# Convergence of Estimation of Distribution Algorithms for Finite Samples

**Heinz Mühlenbein**
Fraunhofer Institut Autonomous intelligent Systems
Schloss Birlinghoven 53757 Sankt Augustin, Germany
heinz.muehlenbein@online.de, http://www.ais.fraunhofer.de/∼muehlen

## Abstract

Estimation of Distribution Algorithms (EDA) have been proposed as an extension of genetic algorithms. Our algorithm *FDA* assumes that the function to be optimized is additively decomposed (ADF). The interaction graph $G_{ADF}$ is used to create exact or approximate factorizations of the Boltzmann distribution. We also discuss the algorithm *LFDA* which learns a Bayesian network from data. For both algorithms estimates of the necessary sample size $N$ to find the optimum are derived. The bounds are based on statistical learning theory and PAC learning. If the assumptions of a factorization theorem are fulfilled the upper bound of the sample size is of order $O(n \ln n)$ where n is the size of the problem. The computational complexity per generation is $O(N * n)$. For *LFDA* a bound cannot be proven because the network learned might be far from optimal. Estimates can be obtained for two new learning methods. The first one learns *factor graphs* instead of Bayesian networks, the second one detects the structure by computing Walsh coefficients. The computational complexity of *LFDA* in order to compute the structure of the Bayesian network is at least $O(N * n^2)$. The computational complexity to compute the Walsh coefficients is only $O(n^2 \ln n$. The networks computed by *FDA* and *LFDA* are analyzed for a set of benchmark functions.

## 1 Introduction

The *Estimation of Distribution* (EDA) family of population based optimization algorithms was introduced in [21] as an an extension of genetic algorithms. They address the problem that the search distributions implicitly generated by recombination and crossover do not use the correlation of the variables in samples of high fitness values. Therefore genetic algorithms have great difficulties solving optimization problems where a number of variables have to be changed simultaneously.

EDA uses probability distributions derived from the function to be optimized to generate search points instead of crossover and mutation as done by genetic algorithms. The other parts of the algorithms are identical. In both cases a population of points is used and points with good fitness are selected either to estimate a search distribution or to be used for crossover and mutation.

Today two major branches of EDA can be distinguished. In the first branch the factorization of the distribution is computed from the structure of the function to be optimized, in the second one the structure is computed from the correlations of the variables in samples of points with good fitness. The second branch has been derived from the theory of belief networks [10].

The outline of the paper is as follows. First we shortly summarize the results of our well-known algorithm *FDA*. Then we describe the algorithm *LFDA* which learns an acyclic Bayesian network from a sample of promising points. For both algorithms theoretical estimates of the sample size needed to compute the optimum are derived from statistical learning theory and PAC learning. The problem of computing good approximate distributions is discussed for both *FDA* and *LFDA*. A

new graphical representation based on *factor graphs* is presented. This representation was recently introduced by Abbeel et al. [1]. As an alternative to statistical learning of Bayesian network the technique from Wright et al. [27] is reviewed. Numerical results for *FDA* and *LFDA* are presented in section 8.

The paper is intended to inform about new developments in machine learning which might be useful for EDA algorithms. Many researchers are too much biased towards Bayesian networks and use an analyis mainly based on genetic algorithm research.

## 2 Factorization of the Search Distribution

EDA algorithms have been derived from a search distribution point of view. We just recapitulate the major steps published in [20, 17]. We will use in this paper the following notation. Capital letters denote variables, lower cases instances of variables. If the distinction between variables and instances is not necessary, we will use lower case letters. Vectors are denoted by $\mathbf{x}$, a single variable by $x_i$.

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

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

A good candidate for optimization using a search distribution is the Boltzmann distribution.

**Definition 1** *For $\beta \geq 0$ define the* Boltzmann distribution[1] *of a function $f(\mathbf{x})$ as*

$$p_\beta(\mathbf{x}) := \frac{e^{\beta f(\mathbf{x})}}{\sum_{\mathbf{y}} e^{\beta f(\mathbf{y})}} =: \frac{e^{\beta f(\mathbf{x})}}{Z_f(\beta)} \tag{2}$$

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

The Boltzmann distribution concentrates with increasing $\beta$ around the global optima of the function. Obviously, the distribution converges for $\beta \to \infty$ to a distribution where only the optima have a probability greater than 0 [18]. Therefore, if it were possible to sample efficiently from this distribution for arbitrary $\beta$, optimization would be an easy task. But the computation of the partition function usually needs an exponential effort for a problem of $n$ variables. We have therefore proposed an algorithm which incrementally computes the Boltzmann distribution from empirical data using Boltzmann selection.

**Definition 2** *Given a distribution $p$ and a selection parameter $\Delta\beta$,* Boltzmann selection *calculates the distribution for selecting points according to*

$$p^s(\mathbf{x}) = \frac{p(\mathbf{x})e^{\Delta\beta f(\mathbf{x})}}{\sum_y p(\mathbf{y})e^{\Delta\beta f(\mathbf{y})}} \tag{3}$$

The following theorem is easy to prove.

**Theorem 1** *If $p_\beta(\mathbf{x})$ is a Boltzmann distribution, then $p^s(\mathbf{x})$ is a Boltzmann distribution with inverse temperature $\beta(t+1) = \beta(t) + \Delta\beta(t)$.*

Algorithm 1 describes *BEDA*, the Boltzmann Estimated Distribution Algorithm.

**BEDA**

- **STEP 0:** $t \Leftarrow 1$. Generate $N$ points according to the uniform distribution $p(\mathbf{x}, 0)$ with $\beta(0) = 0$.
- **STEP 1:** With a given $\Delta\beta(t) > 0$ do Boltzmann selection giving the distribution $p^s(\mathbf{x}, t)$.

[1] The Boltzmann distribution is usually defined as $e^{-\frac{E(\mathbf{x})}{T}}/Z$. The term $E(x)$ is called the energy and $T = 1/\beta$ the temperature. We use the inverse temperature $\beta$ instead of the temperature.

- **STEP 2:** Generate $N$ new points according to the distribution $p(\mathbf{x}, t+1) = p^s(\mathbf{x}, t)$.
- **STEP 3:** $t \Leftarrow t+1$ until STOP reached.

*BEDA* is a conceptional algorithm, because the calculation of the distribution $p^s(\mathbf{x}, t)$ requires a sum over exponentially many terms. This problem is investigated next.

### 2.1 Factorization of the Distribution

In this section an efficient numerical algorithm is derived if the fitness function is additively decomposed.

**Definition 3** *Let $S_1, \ldots, S_m$, $S_i \subseteq \{1, \ldots, n\}$ be index sets. Let $f_i$ be functions depending only on the variables $x_j$ with $j \in S_i$. $S_j$ is called the scope of $f_j$. Then*

$$f(\mathbf{x}) = \sum_{i=1}^{m} f_i(\mathbf{x}_{s_i}) \tag{4}$$

*is an* additive decomposition *of the fitness function (ADF).*

**Definition 4** *Let an ADF be given. Then the interaction graph $G_{ADF}$[2] is defined as follows: The vertices represent the variables of the ADF . Two vertices are connected by an arc iff the corresponding variables are contained in a common sub-function.*

**Remark:** *ADF* defines a class, which covers all possible functions. In order to obtain algorithms of polynomial complexity we will later restrict the class. We will assume that the number of variables of the index sets is bounded by a constant independent from $n$.

Given an *ADF* we want to estimate the Boltzmann distribution (2) using a product of marginals. The approximation has to fulfill two conditions

- The approximation should use marginals of low order.
- Sampling from the approximation should be easy.

A class of distributions fulfilling these conditions are the acyclic Bayesian networks (acBN).

$$q(\mathbf{x}) = \prod_{i=1}^{n} \tilde{q}(x_i | \mathbf{pa}_i) \tag{5}$$

where $\mathbf{pa}_i$ are called the parents of $x_i$. For acyclic Bayesian networks sampling can easily be done starting with the root $x_1$. Cyclic Bayesian networks need a complex iterative sampling procedure or even Gibbs sampling.

Note that any distribution can be written in the form of an acyclic Bayesian network because of

$$p(\mathbf{x}) = p(x_1)p(x_2|x_1)\cdots p(x_n|x_1, \ldots, x_{n-1}) \tag{6}$$

But this factorization uses marginal distributions of size $O(n)$, thus sampling from the distribution is exponential in $n$. Therefore we are looking for factorizations where the size of the marginals is bounded independently of $n$.

For *ADF*'s the following procedure can be used to create a factorization of the distribution.

