Evolutionary Optimization and the Estimation of Search Distributions with Applications to Graph Bipartitioning

H. Mühlenbein **Th. Mahnig**
RWCP[1] Theoretical Foundation GMD[2] Laboratory
D-53754 Sankt Augustin
muehlenbein@gmd.de

Abstract
We present a theory of population based optimization methods using approximations of search distributions. We prove convergence of the search distribution to the global optima for the Factorized Distribution Algorithm *FDA* if the search distribution is a Boltzmann distribution and the size of the population is large enough. Convergence is defined in a strong sense–the global optima are attractors of a dynamical system describing mathematically the algorithm. We investigate an adaptive annealing schedule and show its similarity to truncation selection. The inverse temperature β is changed inversely proportionally to the standard deviation of the population. We extend *FDA* by using a Bayesian hyper parameter. The hyper parameter is related to mutation in evolutionary algorithms. We derive an upper bound on the hyper parameter to ensure that *FDA* still generates the optima with high probability. We discuss the relation of the *FDA* approach to methods used in statistical physics to approximate a Boltzmann distribution and to belief propagation in probabilistic reasoning. In the last part, we apply the algorithm to an important practical problem, the bipartioning of large graphs. We assume that the graphs are sparsely connected. Our empirical results are as good or even better than any other method used for this problem.

Keywords
genetic algorithms, linkage equilibrium, factorization of distributions, Boltzmann distribution, adaptive annealing, Kullback-Leibler divergence, advanced mean field methods, stochastic processes

1 Introduction

In this paper we analyze evolutionary algorithms from the perspective of stochastic processes. There are at least three views on stochastic processes– the *microscopic view*, the *mesoscopic view*, and the *macroscopic view*.

In the microscopic view the dynamic behaviour of a population of objects is simulated. In genetic algorithms, for instance, a set of points is generated. From this set promising points (points with high fitness) are selected. These points are used as the "parents" of the next set. Each run is unique, therefore a mathematical analysis is almost impossible.

[1]Real World Computing Partnership
[2]GMD - Forschungszentrum Informationstechnik

In the mesoscopic view a probability distribution $p(\mathbf{x}, t)$ is introduced. From an initial distribution $p(\mathbf{x}, 0)$ a population (ensemble) is generated. Promising points are selected. The corresponding distribution of the selected points $p^s(\mathbf{x}, t)$ is estimated and then used to generate new points. Holland [8] tried a mesoscopic analysis of *genetic algorithms* with his *schema theory*. We will show that using marginal and conditional distributions instead of schemata makes the analysis easier and tractable.

In the macroscopic view one is interested in macroscopic variables only, like the average fitness $E_t[f(x)] = \sum p(\mathbf{x}, t) f(\mathbf{x})$. In many physical applications simplified equations describing the evolution of macroscopic variables can be derived.

In this paper we concentrate on the mesoscopic view. This view has been developed for the analysis of stochastic processes [32]. Despite the fact that the microscopic system and the mesoscopic system are strongly related, the derivation of mesoscopic equations from the microscopic system is very difficult. This is the reason that the microscopic view dominates the field.

We will mainly use the terminology of dynamic stochastic systems. For our algorithms the approximation of the Boltzmann distribution by a product of conditional distributions will be of central importance. We will show the relation of this approach to methods used in probabilistic reasoning and probabilistic logic. Such a relation has been predicted by von Neumann [34]: "This new system of formal logic will move closer to another discipline which has been little linked in the past with logic. This is *thermodynamics*, primarily in the form it was received from Boltzmann, and is that part of theoretical physics which comes nearest in some of its aspects to manipulating and measuring information. Its techniques are much more analytical than combinatorial."

Thus our theory is part of a general theory starting now to unify disciplines like statistical physics, probabilistic reasoning, and probabilistic logic. This paper extends the survey of Larrañaga and Lozano [15] with an interdisciplinary perspective.

The outline of the paper is as follows. In sections 2,3, and 4 we recapitulate the foundations of evolutionary algorithms based on search distributions. In section 5 an adaptive annealing schedule for Boltzmann selection is derived. In section 6 the use of Bayesian hyper parameters is investigated. We discuss in section 7 how good our proposed algorithm approximates Boltzmann distributions. In section 8 we show that our algorithm fulfills an equation derived by Holland for an almost "optimal" search algorithm. Then we discuss algorithms from statistical physics which have been used to approximate a Boltzmann distribution. In the last section we apply our algorithm to an important combinatorial optimization problem – the bipartitioning of graphs.

2 The Simple Genetic Algorithm and $UMDA$

The theory presented is valid for discrete, but also for continuous variables. For simplicity we restrict the discussion to binary variables $x_i \in \{0, 1\}$. Let $\mathbf{x} = (x_1, \ldots, x_n)$ denote a binary vector. We use the following conventions. Capital letters X_i denote variables, small letters x_i assignments.

Definition 1. *Let a function $f : \boldsymbol{X} \to R^{\geq 0}$ be given. We consider the optimization*

problem

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

We will use $f(\mathbf{x})$ as the fitness function for the standard genetic algorithm, also called the Simple Genetic Algorithm (*SGA*). The algorithm is described by Holland [8] and Goldberg [6]. It consists of *fitness proportionate selection, recombination/crossover, mutation*. For the stochastic analysis marginal distributions will be important.

Definition 2. *Let $p(\mathbf{x}, t)$ denote the probability of $\mathbf{x}$ in the population at generation t. Let $\mathbf{z}$ denote a sub-vector of $\mathbf{x}$. Then $p(\mathbf{z}, t) = \sum_{\mathbf{x}, Z_i = z_i} p(\mathbf{x}, t)$ defines a marginal distribution. Of special importance are the univariate marginal distribution, where $\mathbf{z}$ consists of a single variable only. This is abbreviated by $p_i(x_i, t)$. Conditional distributions are defined by the Bayesian rule $p(\mathbf{y}|\mathbf{z}) = p(\mathbf{y}, \mathbf{z})/p(\mathbf{z})$. $\mathbf{y}$ and $\mathbf{z}$ are disjunct and their union is a subset of $\mathbf{x}$.*

We often write $p_i(x_i)$ if just one generation is discussed, and p_i denote $p_i(x_i = 1)$. The average fitness of the population and the variance is given by

$$E_t[f(x)] = \sum_x p(\mathbf{x}, t) f(\mathbf{x}) \tag{2}$$

$$V(t) = \sum_x p(\mathbf{x}, t) \left(f(\mathbf{x}) - \bar{f}(t) \right)^2 \tag{3}$$

Proportionate selection changes the probabilities according to

$$p^s(\mathbf{x}, t+1) = p(\mathbf{x}, t) \frac{f(\mathbf{x})}{E_t[f(x)]} \tag{4}$$

With proportionate selection the average fitness never decreases. This is true for every rational selection scheme. For the analysis of recombination we introduce a special distribution.

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

$$\pi(\mathbf{x}, t) := \prod_{i=1}^{n} p_i(x_i, t) \tag{5}$$

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

Geiringer [5] has shown that all reasonable recombination schemes lead to the same limit distribution.

Theorem 4 (Geiringer). *Recombination does not change the univariate marginal frequencies, i.e. $p_i(x_i, t+1) = p_i(x_i, t)$. The limit distribution of any complete recombination scheme is Robbins' proportions $\pi(\mathbf{x}, 0)$.*

Complete recombination means that for each subset S of $\{1, \ldots, n\}$, the probability of an exchange of genes by recombination is greater than zero. Convergence to the limit distribution is very fast. Robbins' proportions are called the *mean field assumption* [25] in physics.

If recombination is used for a number of times without selection, then the genotype frequencies converge to linkage equilibrium. This means that *all genetic algorithms are identical if after one selection step recombination is done without selection a sufficient number of times.* This procedure keeps the population in linkage equilibrium. A simpler method is used in our *univariate marginal distribution algorithm UMDA*. New search points are generated from the distribution

$$p(\mathbf{x}, t+1) = \prod_{i=1}^{n} p_i^s(x_i, t) \tag{6}$$

Algorithm 1: *UMDA*

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

For mathematical clarity we denote the average fitness, seen as a function of the independent variables p_i as $W(p_1, \ldots, p_n)$. Then the following theorem is valid [21]:

Theorem 5 (Wright's Equation). *For infinite populations and proportionate selection UMDA changes the gene frequencies as follows:*

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

The relation between $f(\mathbf{x})$ and $W(\mathbf{p})$ is simple. One has just to change x_i to p_i. Thus for $f(\mathbf{x}) = \sum_{i,j} a_{ij} x_i x_j$ we obtain $W(\mathbf{p}) = \sum_{i,j} a_{ij} p_i p_j$. A detailed discussion about Wright's equation can be found in [21, 23].

Remark: $p(\mathbf{x}, t)$ describes a *dynamical system* with 2^n variables. The dynamical system is constrained to the unit simplex because of the constraints $0 \leq p(\mathbf{x}) \leq 1$ and $\sum_x p(\mathbf{x}) = 1$. *UMDA* with proportionate selection is related to a dynamical system described by equation (7). The system is defined for discrete time steps. For mathematical clarity we give the dynamical system a different name, UNI_p. The index p indicates that the

equations are derived from *UMDA* with proportionate selection. The dynamical system has attractors.

Theorem 6. *The stable attractors of UNI_p are at the corners, i.e $p_i \in \{0,1\}$ $i = 1,\ldots,n$. In the interior there are only saddle points where grad $W(p)) = 0$. The attractors are local maxima of $f(\mathbf{x})$ according to one bit changes. Thus UNI_p solves the continuous optimization problem* $\text{argmax}\{W(\mathbf{p})\}$ *in S by gradient ascent [21].*

Since the attractors are at the boundary ($p_i \in \{0,1\}$), *UMDA* with proportionate selection will end with a population consisting of a single string $\mathbf{x}$ only, where $x_i = 0$ if $p_i = 0$ and $x_i = 1$ if $p_i = 1$. Another important result is that the average fitness never decreases [19].

