Evolutionary Optimization using Graphical Models

Heinz Mühlenbein and Thilo Mahnig

Theoretical Foundation GMD Lab.
Real World Computing Partnership
GMD FZ Informationstechnik
53754 St. Augustin
{muehlenbein,mahnig}@gmd.de

Received 30 August 1999

Abstract
We have previously shown that a genetic algorithm can be approximated by an evolutionary algorithm using the product of univariate marginal distributions of selected points as search distribution. This algorithm (UMDA) successfully optimizes difficult multi-modal optimization problems. For correlated fitness landscapes more complex factorizations of the search distribution have to be used. These factorizations are used by the Factorized Distribution Algorithm FDA. In this paper we extend FDA to an algorithm which computes a factorization from the data. The factorization can be represented by a Bayes network. The Bayes network is used to generate the search points.

Keywords Genetic Algorithm, Distribution, Bayesian Network, Factorization of Distribution, Boltzmann Distribution

§1 Introduction

Simulating evolution as seen in Nature has been identified as one of the key computing paradigms for the next decade. Today evolutionary algorithms have been successfully used in a number of applications. These include discrete

[*0] Published in: *New Generation Computing*, 18.2, 2000, pp. 157–166

and continuous optimization problems, synthesis of neural networks, synthesis of computer programs from examples (also called genetic programming) and even evolvable hardware. But in all application areas problems have been encountered where evolutionary algorithms performed badly. Therefore a mathematical theory of evolutionary algorithms is urgently needed. Theoretical research has evolved from two opposite ends; from the theoretical approach there are theories emerging that are getting closer to practice; from the applied side ad hoc theories have arisen that often lack theoretical justification.

Part of our RWC related research was dedicated to a mathematical analysis of evolutionary algorithms for optimization. This analysis has lead to a simplification of genetic algorithms. The corresponding algorithm is called the univariate marginal distribution algorithm UMDA. It uses search distributions instead of the usual recombination/crossover of genetic strings[7]. UMDA has been extended to the factorized distribution algorithm FDA which solves additively decomposed problems where UMDA and genetic algorithms perform badly[8].

The outline of the paper is as follows. We first introduce UMDA and its theoretical analysis. Then FDA is shortly discussed. The main part is dedicated to LFDA. This algorithm computes the factorization from the data.

§2 From Recombination to Distributions

For notational simplicity we restrict the discussion to binary variables $x_i \in \{0, 1\}$. We use the following conventions. Capital letters X_i denote variables, small letters x_i assignments. Let $\mathbf{x} = (x_1, \dots, x_n)$ denote a binary vector. Let a function $f : \boldsymbol{X} \to R^{\geq 0}$ be given. We consider the optimization problem $\mathbf{x}_{opt} = argmax f(\mathbf{x})$. The optimization is done with a set of points called *population*.

Definition 2.1
Let $p(\mathbf{x}, t)$ denote the probability of $\mathbf{x}$ in the population at generation t. Then $p(x_i, t) = \sum_{\mathbf{x}, X_i = x_i} p(\mathbf{x}, t)$ defines the univariate marginal distributions.

$p(x_i, t)$ depends only implicitly on t. We will generally write $p(x_i)$. If different generations are involved, we will denote it by $p(x_i, t)$.
We have shown that any genetic algorithm can be approximated by an algorithm using univariate marginal distributions distributions only[7].

UMDA

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

The simple genetic algorithm uses fitness proportionate selection[4]. In this case the probabilities of the selected points are given by $p^s(\mathbf{x}, t) = p(\mathbf{x}, t) f(\mathbf{x}) / \bar{f}(t)$ where $\bar{f}(t) = \sum_{\mathbf{x}} p(\mathbf{x}, t) f(\mathbf{x})$ denotes the average fitness of the population. For the theoretical analysis we consider $\bar{f}(t)$ to dependent on $p(x_i)$. In order to emphasize this dependency we write

$$W(p(x_1 = 0), p(x1 = 1), \ldots, p(x_n = 1)) := \bar{f}(t) \tag{1}$$

For infinite populations the dynamics of UMDA leads to a deterministic difference equations for $p(x_i)$[7].

