A Markov Network based Factorized Distribution Algorithm for optimization

Roberto Santana
Institute of Cybernetics, Mathematics, and Physics (ICIMAF)
Calle 15, e/ C y D, Vedado
CP-10400, La Habana, Cuba
rsantana@cidet.icmf.inf.cu

Abstract- In this paper we propose a population based optimization method that uses the estimation of probability distributions. To represent an approximate factorization of the probability, the algorithm employs a junction graph constructed from an independence graph. We show that the algorithm is able to extend the representation capabilities of previous algorithms that use factorizations. A number of functions are used to evaluate the performance of our proposal. The results of the experiments show that the algorithm is able to optimize the functions, and it overperforms other evolutionary algorithms that use factorizations.
Keywords. Genetic algorithms, EDA, FDA, evolutionary optimization, estimation of distributions.

1 Introduction

In the application of Genetic Algorithms (GAs) [8, 6] to a wide class of optimization problems is essential the identification and mixing of building blocks. It has been early noticed that the Simple GA (SGA) is in general unable to accomplish these two tasks for difficult problems (e.g. deceptive problems). Perturbation techniques, linkage learners and model building algorithms are among the alternatives proposed to improve GAs. They try to identify the relevant interactions among the variables of the problem, and to use them in an efficient way to search for solutions.

Model building techniques refer to GAs that construct a probabilistic model of the solutions instead of the crossover operator. In this paper we use the term Estimation Distribution Algorithms (EDAs) [16] to call this type of algorithms. These algorithms construct in each generation a probabilistic model of the selected solutions. The probabilistic model must be able to capture a number of relevant relationships in the form of statistical dependencies among the variables. Dependencies are then used to generate solutions during a sampling step.

It is expected that the generated solutions share a number of characteristics with the selected ones. In this way the search is led to promising areas of the search space. EDAs are also known as Iterated Density Estimators Evolutionary Algorithms [1], and Probabilistic Model Building Genetic Algorithms [20]. The interested reader is referred to [9] for a good survey that covers the theory, and a wide spectrum of EDAs applications.

One efficient way of estimating a probability distribution is by means of factorizations. A probability distribution is factorized when it can be computed by a small number of factors. A subclass of EDAs will group the algorithms that use factorizations of the probability distribution. In this paper we call to this subclass Factorized Distribution Algorithms (FDAs)[1] [15].

FDAs have outperformed other evolutionary algorithms in the optimization of complex additive functions, and deceptive problems with overlapping variables [15]. They have been recently applied as well for the solution of the bipartitioning [14], and satisfiability problems [18]. However, a shortcoming of FDAs is that the probabilistic model they are based on is constrained to represent a limited number of interactions. In this paper we investigate the issue of extending the representation capabilities of FDAs. To this end we introduce the Markov Network FDA (MN-FDA), a new type of FDA based on an undirected graphical model, and able to represent the so called "invalid" factorizations [15].

The paper is organized as follows. In section 2 we discuss the problem of obtaining a factorization of the probability. Section 3 presents the main steps for learning an approximate factorization from data. Section 4 explains the way the sampling step has been implemented. We introduce the MN-FDA in section 5. Section 6 presents the functions used in our experiments. We discuss the numerical results of the simulation. Section 7 analyzes the MN-FDA in the context of recent related research on evolutionary computation, we also present in this section the conclusions of our paper.

[1]In the literature the term FDA is frequently used to name a particular type of Factorized Distribution Algorithms. Our definition covers it, and other algorithms that use factorizations.

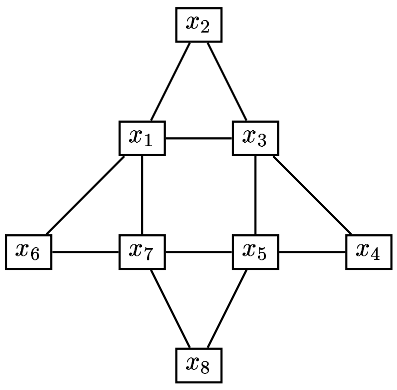

Figure 1: Independence graph

2 Factorization of a probability

The central problem of FDAs is how to efficiently estimate a factorization of the joint probability of the selected individuals. To compute a factorization the theory of graphical models is usually employed. One example of graphical models are Bayesian networks, where the dependencies relationships between the variables of the problem are represented using directed graphs. A number of Bayesian FDAs have been proposed in the literature [5, 19, 13]. We will focus on another type of graphical representation based on undirected graphs. The following definitions will help in the explanation of our proposal.