Theorem 7. *For UNI_p we have $W(\mathbf{p}(t+1)) \geq W(\mathbf{p}(t))$.*

Note that the dynamical system UNI_p can be used as an optimization method by itself. One does iterate the difference equations (7) until convergence. This method is investigated in [23]. Note that the selection method of UNI_p is proportionate selection. The mathematical analysis of *UMDA* with truncation selection or tournament selection is much more difficult. These selection methods can be easily programmed for *UMDA*, but we have not been able to derive the difference equations for the corresponding dynamical system. Thus *UMDA* is the much more flexible optimization method. The interested reader is referred to [19, 23].

There exist many "convergence" theorems in genetic algorithm theory. But most of them rely on the stochastic nature of evolutionary algorithms only. Convergence is derived from the simple fact that all possible configurations are generated with probability greater than zero. This convergence definition is uninteresting from a numerical point of view. It does not specify how long it takes to converge and how convergence can be observed. In contrast, *UMDA* is a very robust numerical algorithm. It usually converges to populations where all individuals are equal. Furthermore, the average fitness increases if the size of the population is large enough.

3 Schema Analysis Demystified

In this section we show that the original analysis of genetic algorithms is based on a mesoscopic view. The theory has been developed by Holland [8]. It analyzes "schemata" and their evolution in a population.

Definition 8. *Let $p(\mathbf{x},t)$ denote the probability of $\mathbf{x}$ in the population at generation t. Let $\mathbf{x}_s = (x_{s_1},\ldots,x_{s_i}) \subset \{x_1,\ldots,x_n\}$. Thus $\mathbf{x}_s$ denotes a sub vector of $\mathbf{x}$ defined by the indices $s_1,\ldots,s_i$. Then the probability of schema $H(\boldsymbol{s})$ and its fitness $f(H(\boldsymbol{s}))$ are defined by*

$$p(H(\boldsymbol{s}),t) = \sum_{X|X_s=x_s} p(\mathbf{x},t) \tag{8}$$

$$f(H(\boldsymbol{s}),t) = \frac{\sum_{X|X_s=x_s} p(\mathbf{x},t) f(\mathbf{x})}{p(H(\boldsymbol{s}),t)} \tag{9}$$

The summation is done by keeping the values of $\mathbf{x}_s$ fixed. Thus the probability of a schema is given by the corresponding marginal distribution $p(\mathbf{x}_s)$. Let us now assume that we have an algorithm which generates new points according to the distribution of selected points. With proportionate selection (equation (4)) we have

$$p(\mathbf{x}, t+1) = p(\mathbf{x}, t)\frac{f(\mathbf{x})}{E_t[f(x)]} \tag{10}$$

This can be seen as the ideal search distribution of *SGA*. The next theorem immediately follows from the definitions.

Theorem 9 (Schema Theorem). *For proportionate selection the probability of schema $H(\boldsymbol{s})$ is given by*

$$p(H(\boldsymbol{s}), t+1) = p(H(\boldsymbol{s}), t))\frac{f(H(\boldsymbol{s}), t)}{E_t[f(\mathbf{x})]} \tag{11}$$

Holland [Theorem 6.2.3][8] computed for *SGA* (a genetic algorithm with recombination and mutation) an inequality

$$p(H(\boldsymbol{s}), t+1) \geq (1-\delta)p\big(H(\boldsymbol{s}, t)\big)\frac{f(H(\boldsymbol{s}), t)}{E_t[f(\mathbf{x})]} \tag{12}$$

δ is a small factor which captures the loss by mutation and crossover. The inequality only complicates the analysis. An application of the inequality (12) is not possible without computing $E_t[f(\mathbf{x})]$. This in turn requires the computation of $p(\mathbf{x}, t)$. Goldberg [6] circumvented this problem by assuming

$$f(H(\boldsymbol{s}), t) \geq (1+c)E_t[f(\mathbf{x})] \tag{13}$$

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

We will not cite all the folklore about the increase of the number of above average schemata. It turns out that equation (10) admits an analytical solution.

Theorem 10 (Convergence). *The distribution $p(\mathbf{x}, t)$ for proportionate selection is given by*

$$p(\mathbf{x}, t) = \frac{p(\mathbf{x}, 0)f(\mathbf{x})^t}{\sum_y p(\mathbf{y}, 0)f(\mathbf{y})^t} \tag{14}$$

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

$$\lim_{t\to\infty} p(\mathbf{x}, t) = \begin{cases} 1/|\mathcal{M}| & \mathbf{x} \in \mathcal{M} \\ 0 & else \end{cases} \tag{15}$$

Equation (14) was already used by Goldberg and Deb [7] in a different context. It enables an exact schema analysis for an ideal genetic algorithm. This is a conceptual algorithm because it needs an exponential amount of computation. But for small problems the increase or decrease of any schema can be exactly computed.

In the next section we will extend the stochastic theory. The theory will require *conditional* distributions. From the analysis we will derive a usable algorithm.

4 The Factorized Distribution Algorithm FDA

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

Definition 11. *For $\beta \geq 0$ define the **Boltzmann distribution** of a function $f(x)$ as*

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

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

The Boltzmann distribution is usually defined as $e^{-\frac{g(\mathbf{x})}{T}}/Z$. The term $g(\mathbf{x})$ is called the energy and $T = 1/\beta$ the temperature. The Boltzmann distribution concentrates around the global optima of the function with increasing β . If it would be possible to sample efficiently from this distribution for arbitrary β, optimization would be an almost trivial task.

4.1 Boltzmann selection

Our proposed algorithm incrementally computes Boltzmann distributions by using Boltzmann selection.

Definition 12. *Given a distribution p and a selection parameter $\Delta\beta$, **Boltzmann selection** calculates the distribution of the selected points according to*

$$p^s(\mathbf{x}) = \frac{p(\mathbf{x})e^{\Delta\beta f(\mathbf{x})}}{\sum_y p(\mathbf{y})e^{\Delta\beta f(\mathbf{y})}} \tag{17}$$

We can now define the *BEDA* (Boltzmann Estimated Distribution Algorithm). *BEDA* is a conceptual algorithm, because the calculation of the distribution requires a sum over exponentially many terms. We have proven the following important convergence theorem for it [24].

Theorem 13 (Convergence). *Let $\Delta\beta(t)$ be an annealing schedule, i.e. for every t increase the inverse temperature β by $\Delta\beta(t)$. Then for BEDA the distribution at time t is given by*

$$p(\mathbf{x}, t) = \frac{e^{\beta(t) f(\mathbf{x})}}{Z_f(\beta(t))} \tag{18}$$

with the inverse temperature

$$\beta(t) = \sum_{\tau=1}^{t} \Delta\beta(\tau). \tag{19}$$

Algorithm 2: *BEDA* – Boltzmann Estimated Distribution Algorithm

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

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

$$\lim_{t\to\infty} p(x,t) = \begin{cases} 1/|\mathcal{M}| & x \in \mathcal{M} \\ 0 & else \end{cases} \tag{20}$$

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

4.2 Factorization of the distribution

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

Definition 14. *Let $s_1, \ldots, s_m$ be index sets, $s_i \subseteq \{1, \ldots, n\}$. Let f_{s_i} be functions depending only on the variables x_j with $j \in s_i$. Then*

$$f(\mathbf{x}) = \sum_{i=1}^{m} f_i(\mathbf{x}_{s_i}) \tag{21}$$

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

We also need the following definitions

Definition 15. *Given $s_1, \ldots, s_m$, we define for $i = 1, \ldots, m$ the sets d_i, b_i and c_i:*

$$d_i := \bigcup_{j=1}^{i} s_j, \qquad b_i := s_i \setminus d_{i-1}, \qquad c_i := s_i \cap d_{i-1} \tag{22}$$

We set $d_0 = \emptyset$.

In the theory of decomposable graphs, d_i are called *histories*, b_i *residuals* and c_i *separators* [16]. We now need the conditional probabilities from definition 2. In [24] we have proven the following theorem.

Theorem 16 (Factorization Theorem). *Let $p_\beta(\mathbf{x})$ be a Boltzmann distribution with*

$$p_\beta(\mathbf{x}) = \frac{e^{\beta f(\mathbf{x})}}{Z_f(\beta)} \tag{23}$$

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

$$b_i \neq \emptyset \quad \forall i = 1, \ldots, l; \quad d_l = \tilde{X}, \tag{24}$$

$$\forall i \geq 2 \; \exists j < i \;\; \textit{such that } c_i \subseteq s_j \tag{25}$$

then

$$p_\beta(\mathbf{x}) = \prod_{i=1}^{m} p(x_{b_i} | x_{c_i}) \tag{26}$$

The constraint defined as equation (25) is called the *running intersection property*. The assumptions of the theorem are formally identical to the general factorization theorem of graphical models [16].

Algorithm 3: *FDA* – Factorized Distribution Algorithm

---

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

With the help of the factorization theorem, we can turn the conceptional algorithm *BEDA* into *FDA*, the Factorized Distribution Algorithm. The factorized distribution is identical to the Boltzmann distribution if the conditions of the factorization theorem are fulfilled. Therefore the convergence proof of *BEDA* applies to *FDA*. *FDA* can in principle be used with any selection scheme, but then the convergence proof is no longer valid. We think that Boltzmann selection is an essential part in using the *FDA*. *FDA* with Boltzmann selection is connected to a dynamical system which we denote $MULTI_{\beta(t)}$. It is defined by equation (23).

Because *FDA* uses finite samples of points to estimate the conditional probabilities, convergence to the optimum will depend on the size of the samples (the population

size). *FDA* has experimentally proven to be very successful on a number of functions where standard genetic algorithms fail to find the global optimum. In [20], the scaling behaviour for various test functions has been studied. The estimation of the probabilities and the generation of new points can be done in polynomial time.

