PERGAMON

Pattern Recognition 35 (2002) 2867–2880

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Inexact graph matching by means of estimation of distribution algorithms

Endika Bengoetxea[a,*], Pedro Larrañaga[b], Isabelle Bloch[c], Aymeric Perchant[c], Claudia Boeres[d]

[a] *Department of Computer Architecture and Technology, University of the Basque Country, P.O. Box 649, 20080 Donostia, Spain*

[b] *Department of Computer Sciences and Artificial Intelligence, University of the Basque Country, P.O. Box 649, 20080 Donostia, Spain*

[c] *Department of Signal and Image Processing, Ecole Nationale Supérieure des Télécommunications, CNRS URA 820, 46 rue Barrault, 75634 Paris Cedex 13, France*

[d] *Departamento de Informática, Universidade Federal do Rio de Janeiro, Brazil*

Received 27 March 2001; accepted 21 November 2001

Abstract

Estimation of distribution algorithms (EDAs) are a quite recent topic in optimization techniques. They combine two technical disciplines of soft computing methodologies: probabilistic reasoning and evolutionary computing. Several algorithms and approaches have already been proposed by different authors, but up to now there are very few papers showing their potential and comparing them to other evolutionary computational methods and algorithms such as genetic algorithms (GAs). This paper focuses on the problem of inexact graph matching which is NP-hard and requires techniques to find an approximate acceptable solution. This problem arises when a nonbijective correspondence is searched between two graphs. A typical instance of this problem corresponds to the case where graphs are used for structural pattern recognition in images. EDA algorithms are well suited for this type of problems.

This paper proposes to use EDA algorithms as a new approach for inexact graph matching. Also, two adaptations of the EDA approach to problems with constraints are described as two techniques to control the generation of individuals, and the performance of EDAs for inexact graph matching is compared with the one of GAs. © 2002 Pattern Recognition Society. Published by Elsevier Science Ltd. All rights reserved.

Keywords: Inexact graph matching; Estimation of distribution algorithms; Bayesian networks; Genetic algorithms; Hybrid soft computing; Probabilistic reasoning; Evolutionary computing

1. Introduction

Graph representations are widely used for dealing with structural information, in different domains such as networks, psycho-sociology, image interpretation, pattern recognition, etc. One important problem to be solved when using such representations is graph matching. In order to achieve a good correspondence between two graphs, the most used concept is the one of graph isomorphism and a lot of work is dedicated to the search for the best isomorphism between two graphs or subgraphs. However in a number of cases, the bijective condition is too strong, and the problem is expressed rather as an inexact graph matching problem. For instance, inexact graph matching appears as an important area of research in the pattern recognition field, where graph matching is used when the recognition is based on comparison with

* Corresponding author. Tel.: +34-943-018058; fax: +34-943-219306.

E-mail addresses: endika@si.ehu.es (E. Bengoetxea), ccplamup@si.ehu.es (P. Larrañaga), isabelle.bloch@enst.fr (I. Bloch), aymeric.perchant@maunakeatech.com (A. Perchant), boeres@inf.puc-rio.br (C. Boeres).

PII: S0031-3203(01)00232-1

a model: one graph represents the model, and another one the image where recognition has to be performed. Because of the schematic aspect of the model (atlas or map for instance) and of the difficulty to segment accurately the image into meaningful entities, no isomorphism can be expected between both graphs. Such problems call for inexact graph matching. Similar examples can be found in other fields.

When the number of features in the image increases the size of graphs increases too, and the matching process becomes more complex. As this is a NP-hard problem, different combinatorial optimization methods have been tested in order to find the best matching. The optimization process through learning and simulation of Bayesian networks is the method proposed in this article. This approach is known as estimation of distribution algorithms (EDAs). This work also compares their performance in a particular graph matching problem to each other and to broadly used genetic algorithms (GAs).

The outline of the article is as follows: Section 2 describes the graph matching problem analyzed in this article, expressing it as a combinatorial optimization problem with constraints. Section 3 explains the theoretical background behind the EDAs. Section 4 proposes some algorithms within the EDAs that are used later on to test their potential in the graph matching problem. Section 5 describes the experiment and the results obtained. Finally, Section 6 shows the conclusions obtained from the experiments and proposes further work.

2. Graph matching as a combinational optimization problem with constraints

Different techniques have been applied to graph matching: combinatorial optimization techniques [1,2], relaxation techniques [3,4], and the EM algorithm [5,6]. We assume here that we have to match the data graph and a model graph, the first one having more nodes than the second one, as it is usual in model-based pattern recognition for image interpretation [7–13].

We call $G_M=(V_M,E_M)$ the graph representing the model, and $G_D=(V_D,E_D)$ the one representing the data that have to be labelled according to the model, where V_i is the set of nodes and E_i is the set of arcs of graph G_i $(i=M,D)$.