$$p(x_i, t+1) = p(x_i, t) \frac{\bar{f}_i(t))}{W} \tag{2}$$

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

$$p(x_i, t+1) = p(x_i, t) \frac{\frac{\partial W}{\partial p(x_i)}}{W} \tag{3}$$

Equation 3 shows that UMDA performs a gradient ascent in the landscape given by W. Despite its simplicity UMDA can optimize difficult multi-modal functions. We discuss only the function BigJump. It is defined as follows, with $|\mathbf{x}|_1 = \sum x_i$ equal to the number of 1-bits:

$$\mathrm{BigJump}(n, m, k, \mathbf{x}) := \begin{cases} |\mathbf{x}|_1 & 0 \leq |\mathbf{x}|_1 \leq n - m \\ 0 & n - m < |\mathbf{x}|_1 < n \\ k \cdot n & |\mathbf{x}|_1 = n \end{cases} \tag{4}$$

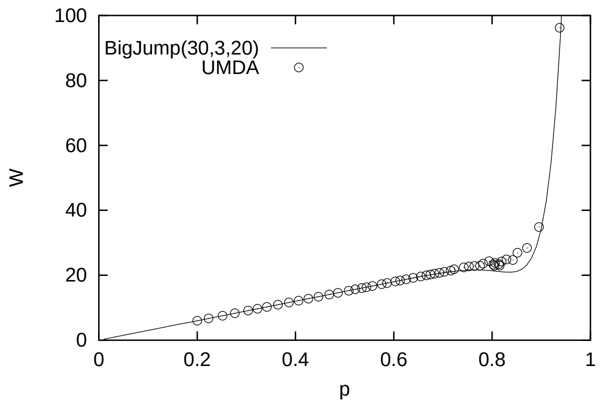

Fig. 1 BigJump(30,3,20), UMDA, p versus average fitness, Popsize=2000

The bigger m, the wider the valley. k can be increased to give bigger weight to the maximum. We have BigJump$(n, 1, 1)$ = OneMax.

BigJump depends only on the number of 1-bits of the genetic string. Therefore we make the assumption that all $p(x_i = 1)$ are identical to a single value denoted as p. Then W depends on a single parameter only. $W(p)$ is shown in Figure 1. $W(p)$ is smooth, it has only a small valley where BigJump has a deep canyon. The open circles are the values of p determined by an UMDA rum. One can see that the probability p changes little when $W(p)$ increases only slightly.

This simple example demonstrates in a nutshell the results of our theory. *Evolutionary algorithms transform the fitness landscape given by $f(\mathbf{x})$ into a fitness landscape defined by $W(\mathbf{p})$. This transformation often smoothes the rugged fitness landscape $f(\mathbf{x})$. There exist difficult multi-modal fitness landscapes $f(\mathbf{x})$ which are transformed into unimodal landscapes $W(p)$. In the $W(p)$ landscapes UMDA performs a gradient ascent.*

But there exist many optimization problems where UMDA is mislead. A well known example are deceptive functions introduced by Goldberg[4)]. Genetic algorithms and UMDA will converge to local optima, because they cannot detect correlations between the variables. This problem can be solved by using higher order distributions for the factorization. This is discussed next.

§3 FDA – The Factorized Distribution Algorithm

For Boltzmann selection, also called exponential proportionate selection, we have proven a factorization theorem for the distribution $p(\mathbf{x},t)$ and convergence for an algorithm using this factorization[8)].

Theorem 3.1
Let $p(\mathbf{x},0)$ be randomly distributed. Let $\beta_1,\ldots,\beta_{t-1}$ be the schedule of the inverse temperature for Boltzmann selection. Then the distribution is given by

$$p(\mathbf{x},t) = \frac{e^{\beta f(\mathbf{x})}}{Z_\beta} \tag{5}$$

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

Unfortunately $p(\mathbf{x},t)$ consists of $2^n - 1$ variables. This seems to make this result useless. But it is very easy to prove that each distribution can be factored into a product, where each variable occurs only once on the left side of a conditional marginal distribution.