Let $X = (X_1, X_2, \cdots, X_n)$ represent a vector of integer random variables, where n is the number of variables of the problem. $x = (x_1, x_2, \cdots, x_n)$ is an assignment to the variables, and $p(x)$ is a joint probability distribution to be modeled. Each variable of the problem has associated one vertex in an undirected graph $G = (V, E)$. The graph G is a conditional independence graph respect to $p(x)$ if there is no edge between two vertices whenever the pair of variables is independent given all the remaining variables.

Definition 1. *Given a graph G, a clique in G is a fully connected subset of V. We reserve the letter C to refer to a clique. The collection of all cliques in G is denoted as $\mathcal{C}$. C is maximal when it is not contained in any other clique. C is the maximum clique of the graph if it is the clique in $\mathcal{C}$ with the highest number of vertices.*

Definition 2. *A junction graph (JG) of the independence graph G is a graph where each node corresponds to a maximal clique of G, and there exists an edge between two nodes if their corresponding cliques overlap.*

Definition 3. *A junction tree (JT) is a single connected junction graph. It satisfies that if the variable X_k is a member of the junction tree nodes i and j, then X_k is a member of every node on the path between i and j. This property is called the* running intersection property.

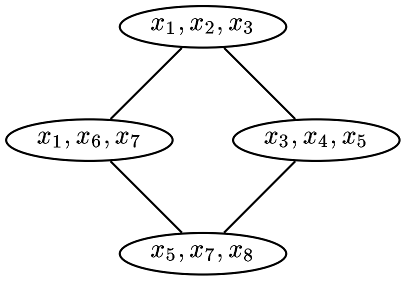

Figure 2: Associated junction graph

Figures 1 and 2 respectively show an example of an independence graph, and its associated junction graph.

If the independence graph G is chordal, an exact factorization of the probability based on the cliques of the graph exists. The factorization can be represented using a JT. If G is not chordal, a chordal supergraph of G can be found by adding edges to G in a process called triangulization. The problem is that we can not guarantee that the maximum clique of the supergraph will have a size that would make feasible the calculation of the marginal probabilities. The problem of finding a triangulization with maximum clique of minimum size is NP-complete.

To obtain an exact factorization of the probability is usually an infeasible task. In such cases we can use approximate factorizations. This is the approach we follow. Our goal is to find an approximate factorization that contains as many dependencies as possible, but without adding new edges to the graph. An exact factorization would comprise all the dependencies represented in the independence graph. We will assume that approximate factorizations of the probability are more precise as they include more of the dependencies represented in the independence graph.

The approximate factorization will be represented using a labeled JG. The algorithm for learning the probabilistic model has four main steps.

Algorithm 1: **Model learning**

```
1  Learn an independence graph $G$ from the data
   (the selected set of solutions).
2  Find the set $L$ of all the maximal cliques of $G$.
3  Construct a labeled $JG$ from $L$.
4  Find the marginal probabilities for the cliques
   in the $JG$.
```

3 Learning of approximate factorization

In this section we consider in detail the different steps for learning a factorization from data.

3.1 Learning of an independence graph

The construction of an independence graph from the data can be accomplished by means of independence tests. To determine if an edge belongs to the graph, it is enough to make an independence test on each pair of variables given the rest. Nevertheless, from an algorithmic point of view it is important to reduce the order of the independence tests. Thus, we have adopted the methodology followed previously by Spirtes [22]. The idea is to start from a complete undirected graph, and then try to remove edges by testing for conditional independence between the linked nodes, but using conditioning sets as small as possible.

To evaluate the independence tests we use the Chi-square independence test. If two variables X_i and X_j are dependent with a 75 percent of significance the corresponding X^2 value of the Chi-square test is used to assign a weight $w(i,j)$ to the edge $i \sim j$. This weight stresses the pairwise interaction between the variables. When the independence graph is known in advance, we assume $w(i,j) = w', \forall i \sim j \in E$. The weight of any subgraph G' of G is calculated as $w(G') = \sum_{i \sim j \in G'} w(i,j)$. In this way the weights of the maximal cliques can be calculated.

