The Factorized Distribution Algorithm and The Junction Tree: A Learning Perspective

Alberto Ochoa Rodríguez * , Marta Soto , Roberto Santana , Julio Madera , Nancy Jorge

Center of Mathematics and Theoretical Physics
Institute of Cybernetics, Mathematics and Physics Calle 15 No 551 e/ C y D. CP 10400. Havana. Cuba.
ochoa@cidet.icmf.inf.cu

This paper extends the FDA - the Factorized Distribution Algorithm - with a structural learning component. The FDA has been extensively investigated for the optimization of additively decomposed discrete functions (ADFs). Now, we are able to deal with more general problems, which are solved by FDA in a blackbox optimization scenario. The key point is the construction of the Junction Tree, which is placed at the centre of the algorithm. Learning the Junction Tree directly from the data is a process that is accomplished by making independency tests of as lower as possible order. The proposed algorithm belongs to the class of Estimation Distribution Algorithms and represents an interesting alternative to approach the Linkage Problem in Genetic Algorithms.

1. Introduction

The notion of Low Cost Evolutionary Algorithm (**LCEA**) introduced in [19] defines a conceptual framework that joins our efforts to design better evolutionary algorithms, which do less evaluations of better functions with less cost, and in better and less points of the solution space.

An obvious way of reducing the cost of a population based algorithm is to do fewer functions evaluations. If the algorithm is stochastic, we have to be sure of sampling the solution space in an optimal way. To accomplish this, one can use knowledge about the probabilistic dependencies among the problem's variables. In some cases, it is possible to design a good evaluation function, which not only indicates which solutions are good, but also which solutions predictably lead to good solutions. Both methods "intelligently" guide the search guaranteeing a faster convergence to the optimum. Furthermore, having a good sampling algorithm and a good designed function, we can be still more efficient, if we are able to construct a model of the function that can be evaluated with less efforts.

The above ideas can be summarized in the following **LCEA** strategies:

1. Learning and using the probabilistic structure of the problem.
2. Learning "good" Evaluation Functions.
3. Partial Evaluation of the evolutionary algorithm.
4. Use of parallel and distributed techniques.

The claim of the first line of research is that detecting and using the most important interactions among the variables is the key to an efficient sampling of the solution space. Recently, this line has deserved a lot of attention from the evolutionary computation community. In [17], for example, was used the theory of Graphical Models to design the Factorized Distribution Algorithm.

Learning Evaluation Functions investigates how statistical machine learning techniques can be used to automatically generate high-quality evaluation functions for practical combinatorial problems. The data for such a learning is gathered by running trajectories through the search space [6].

The idea of Partial Evaluation [19] is similar to the above. One of its directions investigates how statistical machine learning techniques can use knowledge gathered by evolution trajectories, to estimate the effect of the function value on the next action to be taken by the algorithm. Another approach explores the ways one can compare candidate solutions without evaluate them fully.

* a8ar@mailcity.com

Finally, we can use parallel and distributed architectures to do less functions evaluations, spending more computational time in supporting jobs as learning the Bayesian structure of the problem.

This paper addresses the first **LCEA** strategy: How to use knowledge about the probabilistic structure of a problem to make an evolutionary algorithm sample the search space efficiently. This line of research is tightly related to the Linkage Problem of Genetic Algorithms. Our discussion is centered on the Factorized Distribution Algorithm introduced in [17], but in contrast with that research, here we emphasize the learning dimension of the problem.

2. From the Linkage Problem to the EDA

Recently, the current "crisis" of the Evolutionary Algorithms has been faced by a number of authors with proposals that use Graphical Models. The proposed models range from simple structures like chains of variables [9], to more complex like dependency trees [4], polytree structures [21] and chordal graphs [17]. Chordal representations have the best representation power and are good candidates for efficient, low cost algorithms (**LCEA**), provided they come with good learning methods. All these algorithms are members of the general class of Estimation of Distribution Algorithms **EDA** [16].