In the next section we derive an adaptive annealing schedule, which connects Boltzmann selection to truncation selection used by breeders.

5 The adaptive annealing schedule SDS

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

5.1 Taylor expansion of the average fitness

In order to determine an adaptive annealing schedule, we will make a Taylor expansion of the average fitness. The average fitness $E_\beta[f(x)]$ from equation 2 is now seen as a function of the inverse temperature. We have proven [17]:

Theorem 17. *The average fitness $E_\beta[f(x)]$ using Boltzmann distributions has the following expansion in β:*

$$E_{\tilde{\beta}}[f(x)] = E_\beta[f(x)] + \sum_{i \geq 1} \frac{(\tilde{\beta} - \beta)^i}{i\,!} M^c_{i+1}(\beta) \tag{27}$$

where M^c_i are the centered moments

$$M^c_i(\beta) := \sum_x \big[f(x) - E_\beta[f(x)]\big]^i p(x) \tag{28}$$

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

$$M^c_{i+1}(\beta) = \left(\frac{\frac{\partial}{\partial \beta} Z_f(\beta)}{Z_f(\beta)} \right)^{(i)} \qquad \text{for } i \geq 1, \quad M^c_1 = 0 \tag{29}$$

Corollary 18. *We have approximatively*

$$E_{\tilde{\beta}}[f(x)] - E_\beta[f(x)] \approx (\tilde{\beta} - \beta) \cdot \sigma^2_f(\beta) \tag{30}$$

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

The proof of the above theorem can be found in [22]. Equation (30) was already proposed in [13]. It is a macroscopic equation relating the average fitness and the variance. From (30) we can derive an adaptive annealing schedule. We recall that truncation selection has proven to be a robust and efficient selection scheme. For truncation selection the *response to selection* $R(t)$ [21] is approximatively given by equation

$$R(t) := E_{t+1}[f(x)] - E_t[f(x)] \approx I_\tau b(t)\sigma_f(t) \tag{31}$$

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

Definition 19. *The **standard deviation schedule** SDS is defined by* $\beta(t+1) = \beta(t) + c/\sigma_f(\beta(t))$.

Using *SDS* we obtain from equation (30)

$$R(t) = E_{\beta(t+1)}[f(x)] - E_{\beta(t)}[f(x)] \approx c \cdot \sigma_f(t) \tag{32}$$

Thus *SDS* with Boltzmann selection behaves similarly to truncation selection if $c = I_\tau b(t)$. We recently found that *SDS* has already been used for genetic algorithms in [30]. But there *SDS* has been derived from a different perspective.

5.2 Linear functions

We will show the connection between *SDS* and truncation selection for linear functions

$$Linear(\mathbf{x}) = \sum_{i=1}^{n} a_i x_i \tag{33}$$

We easily compute

$$p_\beta(\mathbf{x}) = \frac{e^{\beta f(\mathbf{x})}}{Z_f(\beta)} = \prod_{i=1}^{n} \frac{e^{\beta a_i x_i}}{1 + e^{\beta a_i}} \tag{34}$$

Thus we have

$$p_i(\beta) := p_\beta(x_i = 1) = \frac{e^{\beta a_i}}{1 + e^{\beta a_i}}. \tag{35}$$

For a linear function the variance is just the sum of the variance of the individual variables, therefore

$$\sigma_f^2(\beta) = \sum_{i=1}^{n} \frac{a_i^2 e^{\beta a_i}}{(1 + e^{\beta a_i})^2} = \sum_{i=1}^{n} a_i^2 p_i(\beta)\big(1 - p_i(\beta)\big) \tag{36}$$

The *SDS* schedule is given by

$$\beta(t+1) = \beta(t) + \frac{c}{\sqrt{\sum_i a_i^2 p_i(\beta)\big(1 - p_i(\beta)\big)}} \tag{37}$$

We approximate the difference equation by a differential equation:

$$\frac{d\beta}{dt} \approx \frac{c}{\sqrt{\sum_i a_i^2 p_i(\beta)\big(1-p_i(\beta)\big)}} \tag{38}$$

If we differentiate equation (35) we obtain

$$\begin{aligned}\frac{dp_i(\beta)}{dt} &= \frac{a_i e^{\beta a_i}(1+e^{\beta a_i})\beta' - e^{\beta a_i} a_i e^{\beta a_i}\beta'}{(1+e^{\beta a_i})^2} \\ &= p_i(\beta)\big(1-p_i(\beta)\big)\, a_i \tfrac{d\beta}{dt}\end{aligned} \tag{39}$$

These equations define a dynamical system in *continuous* time. If we insert equation (38) we obtain

$$\frac{dp_i(\beta)}{dt} = c \cdot \frac{p_i(\beta)\big(1-p_i(\beta)\big)a_i}{\sqrt{\sum_i a_i^2 p_i(\beta)\big(1-p_i(\beta)\big)}} \tag{40}$$

The differential equations remains the same if we multiply all a_i by a constant factor. For *Onemax* we have $a_i = 1$. In this case all marginal frequencies are equal and we can set $p_\beta := p_i(\beta)$. We obtain the differential equation

$$\frac{dp_\beta}{dt} = c\sqrt{p_\beta(1-p_\beta)/n} \tag{41}$$

This differential equation has been derived for truncation selection in [19]. There the solution can be found.

6 Mutation and the Hyper Parameter r

UMDA and *FDA* can be run without a parameter corresponding to mutation in genetic algorithm. In order to obtain good solutions, the size of the population has to be chosen accordingly. But there is an easy way to introduce "mutation". Normally the empirical probability is estimated by $p_i = m/N$. Here m denotes the number of occurrences of $x_i = 1$ in a sample of size N. But in the Bayesian approach the estimated probability is set to $p_i = (m+r)/(N+2r)$. The *hyper parameter* r has to be chosen in advance [9]. The hyper parameter is a simple example of a *Bayesian prior*. It is related to *mutation* in genetic algorithms works. Mutation works in the following way: When generating new individuals, with a probability of μ the generated bit is inverted.

Theorem 20. *For binary variables, the expectation value for the probability using a hyper parameter r is the same as mutation with mutation rate $\mu = r/(N+2r)$ and using the maximum likelihood estimate.*

The theorem can easily be proven by calculating the probability of generating a particular bit for both cases. Wright's equation can be extended to include mutation (or equivalently a hyper parameter r) [23]. The extended equation defines a dynamical

system which we call $UNI_p(r)$. $r > 0$ moves the attractors from the boundary of the hypercube into the interior. For $r \to \infty$ there is a unique attractor at $p = 0.5$. The hyper parameter can also be used for *UMDA*. This algorithm we call $UMDA(r)$. The relation between the attractors of $UNI_p(r)$ and the populations generated by $UMDA(r)$ is as follows.

Let $0 < p_i^*(x_i) < 1$ denote the values of an attractor of $UNI_p(r)$. Then $UMDA(r)$ will generate for $t \to \infty$ populations according to

$$p(\mathbf{x}, \infty) = \prod_{i=1}^{n} p_i^*(x_i)$$

The dynamical system $UNI_p(r)$ has converged to an attractor, but $UMDA(r)$ generates populations which can be very different from each other. Thus in order to be able to observe convergence for $UMDA(r)$, we require that the attractor is nearby the boundary. To be more specific: r should be so small that an attractor nearby a global optimum should enable $UMDA(r)$ to generate the optimum with a high probability, say about 30%. Thus we consider mutation to be a background operator.

The problem of determining a suitable r for *UMDA* with proportionate and truncation selection has been investigated in [23]. We obtained the following rule of thumb:

For truncation selection with selection intensity I_τ use a value of $r = I_\tau M/n$. τ is the proportion of individuals selected, $M = \tau N$.

We now compute the attractors of the dynamical system $MULTI_{\beta(t)}$ behaving for a linear function similar to $UMDA(r)$ with Boltzmann selection and a large population. Let the linear function be defined by $Linear(x) = \sum a_i x_i$. For Boltzmann selection we easily compute

$$p_i^s(t) = \frac{p_i(t)e^{a_i \Delta\beta}}{1 + p_i(t)(e^{a_i \Delta\beta} - 1)} \tag{42}$$

We now assume our recommended prior of $r = \alpha N/n$. Then we obtain

$$p_i(t+1) = \frac{p_i^s(t)N + r}{N + 2r} = \frac{np_i^s(t) + \alpha}{n + 2\alpha} \tag{43}$$

This gives

$$p_i(t+1) = \frac{np_i(t)e^{a_i \Delta\beta} + \alpha + \alpha p_i(t)(e^{a_i \Delta\beta} - 1)}{(n + 2\alpha)(1 + p_i(t)(e^{a_i \Delta\beta} - 1))} \tag{44}$$

Equilibrium is reached when $p_i(t+1) = p_i(t)$. This is a quadratic equation. Let $\gamma_i = a_i \Delta\beta$. Then the positive solution is given by

$$p_i^* = \frac{(e^{\gamma_i} - 1)(n + \alpha) - 2\alpha + \sqrt{\left((e^{\gamma_i} - 1)(n + \alpha) - 2\alpha\right)^2 + 4\alpha(e^{\gamma_i} - 1)(n + 2\alpha)}}{2(e^{\gamma_i} - 1)(n + 2\alpha)} \tag{45}$$

In order to compute a numerical example we set $a_i = 0.5$ and $\alpha = 1$. In Figure 1 the probability $P_s^* = \prod_{i=1}^{n} p_i^*$ of generating the optimum is displayed. There is almost

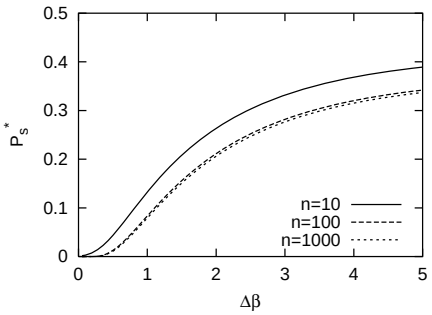

Figure 1: Value of P_s, the probability to generate the optimum, when varying β using equation (44)

no difference between $n = 100$ and $n = 1000$. But we have to use $\Delta\beta = 3$ in order to have a probability of 0.3 to generate the optimum. This demonstrates the weakness of a fixed annealing schedule.