**Definition 5** *Given $S_1, \ldots, S_m$, we define the sets $D_i$, $B_i$ and $C_i$ for $i = 1, \ldots, m$ :*

$$D_i := \bigcup_{j=1}^{i} S_j, \qquad B_i := S_i \setminus D_{i-1}, \qquad C_i := S_i \cap D_{i-1} \tag{7}$$

*A FDA factorization is defined by*

$$q(\mathbf{x}) = \prod_{i=1}^{m} \tilde{q}(\mathbf{x}_{b_i} | \mathbf{x}_{c_i}) \tag{8}$$

*We demand $D_m = \{1, \ldots, n\}$ and set $D_0 = \emptyset$. In the theory of decomposable graphs, $D_i$ are called* histories, *$B_i$* residuals *and $C_i$* separators *[12].*

[2] [28] call it a decomposable Markov graph.

**Remark:** *Any FDA factorization can easily be transformed into an acyclic Bayesian network. Therefore the class of FDA factorizations is identical to the class of acyclic Bayesian networks.*

We next investigate the characteristics of the *FDA* factorization. The definition is stated informally.

**Definition 6** *A set of marginal distributions $\tilde{q}(\mathbf{x}_{b_i}, \mathbf{x}_{c_i})$ is called* consistent *if the marginal distributions fulfill the laws of probability, e.g.*

$$\sum_{\mathbf{x}_{b_i}, \mathbf{x}_{c_i}} \tilde{q}(\mathbf{x}_{b_i}, \mathbf{x}_{c_i}) = 1 \tag{9}$$

$$\sum_{\mathbf{x}_{b_i}} \tilde{q}(\mathbf{x}_{b_i}, \mathbf{x}_{c_i}) = \tilde{q}(\mathbf{x}_{c_i}) \tag{10}$$

**Proposition 1** *Let a consistent set of marginal distributions $\tilde{q}(\mathbf{x}_{b_i}, \mathbf{x}_{c_i})$ be given. If $B_i \neq \emptyset$ then the FDA factorization defines a valid distribution ($\sum q(\mathbf{x}) = 1$). Furthermore*

$$q(\mathbf{x}_{b_i}|\mathbf{x}_{c_i}) = \tilde{q}(\mathbf{x}_{b_i}|\mathbf{x}_{c_i}), \;\; i = 1, \ldots m \tag{11}$$

*whereas in general*

$$q(\mathbf{x}_{b_i}, \mathbf{x}_{c_i}) \neq \tilde{q}(\mathbf{x}_{b_i}, \mathbf{x}_{c_i}), \;\; i = 1, \ldots m \tag{12}$$

The proof follows from the definition of marginal probabilities. The proof of equation (11) is somewhat technical, but straightforward. The inequality (12) is often overlooked. It means that in general *sampling from the factorization does not reproduce the given marginals*. It needs additional assumptions to reproduce the unknown distribution. This was proven in [20] for the Boltzmann distribution.

**Theorem 2 (Factorization Theorem)** *Let $f(\mathbf{x}) = \sum_{i=1}^{m} f_{s_i}(\mathbf{x})$ be an additive decomposition. If*

$$\forall i = 1, \ldots, m; \quad B_i \neq \emptyset \tag{13}$$

$$\forall i \geq 2 \; \exists j < i \;\; \textit{such that } C_i \subseteq S_j \tag{14}$$

*then*

$$p_\beta(\mathbf{x}) = \prod_{i=1}^{m} p_\beta(\mathbf{x}_{b_i}|\mathbf{x}_{c_i}) = \frac{\prod_{i=1}^{m} p_\beta(\mathbf{x}_{b_i}, \mathbf{x}_{c_i})}{\prod_{i=2}^{m} p_\beta(\mathbf{x}_{c_i})} \tag{15}$$

The Factorization Theorem shows that under certain conditions the Boltzmann distribution can exactly be represented by a product of conditional marginals. We have proven the theorem only for the Boltzmann distribution, but from the theory of graphical models it follows that the theorem is valid for all distributions and also for *continuous variables* [12].

**Definition 7** *The constraint defined by equation* (14) *is called the* running intersection property (*RIP*). *The factorization is* polynomially bounded *(PBF) if the size of the sets $\{B_i, C_i\}$ is bounded by a constant independent of $n$.*

The above theorem does not answer the question how to compute a good or even an exact factorization. The construction defined by equation (7) depends on the sequence $S_1, \ldots, S_m$. If the sequence is permutated, it might be possible that the $RIP$ will be fulfilled, even if it was not fulfilled before. Furthermore, we can join two or more sub-functions, resulting in larger sets $\tilde{S}_i$. It might be that using these larger sets, the factorization becomes exact.

Let us discuss a simple example, the loop

$$S_1 = \{X_1, X_2\}, S_2 = \{X_2, X_3\}, S_3 = \{X_3, X_4\}, S_4 = \{X_4, X_1\}$$

The simplest *FDA* factorization would be

$$p(\mathbf{x}) = p(x_1, x_2)p(x_3|x_2)p(x_4|x_3)$$

This factorization leaves out the edge $X_4 \rightarrow X_1$. This problem can be solved by increasing the size of the clique

$$p(\mathbf{x}) = p(x_1, x_2)p(x_3|x_2)p(x_4|x_3, x_1) \tag{16}$$

This factorization violates the RIP because $X_1$ and $X_3$ are not contained in a common clique. A factorization fulfilling the RIP would be

$$p(\mathbf{x}) = p(x_1, x_2)p(x_3|x_2, x_1)p(x_4|x_3, x_1) \tag{17}$$

Testing all these combinations for an arbitrary *FDA* factorization is prohibitive. Actually, it turns out that the computation of an exact factorization is done better by investigating the corresponding interaction graph $G_{ADF}$. A well-known algorithm computes *junction trees* [9]. It obtains an exact factorization with marginals of small size and fulfilling the *RIP* given an arbitrary graph. A short description can be found in [14]. The largest clique of the junction tree gives the largest marginal of the factorization.

The space complexity of exact factorizations has been investigated in [5]. For many interesting problems like functions defined on grids of dimension two and higher *exact factorizations are not bounded polynomially*. Thus for larger problems, one has to use approximate factorizations.

### 2.2 The Factorized Distribution Algorithm

We first describe our algorithm *FDA*. It uses the cycle free *FDA* factorization introduced in definition 5.

**FDA**

- **STEP 0:** Set $t \Leftarrow 0$. Generate $N$ points randomly.
- **STEP 1:** Selection of points with high fitness.
- **STEP 2:** Compute the conditional probabilities $p^s(x_{b_i}|x_{c_i}, t)$ using the selected points.
- **STEP 3:** Generate a new population according to $p(\mathbf{x}, t+1) = \prod_{i=1}^{l} p^s(x_{b_i}|x_{c_i}, t)$
- **STEP 4:** If termination criteria is met, STOP.
- **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 1.

The next theorem follows from the Factorization Theorem and the convergence theorem of *BEDA*.

**Theorem 3** *If the FDA factorization fulfills the assumptions of the Factorization Theorem and FDA is run with Boltzmann selection, then FDA will converge for infinite populations to the optima of the function.*

Note that the theorem gives a *sufficient condition for convergence* which is in practical applications often not necessary. For convergence of EDA a good approximation of the exact distribution is *not* necessary. We only need to use a distribution where the *probabilities of the global optima* are high. We show the problem with a simple example.

**Example 1:**

$$f(\mathbf{x}) = \sum_{i=1}^{n} x_i + \prod_{i=1}^{n} x_i \tag{18}$$

The exact factorization is the complete distribution $p(x_1, \ldots, x_n)$. But obviously $FDA$ using the factorization

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i) \tag{19}$$

will easily find the optimum. The factorization is also a good approximation of the true distribution. Thus instead of using the exact distribution, it is possible to use a *simpler distribution which has the same global optima as the original distribution.*

Now we change the function.

**Example 2:**

$$f(\mathbf{x}) = \sum_{i=1}^{n} (1 - x_i) + (n+1) \cdot \prod_{i=1}^{n} x_i \tag{20}$$

This is a trap function of order n. In this case the factorization (19) is also a good approximation of the exact distribution. But selection will drive $p(x_i) \to 0$. Therefore no EDA algorithm will find the global optimum in polynomial time. Note that both functions are formally a needle-in-a-haystack problem.