Definition 3.1
The conditional probability $p(\mathbf{x}|\mathbf{y})$ is defined as follows

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{x},\mathbf{y})}{p(\mathbf{y})} \tag{6}$$

Theorem 3.2 (Bayesian Factorization)
Each probability can be factored into

$$p(\mathbf{x}) = p(x_1)\prod_{i=2}^{n} p(x_i|pa_i) \tag{7}$$

Proof: By definition of conditional probabilities we have

$$p(\mathbf{x}) = p(x_1)\prod_{i=2}^{n} p(x_i|x_1,\cdots,x_{i-1}) \tag{8}$$

Let $pa_i \subset \{x_1,\cdots,x_{i-1}\}$. If x_i and $\{x_1,\cdots,x_{i-1}\}\setminus pa_i$ are conditionally independent given pa_i, we can simplify $p(x_i|x_1,\cdots,x_{i-1}) = p(x_i|pa_i)$. ■

PA_i are called the parents of variable X_i. This factorization defines a directed graph. In the context of graphical models the graph is called a Bayesian network[6]. This factorization is used by the Factorized Distribution Algorithm FDA.

FDA

- **STEP 0:** Set $t \Leftarrow 0$. Generate $N \gg 0$ points randomly.
- **STEP 1:** Selection
- **STEP 2:** Compute the conditional probabilities $p^s(x_i|pa_i, t)$ using the selected points.
- **STEP 3:** Generate a new population according to $p(x, t+1) = \prod_{i=1}^{n} p^s(x_i|pa_i, t)$
- **STEP 4:** If termination criteria is met, FINISH.
- **STEP 5:** Set $t \Leftarrow t+1$. Go to STEP 2.

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

The amount of computation of FDA depends on the size of the population (N) and the number of variables used for the factors. There exist many problems where the size of the factors is bounded by k independent from n. In this case FDA is computationally efficient[8]. If the fitness function is separable, FDA can be mathematically analyzed[9]. But for the function BigJump an exact factorization needs a factor of size n. Then the amount of computation of FDA is exponential in n. We have seen before that for BigJump UMDA will already find the global optimum. Thus an exact factorization is not a necessary condition for convergence. But it is necessary if we want to be sure that the optimum is found.

From a heuristic point of view we would like to find the smallest factorization leading to convergence of FDA to the global optima. But how can we find such a factorization? Part of this problem is solved by an evolutionary algorithm which computes the factorization during the optimization run.

§4 LFDA - Learning a Bayesian Factorization

Computing the structure of a Bayesian network from data is called learning. Learning gives an answer to the question: *Given a population of selected* *points $M(t)$, what is a good Bayesian factorization fitting the data?* The most difficult part of the problem is to define a quality measure also called scoring measure.

A Bayesian network with more arcs fits the data better than one with less arcs. Therefore our scoring metric should give the best score to the minimal Bayesian network which fits the data. It is outside the scope of this paper to discuss this problem in more detail. The interested reader is referred to the two papers by Heckerman and Friedman et al. in[6].

For Bayesian networks two quality measures are most frequently used - the BDe score and the *Minimal Description Length* (MDL) score. We concentrate on the MDL principle. This principle is motivated by universal coding. Suppose we are given a set D of instances, which we would like to store. Naturally, we would like to conserve space and save a compressed version of D. One way of compressing the data is to find a suitable model for D that the encoder can use to produce a compact version of D. In order to recover D we must also store the model used by the encoder to compress D. Thus the total description length is defined as the sum of the length of the compressed version of D and the length of the description of the model. The MDL principle postulates that the optimal model is the one that minimizes the total description length.

We use Bayesian networks as our model. It is defined as a graph with a probability distribution p. Let $M = |D|$ denote the size of the data set. Then MDL is approximately given by[1]

$$MDL(B,D) = -\mathrm{ld}(P(B)) + M \cdot H(B,D) + \frac{1}{2} PA \cdot \mathrm{ld}(M) \quad (9)$$

with $\mathrm{ld}(x) := \log_2(x)$. $P(B)$ denotes the prior probability of network B, $PA = \sum_i 2^{|pa_i|}$ gives the total number of probabilities to compute. $H(B,D)$ is defined by

$$H(B,D) = -\sum_{i=1}^{n} \sum_{pa_i} \sum_{x_i} \frac{m(x_i,pa_i)}{M} \mathrm{ld} \frac{m(x_i,pa_i)}{m(pa_i)} \quad (10)$$

where $m(x_i, pa_i)$ denotes the number of occurrences of x_i given configuration pa_i. $m(pa_i) = \sum_{x_i} m(x_i, pa_i)$. If $pa_i = \emptyset$, then $m(x_i, \emptyset)$ is set to the number of occurrences of x_i in D.