2.1. Representation of individuals

EDAs require to define the format of the individuals that will be used to represent possible solutions to our problem in a similar way as in GAs. We have chosen an individual representation of a size of $|V_D|$ genes or variables, each one taking any value between 1 and $|V_M|$: $\{x=(x_1,\ldots,x_j,\ldots,x_{|V_D|})\}$, where $x_j=i$ $(1\leqslant i\leqslant|V_M|$ and $1\leqslant j\leqslant|V_D|)$ means that the jth node of G_D is matched with the ith node of G_M.

Usually inexact graph matching problems have constraints that have to be satisfied by the final solution in order to be considered as acceptable. The following typical constraints will be considered as examples to illustrate how EDAs are able to take them into account:

- Every node in G_D must have one and only one corresponding match with a node in G_M.
- All the nodes in G_M must have at least a matched node in G_D.

With the chosen representation of individuals, only the last condition needs to be checked as the other is inherent to the representation itself. More formally, an individual will be considered as correct when the following condition is satisfied:

$$\forall i\in\{1,\ldots,|V_M|\},\quad \exists j\in\{1,\ldots,|V_D|\}\,|\,x_j=i. \tag{1}$$

2.2. Definition of the fitness function

The aim of this paper is not to test the goodness of the fitness function, nor to give a comparison of different fitness functions. Here we use the simple function proposed in [11], which gives to every individual $\boldsymbol{x}=(x_1,\ldots,x_{|V_D|})$ a fitness value as follows:

$$\begin{aligned} &f(\mathbf{x};\rho_\sigma,\rho_\mu,\alpha)\\ &=\left\{\left[\frac{\alpha}{|V_D||V_M|}\sum_{i=1}^{|V_D||V_M|}\sum_{j=1}\{(1-|c_{ij}-\rho_\sigma^{u_M^i}(u_D^j)|)\}\right]\right\}\\ &\quad\times\left\{\left[\frac{1-\alpha}{|E_D||E_M|}\sum_{e_M^l\in E_M}\sum_{e_D^k\in E_D}\{(1-|c_{ij}c_{i'j'}-\rho_\mu^{e_M^1}(e_D^k)|)\}\right]\right\} \end{aligned} \tag{2}$$

where if $x_i=j$ then $c_{ij}=1$, otherwise $c_{ij}=0$. u_M^i and e_M^i are the ith node and lth edge of the graph G_M, respectively, and analogously u_D^j and e_D^k are the jth node and kth edge of the graph G_D. α is a parameter used to adapt the weight of node and edge correspondences in f, and ρ_σ measures the similarity between the nodes of both graphs G_M and G_D. In the same way, ρ_μ measures the similarity between the arcs of both graphs G_M and G_D. Usually these similarities are based on attributes of nodes and edges. The fitness function can be easily understood by having a look to the two main terms: the first one measures the correspondence between nodes of the model and data graphs, and the second the correspondence between edges of both graphs. The value f associated to each individual returns the goodness of the matching it represents.

3. Estimation of distribution algorithms (EDAs)

3.1. Introduction

Generally speaking, all the search strategy types can be classified as complete and heuristic strategies. The difference between them is that complete strategies perform a systematic examination of all possible solutions of the search space whereas heuristic strategies only concentrate on a part of them following a known algorithm.

Heuristic strategies are also divided between deterministic and non-deterministic. The characteristic of deterministic strategies is that under the same conditions the same solution is always obtained. Examples of this type are forward, backward, stepwise, hill-climbing, threshold accepting, and other well known algorithms, and their main drawback is that they can always get stuck in local maximum values. Non-deterministic search is able to escape from these local maxima by means of the randomness and, due to their stochasticity, different executions might lead to different solutions under the same conditions.

Some of the stochastic heuristic searches such as simulated annealing only store one solution in every iteration of the algorithm. The stochastic heuristic searches that store more than a solution every iteration (or every generation) are grouped under the term of population-based heuristics, an example of which is evolutionary computation. On these, each of the solutions is called *individual*. The group of individuals (also known as *population*) evolves towards more promising areas of the search space while the algorithm carries on with the next generation. GAs are examples of evolutionary computation [14,15].

The behaviour of GAs depends to a large extent on associated parameters like operators of crossing and mutation, probabilities of crossing and mutation, size of the population, rate of generational reproduction, the number of generations, and so on. The researcher requires experience in the resolution and use of these algorithms in order to choose the suitable values for these parameters. Furthermore, the task of selecting the best choice of values for all these parameters can be considered itself as an optimization problem [16]. In addition, GAs show a poor performance in some problems (e.g. deceptive problems) in which the designed operators of crossing and mutation do not guarantee that the building block hypothesis is preserved.