For the *SDS* schedule we have $\Delta\beta(t) = c/\sigma(t)$. In this case an analytical solution of equation (44) cannot be obtained. But the fix-points can be obtained numerically by iteration until $p_i(t+1) = p_i(t)$. In the next table the results for different linear functions are displayed. The function $Linear_{inc}$ is defined by $a_i = i$, the function $Exp_{inc\gamma}$ by $a_i = 2^\gamma$. The probability to generate the optimum is about 0.15 – with the exception of $Exp_{inc2.0}$. For this function a smaller prior has to be used. If we use $r = 0.5N/n$ then $P_s^* = 0.165$.

Function	$Onemax(16)$	$Linear_{inc}$	$Exp_{inc1.5}$	$Exp_{inc2.0}$	$Onemax(100)$
P^*	0.214	0.170	0.159	0.053	0.147

We next compare the theoretical results with simulation runs of $UMDA(r)$. In table 1 the univariate marginal frequency p_i^* is shown. Note that the attractors of *SDS* are fairly independent from the size of the population.

Params	Theory	Simulation
$n = 100, \Delta\beta = 1, N = 100$	0.98458	0.9843
$n = 100, \Delta\beta = 1, N = 1000$	0.9846	0.9863
$n = 100, \Delta\beta = 1, N = 30$	0.9846	0.9881
$n = 100, \Delta\beta = 2, N = 1000$	0.9887	0.9891
$n = 100, \Delta\beta = 0.25, N = 1000$	0.9572	0.9766
$n = 100, SDS, N = 100$	0.9814	0.9862
$n = 100, SDS, N = 30$	0.9814	0.9817

Table 1: Attractor p^* for *Onemax* and $MULTI_{\beta(t)}$

6.1 Calculating a bound of the hyper parameter for FDA

The theory of Bayesian hyper parameters can be extended to conditional probabilities [2]. Our analysis will be very crude, giving a rule of thumb to be tested in practice. The chain rule of conditional probabilities says that

$$p(x_1,\ldots,x_k) = p(x_1)\cdot p(x_2|x_1)\cdot p(x_3|x_1,x_2)\cdots p(x_k|x_1,\ldots,x_{k-1}) \tag{46}$$

Using the Bayesian estimates, we get the following equation:

$$\begin{aligned}\frac{N(x_1,\ldots,x_k)+r'}{N+2^k r'} =& \frac{N(x_1)+r_1}{N+2r_1}\cdot\frac{N(x_1,x_2)+r_2}{N(x_1)+2r_2}\cdot\frac{N(x_1,x_2,x_3)+r_3}{N(x_1,x_2)+2r_3}\cdots \\ &\cdot\frac{N(x_1,\ldots,x_k)+r_k}{N(x_1,\ldots,x_{k-1})+2r_k}\end{aligned} \tag{47}$$

where $N(\cdot) = N\cdot p(\cdot)$ is the number of occurrences in the population. The denominators were chosen in such a way that $\sum_{x_1,\ldots,x_k} p(x_1,\ldots,x_k) = 1$ and $\sum_{x_i} p(x_i|x_1,\ldots,x_{i-1}) = 1$.

In order for (47) to hold, the fractions on the right hand side have to cancel each other out. We get the following identities for the parameters:

$$2r_i = r_{i-1} \implies r_i = 2^{-(i-1)}r_1 \quad \text{and} \quad r' = r_k = 2^{-(k-1)}r_1 \tag{48}$$

Thus we have obtained the rule of thumb:

Let r be the hyper parameter for a single binary variable. Then the hyper parameter r' for a marginal distribution $p(x_1,\ldots,x_k)$ and the hyper parameter r^* for a conditional distribution $p(x_k|x_1,\cdots,x_{k-1})$ should be

$$r' = r^* = 2^{-(k-1)}\cdot r \tag{49}$$

It is not possible to evaluate the rule of thumb for real attractors defined by the dynamic equilibrium between selection and mutation. We test our proposal assuming that the selected points are at the boundary. Let the probability distribution be the product of marginal distributions of k_i variables each. Then we have $l = n/\sum k_i$ factors. The probability P_s^* of generating the optimum is at most

$$P_s^* = \prod_{i=1}^{l}\left(\frac{N+r'}{N+2^{k_i}\cdot r'}\right) \tag{50}$$

where $r' = 2^{-(k_i-1)}r$. If we set $r = N/n$ then

$$P_s^* = \prod_{i=1}^{l}\left(\frac{n+2^{-(k_i-1)}}{n+2}\right) = \prod_{i=1}^{l}\left(1-\frac{2(1-2^{-k_i})}{n+2}\right) \geq 0.3 \tag{51}$$

Thus using our rule of thumb we generate the optimum with a probability greater than 0.3.

6.2 UMDA with very small population size

Formally $UMDA(r)$ can run with a very small population size N and a small number of selected points M. It fulfills the requirements of weak convergence: with probability greater than zero it will find the optima. $UMDA(r)$ with a tough selection ($M = 2$) can be seen as a stochastic local search algorithm with an unrestricted neighborhood. The points of the neighborhood are not chosen uniformly, but points with a small Hamming distance to the selected points are chosen more often. In fact, this algorithm can be seen as an instance of an $(2, N)$ evolution strategy [1] adapted to discrete variables.

We will discuss the case $M = 2$ in more detail. With two selected points only three different values of p_i^s are possible, namely $0, 0.5, 1$. Our recommended hyper parameter is $r = I_\tau \cdot 2/n$. The relation between N and M is captured by I_τ. The larger N, the larger will be r. For $N = 4$ we have $\tau = 0.5$ and $I_\tau = 0.8$ [19]. This gives $r = 1.6/n$. Thus $UMDA(r)$ generates new points with $p_i = r/(2 + 2r)$ if $m = 0$, $p_i = 0.5$ if $m = 1$ and $p_i = 1 - r/(2 + 2r)$ if $m = 2$. m denotes the number of instances of $x_i = 1$ in the two selected points.

$$p(\mathbf{x}, t + 1) = \prod_i^n \frac{p_i^s(x, t) \cdot M + r}{M + 2r} \tag{52}$$

We now investigate the behaviour by simulations. The functions to be optimized are *Onemax* and *Jump*. The function *Jump* has a valley of *gap* bits before the global optimum consisting of all bits set to 1. At the bottom of the valley the fitness values are set to 0. Thus *Onemax* can be seen as a *Jump* with a *gap* of 0.

gap	96	64	48	24	12	12(0.5)	6(0.5)	4(0.5)	4
0	801	566	457	348	345	449	697	2527	8020
1	768	627	480	456	610	646	3042	(80)13038	(30)14682
2	860	693	544	769	4996	4555	(90)10917	-	-
3	1113	1213	921	3727	36333	(60)42686)	-	-	-

Table 2: Function evaluations for different population sizes, $n = 50$, $\tau = 0.25$; $\tau = 0.5$ for three cases; number in parentheses is success rate out of 100 runs.

From table 2 we conclude that for $gap = 0$ the best result is obtained with $N = 12$. For $gap = 3$ a larger value, $N = 48$, gives the best result. Small population sizes do not reach the optimum in reasonable time. A small population has difficulties to jump over the valley. With a hyper parameter r a population size too small is much worse than a population size too large. The larger *gap*, the larger is the population size giving the best results.

7 FDA and the approximation of the Boltzmann distribution

FDA approximates the Boltzmann distribution in a way not used before. It starts with a uniform distribution. Then Boltzmann selection is applied to compute the new

parameters of the distribution. New points are generated using these estimates. For an infinite population we get an exact Boltzmann distribution at every generation (step). But as we use a finite sample, the Boltzmann distribution will only be approximated. A hyper parameter makes the algorithms more robust concerning the population size and premature convergence. But it moves the empirical distribution even more away from a desired Boltzmann distribution. We will now investigate the approximation error. The following cases will be distinguished:

- *FDA* with exact factorization and $r = 0$
- *FDA* with exact factorization and the recommended hyper parameter
- *FDA* with approximate factorization and $r = 0$
- *FDA* with approximate factorization and recommended hyper parameter

For the analysis the Kullback-Leibler divergence between the generated distribution (by *FDA*) and a Boltzmann distribution is used. For the Boltzmann distribution we have two choices. We can assume that in every generation β is changed according to Boltzmann selection by $\beta \leftarrow \beta + \Delta\beta$. This value is subsequently used for the Boltzmann distribution. In our second choice we compute β_{opt} giving the smallest Kullback-Leibler divergence of a Boltzmann distribution to the empirical distribution.

The Kullback-Leibler divergence of two distributions is defined as

$$D^{KL}(p \,\|\, q) = \sum_{\mathbf{x}} p(\mathbf{x}) \ln \frac{p(\mathbf{x})}{q(\mathbf{x})} \tag{53}$$

with $p(\mathbf{x}) \ln p(\mathbf{x}) = 0$ when $p(\mathbf{x}) = 0$. The divergence is infinite for $p(\mathbf{x}) \neq q(\mathbf{x}) = 0$.