**Example 3:**

$$f(\mathbf{x}) = \sum_{i=1,4,\ldots}^{n} x_i * x_{i+1} * x_{i+2} \tag{21}$$

Here the factorization (19) is a bad approximation, but EDA algorithms using this distribution will easily find the optimum. The exact factorization is of course

$$p(\mathbf{x}) = \prod_{i=1,4,\ldots}^{n} p(x_i, x_{i+1}, x_{i+2}) \tag{22}$$

If the factorization violates the assumption of the factorization theorem, one has to compute an approximate factorization given the graph $G_{\text{ADF}}$. A good heuristic will minimize the size of the cliques but simultaneously use all dependencies in $G_{\text{ADF}}$. Thus the heuristic generates graphs with $B_i \neq \emptyset$ which violate the RIP only in a few cases.

This is the idea of the *FDA sub-function merger algorithm*. Each new variable is included in a set together with the previous variables on which it depends. However, if another variable depends on a superset of variables, the two sets are merged. After completing the merge phase, the algorithm calculates $\tilde{C}_j$, $\tilde{B}_j$ and $\tilde{D}_j$ analogous to the construction given by (7).

This sub-function merger algorithm might compute too large cliques. Therefore a cut parameter $k$ is needed which bounds the clique size. If the clique size becomes larger than $k$ our implementation will randomly leave out arcs from $G_{ADF}$.

*FDA* has experimentally proven to be very successful on a number of functions where standard genetic algorithms fail to find the global optimum. In [16] the scaling behavior for various test functions has been studied. For recent surveys and a more detailed description of the algorithm, the reader is referred to [17, 19, 14, 15].

## 3 Learning a Factorization from Data

If the symbolic definition of the function is not known, the factorization has to be learned from the data. For learning the maximum likelihood can be used. We consider the class of *acyclic Bayesian networks* defined in equation (5).

In order to find the best network structure, just maximizing the likelihood is not enough. It will lead to densely connected networks. In order to give more weight to sparse Bayesian networks, a weight factor $\alpha$ is used which *penalizes the size of the network*. Let $B$ denote a Bayesian network, $D$ the given data set and $N = |D|$ its size. To score the networks we use the *Bayesian Information Criterion* proposed by Schwarz [24]. Let $H(B, D)$ be the *conditional entropy* of the network structure $B$ and data $D$.

$$H(B, D) = -\sum_{i=1}^{n} \sum_{\mathbf{pa}_i} \sum_{x_i} \frac{m(x_i, \mathbf{pa}_i)}{N} \log \frac{m(x_i, \mathbf{pa}_i)}{m(\mathbf{pa}_i)} \tag{23}$$

Here $m(x_i, \mathbf{pa}_i)$ denotes the number of occurrences of $x_i$ given configuration $\mathbf{pa}_i$. $m(\mathbf{pa}_i) = \sum_{x_i} m(x_i, \mathbf{pa}_i)$. If $\mathbf{pa}_i = \emptyset$, then $m(x_i, \emptyset)$ is set to the number of occurrences of $x_i$ in D. The $BIC$ score is defined as

$$BIC(B, D, \alpha) = -N \cdot H(B, D) - \alpha PA \cdot \log(N) \tag{24}$$

$PA$ is the total number of probabilities to compute. The term $PA \cdot \log(N)$ models the computational cost of computing the probabilities. Under certain assumptions Schwarz computed $\alpha = 0.5$ as the optimal weight. A more detailed derivation using the maximum entropy principle and the log-likelihood principle can be found in [14, 15].

To compute a network $B^*$ which maximizes $BIC$ requires a search through the space of all Bayesian networks. This is very expensive. We use the following greedy algorithm. $k_{max}$ is the maximum number of incoming edges allowed.

| u | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| $f_1$ | m | 0 | 0 | m-1 |
| $f_2$ | 0 | 0 | 0 | m |

Table 1: Definition of $f_1$ and $f_2$; m denotes the number of sub-functions

$$\mathbf{BN}(\alpha, \mathbf{k_{max}})$$

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

Checking whether an arc would introduce a cycle can be easily done by maintaining for each node a list of parents and ancestors, i.e. parents of parents etc. $(x_i \rightarrow x_j)$ introduces a cycle if $x_j$ is ancestor of $x_i$. This simple learning method has been first proposed by Heckerman and Friedman et al. in [10]. It is also used in *BOA* [22].

## 4 The Connection between *FDA* and *LFDA*

An important problem, not often investigated, is the analysis of the Bayesian network computed by the learning algorithm. Most researchers implicitly assume that the network is closely related to the interaction graph $G_{ADF}$. But a theoretical proof turns out to be very difficult. The following conjecture is a first step.

**Conjecture:** *Let the empirical distribution $\hat{p}(\mathbf{x})$ be generated by selection from an ADF . Then for a large sample size $N$ the mutual information is the largest between those variables which are contained in a common sub-function. This means that the graph constructed by LFDA should contain for $N \rightarrow \infty$ the interaction graph $G_{ADF}$.*

Thus for $N \rightarrow \infty$ a good learning algorithm should compute a graph which contains $G_{ADF}$. *LFDA* has a slight advantage compared to *FDA*, because it can use the actual mutual information to leave out less important arcs. But for complex dependencies *the simple greedy learning algorithm is obviously not able to compute the correct network.*

A proof of the above conjecture is difficult. So we decided to do numerical experiments. We will investigate the following test functions.

**F1:** The trap function $f_{Trap:k}$ of order $k$

$$f_{Trap:k} = \sum_{j=1,k+1,\ldots} f_{trap:k}(x_j, x_{j+1}, ..x_{j+k}) \tag{25}$$

$f_{trap:k}$ is a deceptive function of order $k$.

**F2:** The function $f_{Iso}$

$$f_{Iso} = \sum_{j=1,3,\ldots} f_1(x_j, x_{j+1}, x_{j+2}) + f_2(x_{n-2}, x_{n-1}, x_n) \tag{26}$$

$f_1$ has the maximum at $x = (0, 0, 0)$, whereas $f_2$ has the maximum at $x = (1, 1, 1)$. The functions are defined in table 1. The global maximum is $\mathbf{x} = (1, 1, \ldots, 1)$. It is very isolated. The function is very difficult to optimize, because the attractor of $\mathbf{x} = (0, 0, \ldots, 0)$ is much larger than the attractor of the global optimum.

**F3:** The 2-D grid Ising spin glass

$$f_{Ising} = \sum_{i,j} J_{i,j} s_i \cdot s_j \tag{27}$$

$j$ are the four neighbors of $i$ in the 2-D grid. The couplings $J_{i,j}$ are randomly drawn from a Gaussian distribution. The spins $s_i$ have values in $\{-1, 1\}$. The function is symmetric, therefore it has two global optima, which are the binary compliment from each other.

**F4:** Kauffman's $n : k$ function.
Here we have $n$ sub-functions, each with $k$ variables. For each variable $k-1$ variables are randomly connected, defining the scope of sub-function $f_i$. The function values of $f_i$ are randomly chosen in $[0, 1]$

$$f_{n;k} = \sum_i f_i(x_i, \ldots, x_j) \tag{28}$$

The function values of each $f_i$ are uniform distributed in $[0, 1]$.

The interconnection structure of the four functions are representative for a large class of functions. $f_{Trap:k}$ is a separable function. $f_{Iso}$ is defined on the real axis with an overlap of one variable, Kauffman's $n : k$ function has a random interconnection structure and the Ising model is defined on a 2-D grid. $f_{Iso}$ is difficult to optimize, there is a large attractor at $x = (0, 0, \ldots, 0)$, the global optimum at $x = (1, 1, \ldots, 1)$ is very isolated.

The optimization of Kauffman's $n : 3$ function is $NP - hard$. The computational complexity of the Ising spin glass is discussed in the next section.

### 4.1 The computational complexity of optimizing Ising spin glasses

It has been shown that the Ising problem $F3$ can be solved in polynomial time [2]. This poses a challenge for our major convergence result for *FDA* formulated in theorem 3. Any *exact* factorization defined on a 2-D grid needs a clique size of at least $O(\sqrt{n})$. This means that *FDA* needs an *exponential* effort for provable convergence for the Ising problem! Thus it seems that our convergence theorem is not very sharp, classifying problems solvable in polynomial time as not being solvable in polynomial time using EDA algorithms.

But further investigation shows, that the Ising problem with external magnetic field $h$

$$f_{Ising_{field}} = \sum_{i,j} J_{i,j} s_i \cdot s_j + \sum_i h_i \cdot s_i \tag{29}$$

is $NP - hard$ [2]. The interaction graph $G_{ADF}$ is the same for both classes of functions, therefore the *FDA* factorizations are also the same. Thus in *FDA* we cannot distinguish between the two problems. This is the reason that both problems can provable be solved in exponential time only.