All these reasons have motivated the creation of a new approach classified under the name of Estimation of Distribution Algorithms (EDA) [17–19], trying to make easier to predict the movements of the populations in the search space as well as to avoid the need for so many parameters. These algorithms are also based on populations that evolve as the search progresses and, as well as GAs, they have a theoretical foundation on the probability theory. In brief, EDA are population-based search algorithms based on probabilistic modelling of promising solutions in combination with the simulation of the induced models to guide their search.

In EDA the new population of individuals is not generated by using crossover nor mutation operators. Instead, the new individuals are sampled starting from a probability distribution estimated from the database containing only selected individuals from the previous generation. Also, while in other heuristics from evolutionary computation the interrelations between the different variables representing the individuals are kept in mind implicitly (e.g. building block hypothesis), in EDA the interrelations are expressed explicitly through the joint probability distribution associated with the individuals selected in each iteration. In fact, the task of estimating the joint probability distribution associated with the database of the selected individuals from the previous generation constitutes the hardest work to perform. In particular, the latter requires the adaptation of methods to learn models from data that have been developed by researchers in the domain of probabilistic graphical models.

Fig. 1 illustrates the EDA approach.

(1) Firstly, the first population D_0 of N individuals is generated. The generation of these N individuals is usually done by assuming a uniform distribution on each variable, and next each individual is evaluated.
(2) Secondly, a number Se ($Se \leqslant N$) of individuals are selected following a criterion (usually the ones with the best fitness value are selected).
(3) Thirdly, the n-dimensional probabilistic model that better reflects the interdependencies between the n variables is induced.
(4) Finally, the new population constituted by the N new individuals is obtained by carrying out the simulation of the probability distribution learnt in the previous step.

Steps 2–4 are repeated until a stopping condition is verified. Examples of stopping conditions are: achieving a fixed number of populations or a fixed number of different evaluated individuals, uniformity in the generated population, and the fact of not obtaining an individual with a better fitness value after a certain number of generations.

3.2. Notations

This section introduces the notation that will be used to describe EDAs through the rest of the paper.

Let X_i, $i = 1,\ldots,n$, be a random variable. A possible instantiation of X_i will be denoted by x_i. $p(X_i = x_i)$ (or simply $p(x_i)$) will denote the probability for the variable X_i over the point x_i. Similarly, $\boldsymbol{X} = (X_1,\ldots,X_n)$ will represent a n-dimensional random variable, and $\boldsymbol{x} = (x_1,\ldots,x_n)$ one of its possible instantiations. The probability of $\boldsymbol{X}$ will be denoted $p(\boldsymbol{X} = \boldsymbol{x})$ (or simply $p(\boldsymbol{x})$). The conditional probability of the variable X_i given the value x_j of the variable X_j will be written as $p(X_i = x_i|X_j = x_j)$ (or simply as $p(x_i|x_j)$). D will denote a data set, i.e. a set of N instantiations of the variables $(X_1,\ldots,X_n)$.

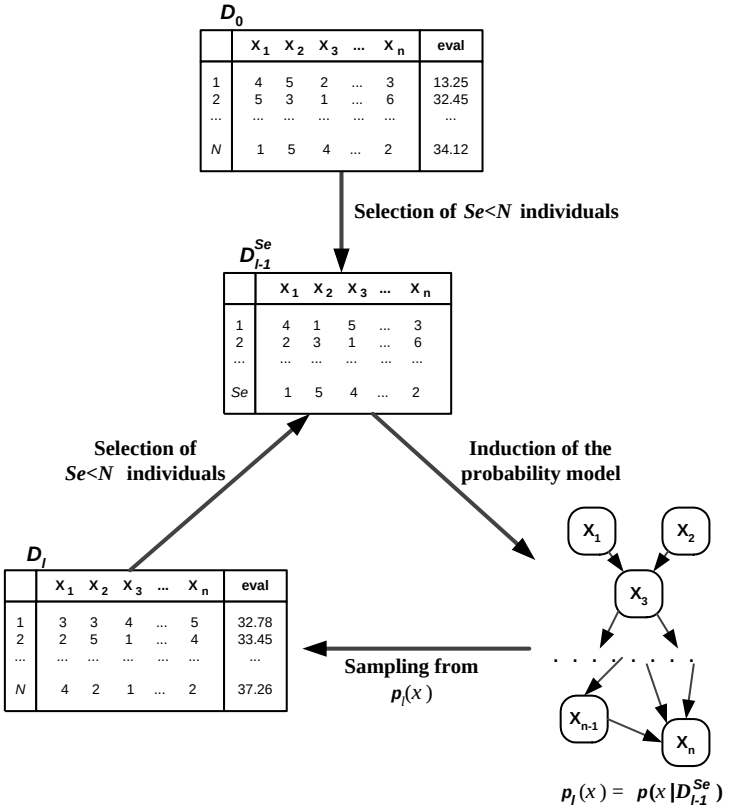

Fig. 1. Illustration of the EDA approach in the optimization process.