One of the main aims behind the development of EDA has been the necessity of designing Evolutionary Algorithms able to detect and exploit the interactions between variables, that arise in the optimization of a given problem. This question is closely related with what have been called the Linkage Problem. In terms of the Building Block Propagation Theory this problem is characterized as the effect caused in the evolution by the disruption of important partial solutions or building blocks.

Like the Linkage Problem is determined no only by the characteristics of the problem but also by its representation, different alternatives have been proposed that manipulate the representation of solutions in order to make them less vulnerable to the disrupting effect of genetic operators. Another solution have been the creation of genetic operators able to cope with the interaction between variables. Nevertheless, these genetic operators have been often designed considering the particular characteristics of the problems, constraining its use to narrow domains. **EDA** use the statistical information contained in the population of selected points to detect dependencies, then the estimated probability distribution of the best individuals is used to sample the points of the next generation. There are no mutation nor crossover operators.

Although Syswerda's Bit Based Simulated Crossover [28] was one of the first attempts to use the probabilistic information in the search, it was the Population Based Incremental Learning [3] the first GA without crossover. PBIL is a combination of evolutionary optimization and hillclimbing. The goal of the algorithm is to create a real valued probability vector which, when sampled reveals high evaluation solution vectors with high probability. Since the creation of PBIL, two main objectives have moved the research on **EDA**. The increase of the expressiveness of the probabilistic models, allowing to represent interactions of higher order, and to reduce the complexity of the methods and algorithms used for the estimation and sampling of the probabilities.

Different classifications of **EDA** may be used to describe these algorithms. In [15] **EDA** are classified by considering the complexity of the models used to capture the interdependencies between the variables. For our analysis we classify the **EDA** according to the way learning is done in the probability graphical model used. One group refers to the algorithms that make a parametric learning of the probabilities and the other is composed by those algorithms where a structural learning of the model is done. Parametric and structural learning are also known as model fitting and model selection.

To the first class belong PBIL, the Univariate Marginal Distribution Algorithm (UMDA) [16] and the Factorized Distribution Algorithm as it was introduced in [17]. In that research a fixed model was assumed for the optimization of additive decomposable functions. Similarly to PBIL, UMDA generates new solutions by only preserving the proportions of the values of all variables independently of the remaining solutions. This approach can work well even for problems where variables are not completely independent.

The FDA uses a factorization of the probability distribution, which does not need to be an exact factorization. In [17] the FDA employs factorizations based in prior knowledge about the structure of the function to be optimized. Results show that the Factorized Distribution Algorithm outperforms other evolutionary algorithms in the optimization

of additive functions. The FDA has been successful also in the optimization of deceptive problems with overlapping variables.

There exist a number of **EDA** that can be grouped into the class of algorithms that make a structural learning of the model. The Mutual Information Maximization for Input Clustering (MIMIC) algorithm uses a greedy search to generate a chain in which each variable is conditioned on the previous one [9]. MIMIC searches in each generation the chain shaped model closest in the Kullback-Leibler distance to the probability distribution of the selected points. In the Bivariate Marginal Distribution Algorithm (BMDA) [23] some pairwise interactions are taken into account to model the probability distribution, the most important acyclic pairwise interactions are considered according to the Pearson's chi-square statistics for independence. For a certain class of problems the behavior of the BMDA is better than the UMDA.

MIMIC's probabilistic model was extended [4] to a larger class of dependency graphs which can be represented using tree shaped networks. The algorithm called COMIT (Combining Optimizers with Mutual Information Trees) searches a Bayesian Network (BN) where each node is constraint to have only one parent. Recently, a number of different **EDA** that use BN models for representing the dependencies of variables have been reported in the literature.

