# Exact Bayesian Network Learning in Estimation of Distribution Algorithms

Carlos Echegoyen, Jose A. Lozano, Roberto Santana, and Pedro Larrañaga

***Abstract*— This paper introduces exact learning of Bayesian networks in estimation of distribution algorithms. The estimation of Bayesian network algorithm (EBNA) is used to analyze the impact of learning the optimal (exact) structure in the search. By applying recently introduced methods that allow learning optimal Bayesian networks, we investigate two important issues in EDAs. First, we analyze the question of whether learning more accurate (exact) models of the dependencies implies a better performance of EDAs. Second, we are able to study the way in which the problem structure is translated into the probabilistic model when exact learning is accomplished.**

## I. Introduction

Estimation of distribution algorithms (EDAs) [1], [2] are evolutionary algorithms that use probability models instead of the typical genetic operators employed by genetic algorithms (GAs) [3]. In EDAs, machine learning methods are used to extract relevant features of the search space. The collected information is represented using a probabilistic model which is later employed to generate new points. In this way, probabilistic models are used to lead the search to promising areas of the search space.

EDAs mainly differ in the class of probabilistic models used and the methods applied to learn and sample these models. One class of the models that has been extensively applied in EDAs is Bayesian networks [4]. One of the benefits of EDAs that use these types of models [5], [6], [7], [8] is that the complexity of the learned structure depends on the characteristics of the data (selected individuals). Additionally, the analysis of the networks learned during the search can provide information about the problem structure.

One important problem in EDAs is to analyze how the choices of the probabilistic models and of the learning and sampling algorithms can influence the adequate balance between exploration and exploitation. There are a number of papers [9], [10], [11], [12] that report on the way in which the performance of EDAs can dramatically change according to changes in the parameters that determine the learning of the models. Although, in the case of EDAs that use Bayesian networks, the role of the parameters that penalize the complexity of the networks has been studied, a similar analysis of the accuracy of the methods used for finding the best network and its influence in the behavior of EDAs has not been conducted.

Intelligent Systems Group, Department of Computer Science and Artificial Intelligence, University of the Basque Country, Paseo Manuel de Lardizábal 1, 20080 San Sebastián - Donostia, Spain. email: (ja.lozano@ehu.es, cechegoyen001@ikasle.ehu.es, rsantana@si.ehu.es, pedro.larranaga@ehu.es)

Another related and important issue in EDAs is how the features of the search space are reflected in the learned probability models. This issue, which has received attention from the EDA community [10], [13], [14], [15], is essential to understand the mechanisms that allow EDAs to efficiently sample the search space during the optimization process. However, the question of analyzing the relationship between the search space and the structure of the learned probabilistic models becomes difficult due to two main reasons: The stochastic nature of EDAs' search, and the fact that methods used for learning the models are, in general, able to find only approximate, suboptimal, structures.

In this paper we present an alternative that allows one to study the effect that learning accurate models of the population produce in the behavior of EDAs based on Bayesian networks. Additionally, our contribution serves as a solution to extract more accurate information about the relationship between the problem structure, the search distributions and the probabilistic dependencies learned during the search.

Our approach is based on the use of recently published methods for learning optimal (exact) Bayesian networks [16], [17], [18], [19]. Methods that do exact Bayesian structure learning compute, given a set of data and a prespecified score (in our case, the $BIC$ score [20]), the network structure that optimizes the score. Since the problem of learning the optimal Bayesian network is NP-hard [21], these methods set constraints on the maximum number of variables and/or cases they can deal with. Usually, dynamic programming algorithms are used to learn the structure.

The paper is organized as follows. In the next section Bayesian networks are presented, the general procedures to learn these networks from data are discussed. In Section III, we focus on the type of search strategies used to find the Bayesian network structure. Approximate and exact learning methods are analyzed. Section IV introduces the EBNA algorithm. In Section V, the functions used to evaluate the exact and local learning methods used by EBNA are introduced. This section presents and discusses the results of the different experiments conducted. Work related to our proposal is analyzed in Section VI. The conclusions of our paper are presented in Section VII.

## II. Bayesian networks

### A. Notation

Let $X$ be a random variable. A value of $X$ is denoted $x$. $\mathbf{X} = (X_1, \ldots, X_n)$ will denote a vector of random variables. We will use $\mathbf{x} = (x_1, \ldots, x_n)$ to denote an assignment to the variables. We will work with discrete variables. The joint

probability mass function of $\mathbf{x}$ is represented as $p(\mathbf{X} = \mathbf{x})$ or $p(\mathbf{x})$. $p(\mathbf{x}_S)$ will denote the marginal probability distribution for $\mathbf{X}_S$. We use $p(X_i = x_i \mid X_j = x_j)$ or, in a simplified form, $p(x_i \mid x_j)$, to denote the conditional probability distribution of $X_i$ given $X_j = x_j$.