Fig. 2 shows the pseudocode of EDA in combinatorial optimization problems using the notations introduced, where $\boldsymbol{x}=(x_1,\ldots,x_n)$ represents the individuals of n genes, and D_l denotes the population of N individuals in the lth generation. Similarly, D_l^{Se} represents the population of the selected Se individuals from D_l. In EDA the main task is to estimate $p(\boldsymbol{x}\,|\,D_l^{Se})$, that is, the probability for one individual $\boldsymbol{x}$ to be among the selected individuals. This probability must be estimated in every generation. We will denote $p_l(\boldsymbol{x}) = p(\boldsymbol{x}|D_{l-1}^{Se})$ the probability of the lth generation.

The most difficult step for EDA is actually to estimate satisfactorily the probability distribution $p_l(\boldsymbol{x})$, as the computation of all the parameters needed to specify the underlying probability model becomes impractical. That is why several approximations propose to factorize the probability distribution according to a probability model.

3.3. Bayesian networks

This section introduces the probabilistic graphical model paradigm [20–22] that has been used during the last decade as a popular representation for encoding uncertainty knowledge in expert systems [23]. Only probabilistic graphical models whose structural part is a directed acyclic graph will be considered, as these adapt properly to EDAs.

Let $\boldsymbol{X}=(X_1,\ldots,X_n)$ be a set of random variables, and let x_i be a value of X_i, the ith component of $\boldsymbol{X}$. Then, a probabilistic graphical model for $\boldsymbol{X}$ is a graphical factorization of the joint generalized probability density function, $\rho(\boldsymbol{X}=\boldsymbol{x})$ (or simply $\rho(\boldsymbol{x})$). The representation of this model is given by two components: a structure and a set of local generalized probability densities.

With regard to the structure of the model, the structure S for $\boldsymbol{X}$ is a directed acyclic graph (DAG) that describes a set of conditional (in)dependencies [24] about the variables on $\boldsymbol{X}$. $\boldsymbol{Pa}_i^S$ represents the set of parents—variables from which an arrow is coming out in S—of the variable X_i in the probabilistic graphical model whose structure is given by S. The structure S for $\boldsymbol{X}$ assumes that X_i and $\{X_1,\ldots,X_{i-1}\} \setminus \{\boldsymbol{Pa}_i^S\}$ are independent given $\boldsymbol{Pa}_i^S$, $i=2,\ldots,n$. Therefore, the factorization can be written as follows:

$$\rho(\boldsymbol{x}) = \rho(x_1,\ldots,x_n) = \prod_{i=1}^{n} \rho(x_i\,|\,\boldsymbol{pa}_i^S). \tag{3}$$

A representation of the models of the characteristics described above assumes that the local generalized probability

EDA

$D_0 \leftarrow$ Generate N individuals (the initial population) randomly

Repeat for $l = 1, 2, \ldots$ until a stopping criterion is met

$D_{l-1}^{Se} \leftarrow$ Select $Se \leq N$ individuals from D_{l-1} according to a selection method

$p_l(\boldsymbol{x}) = p(\boldsymbol{x}|D_{l-1}^{Se}) \leftarrow$ Estimate the probability distribution of an individual being among the selected individuals

$D_l \leftarrow$ Sample N individuals (the new population) from $p_l(\boldsymbol{x})$

Fig. 2. Pseudocode for EDA approach.

densities depend on a finite set of parameters $\boldsymbol{\theta}_S \in \boldsymbol{\Theta}_S$, and as a result the previous equation can be rewritten as follows:

$$\rho(\boldsymbol{x} \mid \boldsymbol{\theta}_S) = \prod_{i=1}^{n} \rho(x_i \mid \boldsymbol{pa}_i^S, \boldsymbol{\theta}_i), \tag{4}$$

where $\boldsymbol{\theta}_S = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n)$.

After having defined both components of the probabilistic graphical model, and taking them into account, the model itself will be represented by $M = (S, \boldsymbol{\theta}_S)$.

In the particular case of every variable $X_i \in \boldsymbol{X}$ being discrete, the probabilistic graphical model is called *Bayesian network*. If the variable X_i has r_i possible values, $x_i^1, \ldots, x_i^{r_i}$, the local distribution, $p(x_i \, \boldsymbol{pa}_i^{j,S}, \boldsymbol{\theta}_i)$ is an unrestricted discrete distribution:

$$p(x_i^k \mid \boldsymbol{pa}_i^{j,S}, \boldsymbol{\theta}_i) = \theta_{x_i^k | \boldsymbol{pa}_i^j} \equiv \theta_{ijk} \tag{5}$$

where $\boldsymbol{pa}_i^{1,S}, \ldots, \boldsymbol{pa}_i^{q_i,S}$ denotes the values of $\boldsymbol{Pa}_i^S$, and q_i is the number of different possible instantiations of the parent variables of X_i. Thus, $q_i = \prod_{X_g \in \boldsymbol{Pa}_i} r_g$. The local parameters are given by $\boldsymbol{\theta}_i = ((\theta_{ijk})_{k=1}^{r_i})_{j=1}^{q_i})$. In other words, the parameter θ_{ijk} represents the conditional probability of variable X_i to take its kth value x_i^k, knowing that the set of its parent variables take their jth combination of values. We assume that every θ_{ijk} is strictly greater than zero.