- The Estimation of Bayesian Network Algorithm (EBNA) [11]. EBNA looks for the BN whose structure has the maximum posterior likelihood, and whose parameter can be computed directly from the data set. It uses the BIC approximation [27] in conjunction with a greedy search heuristic. Preliminary results showed that EBNA outperforms UMDA in the optimization of complex additive functions.
- The Bayesian Optimization Algorithm (**BOA**) [24]. BOA uses the Bayesian Dirichlet scoring metric which allows to combine prior knowledge about the problem and the statistical data from a given data set. A simple greedystrategy is used to construct the network algorithm with only an edge addition allowed when the number of parents of the node is constraint to $k > 1$.
- The Polytree Approximation of Distribution Algorithm (**PADA**) [21]. The learning component of PADA uses a method based in testing dependencies, for the construction of the polytree's skeleton. Then it uses marginal and conditional dependency information to determine head to head patterns. At the end, the algorithm can use a scoring metric to complete the directioning of the edges. This improves the last part of the learning algorithm, namely giving direction to edges. We believe that the combination is superior to both: a score+search method only and the original testing only algorithm.

3. Algorithms for Learning Probabilistic Graphs from data

Graphical models have become common knowledge representation tools capable of efficiently representing and handling independence relationships as well as uncertainty in our knowledge. They comprise a qualitative and a quantitative component. The qualitative component is a graph displaying dependency/independence relationships: the absence of some links means the existence of certain conditional independence relationships between variables, and the presence of links may represent the existence of direct dependency relationships. The quantitative component is a collection of numerical parameters, usually conditional probabilities, which give idea of the strength of the dependencies. Therefore, graphical models provide an intuitive graphical visualization of the available knowledge and encode probabilistic information in an economical way: the storage requirements of a joint probability distribution are usually excessive, whereas the memory requirements of a suitable factorization of this distribution, taking into account the independence relationships displayed by the graph, may be much smaller.

Probabilistic modeling for combinatorial optimization [4,17] is an example where very fast learning algorithms are needed: these methods explicitly maintain a probabilistic model of the good solutions found so far in the search space; these models are sampled to generate next candidate solutions to be evaluated, and the process is repeated; thus, many probabilistic models have to be estimated.

As was mentioned above, there are different ways to classify structural learning methods. In section 3.2 we will discuss a method based on statistical testing of conditional independence of different order. This kind of method tries to get a list of dependence/independence weighted assertions, and then constructs a chordal graph that satisfies

as much as possible the assertions on the list. The second approach searches for good networks using a cost function to compare them. We discuss it in the next section.

3.1. Score + Search Structural Learning

The problem of finding a good Bayesian Network can be seen as the optimization of a score function over the set of all possible graph structures. As far as the solution space is huge, this task is usually complex. It is possible to reduce the complexity of the search, if we constraint the networks in some way. For example, it is easy to optimize functions over networks with at most one parent for each node, than over many-parents networks.

In this task, it is commonly used local search methods like simple greedy or hill-climbing algorithms [13], but obviously it is possible to use more sophisticated procedures like simulated annealing and evolutionary methods. The allowed operations are: edge addition, removal and reversal. There are different metrics or score functions; the most famous one is the K2 metric.

Provided that certain conditions holds (independence of the cases of the database, uniformity of the probability density of the network's parameters given the network, completeness of the data) it is possible to derive a formula that states what is the joint distribution of a network G and a database BD.

$$P(G, BD) = P(G) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}!$$

Here r_i and q_i are the cardinality of variable X_i and the Cartesian product $(\prod_G(X_i))$ of its parents. The kth-value of X_i and the jth-value of $(\prod_G(X_i))$ is represented by x_{ik} and w_{ij} respectively. From the number of cases N_{ijk} when X_i takes its kth-value and $\prod_G(X_i)$ its jth-value, it is obtained $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$.

The K2 metric is a special case of the Bayesian-Dirichlet (BD) [13] metric, which combines the prior knowledge about the problem and the statistical data from a given database. The K2 metric considers that there is not any prior knowledge about the problem. Another interesting metric is the so called BIC approximation [27].

$$\log\ p(BD \mid G) \approx BIC(G, BD) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \log \frac{N_{ijk}}{N_{ij}} - \frac{\log N}{2} \sum_{i} (r_i - 1) q_i$$

The BIC approximation has the nice property for an **LCEA** of selecting simpler structures, which reduce the computational cost.

