H}x,t) = p(H,t)$. The schema theorem of BEDA is only valid for Boltzmann selection, not proportionate selection. Therefore we define the fitness of a schema H and the average of the population for Boltzmann selection. These are given by

$$f_v(H,t) = \frac{1}{p(H,t)} \sum_{x \in H \cap P(t)} p(x,t) v^{f(x)}$$
$$\bar{f}_v(t) = \sum_{x \in P(t)} p(x,t) v^{f(x)}.$$

From Theorem 1 easily follows:

Proposition 4: *For BEDA the distribution of any schema H is given by*

$$p(H,t+1) = p^s(H,t) = p(H,t)\frac{f_v(H,t)}{\bar{f}_v(t)} \tag{45}$$

Proof: We have for all x

$$p(x,t+1) = p(x,t)\frac{v^{f(x)}}{\sum_y p(y,t)v^{f(y)}}$$

Summing over all x which are an element of H gives the assertion. □

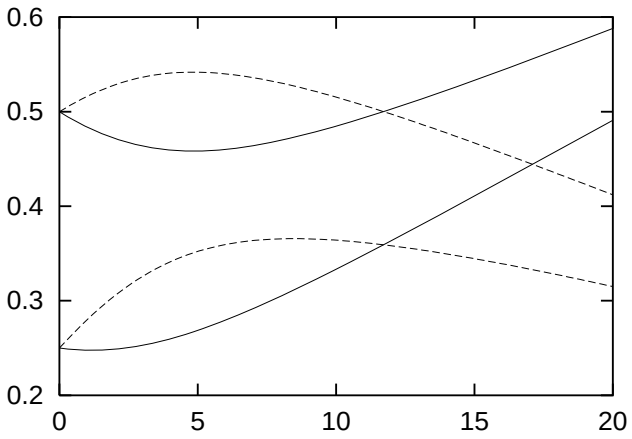

Figure 1. Distribution of schemata $(*, *, 1)$ (solid line), $(*, *, 0)$, $(*, 1, 1)$ (solid line) and $(*, 0, 0)$ for BEDA with $v = 1.5$ optimizing a deceptive function of order three

Proposition 4 can be used to compute the distribution of schemata for any fitness function. In Figure 1 the deceptive function f^3_{dec} is optimized. Even with BEDA the schemata containing the optimum decrease at first, whereas the "wrong schemata" containing not the optimum increase. This means that at first the value of the "good" schemata $f(H, t)$ is less than the value of the corresponding wrong schemata not containing the optimum. But in contrast to genetic algorithms BEDA or FDA cannot be deceived in the long run. After a certain number of generations the good schemata increase and overtake the wrong schemata after generation 12. This example shows how complex the dynamics of the distribution of the schemata can be. The dynamics is defined by the nonlinear difference equation 45.

The same behaviour has already been observed for the genetic algorithm by Goldberg (1989). For a genetic algorithm there are even two bit problems, where the GA will not converge to the optimum.

BEDA is only a conceptual algorithm. Therefore it is unfair to compare BEDA with a genetic algorithm. But FDA can be compared with a GA. This comparison provides new insight into another qualitative argument used in the theory of GAs, the *building block hypothesis* (see Goldberg, 1989). From the schema theorem it is derived that by a GA "short, low-order, and highly fit schemata are sampled, recombined, and resampled to form strings of potentially higher fitness." These schemata are called the building blocks. We investigate this argument for FDA. The following proposition follows from our factorization theorem.

Proposition 5: *Let a valid factorization (21) be given. Then for any schema H which is defined on a set consisting of a union of sets of the factorization we have*

$$p(H, t+1) = p(H, t)\frac{f_v(H, t)}{\bar{f}_v(t)} \tag{46}$$

The proposition shows that for FDA the *important building blocks are given by the factors of exact factorizations*. A lower bound on the size of the building blocks is given by the size of the sets s_i. Building blocks of smaller size are not contained in any valid factorization. If FDA is used with an approximate factorization based on building blocks that are too small, then FDA might not converge to the optimum.

For genetic algorithms with proportionate selection the analysis of the dynamics of the distribution of schemata is very difficult. Only two or three bit problems have been investigated analytically. An in-depth numerical study of the simple genetic algorithm has been made by Forrest and Mitchell (1993). They reached the following conclusions

- Overlap in the defined loci is the principal reason for the difficulty in optimizing the function.
- The lack of information from low-order schemata was a secondary cause of GA's poor performance.