3.2 Maximal cliques of graph

To find all the cliques of the graphs the Ken and Kerbash algorithm [2] is used. This algorithm uses a branch and bound technique to cut off branches that can lead to cliques. Its memory requirements are at most $\frac{1}{2} \cdot M \cdot (n+3)$ storage locations to contain arrays of integers, where M is the size of the largest connected component in the input graph [2]. When the undirected graph is very dense, it can be made sparser before the calculation of the cliques of the graph by removing the edges with lower weights.

3.3 Construction of the labeled JG

Algorithm 2 receives the list of cliques L with their weight, and outputs a list L' of the cliques in the JG. The first clique in L' is the root, and the labels of cliques in the JG correspond to their position in the list. Each clique in the JG is a subset of a clique in L.

Algorithm 2: **Algorithm for learning a** JG

```
1  Order the cliques in $L$ decrementally according to their weight.
2  Add element $L(1)$ to list $L'$
3  Remove element $L(1)$ from $L$
4  While $L$ is not empty
5     Find the first element $C$ in $L$ such that the number of variables in $C \cap (L'(1) \cup L'(2) \cdots \cup L'(NCliques)) \neq C$, and $C \cap L'(1) \cap L'(2) \cdots \cap L'(NCliques)$ is maximized
6     If $C = \emptyset$
7        Remove all the elements in $L$
8     else
9        Insert $C$ in $L'$
```

We focus now on step 5 of algorithm 2. The condition of maximizing the number of variables in $C \cap (L'(1) \cup L'(2) \cdots \cup L'(NCliques))$ states that the clique C in L that has the highest number of overlapping variables with all the variables already in L', will be added to L'. The number of overlapping variables has to be less than the size of the clique, constraint meaning that at least one of the variables in C has not appeared before. If there exist many cliques with maximum number of overlapped variables, the one that appears first in L is added to L'. On the other hand, if the maximum number of overlapped variables is zero, then in the JG there exists more than one connected component. In this case we have a set of junction graphs, however we have preferred to abuse the notation and call it JG, whether it has one or more connected components. Finally, the addition of cliques stops when all the variables are already in the JG.

3.4 Calculation of the marginal probabilities

Marginal probabilities are found by calculating the number of counts associated to each configuration, and normalizing. In the implementation, the learned model's parameters can be changed by adding a perturbation in the form of probabilistic priors [10]. The effect of this type of priors is similar to the effect of mutation in GAs. In many problems they improve the behavior of the algorithm. This fact can be explained by the important role played by mutation in avoiding premature convergence, and allowing the exploration of new regions during the search.

4 Sampling of the approximate factorization

To create the new generation of solutions FDAs sample points from the probabilistic model. The MN-FDA sampling algorithm follows the order determined by the labels of the JG. The variables corresponding to the first clique in the JG are instantiated sampling from the marginal probabilities. For the rest of cliques, each subset of variables that has not been instantiated is sampled conditionally on the variables already instantiated that belong to the clique. The process is very similar to Probabilistic Logic Sampling (PLS) [7] when it is used in junction trees. There exists however an important difference. The definition of JT discards the existence of cycles. A labeled JG can contain cycles. This fact allows the representation of more interactions, but it does not essentially change the performance of the sampling algorithm. The reason is that in every step of the JG sampling algorithm, the conditioning and conditioned subsets of variables will belong to the clique whose variables are being sampled.

5 MN-FDA

FDAs based on undirected graphical models represent the factorizations using a JT. The FDA^* [15] uses a fixed model of interactions, only the parameters of the cliques are learned in each generation. The simplest FDA is the Univariate Marginal Distribution Algorithm (UMDA) [11]. The UMDA assumes that all the variables are independent, and in every step it makes a parametric learning of the univariate probabilities.

The FDA-learning [17] is based on a theoretical algorithm [4] that learns a chordal independence graph straight from the data, allowing to change the model structure in each generation. If the underlying probability model is decomposable, the algorithm recovers it, if not it recovers a chordal supergraph. The JT is constructed from the chordal graph found.

Our algorithm will be called Markov Network FDA (MN-FDA), its pseudo-code is presented in algorithm 3. The main difference between it and previous FDAs based on undirected models is that it uses as its probabilistic model a JG, allowing factorizations that have not to be correct.

Analogously to GAs, different replacing strategies can be incorporated to the MN-FDA. Additionally, in the case of proportional selection the learning step can be done straight on the probabilities determined by the selection, without the need of constructing a selected set [21].