The formula has an interpretation which can be easily understood. If no prior information is available, $P(B)$ is identical for all possible networks. For minimizing, this term can be left out. $0.5PA \cdot \mathrm{ld}(M)$ is the length required to code the parameter of the model with precision 1/N. Normally one would need $PA \cdot \mathrm{ld}(M)$ bits to encode the parameters. However, the central limit theorem says that these frequencies are roughly normally distributed with a variance of $N^{-1/2}$. Hence, the higher $0.5\mathrm{ld}(M)$ bits are not very useful and can be left out. $M \cdot H(B,D)$ has two interpretations. First, it is identical to the logarithm of the maximum likelihood ($\mathrm{ld}(L(B:D))$). Thus we arrive at the following principle:

Choose the model which maximizes $\mathrm{ld}(L(B:D) - 0.5PA \cdot ld(M)$.

Second, $H(B,D)$ is the conditional entropy of the network structure B, defined by PA_i and the data D. The above formula is appealing, because it has no parameter to be tuned. But the formula has been derived under many simplifications. In practice, one needs more control about the quality vs. complexity tradeoff. Therefore we use a weight factor α.

$$BIC(B,D,\alpha) = -M \cdot H(B,D) - \alpha PA \cdot \mathrm{ld}(M) \qquad (11)$$

This measure with $\alpha = 0.5$ has been first derived by Schwarz[12] as *Bayesian Information Criterion.* The problem of how to control α has been intensively studied by Zhang and Mühlenbein in the context for neural networks[13].

To compute a network B^* which maximizes BIC requires a search through the space of all Bayesian networks. Such a search is more expensive than to search for the optima of the function. Therefore the following greedy algorithm has been used. k_{max} is the maximum number of incoming edges allowed for each node.

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(α) 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 will 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.

The BOA algorithm of Pelikan[11] uses the BDe score. This measure has the following drawback. It is more sensitive to coincidental correlations implied by the data than the MDL measure. As a consequence, the BDe measure will prefer network structures with more arcs over simpler networks[1]. The BIC measure with $\alpha = 1$ has also been proposed by Harik[5]. But Harik allows only factorizations without conditional distributions. This restricts the fitness functions to separable functions. Simple tree structures for the search distribution have been used by Baluja[2] and De Bonet et al.[3].

Given the BIC score we have several options to extend FDA to LFDA which learns a factorization. Due to limitations of space we can only show results of an algorithm which computes a Bayesian network at each generation using algorithm $BN(\alpha, k_{max})$. FDA and LFDA should behave fairly similar, if LFDA computes factorizations which are in probability terms very similar to the FDA factorization. FDA uses the same factorization for all generations, whereas LFDA computes a new factorization at each step which depends on the given data M.

We have applied LFDA to a number of problems. Here we only discuss the functions OneMax, BigJump and Deceptive-4[11]. Additional examples can be found in[10].

Function	n	α	N	τ	Succ.%	SDev
OneMax	30	UMDA	30	0.3	75	4.3
	30	0.25	100	0.3	2	1.4
	30	0.5	100	0.3	38	4.9
	30	0.75	100	0.3	80	4.0
	30	0.25	200	0.3	71	4.5
BigJump(30,3,1)	30	UMDA	200	0.3	100	0.0
	30	0.25	200	0.3	58	4.9
	30	0.5	200	0.3	96	2.0
	30	0.75	200	0.3	100	0.0
	30	0.25	400	0.3	100	0.0
Deceptive-4	32	UMDA	800	0.3	0	0.0
	32	FDA	100	0.3	81	3.9
	32	0.25	800	0.3	92	2.7
	32	0.5	800	0.3	72	4.5
	32	0.75	800	0.3	12	3.2