Both problems are solved with our FDA. Overlap is not a problem as long as an exact factorization can be computed. The factorization also determines the schemata which guide the FDA to obtain the optimum.

We would like to mention that Mühlenbein (1998) has given many arguments that string based Mendelian recombination used in genetic algorithms is not able to detect higher order schemata necessary to optimize difficult ADFs. This fact explains all the "anomalous" results of genetic algorithms.

10. Approximate Convergence Results

For Boltzmann selection we have analytically derived exact difference equations for the marginal distributions. FDA mainly depends on the factorization, not on the *f*unction values. Numerical experiments have confirmed that the behaviour of FDA is very similar for functions having a similar factorization. This can be explained as follows. If the subsets s_i are disjoint, then the fitness function can be mapped to a generalized linear function with macro variables m_i which can be defined as integer representations of s_i. Therefore FDA should behave similar to an UMDA used with integer variables (Mühlenbein and Mahnig (1999a)) if the ADF is separable. This should also be true for ADFs where the sets s_i overlap in a chain-like manner.

For UMDA the *OneMax* function plays a central role. It leads to a "typical" fitness distribution for ADFs. For OneMax we are able to compute the distribution $p(x, t)$ both for Boltzmann selection and truncation selection.

THEOREM 8 *For Boltzmann selection with basis v the probability distribution for $OneMax$ is given by*

$$p(x,t) = \frac{v^{tf(x)}}{(1+v^t)^n} \tag{47}$$

The number of generations needed to generate the optimum with probability $1-\epsilon$ is given by

$$GEN_\epsilon \approx \frac{ln\frac{n}{\epsilon}}{ln(v)} \tag{48}$$

Proof: The first two assertions of the theorem follow from Theorem 1. We note that for *OneMax*

$$\sum_y v^{tf(y)} = \sum_{i=0}^{n} \binom{n}{i} v^{ti} = (1+v^t)^n$$

The third assertion is obtained by solving the equation

$$\frac{v^{tn}}{(1+v^t)^n} = 1-\epsilon$$

Using a Taylor expansion we obtain from

$$v^t = \frac{1}{1-\sqrt[n]{1-\epsilon}} - 1$$

the result $v^t \approx n/\epsilon$. □

For truncation selection an approximate analysis was already done in (Mühlenbein et al. 1994, Mühlenbein, 1998). For simplicity we assume that in the initial population all univariate marginal distributions are equal ($p_i(x_i = 1, t = 0) = p_0$). Then $p_i(x_i = 1, t) := p(t)$ for all t.

THEOREM 9 *For truncation selection τ with selection intensity I_τ the marginal probability $p(t)$ obeys for $OneMax$*

$$p(t+1) = p(t) + \frac{I_\tau}{n}\sqrt{np(t)(1-p(t))}. \tag{49}$$

This equation has the approximate solution ($p_0 := p(0)$)

$$p(t) = 0.5\left(1 + sin\left(\frac{I_\tau}{\sqrt{n}}t + arcsin(2p_0-1)\right)\right) \tag{50}$$

where

$$t \leq \left(\frac{\pi}{2} - arcsin(2p_0-1)\right)\frac{\sqrt{n}}{I_\tau}$$

The number of generations till convergence is given by

$$GEN_e = \left(\frac{\pi}{2} - arcsin(2p_0 - 1)\right) \frac{\sqrt{n}}{I_\tau}. \tag{51}$$

The relation between τ and I_τ depends on the fitness distribution (Mühlenbein, 1998). Assuming that the fitness distribution is normal, I_τ can be computed from the error integral. For normal distribution we obtain the values shown in Table 4.

Table 4. Selection intensity

τ	0.75	0.5	0.25	0.125	0.06
I	0.42	0.8	1.27	1.65	1.97

A very different fitness distribution is generated by the function

$$Int(n) = \sum_{i=1}^{n} 2^{i-1} x_i \tag{52}$$

Here all 2^n assignments of x have a different fitness value. We first consider truncation selection with $\tau = 0.5$ and a large population size. After one generation of selection the n-th bit will be fixed. The other bits will not be affected by selection. After the next generation bit $(n-1)$ will be fixed etc. Convergence to the optimum is achieved after n generations.