Formally, a Bayesian network [22] is a pair $(S, \boldsymbol{\theta})$ representing a graphical factorization of a probability distribution. The structure $S$ is a directed acyclic graph which reflects the set of conditional (in)dependencies among the variables. The factorization of the probability distribution is codified by $S$:

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i|\mathbf{pa}_i)$$

where $\mathbf{pa}_i$ denotes a value of variable $\mathbf{Pa}_i$, the parent set of $X_i$ (variables from which there exists an arc to $X_i$ in the graph $S$). On the other hand, $\boldsymbol{\theta}$ is a set of parameters for the local probability distributions associated with each variable. If the variable $X_i$ has $r_i$ possible values, $x_i^1, \ldots, x_i^{r_i}$, the local distribution $p(x_i|\mathbf{pa}_i^j, \boldsymbol{\theta}_i)$ is an unrestricted discrete distribution:

$$p(x_i^k|\mathbf{pa}_i^j, \boldsymbol{\theta}_i) \equiv \theta_{ijk}$$

where $\mathbf{pa}_i^1, \ldots, \mathbf{pa}_i^{q_i}$ denote the values of $\mathbf{Pa}_i$ and the term $q_i$ denotes the number of possible different instances of the parent variables of $X_i$. In other words, the parameter $\theta_{ijk}$ represents the probability of variable $X_i$ being in its $k$-th value, knowing that the set of its parent variables is in its $j$-th value. Therefore, the local parameters are given by $\boldsymbol{\theta}_i = (((\theta_{ijk})_{k=1}^{r_i})_{j=1}^{q_i})$.

### B. Learning Bayesian networks from data

There are different strategies to learn the structure of a Bayesian network. We focus on a method called "score + search" which is the one used in the experiments presented in this paper. In this strategy, given a database $D$ and a Bayesian network whose structure is denoted by $S$, a value (score) which evaluates how well the Bayesian network represents the probability distribution of the database $D$ is assigned. Different scores can be used. In this work we have used the Bayesian Information Criterion score ($BIC$) [20] (based on penalized maximum likelihood).

A general formula for a penalized maximum likelihood score can be written as follows:

$$\log p(D|S, \hat{\boldsymbol{\theta}}) - f(N)dim(S)$$

where $dim(S)$ is the dimension –number of parameters needed to specify the model– of the Bayesian network with a structure given by $S$. Thus:

$$dim(S) = \sum_{i=1}^{n} q_i(r_i - 1)$$

and $f(N)$ is a non negative penalization function. The Jeffreys-Schwarz criterion, sometimes called BIC [20] takes into account $f(N) = \frac{1}{2}\log N$. Thus the $BIC$ score can be written as follows:

$$BIC(S, D) = \log \prod_{w=1}^{N} \prod_{i=1}^{n} p(x_{w,i}|\mathbf{pa}_i^S, \hat{\boldsymbol{\theta}}_i) - \frac{1}{2}\log N \sum_{i=1}^{n} q_i(r_i - 1) \quad (1)$$

To find the Bayesian network that optimizes the score implies solving an optimization problem. This can be done with exhaustive or heuristic search algorithms. In Section III, we analyze two variants for finding the Bayesian network structures. Each structure is evaluated using the maximum likelihood parameters.

### C. Learning of the parameters

Once the structure has been learned, the parameters of the Bayesian network are calculated using the Laplace correction:

$$\hat{\theta}_{ijk} = \frac{N_{ijk} + 1}{N_{ij} + 2} \ . \quad (2)$$

where $N_{ijk}$ denotes the number of cases in $D$ in which the variable $X_i$ has the value $x_i^k$ and $\boldsymbol{Pa}_i$ has its $j^{th}$ value, and $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$.

## III. Methods for learning Bayesian networks

Once we have defined a score to evaluate Bayesian networks, we have to set a search process to find the Bayesian network that maximizes the score given the data. Approximate and exact methods can be used.

### A. Learning an approximate model

In practical applications, we need to find an adequate model structure as quickly as possible. Therefore, a simple algorithm which returns a good structure, even if not optimal, is preferred. An algorithm that fulfills these criteria is Algorithm B [23] which is typically used by most of Bayesian-network based EDAs. Algorithm B is a greedy search which starts with an arc-less structure and, at each step, adds the arc with the maximum improvement in the score. The algorithm finishes when there is no arc whose addition improves the score.

### B. Learning the exact model

Since learning the Bayesian network structure is an NP-hard problem, for a long time the goal of learning exact Bayesian networks was constrained to problems with a very reduced number of variables. In [17], an algorithm for learning the exact structure in less than super-exponential complexity with respect to $n$ is introduced for the first time. The time complexity of this method is $O(n2^n + n^{k+1}C(m))$ where $k$ is a constant maximum in-degree, and $C(m)$ is the cost of computing a single local marginal conditional likelihood for $m$ instances.

Singh and Moore [19] present a more efficient method called OPTORD which is feasible for $n < 22$ and is shown to work with $n = 22$. The method is compared to local search heuristic to learn Bayesian networks in a number of datasets, obtaining better scoring solutions that the alternatives tested.

In [18], a more efficient method is presented. The algorithm was shown to learn a best network for a data set of 29 variables. Algorithm 1 presents the main steps of the method.

Algorithm 1: **Exact learning algorithm**

1 Calculate the local scores for all $n2^{n-1}$ different (variable, variable set)-pairs.
2 Using the local scores, find best parents for all $n2^{n-1}$ (variable, variable set)-pairs.
3 Find the best sink for all $2^n$ variables.
4 Using the results from Step 3, find a best ordering of the variables.
5 Find a best network using results computed in Steps 2 and 4.

The total score of the Bayesian structure can be decomposed in the computation of local scores. Therefore, in the first step only local scores are computed. On the other hand, the concept of sink, and specifically the best sink of the Bayesian network, plays an important role in the algorithm. A sink is a node with no outgoing arcs (i.e. a node that is not a parent of any other node). Every directed acyclic graph (DAG) has at least one sink. The identification of sinks by Algorithm 1 allows one to obtain best ordering in reverse order and this fact is used to find the best parents following the order computed. More details about the algorithm can be found in [18].