Despite the many positive results in numerically solving the 2-D Ising problem, I conjecture that one cannot prove the convergence for EDA algorithms in polynomial time. Reporting numerical scaling results for the 2-D Ising problem without a caveat (see e.g. [23] is misleading the research community.

### 4.2 Investigation of the learned Bayesian network

In this section we numerically analyze the structure of the networks learned. Let us start with a simple artificial example, the function $Iso$ with 9 variables. We restrict the number of parents to 1. The first two equations show optimal networks within this class, the next three equations are networks learned by *BOA* and *LFDA*. The representation is compact, $(0|1)$ meaning $p(x_0|x_1)$

$$p(x) = (0)(1|0)(2|0)(3|2)(4|3)(5|3)(6|5)(7|6)(8|6) \tag{30}$$
$$p(x) = (0|2)(1|2)(2|3)(3|5)(4|5)(5|6)(6|7)(7|8)(8) \tag{31}$$
$$p(x) = (0|1)(1|2)(2|6)(3|1)(4))(5|8)(6|2)(7|0)(8|2) \quad \text{BOA} \tag{32}$$
$$p(x) = (0|1)(1|3)(2|1)(3)(4|1)(5|6)(6|4)(7)(8|6) \quad \text{BOA} \tag{33}$$
$$p(x) = (0|1)(1|2)(2|3)(3|6)(4|2)(5|0)(6|8)(7|1)(8) \quad \text{LFDA} \tag{34}$$

The maximal number of correct edges can be at most 8, two examples are given in equations (30) and (31). Both networks have only one arc in common! The next two lines are the networks generated by *BOA* in generation 1 and 2. Note that in generation 2 two correct conditionals for the critical sub-function $f_2(x_6, x_7, x_8)$ appear. This seems to be necessary to find the optimum. *BOA* generates

only 2 correct edges in generation 1 but 5 in generation 2. The number of correct edges are 5 for *LFDA* (equation (34)). Both runs find the optimum.

This simple example shows the fundamental problem of the theoretical analysis of learning algorithms in EDA algorithms. *Many factorizations might give distributions which generate the global optima with sufficient probability*. The approximation of the exact distribution might be very bad.

Table 2 shows numerical results for problem sizes of $n = 49$ or $n = 50$. Note that *LFDA* and *BOA* compute the network new in each generation, so the table shows a typical networks, usually after about the first five generations. The results confirm our conjecture, that with large population sizes the number of missed edges from the interaction graph gets very small.

Both *LFDA* and *BOA* use a greedy heuristic which adds a single arc at each step. The learning algorithm does not penalize $RIP$ violations, so there are lots of $RIP$ violations. We show the results for two structure penalty factors $\alpha$, namely the standard $\alpha = 0.5$ indicated in the table by (1) and the small penalty $\alpha = 0.1$ indicated by (2). *BOA* is run with the standard setting.

For $f_{Trap:5}$ the number of parents is restricted to $k = 4$. In order to find the optimum about $80\% - 90\%$ of the edges have to be correctly identified. The factorizations computed with $\alpha = 0.5$ and $\alpha = 0.1$ are very different. We show a typical part of both factorizations.

$$\begin{aligned} p(\mathbf{x}) &= (5)(6|5,7,8)(7|5)(8|5,7,9)(9|5)\cdots \\ p(\mathbf{x}) &= (0|2,3,35,45)(1|0,2,3,44)\cdots \end{aligned}$$

The standard penalty factor $\alpha = 0.5$ gives a sparse network, where 80 edges of a total of 87 are correct. Only the highest order dependencies are often missing. In contrast, the small weight $\alpha = 0.1$ gives a dense network where *every variable has the maximum number of allowed parents*, namely 4. The same is true for *BOA*. Shown is also a run with only 2 allowed parents per variable. With $\alpha = 0.1$ the optimum is found with high probability. This is a numerical example showing that often the high order dependencies are not necessary for finding the optimum!

| problem | pop. | $\alpha$ | $k$ | (c/t/m) edges | RIP viol. | best |
|---|---|---|---|---|---|---|
| $f_{Trap:5}$ | 1000 | 0.1 | 4 | 20/45/80 | 7 | no |
| | 4000 | 0.1 | 4 | 80/87/20 | 0 | yes |
| | 2500 | 0.5 | 4 | 93/182/7 | 40 | yes |
| (*BOA*) | 3000 | | 4 | 94/190/6 | 44 | yes |
| | 4000 | 0.5 | 2 | 70/95/30 | 17 | yes |
| $F_{Iso}$ | 1000 | 0.1 | 2 | 54/92/14 | 29 | no |
| (*BOA*) | 1500 | | 2 | 52/95/16 | ? | yes |
| | 3000 | 0.1 | 2 | 63/78/9 | 6 | yes |
| | 2000 | 0.5 | 2 | 60/94/12 | 24 | yes |
| $Ising$ | 500 | 0.1 | 4 | 32/59/52 | 7 | yes |
| | 500 | 0.5 | 4 | 15/167/69 | 43 | no |
| | 4000 | 0.1 | 4 | 56/73/28 | 16 | yes |
| | 4000 | 0.5 | 4 | 61/174/23 | 45 | yes |
| $n:3$ | 1000 | 0.1 | 4 | 43/63/75 | 16 | no |
| | 1000 | 0.5 | 4 | 57/179/61 | 44 | yes |
| | 10000 | 0.1 | 4 | 88/95/30 | 23 | yes |
| | 10000 | 0.5 | 4 | 100/177/18 | 77 | yes |

Table 2: Typical *LFDA* network: $\alpha$ penalty, $k$ number of parents, graph:(correct(c)/total(t)/missed(m) edges), RIP violations, best: optimum found;

For the $7 * 7$ Ising problem, *LFDA* with an approximate factorization finds the optimum with a population size of $500$. In this example only 32 edges of the Bayesian network are contained in $G_{ADF}$, 52 are missing. Here the run with $\alpha = 0.1$ does not find the optimum. For Kauffman's $n : 3$ function the situation is different. Here the run with $\alpha = 0.1$ finds the optimum with $N = 1000$, but the standard setting not. Only $50\%$ of the edges are contained in $G_{ADF}$.

So far the analysis of *BOA* concentrated on the scaling of the number of function evaluations only [22]. The results of *BOA* are comparable to running *LFDA* with $\alpha = 0.1$. Recently an analysis of the networks computed by the successor program *hBOA* appeared in [7]. *hBOA* generates more

sparse networks than *BOA*. The results of *hBOA* are more similar to using $\alpha = 0.5$. For the $Ising$ problem results similar to *LFDA* are reported. *hBOA* detects only $60\%$ edges of $G_{ADF}$, nevertheless the algorithm converges.

## 5 Factorization of Factor Graphs

Acyclic Bayesian networks are an ideal representation for sampling. But the exact factorization might lead to cliques of size $O(n)$. Furthermore, the computation of the best approximation by an learning algorithm is in $NP$. Both problems are solved by the new factorization published in [1]. It is based on the concept of *factor graphs*. I believe that this proposal is of great interest to EDA researchers, therefore I summarize the results here.

**Definition 8** *A factor graph $\mathcal{FG}$ is a graph with two kinds of vertices, the set of factors $\{F_j\}_{j=1}^m$ with scopes $\{S_j\}_{j=1}^m$, and the set of random variables $\mathcal{X} = \{X_1, \ldots, X_n\}$. Each variable is connected to those factors where it is contained in the scope. The Gibbs distribution of $\mathcal{FG}$ is defined as*

$$P(X) = \frac{1}{Z} \prod_{j=1}^{m} F_j(S_j) \tag{35}$$

Factor graphs are the natural graphical representation for ADFs. The Gibbs distribution is an extension of the Boltzmann distribution.

**Example 4: 2-D Grid**

$$\begin{aligned} f(x) \quad = \quad & f_{1,2}(x_1, x_2) + f_{2,3}(x_2, x_3) + f_{1,4}(x_1, x_4) + f_{2,5}(x_2, x_5) \\ & + f_{4,5}(x_4, x_5) + f_{3,6}(x_3, x_6) + f_{5,6}(x_5, x_6) \end{aligned}$$

For this function the Boltzmann distribution is given by

$$p_\beta(\mathbf{x}) = \frac{1}{Z} exp(\beta \sum_i f_i(,)) = \frac{1}{Z} \prod_i exp(\beta f_i(,)) \tag{36}$$

$F_i(,) = exp(\beta f_i(,))$ are the factors of the Gibbs distribution. The computation of the partition function $Z$ is exponential in $n$, therefore a different factorization is needed.