3.4. Existent EDA in combinatorial optimization

In this subsection some EDA approaches for combinatorial optimization problems that can be found in the literature are commented. All the algorithms and methods are classified depending on the maximum number of dependencies between variables that they accept (maximum number of parents that a variable X_i can have in the probabilistic graphical model). The reader can find in [18] a more complete review of this topic.

3.4.1. Without interdependencies

All the papers belonging to this category assume that the n-dimensional joint probability distribution factorizes like a product of n univariate and independent probability distributions. This assumption appears to be inexact from the nature of any difficult optimization problem, where interdependencies between the variables will exist to some degree. Nevertheless, this approximation can lead to a good enough behavior in EDAs in some problems.

Several approaches that correspond to this category can be found in the literature, such as bit-based simulated crossover (BSC) [25], population-based incremental learning (PBIL) [26], the compact genetic algorithm [27], and the univariate marginal distribution algorithm (UMDA) [28]. Section 4.2.1 explains this algorithm in more detail.

3.4.2. Pairwise dependencies

In an attempt to express the simplest possible interdependencies between variables, all the papers in this category propose that the joint probability distribution can be estimated well and fast enough by only taking into account dependencies between pairs of variables.

Algorithms in this category require therefore an additional step that was not required in the previous class, which is the construction of a structure that best represents the probabilistic model. In other words, the parametric learning of the previous category—where the structure of the model remains fixed—is extended to structural learning.

An example of this second category is the greedy algorithm called MIMIC (mutual information maximization for input clustering) proposed in Ref. [29]. Later on in Section 4.2.2 the algorithm MIMIC will be explained in more detail. Other approaches in this group are the ones proposed in Ref. [30] and the one called BMDA (bivariate marginal distribution algorithm) [31].

3.4.3. Multiple interdependencies

Several other EDA approaches in the literature propose the factorization of the joint probability distribution to be done by statistics of order greater than two. As the number of dependencies between variables is greater than in the previous categories, the complexity of the probabilistic structure as well as the task of finding the best structure that suits the model is higher. Therefore, these approaches require a more complex learning process.

The most important EDA approaches that can be found in the literature within this category are as follows: FDA (factorized distribution algorithm) introduced in Ref. [32], EBNA (estimation of Bayesian networks algorithm) [33], BOA (Bayesian optimization algorithm) [34], LFDA (learning factorized distribution algorithm) introduced in Ref. [35]

that follows essentially the same approach as in EBNA, and the extend compact genetic algorithm (EcGA) proposed in Ref. [36].

4. Proposed EDA approaches for inexact graph matching

4.1. Notation of EDAs applied to graph matching

We will define more formally the inexact graph matching problem and the way of facing it in an EDA approach.

Let $G_M=(V_M,E_M)$ be the model graph, and $G_D=(V_D,E_D)$ the data graph to be matched. The individuals have $n=|V_D|$ (that is, $\boldsymbol{X}=(X_1,\ldots,X_{|V_D|})$) variables, each of them taking $|V_M|$ possible values. We denote by $x_i^1,\ldots,x_i^{|V_M|}$ the possible values that the ith variable, X_i, can take.

In the same way, applying the notation of Section 3.3 for this problem, we have that the number of possible values for each variable $i=1,\ldots,|V_D|$ is $r_i=|V_M|$. Therefore, for the unrestricted discrete distribution θ_{ijk} the range of i, j and k is as follows: $i=1,\ldots,|V_D|$, $k=1,\ldots,|V_M|$, and $j=1,\ldots,q_i$, where $q_i=|V_M|^{npa_i}$ and npa_i denotes the number of parents of X_i.

4.2. Estimating the probability distribution

We propose three different EDAs to be used in inexact graph matching. Due to the fact that the different behaviors of the algorithms are to a large extent due to the complexity of the probabilistic structure that they have to build, these algorithms can be seen therefore as representatives of the three categories of EDA introduced in Section 3.4: (1) UMDA [28] as an example of an EDA that considers no interdependencies between the variables (i.e. the learning is only parametrical, not structural); (2) MIMIC [29] as an example of algorithms that consider pairwise dependencies and (3) EBNA [33] as an example of the category of EDAs where multiple interdependencies are allowed between the variables, and for which the structural learning is even more complex than in the previous algorithm.