If p_β is the Boltzmann distribution and q the empirical distribution, we compute

$$D^{KL}(q \,\|\, p_\beta) = \sum_{\mathbf{x}} q(\mathbf{x}) \ln q(\mathbf{x}) + \ln Z_\beta - \beta \sum_{\mathbf{x}} q(\mathbf{x}) f(\mathbf{x}) \tag{54}$$

$$\frac{\partial D^{KL}(q \,\|\, p_\beta)}{\partial \beta} = \frac{\frac{\partial}{\partial \beta} Z_\beta}{Z_\beta} - \sum_{\mathbf{x}} q(\mathbf{x}) f(\mathbf{x}) = E_\beta[f(\mathbf{x})] - E_q[f(\mathbf{x})] \tag{55}$$

where $E_\beta[f(\mathbf{x})]$ is the average fitness according to the p_β distribution and $E_q[f(\mathbf{x})]$ is the average fitness according to the q distribution. We have numerically solved equation $\partial D^{KL}/\partial\beta = 0$ to obtain the values β_{opt} in table 7.

We use two functions, *Decep*15, and *Grid*16 as example. *Decep*15 is a separable function. It consists of five blocks of three variables. *Grid*16 is defined on a 4*4 grid. For this function we have used an approximate factorization using factors of four or three variables. The exact definition of the functions is not necessary.

For both problems D^{KL} first increases slightly. It decreases when the algorithm approaches an attractor. D^{KL} is larger for the approximate factorization and when *FDA* is used with a hyper parameter. But in all cases D^{KL} is surprisingly small. For comparison we also show the difference of the *UMDA* factorization to a Boltzmann distribution. It is much higher. This factorization is not able to approximate the Boltzmann distribution.

$Func$	N	$prior$	$iter$	β	D^{KL}	β_{opt}	D^{KL}
$Grid16$	100	no	1	0.736	1.051	0.728	1.051
			2	1.630	1.779	1.362	1.757
			6	8.924	1.596	7.437	1.544
			7	15.505	0.798	14.158	0.790
	100	yes	1	0.797	1.061	0.541	1.001
			2	1.699	1.705	1.124	1.453
			6	5.378	2.578	3.113	1.425
			7	6.359	3.023	3.597	1.469
$Decep15$	100	no	1	0.389	0.673	0.467	0.672
			2	0.829	0.787	0.829	0.787
			6	4.159	0.085	4.065	0.085
			7	8.349	0.001	12.022	0.001
	100	yes	1	0.428	0.341	0.351	0.315
			2	0.864	0.907	0.600	0.672
			6	2.738	2.972	1.320	0.421
			7	3.216	2.851	1.551	0.392
$Decep15(UMDA)$	100	no	1	0.472	1.021	0.004	0.153
			2	0.934	3.780	0.020	0.311
			6	2.551	17.945	0.343	2.584
			7	2.921	16.435	0.679	3.543

Table 3: Results of Kullback-Leibler divergences, $n = 15/16$

8 Holland's schema analysis and the Boltzmann distribution

We will now turn back to the analysis of genetic algorithms made by Holland. We will use Holland's notation. ξ denotes a schema, the probability $P(\xi, t)$ has been defined in equation (8), and the average fitness $\hat{\mu_\xi}(t)$ is given by equation (9). Holland makes the following conjecture about a good population based search algorithm

Holland ([8],p.88): *Each (schema) ξ represented in (the current population) $B(t)$ should increase (or decrease) in a rate proportional to its observed usefulness $\hat{\mu_\xi}(t) - \hat{\mu}(t)$ (average fitness of schema ξ minus average fitness of the population)*

$$\frac{dP(\xi,t)}{dt} = (\hat{\mu_\xi}(t) - \hat{\mu}(t))P(\xi,t) \tag{56}$$

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

Theorem 21. *The Boltzmann distribution $p(\mathbf{x},t) = e^{tf(\mathbf{x})}/Z_t$ with $P(\xi,t) = \sum_{X|X_\xi = x_\xi} p(\mathbf{x},t)$ fulfills Holland's equation* (56).

Proof: Taking the derivative we easily obtain

$$\frac{p(\mathbf{x},t)}{dt} = p(\mathbf{x},t)(f(\mathbf{x}) - \bar{f}(t)) \tag{57}$$

Let now $\mathbf{x}_\xi$ define a marginal distribution. Then

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

Thus the Boltzmann distribution with the *annealing schedule* $\beta(t) = t$ fulfills Holland's equation. According to Holland's analysis *FDA* with this schedule should be an almost optimal algorithm. The problem is to define in a precise manner what is meant by an optimal algorithm. Holland has derived equation (56) from an information theoretic analysis. We state the result as a conjecture:

Conjecture: *Generating search points according to a Boltzmann distribution seems a very good search strategy for optimization.*

9 A kingdom for approximating the Boltzmann distribution

With *FDA* we try to sample efficiently from a Boltzmann distribution. But also other disciplines need to estimate or sample a distribution. The relation becomes clear if we summarize the major tasks:

- to estimate and sample $p(\mathbf{x})$ (density estimation)
- to estimate and sample N points with high $p(\mathbf{x})$ (optimization)
- to estimate $\mathbf{y}$ given $\mathbf{z}$, e.g to compute $p(\mathbf{y}|\mathbf{z})$ (probabilistic reasoning)
- to estimate the probability of $\mathbf{y}$ being true given $\mathbf{z}$, e.g to compute $p(\mathbf{y}|\mathbf{z})$ in a probabilistic logic setting

The difference between probabilistic reasoning and probabilistic logic is as follows: probabilistic reasoning uses statistical dependencies between variables, whereas probabilistic logic creates a graph from rules.

In this paper we have mainly dealt with optimization. In statistical physics there exist some very old and almost forgotten algorithms to effectively calculate the Boltzmann distribution if the energy function ($E = -f(\mathbf{x})$) is known [25]. Thus a fascinating interdisciplinary research is well on the way – bringing together such diverse fields as population based optimization, probabilistic reasoning, and statistical physics. The core of the theory is the same: the factorization of the distribution if the corresponding factor graph is singly connected.

We will explain the approach of statistical physics in more detail. If the function is given in an analytical form, why do we compute the Boltzmann distribution by sampling? It is possible to compute an optimal Boltzmann distribution directly by minimizing the Kullback-Leibler divergence. The minimization takes the parameters of the factorization as *variables* to be determined. We will explain the approach with the simplest example.

9.1 The mean field approach

In the *mean field approach* one assumes that the distribution is given by the product also used by *UMDA*

$$q(\mathbf{x}) = \prod_{i=1}^{n} q_i(x_i) \tag{58}$$

First we compute

$$\begin{aligned}\sum_{\mathbf{x}} q(\mathbf{x}) \ln q(\mathbf{x}) &= \sum_{x_1} \sum_{x_2,\ldots,x_n} q_1(x_1) \prod_{i=2}^{n} q_i(x_i) \cdot \big[\ln q_1(x_1) + \sum_{j=2}^{n} \ln q_j(x_j)\big] \\ &= \sum_{x_1} q_1(x_1) \ln q_1(x_1) \underbrace{\sum_{x_2,\ldots,x_n} \prod_{j=2}^{n} q_j(x_j)}_{=1} + \underbrace{\sum_{x_1} q_1(x_1)}_{=1} \sum_{x_2,\ldots,x_n} \prod_{i=2}^{n} q_i(x_i) \sum_{j=2}^{n} \ln q_j(x_j) \\ &= \ldots = \sum_{i=1}^{n} \big(q_i \ln q_i + (1-q_i)\ln(1-q_i)\big)\end{aligned}$$

For the derivation we have used an obvious recursion in n and the fact that $q_i(0) = 1 - q_i$. The Kullback-Leibler divergence of $q(\mathbf{x})$ to a Boltzmann distribution can be written as

$$\begin{aligned}D^{KL}(q \,\|\, p_\beta) &= \sum_{\mathbf{x}} q(\mathbf{x}) \big[\ln q(\mathbf{x}) - \beta f(\mathbf{x}) + \ln Z_\beta\big] \\ &= \ln Z_\beta + \sum_{\mathbf{x}} q(\mathbf{x}) \ln q(\mathbf{x}) - \beta \sum_{\mathbf{x}} q(\mathbf{x}) f(\mathbf{x}) \\ &= \ln Z_\beta + \sum_{i=1}^{n} \big(q_i \ln q_i + (1-q_i)\ln(1-q_i)\big) - \beta \cdot W(q_1, \ldots, q_n)\end{aligned}$$

W denotes the average fitness, seen as a function of $\mathbf{q}$. A local minimum of the divergence can be obtained by setting the derivatives to 0. We obtain:

$$\frac{\partial D^{KL}(q \,\|\, p_\beta)}{\partial q_i} = \ln \frac{q_i}{1-q_i} - \beta \frac{\partial W}{\partial q_i} = 0 \tag{59}$$

This has the solution

$$q_i = \frac{1}{1 + e^{-\beta \frac{\partial W}{\partial q_i}}} \tag{60}$$

Equations (60) are called the *Mean Field Equations* in statistical physics[10]. They can be solved numerically if the expression of W is given. We have computed in [21] the analytical expression of W if the analytical expression of f is given. W is simply obtained by an exchange of variables. Thus, if $f(\mathbf{x}) = \sum_i a_{ii} x_i + \sum_{i \neq j} a_{ij} x_i x_j$ then we have $W(q) = \sum_i a_{ii} q_i + \sum_{i \neq j} a_{ij} q_i q_j$. We will discuss two examples.

9.1.1 Linear Fitness

For the linear function $Linear = \sum_i a_i x_i$ equation (60) has the closed solution

$$q_i = \frac{1}{1 + e^{-a_i \beta}}$$

The solutions are identical to the exact marginal distributions of a Boltzmann distribution (see in equation (35). Thus for linear functions we have $D^{KL} = 0$.