**Definition 9 (Markov Blanket)** *Let a set of scopes be given. The Markov blanket of a set of variables $D \subseteq \mathcal{X}$ is defined as*

$$MB(D) = \bigcup \{S_j : S_j \in S, S_j \cap D \neq \emptyset\} \tag{37}$$

Thus, the Markov blanket is the minimal set of variables that separates $D$ from the other variables in the factor graph.

For the factorization theorem additional definitions are needed.

**Definition 10** *Let $\bar{\mathbf{x}} = (\bar{x_1}, \ldots, \bar{x_n})$ be an assignment. Let any subset of variables $D = \{x_{i_1}, \ldots, x_{i_{|D|}}\}$ be given. Let $U \subseteq D$. Then*

$$(\sigma_U(d))_i = \begin{cases} x_i & \text{if } x_i \in U \\ \bar{x_i} & \text{if } x_i \ni U \end{cases}$$

*The canonical factor $g^*$ is defined as follows*

$$g_D^*(d) = exp(\sum_{U \subseteq D} (-1)^{|D-U|} \log p(\sigma_U(d)) \tag{38}$$

The following theorem extends the Hammersley-Clifford theorem (which applies to Markov networks) to factor graphs.

**Theorem 4 (Hammersley-Clifford)** *Let $p$ be a positive Gibbs distribution with factor scopes $\{S_j\}_{j=1}^{m}$. Let $\{S_j^*\}_{j=1}^{m^*} = \bigcup_{j=1}^{m^*} 2^{S_j} - \emptyset$. Then*

$$p(\mathbf{x}) = p(\bar{\mathbf{x}}) \prod_{j=1}^{m^*} g_{S_j}^*(s_j^*) \tag{39}$$

The partition function $Z$ has vanished, but the computation of $g_{S_j}^*$ is now not local. By a rearrangement of the terms and introducing conditional probabilities the following factorization can be obtained.

**Proposition 2 (Factorization of Factor Graphs)** *Let $p$ be a positive Gibbs distribution with factor scopes $\{S_j\}_{j=1}^{m}$. Let $\{S_j^*\}_{j=1}^{m^*} = \bigcup_{j=1}^{m^*} 2^{S_j} - \emptyset$. Then*

$$p(\mathbf{x}) = p(\bar{\mathbf{x}}) \prod_{j=1}^{m^*} g_{S_j^*|MB(S_j^*)}^*(s_j^*) \tag{40}$$

*where*

$$g_{D|Y}^*(d) = exp(\sum_{U \subseteq D} (-1)^{|D-U|} \log p(\sigma_U(d)|Y = \bar{\mathbf{y}}) \tag{41}$$

The detailed proof can be found in [1]. The theorem is valid for any factor graph. No additional assumptions are needed, like the $RIP$ for the *FDA* factorization theorem.

The factor graph can also be learned from data. Abbeel et al. [1] propose the following simple *Factor-Graph-Learn-Algorithm*. It is based on conditional entropy, which was also used for learning of Bayesian networks.

**Proposition 3 (Cover and Thomas [3])** *Let $p$ be a probability distribution over $X, Y, Z$. Then*

$$H(X|Y,Z) \leq H(X|Y) \tag{42}$$

The *Factor-Graph-Learn-Algorithm* is a simple enumeration. The algorithm receives as input the sample; $k$ the maximum number of variables per factor; $b$ the maximum number of variables per Markov blanket; and a base instantiation $\bar{\mathbf{x}}$. We define the sets

$$S = \{S_j^* : S_j^* \subseteq X, S_j^* \neq \emptyset, |S_j^*| \leq k\} \tag{43}$$

$$Y = \{Y : Y \subseteq X, |Y| \leq b\} \tag{44}$$

Proposition 3 shows that conditional entropy can be used to find the Markov blanket for any set of variables because for any $Y \subseteq X$ with $S_j^* \cap Y = \emptyset$

$$H(S_j^*|MB(S_j^*)) \leq H(S_j^*|Y) \tag{45}$$

Thus one can select the set of variables $Y$ that minimize $H(S_j^*|Y)$ as the Markov blanket for $S_j^*$. Now we can formulate the algorithm, which is just an enumeration of the candidate sets.

$$\mathbf{LEARN - FG(k, b)}$$

- **STEP 1:** For all $S_j^*$ find $MB(S_j^*)$.
- **STEP 2:** For all $S_j^*$ compute the estimates $g_{S_j^*|MB(S_j^*)}^*((s_j^*)$
- **STEP 3:** Threshold to one the factors satisfying $|logg_{S_j^*|MB(S_j^*)}^*(s_j^*) \leq \frac{\epsilon}{2^{k+2}}$
- **STEP 4:** Discard the factors that have all entries equal to one.

The algorithm uses the thresholding techniques to generate more sparse networks. A measure similar to the $BIC$ measure could be used to bound the complexity of the structure in a theoretical better founded manner.

The factor graph representation seems to be the ideal representation for EDA algorithms. But there is of course a problem. Sampling using this factorization is difficult. It requires Gibbs sampling. For difficult distributions Gibbs sampling might need an exponential computational effort!

## 6 An adaptive annealing schedule for the Boltzmann distribution

Usually *FDA* and *LFDA* are run with truncation selection of $\tau = 0.3$. But the convergence theory is based on a Boltzmann distribution and Boltzmann selection. The Boltzmann selection needs an annealing schedule. We have derived an *adaptive* annealing schedule *SDS* which is almost as fast as truncation selection, but is more robust than truncation selection. The theoretical derivation can be found in [13].

**Definition 11** *The **average fitness** of a fitness function and a distribution is*

$$W_f(p) = \sum_x f(\mathbf{x})p(\mathbf{x}) \tag{46}$$

*The variance is defined as*

$$\sigma_f^2(\beta) = \sum_x \big[f(x) - W_f(\beta)\big]^2 p_\beta(x) \tag{47}$$

*The response to selection is defined as*

$$R_f(t) = W_f\big(\beta(t+1)\big) - W_f\big(\beta(t)\big) \tag{48}$$

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

The *response to selection* $R_f(t)$ can be computed exactly by using the centered moments $M_i^c$ [13].

$$M_i^c(\beta) := \sum_x \big[f(x) - W_f(\beta)\big]^i p(x) \tag{49}$$

**Definition 12** *The SDS schedule is given by*

$$\Delta\beta(t) = c \,/\, \sqrt{\sigma_f^2(\beta(t))} \tag{50}$$

The schedule is proportional to the inverse of the square root of the variance.

**Theorem 5** *Let $\sigma(t) := \sqrt{\sigma_f^2(\beta(t))}$ be the standard deviation. Then the response to selection for Boltzmann selection with the SDS schedule is given by*

$$R_f(t) = c \cdot \sigma(t) + \frac{c^2 M_3^c}{2\,\sigma(t)^2} + \frac{c^3 M_4^c}{6\,\sigma(t)^3} + \ldots \tag{51}$$

Thus the response to selection is approximately $o(\sqrt{\sigma_f^2(\beta(t))})$, the same is true for truncation selection [13]. For the optimization of the function $Iso$ Boltzmann selection is essential, truncation selection will not generate the optimum for very large $n$.

## 7 Convergence of *FDA* with Finite Samples

The convergence theorems are valid for infinite populations only. In this section we investigate convergence for finite samples. *FDA* is a probabilistic algorithm. Obviously for finite populations convergence to the optimum cannot be guaranteed. Therefore convergence can only be probabilistic. In this section we will use the $(\epsilon, \delta)$ convergence concept first applied in $PAC$ learning [11]. It is defined as follows:

**Definition 13** *Let $\epsilon > 0$,$\delta > 0$. Let $P$ be the true distribution, $Q$ an approximation. Then we speak of $(\epsilon, \delta)$ convergence if*

$$prob\,(error(P, Q) > \epsilon) < \delta \tag{52}$$

$error$ denotes any distance measure.

To get a feeling for sample complexity bounds, consider the following problem: suppose we have a coin whose heads probability $p$ we wish to determine. Letting 0 denote tails and 1 denote heads we obtain a sample $S_N =< x_1, \ldots, x_N >$. The maximum likelihood approximation gives $\hat{p} = 1/N \sum_i \mathbf{x}_i$. The question then arises: how large must $N$ be for $|\hat{p} - p| < \epsilon$ with probability at least $1 - \delta$. An application of Chernoff-type bounds [11] shows that $N = \frac{1}{2\epsilon^2} ln \frac{2}{\delta}$ will suffice. This *upper bound* is distribution free and makes no assumption about the value of $p$.