4.2.1. UMDA—univariate marginal distribution algorithm

This algorithm assumes all the variables to be independent in order to estimate the probability distribution. More formally, the UMDA approach can be written as

$$p_l(\boldsymbol{x};\boldsymbol{\theta}^l)=\prod_{i=1}^{n}p_l(x_i;\boldsymbol{\theta}) \tag{6}$$

where $\boldsymbol{\theta}^l=\{\theta_{ijk}^l\}$ is recalculated every generation by its maximum likelihood estimation, i.e. $\hat{\theta}_{ijk}^l=N_{ijk}^{l-1}/N_{ij}^{l-1}$. N_{ijk}^{l-1} is the number of cases in which the variable X_i takes the value x_i^k when its parents take their jth combination of values for the $l-1^{\text{th}}$ generation, and $N_{ij}^{l-1}=\sum_k N_{ijk}^{l-1}$.

4.2.2. MIMIC—mutual information maximization for input clustering

The main idea in MIMIC [29] is to describe the true probability as closely as possible by using only one univariate marginal probability and $n-1$ pairwise conditional probability functions.

Given a permutation $\pi=(i_1,\ldots,i_n)$,

$$p_\pi(\boldsymbol{x})=p(x_{i_1}|x_{i_2})\cdot p(x_{i_2}|x_{i_3})\cdot\ldots\cdot p(x_{i_{n-1}}|x_{i_n})\cdot p(x_{i_n}), \tag{7}$$

where $p(x_{i_n})$ and $p(x_{i_j}|x_{i_{j+1}})$, $j=1,\ldots,n-1$, are estimated by the marginal and conditional relative frequencies of the correspondent variables within the subset of selected individuals D_{l-1}^{Se} in the lth generation. The goal for MIMIC is to choose the appropriate permutation π^* such that $p_{\pi^*}(\boldsymbol{x})$ minimizes the Kullback–Leibler information divergence between the true probability function, $p(\boldsymbol{x})$, and the probability functions, $p_\pi(\boldsymbol{x})$.

This Kullback–Leibler information divergence can be expressed by means of the Shanon entropy of a probability function, $h(p(\boldsymbol{x}))$, in the following way:

$$\begin{aligned}D_{K-L}(p(\boldsymbol{x}),p_\pi(\boldsymbol{x}))&=\sum_{\boldsymbol{x}}p(\boldsymbol{x})\log\frac{p(\boldsymbol{x})}{p_\pi(\boldsymbol{x})}\\&=-h(p(\boldsymbol{x}))+h(X_{i_1}|X_{i_2})+h(X_{i_2}|X_{i_3})\\&\quad+\cdots+h(X_{i_{n-1}}|X_{i_n})+h(X_{i_n})\end{aligned} \tag{8}$$

where $h(X\mid Y)$ denotes the mean uncertainty in X given Y, that is $h(X\mid Y)=\sum_y h(X|Y=y)p_Y(y)$ and $h(X|Y=y)=-\sum_x h(X=x|Y=y)\log p_{X|Y}(x|y)$ expresses the uncertainty in X given that $Y=y$.

The latter equation can be rewritten considering that $D_{K-L}(p(\boldsymbol{x}),p_\pi(\boldsymbol{x}))$ does not depend on π. Therefore, the task to accomplish is to find the sequence π^* that minimizes the expression

$$J_\pi(\boldsymbol{x})=h(X_{i_1}\mid X_{i_2})+\cdots+h(X_{i_{n-1}}\mid X_{i_n})+h(X_{i_n}). \tag{9}$$

In Ref. [29] the authors prove that it is possible to find an approximation of π^* avoiding the need to search over all $n!$ permutations by using a straightforward greedy algorithm. The idea consists in selecting firstly X_{i_n} as the variable with the smallest estimated entropy, and then in successive steps to pick up the variable—from the set of variables not chosen so far—whose average conditional entropy with respect to the previous is the smallest.

4.2.3. EBNA—estimation of Bayesian network algorithm

EBNA is an EDA proposed in Ref. [33] that belongs to the category of algorithms that take into account multiple interdependencies between variables. This algorithm proposes the construction of a probabilistic graphical model with no restriction in the number of parents that variables can have.