9.1.2 Quadratic Fitness

Let $A = (a_{ij})$ be a symmetric matrix. Consider the quadratic function $f(\mathbf{x}) = \sum_i a_{ii}x_i + \frac{1}{2}\sum_{(ij),i\neq j} a_{ij}x_ix_j$. Here we have $W(\mathbf{q}) = \sum_i a_{ii}q_i + \frac{1}{2}\sum_{(ij),i\neq j} a_{ij}q_iq_j$. From equation (60) we obtain

$$q_i = \frac{1}{1+\exp[-\beta(a_{ii} + \sum_{j\neq i} a_{ij}\cdot q_j)]} \tag{61}$$

This is a system of nonlinear equations in the parameters q_i. There is no closed solution, the equations have to be iterated to find a numerical solution. It is difficult to precisely describe the relation of solutions of equation (60) to solutions of the given optimization problem. We can informally derive our next conjecture from equation (59). For large β the solutions have to fulfill $\partial W/\partial q_i \approx 0$. The conjecture is difficult to specify precisely. Therefore we state informally:

Conjecture: *For β large enough, the solutions of equation (61) are given by either $q_i \leq \epsilon$ or $q_i \geq 1-\epsilon$. ϵ can be made as small as wanted by increasing β. If we set $\epsilon = 0$ then the solutions are local optima of the function $f(\mathbf{x})$ concerning 1-bit changes.*

Proof: Let q_i^* be a solution of (61). We can assume $q_i^* \leq \epsilon\ \forall i$. Then from equation (61) it follows with a large enough β that

$$\forall i: \quad a_{ii} + \sum_{j\neq i} a_{ij}q_j^* < 0 \tag{62}$$

Let us assume that we have another solution r^* where just one r_k^* is different from q_k^*, thus $r_k^* \geq 1-\epsilon$. Then from equation (61) it follows

$$a_{kk} + \sum_{j\neq k} a_{kj}r_j^* = a_{kk} + \sum_{j\neq k} a_{kj}q_j^* > 0$$

But this is a contradiction to our assumption because of equation (62). □

At least for quadratic functions the mean field approach seems to be as powerful as the *UMDA* algorithm. In theory, the mean field approach needs just one step. For optimization one chooses just a very large β. But in practice, the quality of the solutions depend on the stability of the numerical procedure. This is shown in figure 2.

The function to be optimized was a quadratic function of 100 variables on a 10×10 grid where the coefficients have been drawn randomly. The estimate of a local optimum has been determined by a simple procedure: if $p_i < 0.5$ set $p_i = 0$, and if $p_i \geq 0.5$ set $p_i = 1$. The best local optima are obtained for $\beta = 3$. Increasing β further gives worse results. Thus also the mean field approach seem to profit from a good annealing schedule.

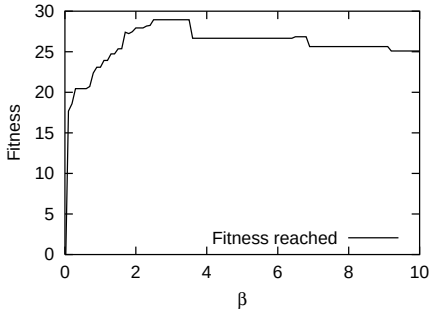

Figure 2: Maximum fitness generated after solving equation (61)

9.2 Advanced mean field methods

For many difficult practical problems, the mean field solutions are just local optima, sometimes far away from the global optima. It is now obvious how to obtain better approximations – use marginal distributions of higher order. We take the quadratic fitness function $f(\mathbf{x}) = \sum_{i,j} a_{ij} x_i x_j$ as example. In general the Boltzmann distribution for this function cannot be exactly factorized using bivariate distributions only. Nevertheless in statistical physics the *ansatz* has been made

$$p_\beta(\mathbf{x}) \approx \frac{1}{Z_\beta} \prod_{i,j}^{n} \Psi_{ij}(x_i, x_j) \prod_{i=1}^{n} \Psi_i(x_i) \tag{63}$$

Now one proceeds as in the mean field approach. The parameters of Ψ_{ij} are determined by minimizing the Kullback-Leibler divergence to the Boltzmann distribution. The equations are really difficult. But several local iteration algorithms have been proposed. For the quadratic fitness function we start the iteration with

$$\Psi_{ij}(x_i, x_j) = e^{\beta a_{ij} x_i x_j} \tag{64}$$

For singly connected graphs the *belief propagation* algorithm of Pearl[26] is the most elegant and efficient iteration algorithm. The following important theorem has been proven [25]:

Theorem 22. *If the graph structure defined by Ψ_{ij} is singly connected then there exist solutions with $D^{KL} = 0$. Furthermore, the solutions can be obtained by using the belief propagation algorithm of Pearl [26].*

The theorem states that for singly connected graphs the solutions are exact Boltzmann distributions. A singly connected graph obviously fulfills the running intersection property. Thus in these cases both *FDA* and advanced mean field methods give the same result. The theorem is valid for any graph fulfilling the running intersection property. Pearl's algorithm has to be modified accordingly (see [16]).

If the graph contains cycles, then Pearl's algorithm does not necessarily converge. The extension of Pearl's algorithm to general 2-d graphs is an area of active research. The interested reader is referred to [25].

9.3 Approximation of the distribution – structure vs. data

If the graph structure defined by the function fulfills the running intersection property then *FDA* as well as advanced mean field methods have for large β attractors nearby the global optima of the function. In principle, one can use a very large β, apply Pearl's algorithm and obtain a distribution which generates the optima with high probability.

Thus we can obtain the optima in one step. But there exists another method, which is for such problems even more effective – it is an extension of dynamic programming. Thus the optimization problems left are those which do not allow an exact factorization with a polynomial number of parameters.

There are at least two methods to compute an approximate distribution for the above problems. In the first approach the structure of the function is used to compute an approximate factorization. This method is used by *FDA* and advanced mean field methods. *FDA* uses a population to determine the parameters, the advanced mean field methods use generalization of Pearl's belief propagation to determine the parameters.

In the second approach we determine the structure from data. Points with high fitness are collected. From the empirical data a Bayesian network is computed. This is called *learning* in Bayesian networks [9]. Early examples of this method are *EBNA* [3, 14], *LFDA* [20], and *BOA* [27].

In the next section we will use a combination of these two methods to solve the graph bipartioning problem. We will consider only those edges as candidates for the Bayesian network which are contained in the given graph.

10 The graph bipartitioning problem

The graph bipartitioning is defined as follows: Given an undirected graph (V, E) with an even number of nodes $|V| = n$, find the partition of the nodes in equal sized sets, such that the cut size is minimal. The cut size is defined as the number of edges *between* the two partitions A and B:

$$\begin{aligned} \min_{A,B \subseteq V} \{cs(A,B) \;\big|\; |A| = |B|\} \qquad & \text{with} \\ cs(A,B) := \left|\left\{(v \leftrightarrow w) \in E \,\middle|\, (v \in A \wedge w \in B) \vee (v \in B \wedge w \in A)\right\}\right| & \end{aligned} \tag{65}$$

In this paper we concentrate on graph bipartitioning. The general M-partitioning problem has been investigated with parallel genetic algorithms in [33].

10.1 UMDA for the bipartioning problem

There is a simple mapping from binary vectors to solutions of the graph bipartitioning problem: the value of x_i is 1 if node $i \in A$ and $x_i = 0$ if node $i \in B$. A graph

bipartitioning problem with n nodes can be represented using an individual with n bits. But there are two problems with this simple approach:

- Most bit strings do not correspond to *feasible* solutions, we need to have exactly $n/2$ bits with value 0.
- Fitness remains constant when all bits are inverted.

We try to solve both problems by using a local search procedures. We start with the second problem. We break the symmetry with the following procedure. We define the best solution in our population as the reference point. Inversion of bits leave the fitness unchanged. Thus we compute the Hamming distance to the reference solution for the original string and the inverted string. We put the string with the smallest Hamming distance into the population.

The calculation of the cut size can be done in the bit string representation by

$$cs(x) = \sum_{(i \leftrightarrow j) \in E} x_i + x_j - 2x_i x_j \tag{66}$$

Obviously, this is a quadratic function.

10.2 The Kernighan-Lin algorithm

The Kernighan-Lin algorithm [11] is an efficient heuristic to find a solution for the graph bipartitioning problem. It uses several passes. In every pass, the current solution is improved by swapping pairs of nodes to get a new solution. This is iterated until a pass does not give an improvement.

In [4] a similar algorithm was introduced that reduced the complexity per pass from $O(|V|^2)$ to $O(|E|)$. A conceptional difference is that in every step a single node may move into the other partition. This violates the constraint $|A|=n/2$. Therefore in the next step we will choose an element from the larger partition to move to the smaller partition.

The speed gain is possible by using an additional data structure. This data structure makes it possible to calculate the edge with maximum gain in constant time. This is done by storing for every possible gain a linked list of corresponding nodes. This list can be updated in time $O(|E|)$ [4]. The gain g_c is the increase (or decrease) in the cut size when node c changes from one set to the other, so

$$v \in A: \qquad g_v := \left|\left\{w \in B \middle| (v \leftrightarrow w) \in E\right\}\right| - \left|\left\{w \in A \middle| (v \leftrightarrow w) \in E\right\}\right| \tag{67}$$

and analogously for $v \in B$.

By considering single nodes and not pairs, it is possible to start with non feasible solutions. In the beginning, only nodes from the larger partition are considered for movement, until both partitions are equal. The details can be seen in algorithm 4. The algorithm has an unknown number of cycles (the outer while loop) until it converges. In every cycle, two lists of candidate edges Q_A and Q_B are maintained, with initially

$Q_A = A$ and $Q_B = B$. Then the gains are calculated. In the inner loop, while there are still elements in Q_A or Q_B, the element with highest gain is chosen from the larger of the candidate sets. The set name is stored in M_i, the element in c_i. Then it is removed from the candidate set and the gains are updated. In f_i we mark if we have a bipartitioning state.

Finally, of the sequence of moves c_1 till $c_{|V|}$, one sequence which has the bipartitioning property and lowest cut size is chosen and those moves are performed.