In this case a lower bound can also be obtained. We set $p = 0.5$. By bounding the relevant binomial coefficients, we obtain inequalities for the probability that the estimate is more than $\epsilon$ away from $p$. One obtains $N \geq \frac{1}{5\epsilon^2} ln \frac{2}{\delta}$. Thus in this simple example the upper and lower bound differ only by a small multiplicative constant.

### 7.1 Sample size for a given network structure

Let an acyclic Bayesian network be given. Let $S_N =< \mathbf{x}_1, \ldots, \mathbf{x}_N >$ where $\mathbf{x}_i = (x_1^{(i)}, \ldots, x_n^{(j)})$ be the empirical sample. It is easy to show that the the maximum likelihood approximation of true probabilities are the long-run frequencies of the sample. In this section we derive upper bounds on the necessary sample size. Later we will compare these with numerical results. In the following analysis we will use the Kullback-Leibler divergence as error measure.

There exist a number of papers deriving bounds on the sample size. We just mention [4]. In this paper we present a simple proof, based on the factorization theorem.

**Theorem 6** *Let f be a separable ADF function with m sub-functions of k variables. Let* $0 < \epsilon < 0.5$,$0 < \delta < 0.5$. *Let* $p$ *be the true Boltzmann distribution,* $q$ *the approximation by finite samples. Then in order that*

$$prob\,(KLD(p,q) < \epsilon) > 1 - \delta \tag{53}$$

*the sample size should be*

$$N \geq O(\frac{\ln 2 * m}{\epsilon}(2^k + log_2(\frac{m}{\delta}))) \tag{54}$$

**Proof:** Let $p = \prod p(\mathbf{x}_{s_i})$ be the exact distribution and $q$ the numerical approximation using the same factorization. We first consider a single sub-function $i$, and $p_i = p(\mathbf{x}_{s_i})$, $q_i = q(\mathbf{x}_{s_i})$, the restrictions of the distribution to sub-function $i$. Let $0 < \tilde{\epsilon} < 0.5$, $0 < \tilde{\delta} < 0.5$ then the standard theorem from PAC learning gives [11]

$$prob\left(KLD(p_i, q_i) < \tilde{\epsilon})\right) > 1 - \tilde{\delta} \tag{55}$$

for

$$N \geq O\left(\frac{1}{\tilde{\epsilon}}(ln(|H|) + ln(\frac{1}{\tilde{\delta}}))\right) \tag{56}$$

$|H|$ is the size of the space of hypotheses which is in our case the space of binary functions with $k$ variables. We have $|H| = 2^{2^k}$. Using the decomposition we obtain

$$prob\left(KLD(p,q) = \sum_{i=1}^{m} KLD(p_i, q_i) < m * \tilde{\epsilon})\right) > (1 - \tilde{\delta})^m$$

Because $(1 - \tilde{\delta})^m \leq e^{-m\tilde{\delta}}$ we can substitute the latter term. We now set $\tilde{\epsilon} = \frac{\epsilon}{m}$ and $\tilde{\delta} = \frac{\delta}{m}$ in equation 56 and obtain the upper bound. **qed**

The extension of this proof to general *FDA* factorizations with conditional marginals is technical. It is omitted because in [1] a sharper bound has been proven.

**Theorem 7** *Let the assumptions of the factorization theorem be fulfilled. Let* $0 < \epsilon$,$0 < \delta$ *be given. Let* $q$ *denote the probability distribution obtained by fitting the conditional probability tables via maximum likelihood and then clipping each entry to the interval* $[\frac{\epsilon}{4}, 1 - \frac{\epsilon}{4}]$ *Then in order that*

$$prob\,(KLD(p,q) < \epsilon) > 1 - \delta \tag{57}$$

*the sample size should be* $O(n)$

The bound $O(n)$ is astonishing. In [4] an upper bound of $N = O(n \ln n)$ has been proven, similar to our bound. The $O(n)$ bound might be a result of restricting the conditionals to the interval $[\ \frac{\epsilon}{4}, 1 - \frac{\epsilon}{4}]$.

The application of these theorems to $FDA$ needs some additional thoughts. The $(\epsilon, \delta)$ estimates bound the error for the approximation of just a single distribution. In *FDA* a sequence of distributions have to be estimated. *FDA* consists of the following iterative scheme

1. Start with a uniform distribution
2. Perform Boltzmann selection with parameter $\Delta\beta(t)$
3. Compute the marginals of the *FDA* factorization
4. Use the *FDA* factorization to sample the distribution
5. If $termination - condition$ not fulfilled, continue with Step2.

But $\epsilon$ can be made so small, that the errors of all estimates during the iteration will be very small.

**Summary:** The necessary sample size for the convergence of *FDA* using a factorization fulfilling the *RIP* is bounded by $O(n \ln n)$ or maybe $O(n)$. The time complexity per generation is $O(N * n)$. This gives a total time complexity of $O(n^2 \ln n)$ per generation.

### 7.2 Sample complexity for learning of the structure

If the structure of the ADF function is not known, it has to be inferred from examples. There are different techniques to infer the structure. We will discuss the two most important ones.

1. Compute the ADF structure by computing the Walsh coefficients. Then use *FDA* for optimization
2. Compute the statistical dependencies of the variables and use this information to learn a Bayesian network.

The second approach is used by *LFDA* and *BOA*. The first approach computes the structure of the function. This method has been proposed by Wright et al. [27]. Earlier work for separable functions has been reported by Streeter [25]. It is then the task of the *FDA* factorization algorithm to compute a good factorization. There is another major difference. In the first approach, the structure is computed only once, whereas in the second method the Bayesian network is learned after each selection step.

### 7.3 Learning the *ADF* structure symbolically

The interested reader is referred to [27].

**Theorem 8 (Wright[27])** *Assume a class of ADF functions where each sub-function has at most $k$ variables. Let $\delta > 0$ be a constant. Then the number of function evaluations required by the DETECT-LINKAGE algorithm of order 2 to detect the scope of all sub-functions with probability at least $\delta$ is bounded by*

$$N = O(2^k n^2 ln(n)) \tag{58}$$

For *separable ADF* functions the bound can be made smaller.

**Theorem 9 (Streeter[25])** *Let f be an order-k separable ADF. Then the number of function evaluations of algorithm ASFOPTIMIZE to return the globally optimal string with probability at least $\delta$ can be bounded by*

$$N \geq 2^k n \ln n \tag{59}$$

Algorithm ASFOPTIMIZE detects the structure *and* computes the optimum on the fly. For separable functions an upper bound of $N = O(m \ln(m))$ has also been reported for the extended compact genetic algorithm $ECGA$ in [6]. But the error measure used is weaker. It is assumed that the probability of a sub-function being not correct is $(O(1/m)$. This means that only $m - 1$ sub-functions are correct, i.e the global optimum is *not* reached on the average.

**Summary:** The sample size for learning the structure of the ADF is upper bounded by $O(n^2 \ln n)$. For separable functions an upper bound is $O(n \ln n)$.

### 7.4 Learning Bayesian networks with statistical techniques

We have already presented the learning algorithm used by *LFDA*. The theoretical analysis has to deal with two problems:

**P1:** Estimate the necessary sample size so that the *best* network in the class of networks considered fulfills the $(\epsilon, \delta)$ convergence.
**P2:** Estimate the error introduced by the learning algorithm used.

Problem P1 has been solved, but the computation of the best Bayesian network is $NP - hard$. This means that problem P2 cannot be solved in the class of Bayesian networks. Thus for Bayesian networks we can only review important results for problem P1. Theoretical progress of problem P2 can only be obtained if a different class of graphical models is considered. We will report recent results using factor graphs at the end of the section.

In the literature a number of upper bounds have been calculated. We only cite one of the first papers [8].

**Theorem 10** *Let the unknown Bayesian network have bounded in-degree $k$. Let $\epsilon, \delta > 0$ be given. Let $p_{best}$ be the best Bayesian network with at most $k$ parents, i.e. the one with minimum Kullback-Leibler divergence KLD. Then for*

$$prob\,(KLD(p, p_{best}) < \epsilon) < 1 - \delta$$

*to hold, it suffices that*

$$N \geq O(n \ln(n) 2^k (\frac{1}{\epsilon} \ln \epsilon)^2 \ln \frac{1}{\delta}) \tag{60}$$

The bound of $N = O(n \ln n)$ should not be a surprise, because the learned network is assumed to be the best in its class. For the *FDA* factorization we have proven the same upper bound. Thus the computational complexity is dominated by the learning algorithm, not the sample size! The computational complexity of the known learning algorithms is at least $O(N * n^2)$, so the researchers concentrate on making the constants smaller. A recent numerical evaluation of the efficiency of popular learning algorithms can be found in [26].