Algorithm 3: **MN-FDA**

```
1  Set $t \Leftarrow 0$. Generate $N \gg 0$ points randomly.
2  do {
3      Select a set $S$ of $k \leq N$ points according to
       a selection method.
4      Learn a $JG$ from the data
5      Calculate the marginal probabilities for all
       the cliques in the $JG$.
6      Generate a the new population sampling
       from the $JG$.
7      $t \Leftarrow t + 1$
8  } until Termination criteria are met
```

6 Experiments

In our experiments we compare the behavior of the MN-FDA with other FDAs. First, a number of functions commonly used to evaluate evolutionary algorithms are presented. Also a practical problem used in the experiments is described. All the problems used in the experiments are defined on binary variables. The numerical results and the analysis of the experiments are presented afterwards.

6.1 Functions used in the experiments

Deceptive functions were introduced by Goldberg to show the deceptive nature of the GAs behavior, and to address the problems given by the convergence to local optima of the function. The following 4 elementary deceptive functions of k variables are used to defined some of the additive functions used in our experiments. They are defined in terms of the unitation value $u(x) = \sum_{i=1}^{n} x_i$.

u	0	1	2	3	4	5
f^3_{dec}	0.9	0.8	0	1		
f^4_{dec}	3	2	1	0	4	
$IsoT_1$	m	0	0	0	0	0
$IsoT_2$	m	0	0	0	0	$m-1$

$$Onemax(x) = \sum_{i=1}^{n} x_i \tag{1}$$

$$f_{3deceptive}(x) = \sum_{i=1}^{i=\frac{n}{3}} f^3_{dec}(x_{3i-2}, x_{3i-1}, x_{3i}) \tag{2}$$

$$Deceptive_4(x) = \sum_{i=1}^{i=\frac{n}{4}} f^4_{dec}(x_{4i-3}, x_{4i-2}, x_{4i-1}, x_{4i}) \tag{3}$$

$$F_{IsoP}(n,m,k,x) = \left(\sum_{i=1}^{n} x_i\right) \tag{4}$$
$$+ k \cdot (m+1)((1-x_1)\cdots(1-x_m)x_{m+1}\cdots x_n)$$

OneMax				*BigJump*(30, 3, 1)				*Deceptive*4			
n	*Alg.*	*N*	*succ.*	*n*	*Alg.*	*N*	*succ.*	*n*	*Alg.*	*N*	*succ.*
30	UMDA	30	75	30	UMDA	200	100	32	UMDA	800	0
30	$\text{LFDA}_{0.25}$	100	2	30	$\text{LFDA}_{0.25}$	200	58	32	FDA	100	81
30	$\text{LFDA}_{0.5}$	100	38	30	$\text{LFDA}_{0.5}$	200	96	32	$\text{LFDA}_{0.25}$	800	92
30	$\text{LFDA}_{0.75}$	100	80	30	$\text{LFDA}_{0.75}$	200	100	32	$\text{LFDA}_{0.5}$	800	72
30	$\text{LFDA}_{0.25}$	200	71	30	$\text{LFDA}_{0.25}$	400	100	32	$\text{LFDA}_{0.75}$	800	12
30	MN-FDA	30	72	30	MN-FDA	100	92	32	MN-FDA	600	90
30	MN-FDA	100	98	30	MN-FDA	200	100	32	MN-FDA	800	100

Table 1: Comparison between the MN-FDA with the UMDA and the LFDA for different functions

When analyzing interactions between variables it is important to consider interactions that do not depend on the linear codification of solutions. To this end we considered function $F_{IsoTorus}$ (5) where x_{up}, x_{left}, etc., are defined as the appropriate neighbors, wrapping around.

$$F_{IsoTorus}(x) = IsoT_1(x_{1-m+n}, x_{1-m+n}, x_1, x_2, x_{1+m}) + \sum_{i=2}^{n} IsoT_2(x_{up}, x_{left}, x_i, x_{right}, x_{down}) \quad (5)$$

Function *BigJump* (6) was introduced in [12]. A valley has to be crossed in order to reach the global optimum of this function. The bigger the parameter m is for this function, the wider the valley. k can be increased to give bigger weight to the maximum.

$$BigJump(u, n, m, k) = \begin{cases} u & for \quad 0 \le u \le n - m \\ 0 & for \quad n - m \le u \le m \\ k \cdot n & for \quad u = m \end{cases} \quad (6)$$