We use an implementation[1] of Algorithm 1. The computational complexity of the algorithm is o($n^2 2^{n-2}$). The memory requirements of the method is $2^{n+2}$ bytes and the disk-space requirement is $12n2^{n-1}$ bytes.

## IV. Estimation of distribution algorithms based on Bayesian networks

The estimation of Bayesian networks algorithm (EBNA) allows statistics of unrestricted order in the factorization of the joint probability distribution. This distribution is encoded by a Bayesian network that is learned from the database containing the selected individuals at each generation. It has been applied with good results to a variety of problems [24], [25], [26], [27], [28]. Other algorithms based on the use of Bayesian networks have been proposed in [6], [7], [8]. A pseudocode of EBNA is shown in Algorithm 2.

In the experiments presented in this paper, EBNA uses truncation selection and the number of selected individuals equals half of the population. The best solution at each generation is passed to the next population, therefore, at each generation $N-1$ new solutions are sampled. The stop criteria changed according to the type of experiments conducted.

[1]The c++ code of this implementation is available from http://www.cs.helsinki.fi/u/tsilande/sw/bene/download/

Algorithm 2: **EBNA**$_{BIC}$

$BN_0 \leftarrow (S_0, \boldsymbol{\theta}^0)$ where $S_0$ is an arc-less DAG, and $\boldsymbol{\theta}^0$ is uniform
$p_0(\mathbf{x}) = \prod_{i=1}^{n} p(x_i) = \prod_{i=1}^{n} \frac{1}{r_i}$.
$D_0 \leftarrow$ Sample $M$ individuals from $p_0(\mathbf{x})$.
t $\leftarrow 1$
**do** {
  $D_{t-1}^{Se} \leftarrow$ Select $N$ individuals from $D_{t-1}$.
  $S_t^* \leftarrow$ Using a search method find one network structure according to the $BIC$ score.
  $\boldsymbol{\theta}^t \leftarrow$ Calculate $\theta_{ijk}^t$ using $D_{t-1}^{Se}$ as the data set.
  $BN_t \leftarrow (S_t^*, \boldsymbol{\theta}^t)$.
  $D_t \leftarrow$ Sample $M$ individuals from $BN_t$.
} **until** Stop criterion is met.

## V. Experiments

The experiments are oriented to compare the EBNA versions that use the two different Bayesian network learning schemes described in Section III. We call them EBNA-Exact and EBNA-Local.

We used three different criteria to compare the algorithms. The time complexity, the convergence reliability and the way in which probabilistic dependencies are represented in the structure of the Bayesian network. A set of test functions that represent different classes of problems are chosen for the experiments. First, we introduce the functions that will be used in our experiments. Then, experiments that illustrate the relationship between the number of evaluations and the behavior of the EBNA with the different learning algorithms are shown. We then present experiments on the convergence reliability of the algorithms. Finally, experiments that show the evolution of the networks learned at different iterations of the algorithm are presented.

### A. Function benchmark

Let $u(\mathbf{x}) = \sum_{i=1}^{n} x_i$, $f(\mathbf{x})$ is a unitation function if $\forall \mathbf{x}, \mathbf{y} \in \{0,1\}^n, u(\mathbf{x}) = u(\mathbf{y}) \Rightarrow f(\mathbf{x}) = f(\mathbf{y})$. A unitation function is defined in terms of its unitation value $u(\mathbf{x})$, or in a simpler way $u$.

Function $OneMax$:

$$OneMax(\mathbf{x}) = \sum_{i=1}^{n} x_i = u(\mathbf{x}) \tag{3}$$

Unitation functions are also useful for the definition of a class of functions where the difficulty is given by the interactions that arise among subsets of variables. One example of this class of deceptive functions is $f_{3deceptive}$ [3].

$$f_{3deceptive}(\mathbf{x}) = \sum_{i=1}^{i=\frac{n}{3}} f_{dec}^3(x_{3i-2}, x_{3i-1}, x_{3i}) \tag{4}$$

where $f_{dec}^3$ is defined as:

$$f^3_{dec}(u) = \begin{cases} 0.9 & for \quad u = 0 \\ 0.8 & for \quad u = 1 \\ 0.0 & for \quad u = 2 \\ 1.0 & for \quad u = 3 \end{cases}$$

For function $Checkerboard$ the goal of the problem is to create a checkerboard pattern of 0's and 1's in an $NxN$ grid. Only the primary four directions are considered in the evaluation. For each position in an $(N-2)(N-2)$ grid centered in an $NxN$ grid, +1 is added for each of the four neighbors that are set to the opposite value. The maximum evaluation for the function is $4(N-2)(N-2)$.

Function $FourPeaks$ is a modification of the $SixPeaks$ problem [29] and it can be defined mathematically as:

$$F_{FourPeaks}(\mathbf{x}, t) = \tag{5}$$
$$max\{tail(0, \mathbf{x}), head(1, \mathbf{x}), tail(1, \mathbf{x}), head(0, \mathbf{x})\} + \mathcal{R}(\mathbf{x}, t) \tag{6}$$

where

$tail(b, \boldsymbol{x})$ = number of trailing $b$'s in $\boldsymbol{x}$
$head(b, \boldsymbol{x})$ = number of leading $b$'s in $\boldsymbol{x}$

$$\mathcal{R}(\boldsymbol{x}, t) = \begin{cases} n & \text{if } tail(0, \boldsymbol{x}) > t \text{ and } head(1, \boldsymbol{x}) > t \text{ or} \\ & \quad tail(1, \boldsymbol{x}) > t \text{ and } head(0, \boldsymbol{x}) > t \\ 0 & \text{otherwise .} \end{cases}$$

The goal is to maximize the function. For an even number of variables this function has 2 global optima, located at the points:

$$(\overbrace{0, 0, \ldots, 0}^{t} 1, 1, \ldots, 1) \qquad (\overbrace{1, 1, \ldots, 1}^{t} 0, 0, \ldots, 0)$$

These points are very difficult to get because they are isolated. On the other hand, two local optima $(0, 0, \ldots, 0)$, $(1, 1, \ldots, 1)$ are very easily reachable. The value of $t$ was set to $\frac{n}{2} - 1$.