Algorithm 4: Kernighan-Lin with single swaps

1 Start with an arbitrary partition A, B.
2 **do** {
3 $Q_A \Leftarrow A$, $Q_B \Leftarrow B$, $i \Leftarrow 1$, initialize the gain lists g_c.
4 **do** {
5 **if** $|Q_A| > |Q_B|$: $M_i \Leftarrow A$, choose $c_i \in Q_A$ with max. g_{c_i}
6 **if** $|Q_A| < |Q_B|$: $M_i \Leftarrow B$, choose $c_i \in Q_B$ with max. g_{c_i}
7 **if** $|Q_A| = |Q_B|$: Choose c_i from $Q_A \cup Q_B$ with max. g_{c_i}, set M_i correspondingly to A or B
8 $Q_{M_i} \Leftarrow Q_{M_i} \setminus \{c_i\}$; $i \Leftarrow i + 1$; $\tilde{g}_i \Leftarrow g_{c_i}$; update the gain lists.
9 **if** $|Q_A| = |Q_B|$: $f_i \Leftarrow 1$, **else**: $f_i \Leftarrow 0$
10 } **while** $(Q_A \cup Q_B \neq \emptyset)$
11 Select $k \in \{1, \ldots, |V|\}$, such that $f_k = 1$ and $\sum_{i=1}^{k} \tilde{g}_i$ maximal.
12 Move the elements $\{c_1, \ldots, c_k\}$ to the other set.
13 } **while** (something was swapped)

On top of Kernighan-Lin we can put *UMDA* to get algorithm 5. Note that *UMDA* does not use the connection structure of the graph to be partitioned.

Algorithm 5: *UMDA* for graph bipartitioning

1 Generate a random population with N individuals. $t \Leftarrow 0$.
2 **do** {
3 Run algorithm 4 for every individual.
4 Select $\hat{N} \leq N$ points. Let $\hat{\mathfrak{X}}^*$ be the best individual.
5 For all $\hat{\mathfrak{X}}^i$: When $\hat{\delta}(\hat{\mathfrak{X}}^*, \hat{\mathfrak{X}}^i) > n/2$: $\hat{\mathfrak{X}}^i \Leftarrow \neg\hat{\mathfrak{X}}^i$.
6 Calculate bit frequencies $p_i(x_i, t)$ from the selected points.
7 Generate new points according to $p(x, t+1) = \prod_{i=1}^{n} p_i(x_i, t)$.
8 $t \Leftarrow t + 1$.
9 } **until** (stopping criterion reached)

The graph bipartitioning problem can be formulated as a quadratic optimization problem (see equation (66)). Therefore another possibility is to solve the mean field equations (61). We are currently evaluating this approach.

10.3 Using LFDA for the graph bipartitioning

Because of the graph structure of the problem, we will use the following modification of *LFDA*. From equation (66) it follows that only variables that are connected in E give rise to a nonlinear term $x_i x_j$. Thus we consider for our Bayesian network BN only those edges which are also edges of the given the graph. This modification makes the implementation a hybrid between *FDA* and *LFDA*. While learning, the list of allowed edges is initialized from the list of edges E instead of the full network. This leads to algorithm 6.

Algorithm 6: *LFDA* for graph bipartitioning

1. Let E be the edges of the graph from the problem definition. Generate a random starting population of N individuals. $t \Leftarrow 0$.
2. **do** {
3. Apply algorithm 4 (Kernighan-Lin with single swaps) to every individual.
4. Select $\hat{N} \leq N$ points. Let $\hat{\mathfrak{X}}^*$ be the best individual.
5. For all $\hat{\mathfrak{X}}^i$: When $\hat{\delta}(\hat{\mathfrak{X}}^*, \hat{\mathfrak{X}}^i) > {}^n/_2$: $\hat{\mathfrak{X}}^i \Leftarrow \neg\hat{\mathfrak{X}}^i$.
6. $F \Leftarrow \{(X_i \to X_j) | (X_i \leftrightarrow X_j) \in E\}$, admissible edges must be contained in the original graph.
7. BN$\Leftarrow \emptyset$.
8. **do** {
9. Choose $(X_i \to X_j) \in F$, such that MDL_α is maximally reduced.
10. BN$\Leftarrow$BN$\cup(X_i \to X_j)$.
11. Remove $(X_i \to X_j)$ and $(X_j \to X_i)$ from F as well as all edges that could introduce a cycle or more than $k_{\max}$ parents.
12. } **while** (there is an edge in F that reduces BIC_α)
13. Calculate a factorization from the graph).
14. Calculate the conditional probabilities $p(x_{b_i}|x_{c_i}, t)$ from the selected points.
15. Generate new points according to $p(x, t+1) = \prod_{i=1}^{l} p(x_{b_i}|x_{c_i}, t)$.
16. $t \Leftarrow t + 1$.
17. } **until** (stopping criterion reached)

In *LFDA* we use a measure which is a tradeoff between goodness of fit and complexity of the model. It has been first proposed by Schwarz [31] as *Bayesian Information Criterion BIC*. Let $M = |D|$ denote the size of the data set. Then

$$BIC_\alpha = -M \cdot H(B, D) - \alpha PA \cdot \log_2(M) \tag{68}$$

$PA = \sum_i 2^{|pa_i|}$ gives the total number of probabilities to compute. pa_i denotes the

parents of node i in the Bayesian network BN. $H(B,D)$ is defined by

$$H(B,D) = -\sum_{i=1}^{n} \sum_{pa_i} \sum_{x_i} \frac{m(x_i,pa_i)}{M} \log_2 \frac{m(x_i,pa_i)}{m(pa_i)} \tag{69}$$

where $m(x_i,pa_i)$ denotes the number of occurrences of x_i given configuration pa_i. $m(pa_i) = \sum_{x_i} m(x_i,pa_i)$. If $pa_i = \emptyset$, then $m(x_i,\emptyset)$ is set to the number of occurrences of x_i in D. Schwartz proposed $\alpha = 0.5$ For a discussion of this and other measures the reader is referred to [20].

11 Benchmark results

There exist at least two other implementations using Bayesian networks to solve the graph bipartitioning problem [28, 29]. But both papers are proof of concepts only. For the test two easy problems have been used. The maximum number of vertices was 144. We decided to test the algorithm on state-of-the-art benchmarks defined by the community (`ftp://dimacs.rutgers.edu/pub/dsj/partition/`).

In [18], the best results so far for a benchmark suite have been published. The authors used a genetic algorithm extended by the Kernighan-Lin local search. Furthermore they compared the results with several other algorithms, among them a multi-start and an iterated Kernighan-Lin. Most difficult was the class of randomly generated graphs G$n.p$. In this class, for a given n and p a random graph was generated having n edges and edges with a probability of p. For several values of n and p an instance was generated and made available for download.

Algo.	N	Avg	σ	FE	Mx	Algo.	N	Avg	σ	FE	Mx
	G500.005 (49)						**G500.01** (218)				
DG		52.0	0.26	15k	0	*DG*		219.3	0.84	9k	5
IKL		55.8	2.11	26k	0	*IKL*		229.7	5.21	19k	0
MA-GX	40	**49.1**	0.37	50k	29	*MA-GX*	40	218.1	0.51	38k	28
M50	40	50.9	0.40	28k	1	*M200*	40	**218.0**	0.00	19k	30
Kim	50	50.4		26k		*Kim*	50	218.0		29k	
UMDA	40	**49.4**	0.57	34k	18	*UMDA*	100	**218.0**	0.0	2k	30
	G500.02 (626)						**G500.04** (1744)				
DG		627.8	1.45	6k	8	*DG*		1747.1	2.12	3k	3
IKL		638.8	4.26	13k	0	*IKL*		1763.8	8.67	7.0k	0
MA-GX	40	627.5	1.14	25k	6	*MA-GX*	40	1745.4	1.50	13k	15
M50	40	626.7	0.71	14k	13	*M50*	40	1745.3	1.54	7.8k	12
Kim	50	626.9		29k		*Kim*	50	1745.6		32k	
UMDA	40	**626.0**	0.18	6k	29	*UMDA*	40	**1744.0**	0.00	2k	30

Table 4: Results for graphs with 500 variables. N is the population size, Avg the average cut size, σ the standard deviation, FE the number of evaluations and Mx counts how often the maximum was found in 30 runs.

Algo.	N	Avg	σ	FE	Mx	Algo.	N	Avg	σ	FE	Mx
	G1000.0025 (93)						**G1000.05** (445)				
DG		101.4	1.45	$11k$	0	*DG*		459.9	2.23	$7k$	0
IKL		99.5	2.87	$10k$	0	*IKL*		452.9	4.09	$7k$	0
MA-GX	40	**94.5**	1.33	$41k$	10	*MA-GX*	40	**447.7**	0.99	$32k$	2
M100	40	96.3	1.03	$24k$	0	*M150*	40	448.9	1.48	$14k$	0
Kim	50	96.2		$28k$		*Kim*	50	449.5		$35k$	
UMDA	40	**95.3**	0.75	$40k$	1	*UMDA*	40	449.1	1.26	$30k$	0
LFDA	100	**94.8**	0.70	$41k$	1	*LFDA*	100	**447.8**	1.53	$29k$	4
	G1000.01 (1362)						**G1000.02** (3382)				
DG		1378.1	2.62	$4k$	0	*DG*		3397.5	5.17	$2.0k$	0
IKL		1370.8	4.66	$4k$	2	*IKL*		3399.5	14.73	$2.3k$	1
MA-GX	40	**1363.1**	1.04	$21k$	9	*MA-GX*	40	**3384.0**	0.49	$11.4k$	0
M150	40	1364.6	2.75	$8k$	7	*M150*	40	**3383.2**	0.81	$4.4k$	6
Kim	50	1364.4		$42k$		*Kim*	50	3384.5		$40.1k$	
UMDA	40	1363.8	1.03	$19k$	1	*UMDA*	100	**3384.1**	0.80	$8.0k$	2
LFDA	100	**1362.7**	0.70	$21k$	13	*LFDA*	100	**3383.6**	0.90	$7.8k$	5

Table 5: Results for graphs with 1000 variables, see table 4 for the column descriptions.