**Remark:** The published scaling results of *BOA* and *hBOA* report mainly the number of function evaluations. This is misleading because this analysis leaves out the most important factor of the computational complexity, namely the learning of the network.

In EDA algorithms there is an option to reduce the sample size $N$ by *local hill climbing* algorithms. This was introduced in [17]. *LFDA* can be run with a sophisticated hill climber. A special method has been implemented to solve large graph bi-partitioning problems with up to 1000 nodes. The candidate edges for the learning algorithm are restricted to the graph connections. With suitable hashing techniques the implementation has a bound for the time complexity of $O(N * n)$ [17]. Hill-climbing is also intensively used by Pelikan [23].

**Summary:** The sample size for learning the structure of the *ADF* by statistical methods is upper bounded by $O(n \ln n)$. Computing the structure has a time complexity of at least $O(N * n^2)$ per generation. This gives a total time complexity per generation of at least $O(n^3 \ln(n)$.

### 7.5 Learning factor graphs

For factor graphs bounds can be computed, both for the sample size and learning the structure. We just cite the theorems.

**Theorem 11 (Computational Complexity [1])** *Let $k$ be the maximum number of variables per factor; let $b$ be the maximum number of variables per Markov blanket. Then the running time of* $\mathrm{LEARN-FG}$ *is $O(N * k * b * n^{k+b})$.*

**Theorem 12 (Sample Complexity [1])** *Let $\tilde{p}$ be the distribution computed by* $\mathrm{LEARN-FG}$. *Then for*

$$prob\,(KLD(p, \tilde{p}) < \epsilon) > 1 - \delta$$

*to hold, it suffices that*

$$N \geq O(n \ln(n)) \tag{61}$$

The computational complexity is dominated by the learning algorithm, the sample complexity is as before $O(n \ln n)$. Bayesian networks are obviously special cases of factor graphs. If the number of parents per variable is bounded by $p_k$ and the number of children bounded by $c$ we have $k \leq p_k + 1$ and $b \leq (c+1)k^2$. Thus the learning complexity is upper bounded by a polynomial in $n$.

In summary, the total complexity is polynomial in $n$ for all factor graphs with factors of bounded size and bounded Markov blankets. The exact determination of the maximum blanket size might be a problem. Normally $b = O(k^2)$ is enough. For grid problems we have $b = O(k)$.

For practical algorithms the polynomial bound is too high. The bound can be reduced by using a learning scheme not based on enumeration.

## 8 Numerical Results for *FDA*

Using *FDA* we are interested to find the global optimum. The theoretical results show that for $(\epsilon, \delta)$ convergence for functions fulfilling the assumptions of the factorization theorem with bounded clique size $k$ a sample size scaling with at most $O(n \ln n)$ is sufficient to obtain convergence.

We test the theoretical result numerically. The computation of the $KLD$ metric is expensive, so we use a different criterion, the probability of finding the optimum. This criterion is the interesting one for optimization tasks.

$$prob(\textit{not finding the optimum}) < \delta \tag{62}$$

| problem | $n$ | $N$ | $gen.$ | $FE$ | $\delta$ | $time/gen.(t)$ |
|---|---|---|---|---|---|---|
| $f_{Trap:5}$ | 50 | 325 | 8.9 | 3200 | 0.05 | 7.7(t) |
| | 100 | 500 | 15.1 | 8070 | 0.04 | 17.4(t) |
| | 200 | 800 | 24.3 | 20200 | 0.03 | 56.9 (t) |
| | 400 | 1200 | 37.2 | 45750 | 0.04 | 195.8(t) |
| (t) | 200 | 800 | 15.3 | 13000 | 0.05 | |
| $f_{Iso}$ | 49 | 440 | 9.1 | 4430 | 0.07 | 4.1 |
| | 99 | 940 | 15.1 | 15100 | 0.07 | 15.8 |
| | 201 | 1900 | 24.3 | 47400 | 0.07 | 67.9 |
| | 401 | 3800 | 36.2 | 138000 | 0.05 | 380.1 |
| | 201(t) | 90000 | 12.8 | 1350000 | 0.60 | 3086.1(t) |
| $2-D\ Ising$ | 49 | 500 | 10.4 | 5670 | 0.05 | 9.5(t) |
| | 100 | 1300 | 22.4 | 34600 | 0.06 | 42.7(t) |
| (1) | 225 | 3500 | 35.6 | 127000 | 0.06 | 300.7(t) |
| (2) | 225 | 15000 | 32.1 | 496000 | 0.06 | 1327.3(t) |
| (t) | 400 | 12000 | 29.3 | 371000 | 0.60 | 2666.2(t) |
| $n:3$ | 25 | 500 | 5.6 | 3300 | 0.05 | 6.4(t) |
| (1) | 50 | 900 | 10.9 | 10600 | 0.01 | 23.3(t) |
| (2) | 50 | 2000 | 12.1 | 26300 | 0.07 | 48.5(t) |
| (1) | 100 | 3500 | 20.1 | 75500 | 0.01 | 204.1(t) |
| (2) | 100 | 9000 | 21.0 | 219800 | 0.05 | 522.7(t) |
| | 200 | 12000 | 35.2 | 1086000 | 0.07 | 1700.4(t) |

Table 3: Numerically determined sample size bound; the Ising factorization violates the $RIP$; for the Ising and the Kauffman problem the results for two different instances, denoted by (1) and (2) are displayed. (t) denotes truncation selection

For F1 and F2 *FDA* uses the exact factorization, for F3 and F4 approximate factorizations. Thus the convergence theorem is valid only for F1 and F2. The *FDA* factorization of the $2-\mathrm{D\ Ising}$ problem uses cliques of size 5. For Kauffman's $n:3$ *FDA* computes a factorization by using the sub-function merge algorithm with a clique size of at most 6. The performance criterion is the percentage of finding the optimum in 100 or 1000 runs, depending on the size of the problem. The results are displayed in table 3.

The sample size scales less than linear for the decomposable function $f_{Trap:5}$. The number of function evaluations scales about $O(m \ln m)$ where $m$ is the number of sub-functions. This seems to be the best possible scaling for a random heuristic [25].

For the function $Iso$ the sample size scales about $O(n)$. Thus the upper bound given in theorem 7 is sharp. For this function the *SDS* selection method is essential. For all other problems truncation selection is numerically more efficient.

The scaling of the 2-D Ising problem and the Kauffman function is erratic, for higher problem sizes the necessary sample size depends on the problem instance. So for problem instance 1 of the 2-D Ising problem of $n = 225$ we need a sample size of only $N = 3500$, whereas problem instance 2 needs $N = 15000$. But it is surprising that *FDA* up to $n = 400$ finds the optimum in 4 out of 6 runs.

In order to estimate the total time complexity we have given in table 3 the results for truncation selection (t) also. A good predictor is the *time complexity per generation*. The results can be fitted by $time_{compl} = c * N * m * k_{max}$. Here $k_{max}$ is the number of parents or $k_{max} = cliquesize - 1$.