Function $Cuban5$ [30] is a non-separable additive function. The second best value of this function is very close to the global optimum.

$$Cuban5(\mathbf{x}) =$$
$$F^5_{cuban1}(s_0) + \sum_{j=0}^{m} (F^5_{cuban2}(s_{2j+1}) + F^5_{cuban1}(s_{2j+2})) \tag{7}$$

where

$s_i = x_{4i}x_{4i+1}x_{4i+2}x_{4i+3}x_{4i+4}$ and $n = 4(2m+1)+1$

$$F^3_{cuban1}(\mathbf{x}) = \begin{cases} 0.595 & for \quad \mathbf{x} = 000 \\ 0.200 & for \quad \mathbf{x} = 001 \\ 0.595 & for \quad \mathbf{x} = 010 \\ 0.100 & for \quad \mathbf{x} = 011 \\ 1.000 & for \quad \mathbf{x} = 100 \\ 0.050 & for \quad \mathbf{x} = 101 \\ 0.090 & for \quad \mathbf{x} = 110 \\ 0.150 & for \quad \mathbf{x} = 111 \end{cases}$$

$$F^5_{cuban1}(\mathbf{x}) = \tag{8}$$
$$\begin{cases} 4F^3_{cuban1}(x_1, x_2, x_3) & if \ x_2 = x_4 \ and \ x_3 = x_5 \\ 0 & otherwise \end{cases}$$

$$F^5_{cuban2}(\mathbf{x}) = \begin{cases} u(\mathbf{x}) & for \quad x_5 = 0 \\ 0 & for \quad x_1 = 0, x_5 = 1 \\ u(\mathbf{x}) - 2 & for \quad x_1 = 1, x_5 = 1 \end{cases}$$

### B. Time complexity analysis

The time complexity analysis experiments were conducted for functions $OneMax$ and $Checkerboard$. The objective is to evaluate the average number of generations to find the optimum needed by EBNA-Local and EBNA-Exact. We start with a population of 10 individuals and the population size is increased by 10 until a maximum population size of 150 is reached. For each possible combination of function, number of variables $n$, and population size $N$, 100 experiments are conducted.

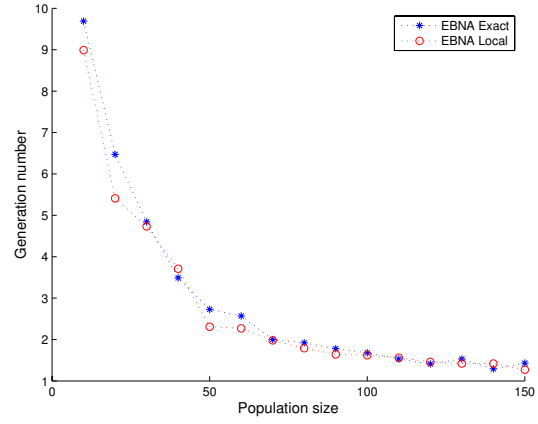



Fig. 1. Time complexity analysis for function $OneMax$, $n = 10$.

For the $OneMax$ function we conducted experiments for $n \in \{10, 12, 15, 20\}$. The idea was to evaluate, under the dimension constraints imposed by the exact learning algorithm, the scalability of both EBNA versions. The results of the experiments for $n \in \{10, 12, 15\}$ are shown in Figures 1, 2 and 3, respectively. For $n = 20$, the computational cost of the experiments was very high and therefore only 50 experiments were conducted. The average results are shown in Figure 4.

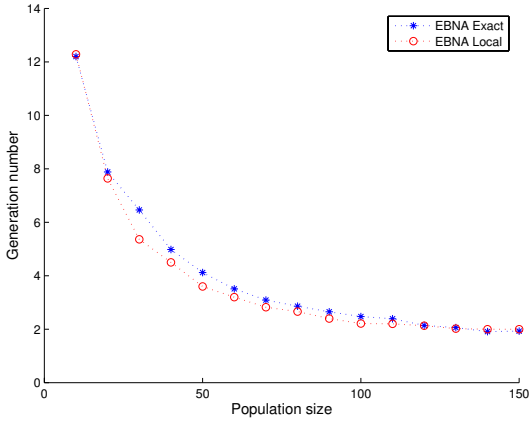



Fig. 2. Time complexity analysis for function $OneMax$, $n = 12$.

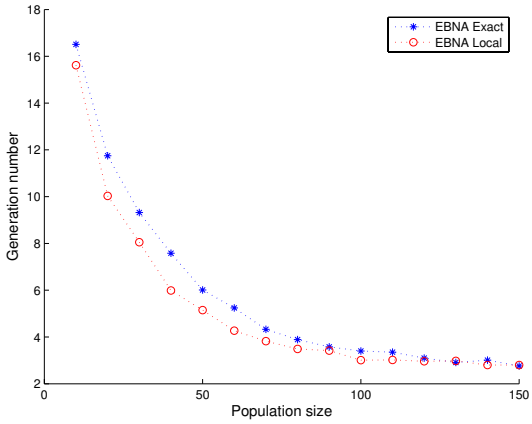



Fig. 3. Time complexity analysis for function $OneMax$, $n = 15$.