For truncation selection with $\tau = 0.25$ two bits will be fixed in every generation. Convergence will be reached after $n/2$ generations. Therefore we obtain for Int

THEOREM 10 *For truncation selection with $\tau = 2^{-k}; k \geq 1$ the number of generations to converge to the optimum for Int is given by*

$$GEN_e = \frac{n}{k} \tag{53}$$

For the fitness distribution derived from Int the selection intensity is $I_\tau = k$ for $\tau = 2^{-k}$. Therefore GEN_e scales inversely proportionate to I_τ. The same result was obtained for $OneMax$. But for this function GEN_e scales proportionate to n.

A big problem for all population based search methods is their dependency on the population size. Here UMDA and FDA have a nice numerical property. If the population is larger than a population $N^*(\tau)$ then

$$GEN_e(N) = GEN_e(N = \infty) \quad N \geq N^*(\tau) \tag{54}$$

This behaviour has been confirmed by many numerical experiments. It means that the number of generations to converge remains constant for $N \geq N^*(\tau)$. $N^*(\tau)$ is

called the *critical population size*, defined as the minimal population size needed to find the optimum with high probability, e.g. equal to 99%. The determination of the critical population size $N^*(\tau)$ is very difficult. We did not yet succeed with an analytical formula.

For determining an "optimal" τ, we have to compute for each τ the critical population size $N^*(\tau)$.

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

$$\tau_{opt} = \min_{\tau} FE(\tau) = \min_{\tau} GEN_e(\tau) * N^*(\tau) \tag{55}$$

It is obvious that strong selection reduces GEN_e. But for strong selection the population size $N^*(\tau)$ has to increase in order that FDA does not converge prematurely. The relation between selection and population size N^* has been investigated by Mühlenbein and Schlierkamp-Voosen (1993b) for genetic algorithms and by Mühlenbein and Mahnig (1999b) for FDA. In both cases the same result was obtained.

Rule of Thumb: *The optimal truncation threshold for FDA is contained in* $0.125 \leq \tau \leq 0.4$. *A good choice is* $\tau \approx 0.3$.

Asymptotically truncation selection needs more number of generations to convergence than Boltzmann selection. GEN_e is of order $O(ln(n))$ for Boltzmann selection and of order $O(\sqrt{n})$ for truncation selection. But if the basis v is small (e.g. $v = 1.2$), and $\epsilon = 0.01$ then even for $n = 1000$ truncation selection converges faster than Boltzmann selection.

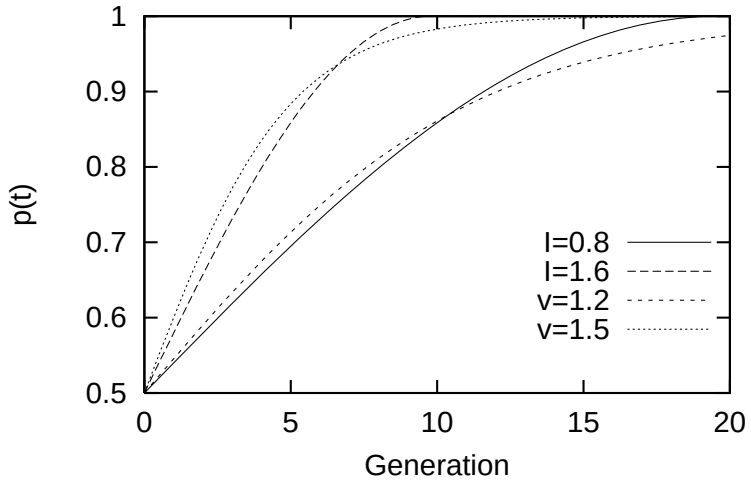

Figure 2. Probability $p(t)$ for OneMax with Truncation selection and Boltzmann selection

The different behaviour of Boltzmann selection and truncation selection is shown in Figure 2. The two equations 47 and 50 are plotted for reasonable values of v and I. For $v = 1.2$ Boltzmann selection selects slightly more severe than truncation selection with $I = 0.8$ at the beginning. It gets weak when the population approaches the optimum. The same behaviour can be observed for $v = 1.5$. In fact, all selection methods using proportionate or even exponential proportionate selection like Boltzmann selection have this problem. If the fitness values in the population differ only slightly, selection gets weak. Truncation selection does not have this problem. It selects much stronger than Boltzmann selection when approaching the optimum. Therefore truncation selection with $I = 1.6$ converges faster than Boltzmann selection for $v = 1.5$.

The convergence of Boltzmann selection can be speeded up if an annealing schedule is used. This means that the basis v has to be changed during the run. But how and when should this be done?