## 9 Numerical results for *LFDA*

The *SDS* selection method does not work efficiently with *LFDA*, it selects too weak at the beginning, therefore the learned network is far away from the $G_{ADF}$ network. Thus we use *LFDA* with truncation selection with standard parameter $\tau = 0.3$. For *BOA* only 10 runs have been made.

| problem | $n$ | $N$ | $gen.$ | $FE$ | $\delta$ |
|---|---|---|---|---|---|
| $f_{Trap:5}$ | 15 | 800 | 3.0 | 3200 | 0.04 |
| | 30 | 1800 | 4.3 | 9500 | 0.04 |
| | 50 | 3500 | 6.7 | 26400 | 0.06 |
| | 60 | 4800 | 7.6 | 40200 | 0.03 |
| | 120 | 10000 | 11.9 | 118000 | 0.04 |
| *BOA* | 120 | 6000 | 24 | 78000 | |
| *BOA* | 240 | 14000 | 34 | 252000 | |
| $f_{Iso}$ | 15 | 450 | 2.5 | 1570 | 0.06 |
| | 29 | 1100 | 4.1 | 4700 | 0.06 |
| | 49 | 2200 | 5.9 | 15000 | 0.06 |
| | 59 | 2500 | 7.1 | 19400 | 0.05 |
| | 121 | 5000 | 11.4 | 62800 | 0.09 |
| *BOA* | 121 | 7500 | 20 | 82500 | 0.00 |
| *BOA* | 201 | 15000 | 28 | 240000 | 0.00 |
| *BOA* | 401 | 60000 | 41 | 1260000 | 1.00 |
| $/mathrm{2-D\ Ising}$ | 25 | 700 | 3.8 | 3300 | 0.05 |
| | 49 | 1100 | 7.5 | 9200 | 0.04 |
| | 100 | 2400 | 13.6 | 35000 | 0.04 |
| | 121 | 3000 | | | 0.05 |
| $n:3$ | 25 | 1000 | 4.7 | 5700 | 0.07 |
| | 50 | 2500 | 8.1 | 24500 | 0.05 |
| | 100 | 6000 | 15.2 | 96000 | 0.01 |
| | 121 | 10000 | 15.0 | 160000 | – |

Table 4: Numerically determined necessary sample size, truncation selection $\tau = 03$, dense network with $\alpha = 0.1$, *BOA* 10 runs only.

The sample sizes are bounded by $O(n \ln n)$, both for Trap:5 and Iso up to $n = 201$. *BOA* or *LFDA* have not been able to solve Iso of size $n = 401$ with a sample size of up to $N = 60000$. This result shows that one has to be very careful in extrapolating numerical results to large dimensions!

The sample size for $2 - \mathrm{D\ Ising}$ and $n : 3$ seems also upper bounded by $O(n \ln n)$. The computational complexity per generation of *LFDA* is bounded by $time_{compl} = O(N * n^3)$. This implementation is not optimized because many different algorithms are implemented using the same data data structures.

The results of *BOA* are comparable, for Trap:5 better and for Iso worse. This is not surprising, because both use similar algorithms. The computational complexity of BOA is $O(N * n^2)$ per generation.

Despite they have the same order of the upper bound, the actual sample sizes needed for statistical learning a Bayesian network are considerably higher than for *FDA*, especially for $\mathrm{Trap}:5$. This leads to a much larger computational effort.

## 10 Conclusion and Outlook

The family of EDA algorithms are well founded in statistical learning. Using known results from Bayesian networks and factor networks we have been able to bound the sample size needed for convergence. The numerical experiments confirm the results, both for fixed networks and algorithms which learn the networks.

The results of the learning algorithms are astonishingly good. We have shown that for convergence the learned network should normally contain a substantial fraction of the edges of the interaction graph $G_{ADF}$. An exception is the regular 2-D grid used by the Ising function. Here only $60\%$ correctly identified edges are enough for finding the optimum. The major problem of the learning algorithms is their computational complexity. It is at least $O(N * n^2 \ln n)$.

Our results show that a promising new development is to combine *FDA* with a program which computes the graph $G_{ADF}$ symbolically. A good candidate is the algorithm of Wright et al. [27]. The new representation based on factor graphs should also be explored. For EDA applications sampling of the factor graph distribution has to be improved.

The interested reader can download our free software from the WWW site
http://www.ais.fraunhofer.de/∼muehlen/.

## References


[1] P. Abbeel, D. Koller, and A.Y. Ng. Learning factor graphs in polynomial time & sample complexity. *Journ. Machine Learning Research*, 7:1743–1780, Aug 2006.

[2] F. Barahona. On the computational complexity of the ising spin glass models. *J. Phys. A: Math. Gen.*, 15:3241–3253, 1982.

[3] Th. M. Cover and J.A. Thomas. *Elements of Information Theory*. Wiley, New York, 1989.

[4] S. Dasgupta. The sample complexity of learning fixed structure bayesian networks. *Machine Learning*, 29(2-3):165–180, 1997.

[5] Y. Gao and J. Culberson. Space complexity of estimation of distribution algorithms. *Evolutionary Computation*, 13(1):125–143, 2005.

[6] G.R. Harik, F.G. Lobo, and Sastry K. Linkage learning via probabilistc modeling in the extended compact genetic algorithm (ecga). In M. Pelikan, K. Sastry, and E. Cantu-Paz, editors, *Scalable Optimization via Probabilistic Modeling*, Studies in Computational Inteligence, pages 39–61. Springer, Berlin, 2006.

[7] M. Hauschild, M. Pelikan, C.F. Lima, and Kumara Sastry. Analyzing probabilistic models in hierarchical boa on trap and spin glasses. Technical Report 2007001, University of Missouri-MEDAL, January 2007.

[8] K.U. Höffgen. Learning and robust learning of produkt distributions. In *Proceedings of the 6th Annual Workshop COLT*, pages 77–83, Santa Cruz, 1993.

[9] F. V. Jensen and F. Jensen. Optimal junction trees. In *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*, pages 360–366, Seattle, 1994.

[10] M. I. Jordan, editor. *Learning in Graphical Models*. MIT Press, Cambrigde, 1999.

[11] M. Kearns and U.V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, Cambridge, 1994.

[12] S. L. Lauritzen. *Graphical Models*. Clarendon Press, Oxford, 1996.

[13] Th. Mahnig and H. Mühlenbein. A new adapive boltzmann selection schedule sds. In *Proceedings Congress on Evolutionary Computation 2001*, pages 121–128. IEEE, 2001.

[14] H. Mühlenbein and R. Höns. The estimation of distributions and the minimum relative entropy principle. *Evolutionary Computation*, 13(1):1–27, 2005.

[15] H. Mühlenbein and R. Höns. The factorized distribution algorithm and the minimum relative entropy principle. In M. Pelikan, K. Sastry, and E. Cantu-Paz, editors, *Scalable Optimization via Probabilistic Modeling*, Studies in Computational Inteligence, pages 11–37. Springer, Berlin, 2006.

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

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

[18] H. Mühlenbein and Th. Mahnig. Mathematical analysis of evolutionary algorithms. In C. C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, Operations Research/Computer Science Interface Series, pages 525–556. Kluwer Academic Publisher, Norwell, 2002.

[19] H. Mühlenbein and Th. Mahnig. Evolutionary algorithms and the Boltzmann distribution. In K. De Jong, R. Poli, and J. C. Rowe, editors, *Foundations of Genetic Algorithms 7*, pages 525–556. Morgan Kaufmann Publishers, San Francisco, 2003.

[20] H. Mühlenbein, Th. Mahnig, and A. Ochoa. Schemata, distributions and graphical models in evolutionary optimization. *Journal of Heuristics*, 5(2):213–247, 1999.

[21] H. Mühlenbein and G. Paaß. From recombination of genes to the estimation of distributions I. binary parameters. In H.-M Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, *Lecture Notes in Computer Science 1141: Parallel Problem Solving from Nature - PPSN IV)*, pages 178–187, Berlin, 1996. Springer-Verlag.

[22] M. Pelikan and D.E. Goldberg. Hierarchical bayesian optimization algorithm. In M. Pelikan, K. Sastry, and E. Cantu-Paz, editors, *Scalable Optimization via Probabilistic Modeling*, Studies in Computational Inteligence, pages 63–90. Springer, Berlin, 2006.

[23] M. Pelikan and A.K. Hartmann. Searching for ground states of ising spin glasses with hierarchical boa and cluster exact approximation. In M. Pelikan, K. Sastry, and E. Cantu-Paz, editors, *Scalable Optimization via Probabilistic Modeling*, Studies in Computational Inteligence, pages 315–332. Springer, Berlin, 2006.

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

[25] M.J. Streeter. Upper bounds on the time and space complexity of optimizing additively separable functions. In Kalyanmoy Deb et al., editor, *Proceedings of GECCO-2003*, Lecture Notes in Computer Science 3103, pages 186–197. Springer Press, 2003.

[26] I. Tsamardinos, L.E. Brown, and C.F. Alifernis. The max-min hill-climbing bayesian network structure algorithm. *Machine Learning*, 65(1):31–78, Oct 2006.

[27] A.H. Wright and S.V.P.M. Sandeep Pulavarty. Estimation of distribution algorithm based on linkage discovery and factorization. In H.G. Beyer et al., editor, *Proceedings of GECCO-2005*, pages 695–703. ACM Press, 2005.

[28] Y. Xiang, S. K. M. Wong, and N. Cercone. A 'microscopic' study of minimum entropy search in learning decomposable Markov networks. *Machine Learning*, 26:65–92, 1997.