The analysis of Figures 1, 2, 3 and 4, respectively, reveals that both algorithms exhibit the same time complexity pattern. However, EBNA-Exact needs, in general, a higher number of evaluations than EBNA-Local to find the optimal solution for the first time. The difference in the number of generations is less evident when the population size approaches 150. For this simple function, it seems that the error in the learning of the model, introduced by the approximate learning algorithm, is beneficial for the search.

For the $Checkerboard$ function, we conducted experiments with $n = 9$ and $n = 16$. The total number of experiments was 100, and the average results are shown in Figure 5 and Figure 6. Also, in this case, EBNA-Exact needed a higher number of evaluations than EBNA-Local. $Checkerboard$ is a function with interactions but, at least for the number of variables considered, it can be optimized with very simple models (data not shown).

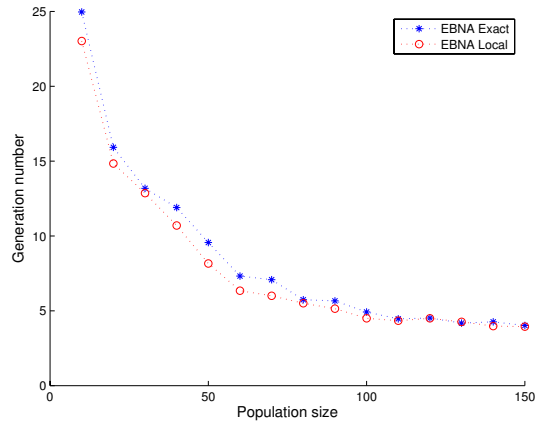



Fig. 4. Time complexity analysis for function $OneMax$, $n = 20$.

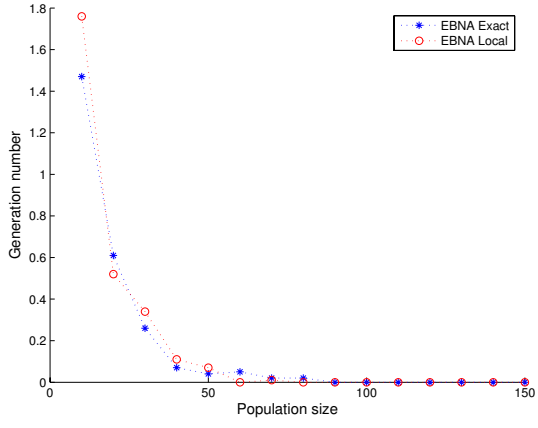



Fig. 5. Time complexity analysis for function $Checkerboard$, $n = 9$.

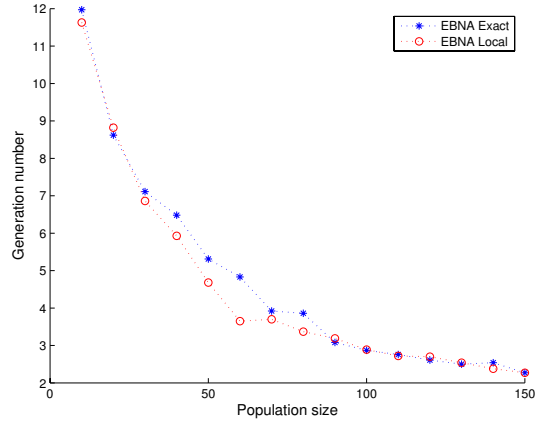



Fig. 6. Time complexity analysis for function $Checkerboard$, $n = 16$.

### C. Convergence reliability

The goal of the following experiments was to find the minimum population size needed by the two different variants of EBNA to find the optimum in 20 consecutive experiments. We investigated the behavior of the algorithms for functions $Cuban5$ ($n = 13$), $FourPeaks$ ($n \in \{10, 12, 14\}$) and $f_{3deceptive}$ ($n \in \{9, 12, 15\}$). The algorithm begins with a population size $N = 16$ which is doubled until the optimal solution has been found in 20 consecutive experiments. The maximum number of evaluations allowed is $10^4$. For each function and value of $n$, 25 experiments are carried out. Table I shows the mean and standard deviation of the critical population size found.

TABLE I

MEAN AND STANDARD DEVIATION OF THE CRITICAL POPULATION SIZE FOR DIFFERENT FUNCTIONS AND NUMBER OF VARIABLES.

| $function$ | $n$ | $EBNA-$Exact | | $EBNA-$Local | |
|---|---|---|---|---|---|
| | | $mean$ | $std$ | $mean$ | $std$ |
| $Cuban5$ | 13 | 118.40 | 53.07 | **109.44** | 57.26 |
| $FourPeaks$ | 10 | **153.60** | 52.26 | 215.04 | 109.11 |
| $FourPeaks$ | 12 | **209.92** | 110.11 | 389.12 | 249.19 |
| $FourPeaks$ | 14 | **312.32** | 133.64 | 604.16 | 318.97 |
| $f_{3deceptive}$ | 9 | **135.68** | 38.40 | 168.96 | 60.94 |
| $f_{3deceptive}$ | 12 | **168.96** | 60.94 | 261.12 | 86.50 |
| $f_{3deceptive}$ | 15 | **220.16** | 58.66 | 296.96 | 95.79 |