Mühlenbein and Mahnig (1999b) have taken a different approach. They show that FDA used with truncation selection leads for the fitness distribution of $OneMax$ to a sequence of Boltzmann distributions. This means that truncation selection implicitly generates an annealing schedule. Table 5 gives the annealing schedule for two values of τ. We conjecture that the annealing schedule for $\tau = 0.3$ is almost optimal.

Table 5. Annealing schedule $T(t)$ for $OneMax$ and $n = 32$

t	1	2	3	4	5	6	7	8	9	10
$\tau = 0.5$	3.51	1.73	1.13	0.82	0.63	0.49	0.39	0.30	0.21	0
$\tau = 0.3$	2.41	1.16	0.73	0.51	0.35	0.21	0			

We summarize the major results for truncation selection.

- GEN_e is bounded by n/I_τ for $\tau \leq 0.5$.
- For "typical" fitness distributions GEN_e is proportionate to $\sqrt{n}/I_\tau$ for $\tau \leq 0.5$.

This means that FDA will converge in at most n steps for $\tau \leq 0.5$. Usually it will converge in about $O(\sqrt{n})$ steps. FDA will not converge to the optimum if the population size is too small.

In the next section we show that the approximate analysis predicts the numerical results exactly.

11. Numerical results

This section has two purposes. First, we want to show that FDA behaves like the theory predicts. Second, we will show that it can solve difficult optimization problems.

We will first investigate the conjecture concerning the number of generations until equilibrium. In addition to $F_1(X) = OneMax(n)$ the following two functions will be analyzed:

$$F_2(X) = \sum_{i=1}^{l} f_2(\Pi_{s_i} x) \quad s_i = \{x_{3i-2}, x_{3i-1}, x_{3i}\}$$

$$F_3(X) = \sum_{i=1}^{l} f_3(\Pi_{s_i} x) \quad s_i = \{x_{3i-2}, x_{3i-1}, x_{3i}\}$$

In F_2 we set $f_2(\Pi_{s_i} x)$ to the values of the *OneMax* function of order three. F_2 is thus identical to F_1. For F_3 we set $f_3(1,1,1) = 10$ and all other values to zero.

Given our theory we expect the following results. GEN_e should be equal for $F1$ and $F2$. GEN_e should be smaller for $F3$ because here FDA has to test only two main alternatives — $(1,1,1)$ and all the rest. For FDA F_3 is just like a $OneMax$ function of size $n/3$, where the probability of generating the important substring $(1,1,1)$ is smaller. With random initialization the string with $(1,1,1)$ will be generated with probability $p_0 = 1/8$. For all cases the expected number of generations Gen_e can be computed from Equation (51).

Table 6. Generations GEN_e until convergence, truncation threshold 0.3

n	F_1	F_2	F_3	$GEN_e(p_0 = 0.5))$	$GEN_e(n/3, p_0 = 0.125)$
30	7.0	7.0	6.2	7.2	6.4
60	10.0	10.0	9.0	10.1	9.0
90	12.2	12.3	11.0	12.4	11.0
120	14.2	14.4	12.9	14.4	12.7
120_{GA}	18.8	18.8	21.3		
150	16.0	16.3	14.1	16.0	14.3
180	17.2	17.8	15.9	17.5	15.6

The results from Table 6 confirm our prediction. Note how precisely Equation 51 predicts GEN_e obtained from actual simulation with FDA. GA is a genetic algorithm with truncation selection and uniform crossover. It needs slightly more generations for *OneMax* than UMDA. This was already observed in (Mühlenbein et al., 1993b). For the function F_3 the genetic algorithm needs almost twice as many generations as FDA, which has knowledge about the micro-structure of $F3$.

We will now turn to more difficult fitness functions. The following test suite is used. The deceptive sub-function f^3_{dec} has been defined before. It is used to define the separable *deceptive* function of order three

$$F_{Dec}(x) = \sum_{i=1}^{l} f^3_{dec}(\Pi_{s_i} x).$$

s_i are non-overlapping sets of three variables. The factorization of the distribution is obvious, because the function is separable.