EBNA is based on the penalized maximum likelihood score. In this algorithm, given a database D with N cases,

$D = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, a measure of the success of any structure S to describe the observed data D is proposed. This measure is obtained by computing the maximum likelihood estimate $\hat{\boldsymbol{\theta}}$ for the parameters $\boldsymbol{\theta}$ and the associated maximized log likelihood, $\log p(D|S, \hat{\boldsymbol{\theta}})$. The main idea in EBNA is to search for the structure that maximizes $\log p(D|S, \boldsymbol{\theta})$ using an appropriate search strategy. This is done by scoring each structure by means of its associated maximized log likelihood. The theoretical foundations of this intuitively appealing approach are based on the consistency and the asymptotic efficiency properties of the maximum likelihood estimates. Using the notations introduced in Section 3.3, we obtain

$$\begin{aligned}&\log p(D \mid S, \boldsymbol{\theta})\\ &= \log \prod_{w=1}^{N} p(\boldsymbol{x}_w \mid S, \boldsymbol{\theta}) = \log \prod_{w=1}^{N} \prod_{i=1}^{n} p(x_{w,i} | \boldsymbol{pa}_i^S, \boldsymbol{\theta}_i)\\ &= \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} \log(\theta_{ijk})^{N_{ijk}}, \end{aligned} \tag{10}$$

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

Knowing that the maximum likelihood estimate for θ_{ijk} is given by $\widehat{\theta_{ijk}} = N_{ijk}/N_{ij}$, the maximum of the previous equation can be rewritten as

$$\log p(D \mid S, \hat{\boldsymbol{\theta}}) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \log \frac{N_{ijk}}{N_{ij}}. \tag{11}$$

For the case of complex models, the sampling error associated with the maximum likelihood estimator might turn out to be too big to consider the maximum likelihood estimate as a reliable value for the parameter—even for a large sample. A common response to this difficulty is to incorporate some form of penalty depending on the complexity of the model into the maximized likelihood. Several penalty functions have been proposed. A general formula for a penalized maximum likelihood score can be the following:

$$\sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \log \frac{N_{ijk}}{N_{ij}} - f(N) dim(S) \tag{12}$$

where $dim(S)$ is the dimension—number of parameters needed to specify the model—of the Bayesian network with structure given by S. It is computed as $dim(S) = \prod_{i=1}^{n} q_i(r_i - 1)$. This penalization function $f(N)$ is a non negative one. Some examples for $f(N)$ are the Akaike's information criterion (AIC) [37] —where $f(N) = 1$—, and the Jeffreys–Schwarz criterion, sometimes called the Bayesian information criterion (BIC) [38]— where $f(N) = \frac{1}{2} \log N$.

Following the latter criterion, the corresponding BIC score—$BIC(S, D)$—for a Bayesian network structure S constructed from a database D and containing N cases is as follows:

$$\begin{aligned}&BIC(S, D)\\ &= \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \log \frac{N_{ijk}}{N_{ij}} - \frac{\log N}{2} \sum_{i=1}^{n} (r_i - 1) q_i \end{aligned} \tag{13}$$

where N_{ijk} and N_{ij} and q_i are defined as above.

On the other hand, the local probability distributions θ_{ijk} in EBNA are calculated every generation using their expected values as obtained in [39]:

$$E[\theta_{ijk}^{l} \mid S, D_{l-1}^{Se}] = \frac{N_{ijk}^{l-1} + 1}{N_{ij}^{l-1} + r_i}. \tag{14}$$

Unfortunately, to obtain the best model all possible structures must be searched through, which has been proved to be NP-hard [40]. Even if promising results have been obtained through global search techniques [41–43], their computation cost makes them impractical for our problem. As the aim is to find a model as good as possible—even if not the optimal—in a reasonable period of time, a simpler algorithm is preferred. An example of the latter is the so-called Algorithm B [44].

Local search strategies are another way of obtaining good models. These start from a given structure, and every step the addition or deletion of an arc that improves most the scoring measure is performed. Local search strategies stop when no modification of the structure improves the scoring measure. The main drawback of local search strategies is their heavy dependence on the initial structure. Nevertheless, as Ref. [45] showed that local search strategies perform quite well when the initial structure is reasonably good, the model of the previous generation could be used as the initial-break structure when the search is based on the assumption that $p(\boldsymbol{x} \mid D_l^{Se})$ will not differ very much from $p(\boldsymbol{x}|D_{l-1}^{Se})$.

The initial model M_0 in EBNA is formed by its structure S_0—an arc-less DAG— and the local probability distributions given by the n unidimensional marginal probabilities $p(X_i = x_i) = \frac{1}{|V_M|}$, $i = 1, \ldots, n$—that is, M_0 assigns the same probability for all individuals. The model of the first generation $-M_1-$ is learned using Algorithm B, while the rest of the models are learnt by means of a local search strategy that received the model of the previous generation as initial structure.

4.3. Adapting the simulation scheme

The simulation of Bayesian networks can be regarded as an alternative to exact propagation methods that were developed to reason with networks. This method creates a database with the probabilistic relations between the different variables previous to other procedures. In our particular case, the simulation of Bayesian networks is used merely as a tool to generate new individuals for the next population based on the structure learned previously.