Table I shows that for function $Cuban5$, EBNA-Exact requires a slightly higher population size than EBNA-Local. The picture is drastically changed for functions $FourPeaks$ and $f_{3deceptive}$, for which EBNA-Exact needs a much smaller population size. This difference is particularly evident for function $FourPeaks$. Another observation is that the standard deviation of EBNA-Local is always higher than that of EBNA-Exact. Since the only difference between EBNA-Exact and EBNA-Local is in the class of algorithm used to learn the models, the difference of behaviors is due to the ability of EBNA-Exact to learn a more accurate model of the dependencies. Therefore, at least for functions $FourPeaks$ and $f_{3deceptive}$, learning a more accurate model determines a better performance of EBNA.

### D. Analysis of the Bayesian network structures

We go deeply into the analysis of the dependencies learned by the two learning algorithms. The objective of this section is to show how the evolution of the dependencies in both variants of EBNA is and to investigate the way in which the problem structure is encoded in the probabilistic model.

First of all, and in order to illustrate the behavior of EBNA-Exact, Figure 7 shows a typical example of a run of EBNA-Exact for function $FourPeaks$. In the first generation, the pattern of the interactions is not clear. However, as the evolution advances, there is a clear path of dependencies between adjacent variables.

To investigate, the type of dependencies learned by EBNA-Exact and EBNA-Local, we saved the structures of the Bayesian networks learned by both variants of the algorithm in the first and last (population where the optimum was reached) for functions $FourPeaks$, and $f_{3deceptive}$. We chose 30 experiments in which the optimum was found by the algorithms and where they did not converge in the first generation. The frequency in which each arc appeared in the Bayesian network was calculated. Since we are not interested in the direction of the dependencies, we add the frequency of the two arcs that involves the same pair of variables. Frequencies are represented using matrices where lighter colors indicate a higher frequency.

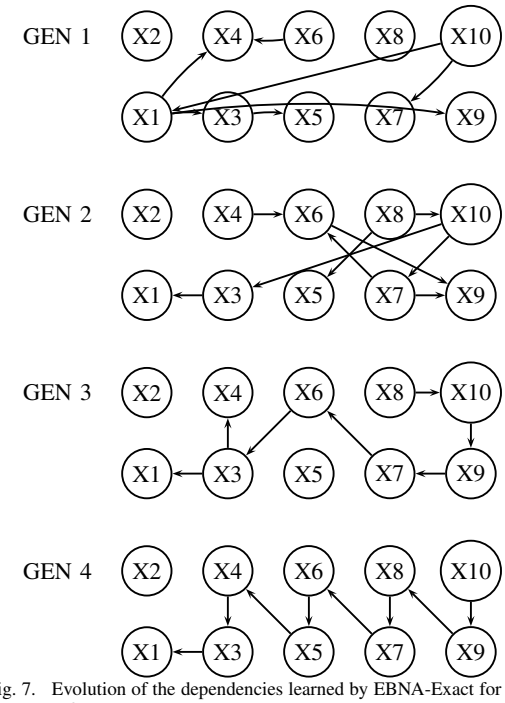
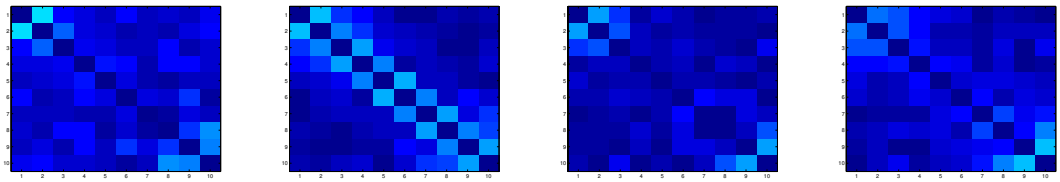



Fig. 7. Evolution of the dependencies learned by EBNA-Exact for function $FourPeaks$.

Figures 8 and 9 show the matrices obtained. The first fact that can be observed is the strong relationship between the structure of the functions and the structure of the Bayesian network. This relationship is specially evident in the networks learned at the last generation of EBNA-Exact for function $FourPeaks$ for which the interactions between adjacent variables are clearly captured. For function $f_{3deceptive}$, both algorithm exhibit a similar behavior. In general, the approximate learning algorithm misses dependencies that are relevant to the search. Therefore, research on more accurate, still computationally simple search strategies is an area worthy of further research.

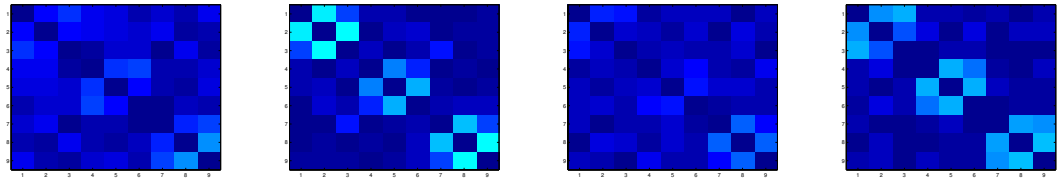
## VI. RELATED WORK

Our work is part of an ongoing research trend that investigates the relationship between the problem structure and the class of structure learned during the search by the probabilistic models. A different but related issue is the effect that the application of EDAs which use probabilistic models



a) b) c) d)

Fig. 8. Matrices of the frequencies of the arcs learned by EBNA-Exact and EBNA-Local for function $FourPeaks$. (a) EBNA-Exact, first population. (b) EBNA-Exact, last population. (c) EBNA-Local, first population. (d) EBNA-Local, last population.



a) b) c) d)

Fig. 9. Matrices of the frequencies of the arcs learned by EBNA-Exact and EBNA-Local for function $f_{3deceptive}$. (a) EBNA-Exact, first population. (b) EBNA-Exact, last population. (c) EBNA-local, first population. (d) EBNA-Local, last population.

with different representation capabilities (e.g. no interactions, bivariate models, Bayesian networks, mixtures of probability distributions, etc.) has for one prespecified problem.