3.2. Learning Chordal Graphs

The important class of graphical models that have been found to be specially useful for **FDA**, is the class of Decomposable Models [22] and can be represented by means of both directed and undirected graphs. The topological property selected to represent independence assertions depends on the type of graph we use: separation for undirected graphs and d-separation for directed acyclic graphs. Two sets of nodes A and B are called to be separated by another set C, if every path between a node in A and a node in B, contains at least one node of C. In this paper we will interested in getting graphs that are I-maps of a given dependency model. Given a dependency model M, an undirected graph G is said to be an I-map of M, if every separation in G implies an independence in M.

The graphical counterpart of the decomposable dependency models are the chordal graphs. An undirected graph is said to be chordal if every cycle of length four or more has a chord. Another crucial property of chordal graphs is that their cliques (i.e the largest subgraphs whose nodes are all adjacent to each other) can be joined to form a tree **JT**, called the Junction Tree, such that any two cliques containing a node α are either adjacent in **JT** or connected by a chain of **JT** made entirely of cliques that contain α. This property is called the running intersection property.

The problem of getting a chordal graph from a data set is relevant to many applications. This can be accomplished by first determining the conditional independence graph of the data and then, if it is not chordal, transforming it into a chordal graph by triangulization. For algorithms that need to learn a model of data many times, the process of making a graph chordal (triangulization) can be too expensive in computational terms. On the other hand, when the above approach is used, resulting graphs might not always be as close to the original graphs as expected. Therefore, it could be more efficient if we obtain the chordal graph straight from the data.

The class of dependency models isomorphic to undirected graphs has been completely characterized in terms of five axioms [22] satisfied by the independence relationships within the model. Re-

cently, two additional axioms have been proven to be enough to characterize decomposable models [10]. The characterization is based on the identification of two different properties:

- Axiom of Strong Chordality. If a separator of α and β is not complete, then it has a proper subset which is still a separator of α and β.
- Axiom of Clique-separability. Whenever two nodes α and β are not adjacent (are independent), we can find a separating set whose nodes are all adjacent to each other

The following theorem was proven in [10]:

Theorem: [de Campos,1996]: A dependency model M is isomorphic to a chordal graph if, and only if, it satisfies the 5 axioms of Pearl and either the axiom of Strong Chordality or the axiom of Clique-separability.

This axiomatic characterization of decomposable models, in terms of independence relationships creates desirata for driving automatic construction of chordal graphs from data. In [10] was developed an algorithm that finds a minimal chordal I-map of any dependency model isomorphic to an undirected graph. The restriction to graph-isomorphic models was necessary to prove the results. It is not clear whether the methodology can be adapted to deal with more general models. We conjecture that for optimization purposes this restriction is not so hard. On the other hand it was assumed that the data is a perfect representation of the underlying model. Because the algorithm uses conditional independence tests of order as low as possible, it could be a good choice for optimization problems with low interaction patterns. By the way, this have been the case for many of the current new Bayesian Optimization Algorithms. For example, in BOA [24] the maximum amount of the edges coming to a node is constraint by a parameter of the algorithm.

3.3. The Junction Tree Sampling Algorithm

In this paper we will consider the Junction Tree Sampling Algorithm (**JTSA**) as the heart of the FDA. This idea will be used as the starting point for the extension of the FDA beyond the scope of the optimization of additive decomposable functions.

In the optimization context the junction tree is used as a sampling tool. However, the junction tree has been used before to tackle general problems of inference and learning within the general framework of probabilistic graphical models. There, the key problem that arises is the problem of computing the posterior probabilities of certains variables given the observed values of other variables. That is the case, for example, of the Hiden Markov Models HMM and the Boltzman Machine. In optimization we are interested also in the generation of a population of individuals according to the computed distributions.

Perhaps the most important property of the junction tree algorithm is that indeed it is generic; it can be applied to any graphical model. So, we can expect that the JTSA is general enough to deal with a large class of optimization problems. Let us to say a few words on this affirmation.