The algorithm *DG* is a diff-greedy algorithm and *IKL* an iterated Kernighan-Lin local search. We include two different *memetic* algorithms (*MA-GX* and *Mx*), all results from [18]. There 10 different memetic algorithms were introduced. Four of these use strong mutation, namely *M50* to *M200*. The number is a parameter of the algorithm, the mutation rate. Of these four the best result is shown for every problem. The worst result was in almost all cases considerably worse, so the mutation rate is a critical parameter for this algorithm. From the other algorithms of the paper, MA-GX was the best one. *Kim* is the algorithm from [12], standard deviation was not published in this paper.

Shown are the best known solutions in parentheses after the problem name, the population size N for population based algorithms, the average best cut size for 30 runs, the standard deviation of the average, the average number of function evaluations and how often the best known solution was found. In bold face are the fitness values of the best algorithm and of those algorithms where the difference to the best is within one standard deviation.

A stochastic algorithm gets better performance by running for a longer time. To be able to compare different algorithms, often the run time in seconds is used. The algorithms in [18] had 60 seconds to optimize the problems with 500 bits and 120 seconds for those with 1000 bits. It turned out that the diff-greedy and the iterated Kernighan-Lin performed worse when given the same amount of CPU time.

As the CPU time depends also on the hardware used, we have used a different criterion to compare the results to *UMDA* and *LFDA*, namely the number of function evaluations. One function evaluations corresponds to a complete run of the Kernighan-Lin algorithm. This is fair, as both memetic algorithms from [18] and the one from [12] use the same principle, they only differ in the method of recombination. As a comparison: *UMDA* took for the 500 bit problems 100-200 seconds and for the 1000 bit problems

roughly 500 seconds, *LFDA* needed 1000 seconds for the 1000 bit problems. *UMDA* and *LFDA* were stopped when a given number of function evaluations was reached. They were never given more evaluations than *MA-GX*.

For the 500 bit problems all population based algorithms give similar results. The simple iterated algorithms are considerably worse than the memetic ones and *UMDA* and *LFDA*. But the size of these problems seems to be too small to show a big difference between the algorithms. *UMDA* is only in one case not the best, but still within one standard deviation. In two cases *UMDA* gave the best results with the same number of function evaluations. *LFDA* is not needed for these problems.

The problems with 1000 bits are more difficult. Both simple search algorithms (DG,IKL) get considerably worse. Algorithm KIM of [12] performs worst in the class of the sophisticated population search methods. *LFDA* and *MA-GX* gave the best results and differ only slightly. But *UMDA* has good results in two cases as well.

LFDA has an overall good performance with bigger computational effort. In [18] the authors mention the importance of interactions: *"'Gene interaction in a given representation can be expressed by a dependency graph... We think that the structure of the dependency graph may have a large impact on the fitness landscape."'* But they do not use this property in their algorithms. The property has been exploited by *LFDA*.

To see the number of edges that were used in a typical *LFDA* run, consider table 6. For the $G1000.01$ problem edges were chosen with a probability of .01. This resulted in a graph with 5064 edges. The table shows the actual number of edges learned by the *LFDA* in a typical run. The number of edges of the Bayesian network is much less than in the given graph. As the population converges, less and less edges are needed to describe the probability structure of the search population.

Gen	1	2	3	4	5	6	7	8	9	10
Edges	984	807	514	362	250	169	128	95	72	56

Table 6: Number of edges of BN out of 5064 used for the **G1000.01** problem.

Both *UMDA* and *LFDA* were not adopted for the problem, only the local search was added. The local search is essential for all population based search methods. The algorithms of [18] and [12] were specifically written for the graph bipartitioning problem. The performance of *UMDA* in conjunction with local optimization using Kernighan-Lin is surprisingly good.

12 Conclusion

We have presented a theory of population based optimization methods using search distributions. We have proven convergence to the global optima for the Factorized Distribution Algorithm *FDA* if the search distribution is a Boltzmann distribution. Convergence has been defined in a strong sense – the limit distribution of *FDA* consists of the distribution of the global optima. *FDA* converges in polynomial time if the search distribution

can be factored so that the number of parameters used is polynomially bounded in n. A general distribution has 2^n parameters.

If *FDA* is used without a Bayesian hyper parameter, then for convergence to the limit distribution the population has to be large enough. The problem of estimating this critical population size can be reduced by using a Bayesian hyper parameter. We have computed upper bounds for Bayesian hyper parameters. They are derived from the constraint that the attractors generate the optima with high probability (e.g $P_s^* > 0.3$). Furthermore we have presented an adaptive annealing schedule for Boltzmann selection.

Thus the mathematical theory is on a solid foundation for optimization problems where the Boltzmann distribution can be exactly factorized using a polynomial number of parameters. The research questions left are connected to finding good approximations for the Boltzmann distribution in the general case. We have shown the relation of our approach to methods used in probabilistic reasoning and statistical physics.

The theory presented here can be extended to general dynamic systems. Whereas in optimization problems we can restrict the search distribution to a Boltzmann distribution, we have to deal in dynamic systems with general time varying distributions.

References

[1] Th. Bäck and H.-P. Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, 1:1–24, 1993.

[2] G.F. Cooper and E.A. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

[3] R. Etxeberria and P. Larrañaga. Global optimization using Bayesian networks. In A.A.O. Rodriguez, M.R.S. Ortiz, and R.S. Hermida, editors, *Proc. of the Second Symposium on Artificial Intelligence Adaptive Systems*, pages 332–339, ICIMAF, La Habana, Cuba, 1999.

[4] C.M. Fiduccia and R.M. Mattheyses. A linear-time heuristic for improving network partitions. In *Proceedings of the Nineteenth ACM/IEEE Design Automation Conference*, pages 175–181, IEEE Computer Society PressPiscataway, New Jersey, 1982.

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

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

[7] D.E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. In G. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 69–93, Morgan Kaufmann, San Mateo, 1991.

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

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

[10] M.I. Jordan, Z. Ghahramani, T.S. Jaakkola, and L.K. Saul. An introduction to variational methods for graphical models. In Jordan [9], pages 105–162.

[11] B.W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 2:291–307, 1970.

[12] Y.-H. Kim and B.-R. Moon. A hybrid genetic search for graph partitioning based on lock gain. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2000*, pages 167–174, Morgan Kaufmann, San Francisco, 2000.

[13] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[14] P. Larrañaga, R. Etxeberria, J.A. Lozano, and J.M. Peña. Combinatorial optimization by learning and simulation of Bayesian networks. In C. Boutilier and M. Goldszmidt, editors, *Procedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 343–352, Morgan Kaufmann, Stanford, 2000.

[15] P. Larrañaga and J.A. Lozano. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Optimization.* Kluwer Academic Press, Boston, 2001.

[16] St. L. Lauritzen. *Graphical Models.* Clarendon Press, Oxford, 1996.

[17] Th. Mahnig and H. Mühlenbein. A new adaptive Boltzmann selection schedule SDS. In *Proceedings of the 2001 Congress on Evolutionary Computation*, pages 183–190. IEEE Press, 2001.

[18] P. Merz and B. Freisleben. Fitness landscapes, memetic algorithms, and greedy operators for graph bipartitioning. *Evolutionary Computation*, 8(1):61–91, 2000.

[19] H. Mühlenbein. The equation for response to selection and its use for prediction. *Evolutionary Computation*, 5(3):303–346, 1998.

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

[21] H. Mühlenbein and Th. Mahnig. Evolutionary algorithms: From recombination to search distributions. In L. Kallel, B. Naudts, and A. Rogers, editors, *Theoretical Aspects of Evolutionary Computing*, Natural Computing, pages 137–176, Berlin, 2000. Springer Verlag.

[22] H. Mühlenbein and Th. Mahnig. Evolutionary computation and beyond. In Y. Uesaka, P. Kanerva, and H. Asoh, editors, *Foundations of Real-World Intelligence*, pages 123–188. CSLI Publications, Stanford, California, 2001.

[23] H. Mühlenbein and Th. Mahnig. Evolutionary computation and Wright's equation. *Theoretical Computer Science*, to be published, 2002.

[24] H. Mühlenbein, Th. Mahnig, and A. Rodriguez Ochoa. Schemata, distributions and graphical models in evolutionary optimization. *Journal of Heuristics*, 5(2):215–247, 1999.

[25] M. Opper and D. Saad, editors. *Advanced Mean Field Methods*, MIT Press, Cambridge, 2001.

[26] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman, San Mateo, 1988.

[27] M. Pelikan, D.E. Goldberg, and E. Cantú-Paz. BOA: The Bayesian Optimization Algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99*, pages 525–532, Morgan Kaufmann, San Francisco, 1999.

[28] M. Pelikan, D.E. Goldberg, and K.Sastry. Bayesian Optimization Algorithm, decision graphs, and Occam's razor. In L. Spector and al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2001*, pages 519–526, San Morgan Kaufmann, Francisco, 2001.

[29] J.M. Peña, J.A. Lozano, and P. Larrañga. Benefits of data clustering in multimodal function optimization via edas. In *Estimation of Distribution Algorithms* [15], pages 99–124.

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

[31] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464, 1978.

[32] N.G. van Kampen. *Stochastic Processes in Physiscs and Chemistry*. Elsevier, Amsterdam, 1992.

[33] G. von Laszewski and H. Mühlenbein. A parallel genetic algorithm for the graph partitioning problem. In R. Maenner and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature*, Lecture Notes in Computer Science 496, pages 165–169. Springer-Verlag, 1991.

[34] J. von Neumann. The general and logical theory of automata. In *The world of mathematics*, pages 2070–2101. Simon and Schuster, New York, 1954.