Table 1 Numerical results for different algorithms, LFDA with $BN(\alpha, 8)$

Table 1 summarizes the results. For LFDA we used three different values of α, namely $\alpha = 0.25, 0.5, 0.75$. The smaller α, the less penalty for the size of the structure. Let us discuss the results in more detail. $\alpha = 0.25$ gives by far the best results when a network with many arcs is needed. This is the case for Deceptive-4. Here a Bayesian network with three parents is optimal. $\alpha = 0.25$ performs bad on problems where a network with no arcs defines a good search distribution. For the linear function OneMax $BIC(0.25)$ has only a success rate of 2%. The success rate can be improved if a larger population size N is used. The reason is as follows. $BIC(0.25)$ allows denser networks. But if a small population is used, spurious correlations may arise. These correlations have a negative impact for the search distribution. The problem can be solved by using a larger population. Increasing the value from $N = 100$ to $N = 200$ increases the success rate from 2% to 71% for OneMax.

For Bigjump a Bayesian network with no arcs is able to generate the optimum. An exact factorization requires a factor with n parameters. We used the heuristic BN with $k_{max} = 8$. Therefore the exact factorization cannot be found. Therefore $\alpha = 0.75$ gives the best results. $BIC(0.75)$ enforces smaller networks. But $BIC(0.75)$ performs very bad on Deceptive-4. Taking all results together, $BIC(0.5)$ gives good results. This result supports the BIC estimate from Schwarz.

§5 Conclusion

Our theory of genetic algorithms has lead to the design of UMDA which uses search distributions instead of recombination of strings. UMDA is able to solve difficult multi-modal optimization problems. This result partly explains the success of genetic algorithms. But UMDA performs badly for functions with strongly correlated variables or if the transformed fitness landscape $W(p)$ has many local minima. Therefore we extended UMDA to the Factorized Distribution Algorithm FDA, which uses more general factorizations of the search distribution.

The theory of FDA uses results from simulated annealing and graphical models. The extension of FDA to an algorithm LFDA, which computes an approximate factorization from the data was done by adapting techniques developed for learning the structure of Bayesian networks to our application.

The theory presented covers discrete optimization problems without constraints. It can be extended to optimization problems with constraints. But in order to give good results, constraints and the fitness function have to be compatible[8]. The theory can also applied to continuous optimization problems.

Application of the theory to genetic programming is also possible.

References

1) Bouckaert, R.R. (1994). Properties of Bayesian network learning algorithms. In R. Lopez de Mantaras and D. Poole, (Eds.) *Proc. Tenth Conference on Uncertainty in Artificial Intelligence* pp. 102-109. San Francisco: Morgan Kaufmann.

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

3) De Bonet, J.S. & Isbell, Ch. L. & Viola, P. (1997). MIMIC: Finding Optima by Estimating Probability Densities. In Mozer,M. & Jordan, M. & Petsche, Th. (Eds) *Advances in Neural Information Processing Systems 9* pp. 424–431

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

5) Harik, G. (1999). *Linkage Learning via probabilistic Modeling in the ECGA* IlliGal Technical Report 99010, University of Illinois at Urbana-Champaign.

6) Jordan, M.I.(ed.) (1999). *Learning in Graphical Models* Cambridge:MIT Press.

7) Mühlenbein, H. (1997). The Equation for Response to Selection and its Use for Prediction. *Evolutionary Computation*, 5:pp. 303-346.

8) Mühlenbein, H. & Mahnig, Th. & Ochoa, R. (1999a). Schemata, Distributions and Graphical Models in Evolutionary Optimization. *Journal of Heuristics*, 5,215-247.

9) Mühlenbein, H. & Mahnig, Th.(1999b). Convergence Theory and Applications of the Factorized Distribution Algorithm. *Journal of Computing and Information Technology* 7:pp 19–32.

10) Mühlenbein, H. & Mahnig, Th.(1999c). FDA – A Scalable Evolutionary Algorithm for the Optimization of Additively Decomposed discrete functions. *to appear in Evolutionary Computation 7.4.*

11) Pelikan, M. & Goldberg, D.E. & Cantu-Paz, E. (1999). *BOA: The Bayesian optimization algorithm.* IlliGAL Technical Report 99003, University of Illinois at Urbana-Champaign.

12) Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics*, 7:pp. 461–464.

13) Zhang, B.-T. & Ohm, P. & Mühlenbein, H. (1997). Evolutionary Induction of Sparse Neural Trees, *Evolutionary Computation*, 5:pp. 213–236.