For decomposable distributions the calculation of the posterior probabilities is straightforward, and can be achieved via a local message-passing algorithm on the junction tree. As it was said above descomposable distributions are isomorphic to triangulated or chordal graphs, where the junction tree is actually defined [29]. Nonchordal graphs can be turned into chordal graphs by addition of edges. This means, that the same algorithm can be used for all undirected graphs, an untriangulated graphs is first triangulated. Finally, it is well known that it is possible to convert direct graphs to undirected in a manner that preserves the probabilistic structure of the original graph.

There are many examples of application of junction tree like algorithms, where it is emphasized the unifying framework of graphical models both for expressing probabilistic dependencies and for describing algorithms that perform the inferential step of calculating posterior probabilities on these graphs. The research work done in [17] extends this unification to the sampling step of generating populations of individuals according to the calculated posterior probabilities.

4. The Factorized Distribution Algorithm

FDA (the Factorized Distribution Algorithm), as was introduced in [17], is an evolutionary algorithm which combines mutation and recombination by using a distribution instead. First the distribution is estimated from a set of selected points. It is then used to generate new points for the next generation. The FDA has been extensively investigated for the optimization of additively decomposed discrete functions (ADFs). It was shown the-

oretically and numerically that the scaling of the FDA depends on the ADF structure and the specific assignment of function values. Difficult functions on a chain or a tree structure are optimized in about $O(n\sqrt{n})$ function evaluations. More standard genetic algorithms are not able to optimize these functions.

The main assumption of previous work done on FDA, is that an ADF and a factorization of the probability distribution is given. The factorization can also be used at the initialization step. For faster convergence a proportion of $r * N$ individuals can be generated with a local approximation of the conditional marginal distributions.

First we define precisely an ADF.

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

$$f(x) = \sum_{s_i \in S} f_i(\Pi_{s_i} x) \qquad S = \{s_1, \ldots, s_l\} \quad s_i \subseteq \tilde{X} \tag{1}$$

where

$$\tilde{X} := \{x_1, \ldots, x_n\} \quad \mathbf{B} := \{0, 1\} \quad X := \mathbf{B}^{|\tilde{X}|}$$
$$X_s \subseteq X \text{ with } s \subseteq \tilde{X}$$
$$\Pi_s x := \text{the projection of } x \in X \text{ onto the subspace } X_s$$

FDA can be used with an exact or an approximate factorization. It uses *finite samples* of points. Convergence of FDA to the optimum will depend on the size of the samples. FDA can be run with any popular selection method. We usually apply truncation selection. In the figure 1 we present the FDA algorithm as was introduced in [17].

The basic assumption of FDA_r is the use of the structure of the ADF function as the structure of the underlying distribution of the selected set of points. However, it is already known that in general this is not true. For exponential selection it can be shown that the assumption holds, but it is a bad selection method. For truncation selection, however there is a considerable departure from this condition.

Nevertheless, the point is that FDA_r outperforms other algorithms when optimizing ADFs. At this moment, there exist a few FDA implementation, but all the reported results are surprisingly very good. Here we only show the results obtained by our implementation in the case of the deceptive sub-function f^3_{dec}. This function has been defined before. It is used to define the separable *deceptive* function of order three

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

$\mathbf{FDA_r}$

- **STEP 0:** Set $t \Leftarrow 0$. Generate $(1-r) * N \gg 0$ points randomly and $r * N$ points according to a local approximation of the conditional marginal distributions.
- **STEP 1:** Selection of promising points.
- **STEP 2:** Compute the conditional probabilities $p^s(\Pi_{b_i} x | \Pi_{c_i} x, t)$.
- **STEP 3:** Generate a new population according to $p(x, t+1) = \prod_{i=1}^{l} p^s(\Pi_{b_i} x | \Pi_{c_i} x, t)$.
- **STEP 4:** If the termination criteria are met, FINISH.
- **STEP 5:** Add the best point of the previous generation to the generated points (elitist).
- **STEP 6:** Set $t \Leftarrow t + 1$. Go to STEP 1.

Figure 1. The FDA with fixed model.