The next function is composed of two sub-functions. Let u denote the number of bits turned on in the sub-string. We define

$$f_1^l = \begin{cases} l & \text{for } u = 0 \\ l-1 & \text{for } u = 3 \\ 0 & \text{for } else \end{cases}$$

Function f_2^l has only one non zero element, $f_2^l(1,1,1) = l$. These two functions are used to define the function $F_{IsoChain}$

$$F_{IsoChain}(x) = \sum_{i=1}^{l-1} f_1^l(\Pi_{s_i} x) + f_2^l(\Pi_{s_l} x)$$

where $s_i \cap s_{i+1} = x_{2i+1}$. This function is not separable, it has a chain like interaction structure. The variables x_{2i+1} are contained in two sets. This function is very difficult to optimize. The global optimum is $x_m = (1, 1, \ldots, 1)$ with $F_{IsoChain}(x_m) = l(l-1)+1$. This optimum is triggered by f_2^l. It is very isolated, the second best optimum is given by $x = (0, 0, \ldots, 0)$ with a function value of $l(l-1)$. For this function a factorization with $b_i = s_i \setminus s_{i-1}$, $s_0 = \emptyset$ and $c_{i+1} = x_{2i+1}, c_1 = \emptyset$ has been used.

For numerical comparison we also consider the separable function

$$F_{Chain}(x) = \sum_{i=1}^{l-1} f_1^l(\Pi_{s_i} x) + f_2(\Pi_{s_l} x),$$

where $s_i \cap s_{i+1} = \emptyset$.

The last example is a two dimensional Ising system, defined in Equation 29. It is known that computing the minimum energy of an 2-D Ising model is an NP problem. But if we restrict $J_{ij} \in \{-1, 1\}$ a polynomial algorithm is known. Toulouse (1977) has reinterpreted the energy minimization of 2-D Ising models as an optimal matching in a graph whose nodes are just the frustrated plaquettes of the underlying Ising lattice. A square plaquette is frustrated if it has exactly one or three couplings $J_{ij} = -1$ on its perimeter. A frustrated plaquette cannot attain the 4-term minimal energy of -4, whereas an unfrustrated one can. It has been shown earlier that there are polynomial algorithms to compute a matching with minimum weight (Lawler, 1976). We have used this method to compute the exact solution of a fairly difficult free boundary problem on a 11*11 grid.

We recall that an exact factorization requires sets of order $O(\sqrt{n})$. Therefore we have used an approximate factorization. The defining sets s_i are given by four neighbors defining a square. If we sequentially number the spins then for a $11 * 11$ grid we obtain

$$\tilde{p}(x) = p(x_1, x_2, x_{12}, x_{13}) p(x_3, x_{14} | x_2, x_{12}) * \ldots * p(x_{121} | x_{109}, x_{110}, x_{120})$$

This factorization violates the running intersection property.

Table 7 gives the numerical results. In order to shorten the computation time, the runs have been stopped after the first occurrence of the optimum, after all individuals are equal, or after a specified maximum number of generations. GEN gives the generation count when stopped.

Table 7. Numerical Results, truncation threshold 0.3

F	n	l	$Alg.$	GEN	$popsize$	$best$
F_{Dec}	90	30	$FDA_{0.0}$	11	1000	30
F_{Dec}	90	30	$FDA_{0.5}$	7	1000	30
F_{Dec}	90	30	GA_T	32	5000	27.1*
F_{Chain}	60	20	$FDA_{0.5}$	5	1000	400
F_{Chain}	90	30	$FDA_{0.5}$	7	1000	900
F_{Chain}	120	40	$FDA_{0.5}$	9	1000	1600
F_{Chain}	90	30	GA_T	25	5000	900
$F_{IsoChain}$	51	25	$FDA_{0.0}$	6	1000	601
$F_{IsoChain}$	101	50	$FDA_{0.0}$	9	2000	2451
$F_{IsoChain}$	151	75	$FDA_{0.0}$	13	5000	5511
$F_{IsoChain}$	151	75	$FDA_{0.5}$	4	1000	5511
$F_{IsoChain}$	101	50	GA_T	30	5000	2450*
F_{Ising}	121		$FDA_{0.5}$	11	1000	178
F_{Ising}	121		$FDA_{0.5}$	11	2000	178
F_{Ising}	121		GA_T	40	5000	174*

GEN mainly depends on n, despite the differences of the fitness functions. This was predicted by the asymptotic theory. The scaling is about $O(\sqrt{n})$, even for $IsoChain$. There is not a large difference between the results of F_{Ising} and of the separable F_{Chain}, if the same number of variables is considered. For separable functions an initialization according to Equation 33 speeds up the convergence. For F_{Dec} the number of generations is reduced from 11 to 7. A dramatic improvement can be observed for $F_{IsoChain}$. For $n = 151$ the number of generations are reduced from 13 to 4. In addition N can be decreased from 5000 to 1000. This result can be explained. By using Equation 33 strings $(0, 0, 0)$ and $(1, 1, 1)$ are mainly generated at initialization because the fitness values of other strings are 0. FDA then only has to evaluate these two alternatives. For random initialization Equation 51 is a very good prediction for GEN_e.