The generalized Ising model (7) is described by the energy functional (Hamiltonian) where L is the set of sites called a lattice. Each spin variable σ_i at site $i \in L$ either takes the value 1 or value -1. A specific choice of values for the spin variables is called a configuration. The constants J_{ij} are the interaction coefficients. In our experiments we take $h_i = 0$, $\forall i \in L$. The ground state is the configuration with minimum energy.

$$H = -\sum_{i<j\in L} J_{ij}\sigma_i\sigma_j - \sum_{i\in L} h_i\sigma_i \quad (7)$$

6.2 Numerical results

In the following experiments N is the population size, *succ* is the number of times the optimum was reached in 100 experiments, *gen* the average number of generations to convergence, $\hat{f}$ the average fitness of solutions, and *eval* is the average number of of evaluations needed to find the optimum.

In table 1 results of the MN-FDA for different functions are compared with results published in [12] for the UMDA and the LFDA (A FDA that uses a Bayesian probabilistic model). For the functions considered, the MN-FDA achieved equal or better results than the LFDA. We have observed that the learning algorithm used by the MN-FDA easily detects variables that are independent. The BN learning algorithms used by Bayesian FDAs may have problems recognizing independency, particularly if the parameter that specifies the density of the network (parameter α in the case of the LFDA) is small.

In table 2 we have included the results for the UMDA, the Tree-FDA, and the LFDA for other functions. For function $f_{3deceptive}$ the results of the MN-FDA are the best, and there is a significant difference with the results achieved by the LFDA. For function $F_{IsoTorus}$ LFDA finds the optimum more times than the MN-FDA, however its average fitness is lower. For function F_{IsoP} the Tree-FDA takes advantage of the chain-shaped structure of function F_{IsoP} to achieve the best results. It can be appreciated that the UMDA is not able to solve the problems with interactions.

We have generated 4 random instances of the Ising model for different number of variables ($n \in \{25, 36, 49, 64\}$). For each of the instances we investigate two different issues. First, the influence of using the prior information about the interactions of the variables. MN-FDA^s is a Markov Network FDA that does not learn the independence graph from the data. In this case the lattice where the Ising model is defined serves as the independence graph. The maximum size of the cliques is equal 2. The parameters of the cliques that belong to the JG are learned from the data. The second issue we study is the scaling of the algorithm.

An analysis of the results shows the convenience of using prior information about the optimization problem for increasing the efficiency of the MN-FDA. The

	$f_{3deceptive}$			$F_{IsoTorus}$			F_{IsoP}		
	succ.	$\hat{f}$	*gen*	*succ.*	$\hat{f}$	*gen*	*succ.*	$\hat{f}$	*gen*
MN-FDA	77	11.97	8.0	78	210.78	5.8	69	1190.69	5.0
UMDA	0	0	0	17	175.01	8.7	0	0	0
Tree-FDA	31	11.90	9.3	70	210.70	6.3	99	1190.99	5.8
LFDA	45	11.91	6.4	85	210.55	6.0	76	1190.75	4.6

Table 2: Comparison between the MN-FDA with other FDAs for different functions.

small population size that is enough for the convergence of the MN-FDA^s is not sufficient for the MN-FDA. As expected, when the number of variables increases a higher population size is needed to solve the problem.

Inst.	N	MN-FDA^s		MN-FDA	
		succ.	*eval.*	*succ.*	*eval.*
I^{25}	200	100	849	43	1163
I^{36}	400	86	2316	41	2453
I^{49}	700	82	3841	36	4201
I^{64}	700	67	6031	28	6641

Table 3: Results of the MN-FDA for different Ising instances.

7 Conclusions

In this paper we have presented a FDA that approximates the probability distribution determined by selection using a labeled JG. The JG is found by calculating the maximal cliques of a Markov Network that can be given as an input or learned from the data. Our work is related with previous work by Muehlenbein et al. [15], where approximate factorizations were recognized as an alternative for modeling probabilistic distributions. Our research has led to to a different way of finding these approximations. It is also related with the work presented by Brown et al. [3] in the application of MRFs to GAs. They have used probabilistic models of GA fitness functions to generate new solutions. Our work shows a number of relevant differences with this approach:

1. The use of statistical test to learn the structure of interactions. In [3] the structure of the interactions is known a priori.
2. The construction of the JG from the MN
3. The use of PLS on the JG. In [3] the Metropolis algorithm is used to generate new solutions.