Table 5 presents for 90 variables, the minimal (or critical) population size, which is defined as the minimal population size guaranteeing that at least 950 of 1000 runs are successful. It is worth noting that the presented results have been obtained without the initialization step of FDA_r. In [18] it is reported an improvement of four generation of $FDA_{0.5}$ with respect to FDA_0.

T	0.05	0.1	0.15	0.2	0.3
N^*	1100	600	**450**	950	1000
Gen	7.2	8.5	9	9.9	11.7

Table 1
Results of our FDA implementation.

The use of the function structure has also the appealing of opening a door to certain class of constraint problems (see [17]). Althouhg in the next section, we extend FDA to deal with a class of constraints, which do not have to be compatible with the structure of the function.

4.1. Dealing with constraints and the FDA

Two basic approaches are known, when Evolutionary algorithms are applied to the solution of

constraint optimization problems.

The first approach searches only in the space of solutions which fulfills the constraints (space of feasible solutions). To do that the applied optimization operator assures that no illegal solutions will be generated during the search. Depending on the type of optimization method applied these operators can become complex and costly in terms of time. The other approach considers the search over a wider space than that formed by the feasible solutions. A penalty function is generally defined which punishes the non feasible solution points. This approach faces then the problem of how to combine the penalty function with the own objective function.

It has been acknowledged that some EDA can easily handle certain optimization problems with constraints. In [17] it is shown that when the structure of the additive function is used for the factorization, and this structure is compatible whith the structure of the constraints, the FDA will work in the optimization of the function by generating only legal points.

In this section we show that a FDA with minor extensions is able to handle the above mentioned, and another kind of constraints, which do not have to be compatible with the structure of the function. Furthermore, the way that these constraints are enforced in the new generated population neither depend on the structure of the function, just on the factorization. We will show an example of this type of constraint problems.

Let f(x) be a function defined over the set a n-dimensional binary vectors x that satisfy the following constraint: $1 \leq a < u(x) \leq b < n$, where $a, b, n \in N$ and $u(x)$ is the function of unitation.

In [26] it is introduced a Constraint Univariated Marginal Distribution Algorithm (CUMDA) for the solution of this class of functions. In order to explain the FDA approach to these problems some notation is necessary.

Definition: Given a set of sets S=$s1, \ldots, sl$, we define for $i = 1, 2, \ldots, l$ sets d_i, and b_i,

$$d_i = \cup_{j=1}^{i} s_j, \qquad b_i = s_i \backslash d_{i-1}$$

We consider that the s_i are sorted following the order imposed by the junction tree in the generation of points, s_1 correspond to the root of the tree, d_0 is set to 0. For each vector to be generated using the FDA, the assignation of values for the subset of variables X_{b_i} has to fulfill the following two conditions:

1) $u(X_{s_i}) \leq b$.
2) $u(X_{s_i}) \geq a - (n - |s_i|)$.

The conditions establish that after generating variables X_{b_i}, the accumulated value of unitation $u(X_{s_i})$ should not exceed the constraint b, and it has to be still possible for the vector X to reach the constraint value a.

To make FDA able to deal with these constraints we only change the generation step. This FDA will restrict the set of possible assignations of X_{b_i} according to their unitation values. The marginal probabilities of those assignations that violate the constraints are redistributed among the 'feasible' ones, proportionally to their own marginal probabilities. In this way, the relative proportions of legal marginal probabilities keep fixed, only their absolute values are changed. Our approach is extensible to a wide set of constraint problems, those for which it is possible to generate new points considering in every step of the generation the fulfillment of constraints. The case when structure of the factorization is compatible which the structure of the constraints is a particular case of the problems we have analyzed.

5. FDA_L - an FDA with Structural Learning

At this point we can introduce our FDA_L, an FDA algorithm that learns the structure of the distribution of the selected set at every generation. The learning algorithm used was commented in section 3.2 and is presented in Figure 2. Figure 3 outlines the FDA_L algorithm.