We have already mentioned that GEN remains constant if the population size is larger than the critical population size. This is shown with the Ising model. GEN_e is 11 both for $N = 1000$ and $N = 2000$.

For FDA the complexity of the optimization problem is only reflected in the population size N. The more difficult the optimization, the larger the population size has to be. The Ising model needs a surprisingly small population size, whereas the critical population size is largest for $F_{IsoChain}$ with random initialization. The

optimization of $F_{IsoChain}$ for $n = 201$ needs a very large population size ($N = 20000$).

For comparison we also note the results of a genetic algorithm GA_T with uniform crossover and truncation selection. The genetic algorithm is not able to find the optimum of the separable function F_{Dec} and the function $F_{IsoChain}$. The solution of the 2-dimensional Ising model seems surprisingly simple. Even GA_T found the optimum once in 10 runs. In all cases FDA outperforms the genetic algorithm by far, in quality of solution obtained and/or in number of function evaluations needed to obtain the optimum.

11.1. Optimum Number of Function Evaluations

In order to determine the minimum number of function evaluations needed to find the optimum, the critical population size has to be computed. Table 8 presents some results on the minimum number of function evaluations for F_{Dec}. To make the problem slightly more difficult, we have simulated F_{Dec} with a 4-bits deceptive problems with one overlapping variable which makes no contribution to the fitness function. The data are averaged over 10 runs, with a constant probability of success of 90%.

Table 8. Minimum number of FE for a deceptive problem of size 60.

I	0.42	0.8	1.27	1.65	2.14
GEN_e	26	15	9	7	5
F_{Dec}	10500	6400	5000	4800	7200

Note that GEN_e is inversely proportionate to the selection intensity I as predicted. The smallest number of function evaluations FE are obtained for $1.0 \leq I \leq 1.65$ corresponding to truncation thresholds of $\tau = 0.4$ and $\tau = 0.125$. Truncation selection is therefore a robust selection method. This experiment confirms our rule of thumb.

Next we investigate this problem with more difficult functions. They have one variable as overlap, the interconnection structure is a chain as before. The function $Fc4$ and $Fc5$ are composed of the following sub functions.

- Function F^3_{cuban1} and F^5_{cuban1}:

$$F^3_{cuban1} = \begin{cases} 0.595 \text{ for } 000 & 1.00 \text{ for } 100 \\ 0.200 \text{ for } 001 & 0.05 \text{ for } 101 \\ 0.595 \text{ for } 010 & 0.09 \text{ for } 110 \\ 0.100 \text{ for } 011 & 0.15 \text{ for } 111 \end{cases}$$

$$F^5_{cuban1}(xyzvw) = \begin{cases} 4 * F^3_{cuban1}(xyz) & if \quad v = y \quad and \quad w = z \\ 0 & otherwise \end{cases}$$

- Function F^5_{cuban2}:

$$F^5_{cuban2} = \begin{cases} \text{u} & \text{for} \quad 0***0 \\ 0 & \text{for} \quad 0***1 \\ \text{u} & \text{for} \quad 1***0 \\ \text{u} - 2 & \text{for} \quad 1***1 \end{cases}$$

- Function Fc4:

$$Fc4(x) = \sum_{j=0}^{L-1} F^5_{cuban1}(\Pi_{s_j} x) \tag{56}$$

  where the s_j are substrings of x containing 5 variables. The last variable of s_j is the first variable of s_{j+1}, L is odd.

- Function Fc5: Here the functions F^5_{cuban1} and F^5_{cuban2} are used alternating.

$$Fc5(x) = F^5_{cuban1}(\Pi_{s_0} x) + \sum_{j=0}^{(L-3)/2} (F^5_{cuban2}(\Pi_{s_{2*j+1}} x) + F^5_{cuban1}(\Pi_{s_{2*j+2}} x)) \tag{57}$$

  where the s_j are defined as for function $Fc4$ and $L = 4 * m + 1$

The optima of functions $Fc4$ and $Fc5$ are not given by summation of local optima. Furthermore the global optimum is not unique for every instance of the problem.