The results of the experiments show that the MN-FDA is able to optimize theoretical functions as well as functions derived from practical problems, overperforming other evolutionary algorithms. The MN-FDA generalizes other FDAs by learning factorizations that have not to be correct. More theoretical investigation is needed to determine bounds for the convergence of the MN-FDA. Other practical optimization problems must be tried to assess the performance of the algorithm.

Bibliography

[1] P. A. N. Bosman and D. Thierens. Linkage information processing in distribution estimation algorithms. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99*, volume I, pages 60–67, Orlando, FL, 1999. Morgan Kaufmann Publishers, San Francisco, CA.

[2] C. Bron and J. Kerbosch. Algorithm 457—finding all cliques of an undirected graph. *Communications of the ACM*, 16(6):575–577, 1973.

[3] D. F. Brown, A. Garmendia-Doal, and J. A. W. McCall. Markov random field modelling of royal road genetic algorithms. In P. Collet, editor, *Proceedings of EA 2001*, volume 2310 of *Lecture Notes in Computer Science*, pages 65–76. Springer Verlag, 2002.

[4] L. M. deCampos and J.F.Huete. Algorithms for learning decomposable models and chordal graphs. Technical Report DECSAI 970213, University of Navarra, 1997.

[5] R. Etxeberria and P. Larrañaga. Global optimization using Bayesian networks. In *Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99)*, pages 151–173, Habana, Cuba, March 1999.

[6] D. E. Goldberg. *Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley, Reading, MA, 1989.

[7] M. Henrion. Propagating uncertainty in Bayesian networks by probabilistic logic sampling. *Uncertainty in Artificial Intelligence*, 2:317–324, 1988.

[8] J. H. Holland. *Adaptation in natural and artificial systems.* University of Michigan Press, Ann Arbor, MI, 1975.

[9] P. Larrañaga and J. A. Lozano. *Estimation Distribution Algorithms. A new tool for Evolutionary Optimization.* Kluwer Academic Publishers, Boston/Dordrecht/London, 2001.

[10] T. Mahnig and H. Mühlenbein. Optimal mutation rate using Bayesian priors for Estimation of Distribution Algorithms. In K. Steinhöfel, editor, *Proceedings of the First Symposium on Stochastic Algorithms: Foundations and Applications, SAGA-2001*, volume 2264 of *Lecture Notes in Computer Science*, pages 33–48. Springer, 2001.

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

[12] H. Mühlenbein and T. Mahnig. *Theoretical Aspects of Evolutionary Computing*, chapter Evolutionary Algorithms: From Recombination to Search Distributions, pages 137–176. Springer Verlag, Berlin, 2000.

[13] H. Mühlenbein and T. Mahnig. Evolutionary synthesis of Bayesian networks for optimization. *Advances in Evolutionary Synthesis of Neural Systems, MIT Press*, pages 429–455, 2001.

[14] H. Mühlenbein and T. Mahnig. Evolutionary optimization and the estimation of search distributions with applications to graph bipartitioning. *International Journal on Approximate Reasoning*, 2002. to appear.

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

[16] H. Mühlenbein and G. Paaß. From recombination of genes to the estimation of distributions I. Binary parameters. In A. Eiben, T. Bäck, M. Shoenauer, and H. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN IV*, pages 178–187, Berlin, 1996. Springer Verlag.

[17] A. Ochoa, M. R. Soto, R. Santana, J. C. Madera, and N. Jorge. The Factorized Distribution Algorithm and the juction tree: A learning perspective. In A. Ochoa, M. R. Soto, and R. Santana, editors, *Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99)*, pages 368–377, Habana, Cuba, March 1999.

[18] M. Pelikan and D. E. Goldberg. Hierarchical BOA solves Ising spin glasses and Max-Sat. IlliGAL Report No. 2003001, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL, January 2003.

[19] 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*, volume I, pages 525–532, Orlando, FL, 1999. Morgan Kaufmann Publishers, San Francisco, CA.

[20] M. Pelikan, D. E. Goldberg, and F. Lobo. A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, 21(1):5–20, 2002.

[21] R. Santana. Factorized distribution algorithms: Selection without selected population. 2003. Submitted for publication.

[22] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction and search.* Lecture Notes in Statistics. Springer-Verlag, New York, 1993.