Learning Algorithm

- **STEP 0:** Starting from a complete graph, delete edges by making independency tests of order 0 and 1. Call this graph G_{01}.
- **STEP 1:** Find the list L of maximal cliques of G_{01} or a list of reasonable sized sets that contain these cliques.
- **STEP 2:** Divide the cliques in L as much as possible using the axioms of section 3.2.
- **STEP 3:** Construct the JT.

Figure 2. Structural Learning Algorithm.

Table 2 presents results obtained for the same deceptive problem, but with 30 variables. We show the minimal population size obtained with a set-

FDA$_L$

- **STEP 0:** Set $t \Leftarrow 0$. Generate $N \gg 0$ points randomly.
- **STEP 1:** Selection of promising points.
- **STEP 2:** Learning the Junction Tree (JT) directly from the data.
- **STEP 3:** Compute the marginal densities associated to the nodes of the JT.
- **STEP 4:** Generate a new population according to the constructed JT.
- **STEP 5:** If the termination criteria are met, FINISH.
- **STEP 6:** Add the best point of the previous generation to the generated points (elitist).
- **STEP 7:** Set $t \Leftarrow t + 1$. Go to STEP 1.

Figure 3. The FDA with Structural Learning.

	N_*	Gen
FDA	150	9
FDA_L	2000	3.5
BOA	900	12

Table 2
Results for a Deceptive 3 problem.

ting of 95 success in 100 runs. Both FDA and FDA_L use truncation selection of 0.15. The result for BOA is only approximated, it was taken from [24] for BOA with K2 metrics.

We have to say that our current implementation of the learning algorithm is still under development. Hence, it could be possible further improvements of these results. Moreover, results of a parallel research indicates us that there is a strong dependency of the minimal population size on the JT. As far as, we can construct different JT for the same distribution, the problem of selecting the optimal JT immediately arises. We will present our results in a forthcoming paper. At this moment, the root of JT is selected randomly at every generation.

6. Conclusions

We have presented an approach that allows the extension of the application of the FDA beyond the scope of problems, for which prior knowledge about the structure of the selected distributions of points is available. For this, we have augmented the classic FDA scheme with a learning component, based on testing independency relationships of order as low as possible. We have also introduced some minor modification of the FDA to manage certain constraint problems. Here the interesting point is that it is not necessary fully compatibility between the constraints and the function structure, as was the case in [17].

References

[1] Acid S. , de Campos L.M. (1995), Approximations of causal networks by polytrees: an empirical study, in: B. Bouchon-Meunier, R.R. Yager, L.A. Zadeh, eds., Advances in Intelligent Computing, Lect. Notes Comput. Sc. 945, Springer Verlag, Berlin, 149–158.

[2] Aho A.V., Hopcroft J.E. and Ullman J.D. (1987), Data Structures and Algorithms ,Addison-Wesley.

[3] Baluja S. (1994), Population-Based incremental learning: A method for intergrating genertic seach based function optimization and competitive learning, Tech Rep. No. CMU-CS-94-163, Pittsburg, PA: Carnegie Mellon University

[4] Baluja S. and Davies S. (1997), Using optimal dependency trees for combinatorial optimization: Learning the structure of the search space, in: Proceedings of the 1997 International Conference on Machine Learning, 30–38.

[5] Beinlich I.A., Suermondt H.J., Chavez R.M. and Cooper G.F. (1989), The Alarm monitoring system: A case study with two probabilistic inference techniques for belief networks, in: Proceedings of the Second European Conference on Artificial Intelligence in Medicine, 247–256.

[6] Boyan J. and More A. (1996), Learning Evaluation Functions for large acyclic domains. In L. Saitta, editor, Machine Learning: Proceedings of the Thirteen International Conference. Morgan Kauffman .

[7] Chow C.K. and Liu C.N. (1968), Approximating discrete probability distribution with dependence trees, IEEE T. Inform. Theory 14, 462–467.

[8] de Campos L.M. (1998), Independency relationships and learning algorithms for singly connected networks, Journal of Experimental and Theoretical Artificial Intelligence 10, 511–549.

[9] De Bonet J. S. , Charles J. , and Viola P. (1997), MIMIC: Finding optima by estimating probability densities. In. M Mozer, M. Jordand T. Petsche, editors, Advances in Neural Information Processing Systems.