First we give numerical results for function $Fc4$ with 25 and 49 sub-functions giving 101 and 197 independent variables. For 25 sub-functions the unique global optimum is 61.6, for 49 components it is 119.2. The next optima are 119.16,119.12,119.08. The fitness values differ only at the fifth place. The requirement to hit the global optimum in every run is very hard. Therefore the number of function evaluations are high, compared to the functions solved before.

The results are shown in Table 9. GEN_e is inversely proportionate to I as predicted. The smallest number of function evaluations are obtained for $1.0 \leq I \leq 1.65$. This is another confirmation ffor our rule of thumb.

Function $Fc5$ was also consistently solved by our algorithm. The global optima for $n = 37, 69, 101, 133, 205$ are $f = 25.6, 47.2, 68.8, 90.4, 138.98$. The fitness values of the second best optima are almost as large as the fitness of the global optimum. Therefore it is a very severe condition to require that the global optimum is found.

Table 10 gives numerical results. Note that GEN_e increases like $\sqrt{n}\ln(n)$ with n. The optimal truncation threshold seems to be $\tau = 0.125$, in contrast to our rule of thumb. But this is due to the fact that not enough runs have been made. A more detailed investigation by Mühlenbein and Mahnig (1999b) gives $0.125 \leq \tau \leq 0.4$ as optimal τ.

Table 9. Numerical results for function $Fc4$

	$n = 101$			$n = 197$		
I	0.8	1.27	1.65	0.8	1.27	1.65
GEN_e	19	12	9	29	19	14
N	2000	2500	3000	2500	3500	4000
$Fc4$	38000	30000	27000	72500	66500	56000

Table 10. Numerical results for function $Fc5$, truncation threshold 0.125

n	37	69	101	133	205
GEN_e	6	8	9	12	14
N	500	850	1000	1400	3500
$Fc5$	3000	6800	10000	16800	49000

12. Conclusion

In this paper we have presented the Factorized Distribution Algorithm FDA. It uses a population of search points to optimize a given function. FDA efficiently optimizes a class of binary functions which have been too difficult for traditional genetic algorithms. For Boltzmann selection FDA behaves like an ideal "schema algorithm", because the schema equation is exactly fulfilled for all schemata which are compatible with a factorization of the probability distribution. The factorization theorem shows which schemata have to be used in order to find an optimum of the function. We have also shown that FDA with Boltzmann selection is an ideal simulated annealing algorithm. FDA uses an exact Boltzmann distribution, whereas simulated annealing only approximates the Boltzmann distribution.

FDA is a well-defined stochastic algorithm. It is a hybrid between genetic algorithms and simulated annealing. It has only one parameter to be set, the population size. This should be contrasted to genetic algorithms, where mutation, recombination and selection are problem specific. The theory of FDA combined with the results from Mühlenbein (1998) is able to explain the "anomalous" results of traditional genetic algorithms based on Mendelian recombination. The factorization theorem shows that overlap of the variables in the sets s_i is not making the optimization difficult by itself, but the difficulty depends on the size of the sets used in the factorization. The factorization theorem requires that the running intersection property is fulfilled. This property leads to inefficient factorizations for 2-D grids. Here the size of the sets of an exact factorization scales like $O(\sqrt{n})$ and therefore the complexity of computation scales like $O(2^{\sqrt{n}})$.

To obtain an efficient factorization sophisticated algorithms from graph theory are needed. So far, we derive the factorization by the simple method used to prove the main theorem. In this paper we have concentrated on FDA using exact factorizations. But we have numerically shown that FDA with approximate factorizations is also an efficient optimization method. The next step will be to detect an appropriate factorization. In the theory of graphical models this is called *learning* the model.

We do not want to give the impression that FDA should always be preferred to UMDA or genetic algorithms. There has to be made a distinction between the *complexity of the structure* of an ADF and the *complexity of the optimization problem.* A complex structure does not necessarily imply a complex optimization problem. For all structures there exist simple functions where the global optimum is given simply by the sum of the optima of the functions f_i. Such functions can be easily optimized by algorithms not using the structure of the ADF.

Our main factorization theorem shows that *all* ADFs with a simple exact factorization can be efficiently optimized by FDA. But FDA can also optimize efficiently using approximate factorizations. This was shown with the 2-D spin system. Determining good approximate factorizations is an area of active research in the theory of graphical models. FDA will profit from any progress concerning this problem.