In [31], the question of the necessary conditions for learning good dependencies (those determined by the interactions in the fitness function) and avoiding bad interactions (those due to the selection operator) are theoretically and empirically investigated. This work is extended in [14], where the probabilistic models learned by hBOA for a separable deceptive function and the spin glasses problem are analyzed. For the spin glasses problem, most of the dependencies found by hBOA are short dependencies between neighbors in the grid but some long range interactions also appear. We point out that the approximate learning algorithm may produce models that are only an approximate representation of the actual dependencies that arise in the population. The error introduced by the learning method in the estimation of the dependencies should be taken into account.

The relationship between problem structure and dependencies is analyzed from two different perspectives in [32]. First, using Pearson's chi-square statistics as a measure of the strength of the interactions between pairs of variables in EDAs, the arousal of dependencies due to the selection operator is shown. Second, it is shown that for some problems, only a subset of the dependencies (those associated to malign interactions [33]) may be needed to solve the problem. It is an open question to investigate which is the composition of the dependencies represented by exact Bayesian networks in terms of malign and benign interactions.

Other researchers have studied the most frequent dependencies learned by the probabilistic models in EDAs and analyzed their mapping with the function structure [9], [10], [13], [34], [35]. Interesting related ideas are the use of the dependency relationships represented by the probabilistic model to define functions with desired degree of interactions [36] and the comparison between different classes of factorizations [37], [38] used for solving a particular function.

## VII. Conclusions

We have proposed the use of exact learning of the Bayesian network structure in the study of EDAs. Although the results presented in this paper are still preliminary, we have shown that the type of learning algorithm (whether exact or approximate) may produce significant differences in the class of models learned and the performance of the EBNA. This fact is important because usually Bayesian models learned using approximate algorithms are thought to accurately reflect the dependencies that arise in the population. Whenever the size of the problem is manageable, exact learning of Bayesian networks is a more appropriate option for theoretical analysis of the probabilistic dependencies. Finally, we emphasize that the study of the relationship between the problem structure and the dependencies captured by the probabilistic model should provide answers for the fundamental question of how to select appropriate probabilistic models to optimize a given problem in the framework of EDAs.

## Acknowledgment

This work was supported by the SAIOTEK-Autoinmune (II) 2006 and Etortek research projects from the Basque Government. It has been also supported by the Spanish Ministerio de Ciencia y Tecnología under grant TIN 2005-03824.

## REFERENCES


[1] P. Larrañaga and J. A. Lozano, Eds., *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Boston/Dordrecht/London: Kluwer Academic Publishers, 2002.

[2] H. Mühlenbein and G. Paaß, "From recombination of genes to the estimation of distributions I. Binary parameters," in *Parallel Problem Solving from Nature - PPSN IV*, H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, Eds. Berlin: Springer Verlag, 1996, pp. 178–187, LNCS 1141.

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

[4] D. Heckerman, D. Geiger, and D. M. Chickering, "Learning Bayesian networks: The combination of knowledge and statistical data," *Machine Learning*, vol. 20, pp. 197–243, 1995.

[5] R. Etxeberria and P. Larrañaga, "Global optimization using Bayesian networks," in *Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99)*, A. Ochoa, M. R. Soto, and R. Santana, Eds., Habana, Cuba, 1999, pp. 151–173.

[6] H. Mühlenbein and T. Mahnig, "Evolutionary synthesis of Bayesian networks for optimization," in *Advances in Evolutionary Synthesis of Intelligent Agents*, M. Patel, V. Honavar, and K. Balakrishnan, Eds. Cambridge, Mass.: MIT Press, 2001, pp. 429–455.

[7] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz, "BOA: The Bayesian optimization algorithm," in *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-1999*, W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, Eds., vol. I. Orlando, FL: Morgan Kaufmann Publishers, San Francisco, CA, 1999, pp. 525–532.

[8] M. R. Soto, A. Ochoa, S. Acid, and L. M. Campos, "Bayesian evolutionary algorithms based on simplified models," in *Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99)*, A. Ochoa, M. R. Soto, and R. Santana, Eds., Habana, Cuba, March 1999, pp. 360–367.

[9] R. Höns, "Estimation of Distribution Algorithms and Minimum Relative Entropy," Ph.D. dissertation, University of Bonn, Bonn, Germany, 2006.

[10] H. Mühlenbein and R. Höns, "The estimation of distributions and the minimum relative entropy principle," *Evolutionary Computation*, vol. 13, no. 1, pp. 1–27, 2005.

[11] R. Santana, A. Ochoa, and M. R. Soto, "The mixture of trees factorized distribution algorithm," in *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2001*. San Francisco, CA: Morgan Kaufmann Publishers, 2001, pp. 543–550.

[12] R. Santana, "An analysis of the performance of the mixture of trees factorized distribution algorithm when priors and adaptive learning are used," Institute of Cybernetics, Mathematics and Physics, Havana, Cuba, Tech. Rep. ICIMAF 2002-180, March 2002.