[10] de Campos, L.M. and Huete, J. F (1997), Algorithms for Learning Decomposable Models and Chordal Graphs. Tech. Rep. DECSAI 970213. University of Granada(http://decsai.ugr.es/gte/tr.html)

[11] Etxeberria R. and Larranhaga P. (1999), Global Optimization using Bayesian networks. II Symposium on Artificial Intelligence CIMAF99. Special Session on Distributions and Evolutionary Optimization

[12] Henrion M. (1988), Propagating uncertainty in Bayesian networks by logic sampling, in: J.F. Lemmer, L.N. Kanal, eds., Uncertainty in Artificial Intelligence 2, North-Holland, Amsterdam, 1988,149–163.

[13] Heckerman, D. Geiger D. and Chicckering M. (1994), Learning Bayesian networks: The combination of knowledge and statistical data. (Technical Report MSR-TR-94-09). Redmond, WA: Microsoft Research.

[14] Lam W. and Bacchus F. (1994), Learning belief networks: an approach based on the MDL principle, Computational Intelligence 10, 269–293.

[15] Larrannaga P.,Etxebarria R., Lozano J.A., Sierra B., Inza I., Penna J. M. (1999), A review of the cooperation between evolutionary computation and probabilistic graphical models. II Symposium on Artificial Intelligence CIMAF99. Special Session on Distributions and Evolutionary Optimization

[16] Mühlenbein H. (1998), The Equation for Response to Selection and its Use for Prediction. *E*volutionary Computation, 5, 303–346.

[17] Mühlenbein H., Manning T. and A. Ochoa (1999), Schemata, Distributions and Graphical Models in Evolutionary Optimization. to appear in the Journal of Heuristic v5. n2.

[18] Mühlenbein H. and Mahnig T. (1999), FDA-An Evolutionary Algorithm for Additively Decomposed Functions. II Symposium on Artificial Intelligence CIMAF99. Special Session on Distributions and Evolutionary Optimization

[19] Ochoa A. (1997), How to deal with costly fitness functions in evolutionary computation. In Proceedings of the 13^{th} ISPE/IEE International Conference on CAD/CAM, Robotics & Factories of the Future. 788–793. Pereira .

[20] Ochoa A. (1999), Finding Wavelet Packet Bases with an Estimation Distribution Algorithm. Presented to the International Conference GECCO99.

[21] Ochoa A., Soto M., Acid S. and de Campos L. M. (1999), Bayesian Evolutionary Algorithms based on Simplified Models, II Symposium on Artificial Intelligence CIMAF99. Special Session on Distributions and Evolutionary Optimization

[22] Pearl J. (1988), Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan and Kaufmann, San Mateo.

[23] Pelikan, M and Mühlenbein H. (1998), The bivariate marginal distribution algorithm. London: Springer-Verlag. In printing

[24] Pelikan, M. Goldber D. E. and Cant-Paz E. (1998), Linkage problem, Distribution estimation and Bayesian Estimation Networks. Illegal Report 98013. November.

[25] Rebane G. and Pearl J. (1989), The recovery of causal polytrees from statistical data, in: L.N. Kanal, T.S. Levitt, J.F. Lemmer, eds., Uncertainty in Artificial Intelligence 3,North-Holland, Amsterdam, 175–182.

[26] Santana, R. and Ochoa, A. (1999), Dealing with Constraints with Estimation Distribution Algorithms: The Univariate case. II Symposium on Artificial Intelligence CIMAF99. Special Session on Distributions and Evolutionary Optimization. (this issue).

[27] Schwarz G. (1978), Estimating the dimension of a model. Annals of Statistics, 7(2):461-464.

[28] Syswerda G. (1989) Uniform Crossover in Genetic Algorithms. International Conference on Genetic Algorithms 3,2-9.

[29] Whittaker, J. (1991), Graphical models in applied multivariate statistics. Wiley Series in Probability and Mathematical Statistics. New York : Wiley.