Notes

1. Also with the Centre of Artificial Intelligence. ICIMAF. Cuba. ochoa@cidet.icmf.inf.cu

References

Aarts, E.H. & Korst, H.M. & van Laarhoven, P.J. (1997). Simulated Annealing. In Aarts, E. & Lenstra, J.K. (Eds.),*Local Search in Combinatorial Optimization.* Chichester:Wiley pp =121-136.

Baluja, S. & Davies, S. (1997). Using Optimal Dependency-Trees for Combinatorial Optimization: Learning the Structure of the Search Space. *Carnegie Mellon Report CMU-CS-97-107.*

De Bonet, J.S.& Isbell, Ch. L. & Viola, P. (1997). MIMIC: Finding Optima by Estimating Probability Densities. In Mozer,M. & Jordan, M. & Petsche, Th. (Eds) *Advances in Neural Information Processing Systems 9*

Dechter, R. & Dechter, A. & Pearl, J. (1990) Optimization in Constraint Networks Oliver R.M. & Smith, J.Q. (Eds). *Influence Diagrams, Belief Nets and Decision Analysis.* pp:411-426, New York:Wiley.

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

Forrest, S. & Mitchell, M. (1993). What Makes a Problem Hard for a Genetic Algorithm? Some Anomalous Results and Their Explanation. *Machine Learning*, 13:285-319.

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

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

Goldberg, D.E & Deb, K. & Kargupta, H. & Harik, G. (1993). Rapid, Accurate Optimization of Difficult Problems Using Fast Messy Genetic Algorithms. In S. Forrest (Ed) *Proc. of the*

Fifth Int. Conf. on Genetic Algorithms pp:56-64, San Mateo, CA: Morgan Kaufman.
Holland, J. (1992). *Adaptation in Natural And Artificial Systems.* Cambridge:MIT Press.
Jordan, M. (1999). *Learning in Graphical Models* Cambridge:MIT Press.
Kargupta, H. & Goldberg, D.E. (1997). SEARCH, Blackbox Optimization, And Sample Complexity. In R.K. Belew & M. Vose (Eds.) *Foundations of Genetic Algorithms 4.* San Mateo, CA: Morgan Kaufman.
Kargupta, H. (1997). *Revisiting The GEMGA: Scalable Evolutionary Optimization Through Linkage Learning.* Personal Communication.
Lauritzen, S.L. (1996) *Graphical Models.* Oxford:Clarendon Press.
Lawler, E.L. (1976) *Combinatorial Optimization: Networks and Matroids.* New York: Holt, Rinehart and Winston.
Mühlenbein, H. & Schlierkamp-Voosen, D. (1993a). Predictive Models for the Breeder Genetic Algorithm I. Continuous Parameter Optimization *Evolutionary Computation*, 1:pp. 25–49.
Mühlenbein, H. & Schlierkamp-Voosen, D. (1993b). The science of breeding and its application to the breeder genetic algorithm. *Evolutionary Computation*, 1:pp. 335–360.
Mühlenbein, H. (1998). The Equation for Response to Selection and its Use for Prediction. *Evolutionary Computation*, 5:pp. 303-346.
Mühlenbein, H. & Mahnig, Th.(1999a). Convergence Theory and Applications of the Factorized Distribution Algorithm. *to appear in Journal of Computing and Information Technology.*
Mühlenbein, H. & Mahnig, Th.(1999b). FDA - A scalable evolutionary algorithm for the optimization of additively decomposed discrete functions. *submitted for publication*
Mühlenbein, H. & Paaß, G. (1996). From Recombination of Genes to the Estimation of Distributions I. Binary Parameters. In Voigt, H.-M et al. (eds.)*Lecture Notes in Computer Science 1141: Parallel Problem Solving from Nature - PPSN IV*, pp. 178-187, Berlin:Springer.
Pelikan, M. & Mühlenbein, H. (1999). The Bivariate Marginal Distribution Algorithm, In Roy, R. & Furuhashi, T. & Chawdhry, P. K. (eds.), *Advances in Soft Computing - Engineering Design and Manufacturing*, pp. 521-535, Berlin:Springer-Verlag.
Tanese, R. (1989). *Distributed genetic algorithms for function optimization.* Unpublished doctoral dissertation, University of Michigan, Ann Arbor.
Toulouse,G. (1977). Optimal graph matching for 2-D Ising models. *Commun. Phys.* 2:pp 115-119.
Whitley, D. & Beveridge, R. & Graves, C. & Mathias,K (1995). Test Driving Three 1995 Genetic Algorithms: New Test Functions and Geometric Matching. *Journal of Heuristics*, 1:77–104.