[13] E. Bengoetxea, "Inexact graph matching using estimation of distribution algorithms," Ph.D. dissertation, Ecole Nationale Supérieure des Télécommunications, 2003.

[14] M. Hauschild, M. Pelikan, C. Lima, and K. Sastry, "Analyzing probabilistic models in hierarchical boa on traps and spin glasses," Missouri Estimation of Distribution Algorithms Laboratory, MEDAL Report No. 2007001, 2007.

[15] C. F. Lima, M. Pelikan, D. E. Goldberg, F. G. Lobo, K. Sastry, and M. Hauschild, "Influence of selection and replacement strategies on linkage learning in boa," University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL, IlliGAL Report No. 2007013, 2007.

[16] D. Eaton and K. Murphy, "Exact Bayesian structure learning from uncertain interventions," in *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, 2007.

[17] M. Koivisto and K. Sood, "Exact Bayesian structure discovery in Bayesian networks," *Journal of Machine Learning Research*, vol. 5, pp. 549–573, 2004.

[18] T. Silander and P. Myllymaki, "A simple approach for finding the globally optimal Bayesian network structure," in *Proceedings of the 22th Annual Conference on Uncertainty in Artificial Intelligence (UAI-2006)*. Morgan Kaufmann Publishers, 2006.

[19] A. Singh and A. Moore, "Finding optimal Bayesian networks by dynamic programming," Carnegie Mellon University, Tech. Rep., June 2005.

[20] G. Schwarz, "Estimating the dimension of a model." *Annals of Statistics*, vol. 7, no. 2, pp. 461–464, 1978.

[21] D. M. Chickering, D. Geiger, and D. Heckerman, "Learning Bayesian networks is NP-hard," Microsoft Research, Redmond, WA, Tech. Rep. MSR-TR-94-17, 1994. [Online]. Available: ftp://research.microsoft.com/pub/Tech-Reports/Summer94/TR-94-17.PS

[22] E. Castillo, J. M. Gutierrez, and A. S. Hadi, *Expert Systems and Probabilistic Network Models*. Springer-Verlag, 1997.

[23] W. Buntine, "Theory refinement in Bayesian networks," 1991, pp. 52–60.

[24] J. A. Lozano, R. Sagarna, and P. Larrañaga, "Parallel estimation of Bayesian networks algorithms," in *Evolutionary Computation and Probabilistic Graphical Models. Proceedings of the Third Symposium on Adaptive Systems (ISAS-2001)*, Habana, Cuba, March 2001, pp. 137–144.

[25] R. Blanco and J. A. Lozano, *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Boston/Dordrecht/London: Kluwer Academic Publishers, 2002, ch. An empirical comparison of discrete Estimation of Distribution Algorithms, pp. 163–176.

[26] R. Sagarna and P. Larrañaga, *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Boston Dordrecht London: Kluwer Academic Publishers, 2002, ch. Solving the 0-1 knapsack problem with EDAs, pp. 191–206.

[27] I. Inza, P. Larrañaga, and B. Sierra, *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Boston/Dordrecht/London: Kluwer Academic Publishers, 2002, ch. Feature subset selection by Estimation of Distribution Algorithms, pp. 265–290.

[28] J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, Eds., *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms*. Springer-Verlag, 2006.

[29] S. Baluja and S. Davies, "Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space," in *Proceedings of the 14th International Conference on Machine Learning*. Morgan Kaufmann, 1997, pp. 30–38.

[30] H. Mühlenbein, T. Mahnig, and A. Ochoa, "Schemata, distributions and graphical models in evolutionary optimization," *Journal of Heuristics*, vol. 5, no. 2, pp. 213–247, 1999.

[31] M. Pelikan, K. Sastry, and D. E. Goldberg, "Scalability of the bayesian optimization algorithm," *International Journal of Approximate Reasoning*, vol. 31, no. 3, pp. 221–258, 2002.

[32] R. Santana, P. Larrañaga, and J. A. Lozano, "Interactions and dependencies in estimation of distribution algorithms," in *Proceedings of the 2005 Congress on Evolutionary Computation CEC-2005*. Edinburgh, U.K.: IEEE Press, 2005, pp. 1418–1425.

[33] L. Kallel, B. Naudts, and R. Reeves, "Properties of fitness functions and search landscapes," in *Theoretical Aspects of Evolutionary Computing*, L. Kallel, B. Naudts, and A. Rogers, Eds. Springer Verlag, 2000, pp. 177–208.

[34] R. Santana, "Estimation of distribution algorithms with Kikuchi approximations," *Evolutionary Computation*, vol. 13, no. 1, pp. 67–97, 2005.

[35] R. Santana, P. Larrañaga, and J. A. Lozano, "The role of a priori information in the minimization of contact potentials by means of estimation of distribution algorithms," in *Proceedings of the Fifth European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*, ser. Lecture Notes in Computer Science, E. Marchiori, J. H. Moore, and J. C. Rajapakse, Eds., vol. 4447, 2007, pp. 247–257.

[36] A. Ochoa and M. R. Soto, "Linking entropy to estimation of distribution algorithms," in *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms*, J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, Eds. Springer-Verlag, 2006, pp. 1–38.

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

[38] R. Santana, E. P. de León, and A. Ochoa, "The edge incident model," in *Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99)*, A. Ochoa, M. R. Soto, and R. Santana, Eds., Habana, Cuba, March 1999, pp. 352–359.