The method used in this paper is the *probabilistic logic sampling* (PLS) [46]. Following this method, the instantia-

tions are done one variable at a time in a forward way, that is, a variable is not sampled until all its parents have already been so. This requires previously to order all the variables from parents to children —any ordering of the variables satisfying such a property is known as ancestral ordering. We will denote $\boldsymbol{\pi} = (\pi(1), \ldots, \pi(|V_D|))$ an ancestral order compatible with the structure to be simulated. The concept of forward means that the variables are instantiated from parents to children. Once the values of the parent variables of a variable $X_i - \boldsymbol{pa}_i$— have been assigned, the values for X_i will be simulated using the distribution $p(x_i|\boldsymbol{pa}_i)$.

4.3.1. Techniques to obtain correct individuals

Ensuring that the final solution is a correct individual is important, as it is necessary to return a solution that satisfies Eq. (1). For this, four techniques are introduced in this section: two techniques that control directly the simulation step, a technique that corrects automatically incorrect individuals, and a technique of changing the fitness value in order to penalize incorrect individuals. GAs need also to face the same problem as EDAs when using the same representation, but from these four techniques only the last two can be applied to GAs.

4.3.2. Controlling directly the simulation step

Up to now in most of the problems where EDAs have been applied no constraints had to be taken into account. This is the reason why very few articles about modifying the simulation step for this purpose can be found [47]. Two different ways of modifying the simulation step are introduced in this paper. It is important to note that altering the probabilities at the simulation step, whichever the way, implies that the learning of the algorithm is also denatured somehow. It is therefore very important to make sure that the manipulation is only performed to *guide* the generation of potentially incorrect individuals towards correct ones.

4.3.2.1. Last time manipulation (LTM) This method consists in altering the simulation step during the generation of the individual. This alteration is not performed until the number of nodes of G_M remaining to be matched and the number of variables to be simulated in the individual are equal. For instance, this could happen when three nodes of G_M have not been matched yet and the value of the last three variables is to be calculated for an individual. In this case, we will force the simulation step so that only these three values could be sampled in the next variable.

In order to force the next variable of the individual to take only one of the values not still appeared, the value of the probabilities that are used to perform the simulation is changed. In this way, the probability of all the values already appeared in the individual is set to 0 and the probabilities of the values not still appeared are normalized accordingly. More formally, the procedure to generate an individual will follow the ancestral ordering $\boldsymbol{\pi} = (\pi(1), \pi(2), \ldots, \pi(|V_D|))$, that is, all the variables will be instantiated in the following order: $(X_{\pi(1)}, X_{\pi(2)}, \ldots, X_{\pi(|V_D|)})$. If we are instantiating the mth variable (we are sampling $X_{\pi(m)}$), we firstly define $NNO(V_M)^m = \{u_M^i \in V_M \,|\, X_j \in \boldsymbol{X}j \in \{\pi(1), \ldots, \pi(m-1)\}, X_j = i\}$ the set that contains all the nodes u_M^i of G_M not yet matched in the individual in the previous $m-1$ steps (NNO stands for nodes not obtained), and secondly we consider $vns^m = |V_D| - m$ (which is the number of variables still to be simulated). According to Eq. (5), $\theta_{\pi(m)lk}$ is the probability of the variable $X_{\pi(m)}$ to take the value k knowing that its parents have already taken their lth possible combination of values —as $\boldsymbol{\pi}$ follows an ancestral ordering, we know that the parent variables of $X_{\pi(m)}$ have already been instantiated in one of the previous $m-1$ steps. Therefore, $P_{Indiv}^m = \sum_{k|u_M^k \in V_M \setminus NNO(V_M)^m} \theta_{\pi(m)lk}$ will be the sum of all the probabilities of variable X_m to take any value already instantiated in one of the previously simulated variables.

Having these definitions, following the LTM method the $\theta_{\pi(m)lk}$ values will be modified while the condition $|NNO(V_M)^m| = vns^m$ is satisfied. When this is the case, the probability of the variable $X_{\pi(m)}$ to take the value k knowing that its parents take their lth combination of values, $\theta^*_{\pi(m)lk}$, will be modified as follows:

$$\theta^*_{\pi(m)lk} = \begin{cases} \theta_{\pi(m)lk} \cdot \dfrac{1}{1 - P_{Indiv}^m} & \text{if } u_M^{\pi(m)} \in NNO(V_M)^m, \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

Once the probabilities have been modified, it is guaranteed that the only values assigned to $X_{\pi(m)}$ will be one of the nodes of V_M not still obtained (a node from $NNO(V_M)^m$), as the probability to obtain any node from $V_M \setminus NNO(V_M)^m$ has been set to 0. This modification of the $\theta_{\pi(m)lk}$ has to be repeated for the rest of the variables that remain to be instantiated in the individual $(X_{\pi(m+1)}, \ldots, X_{\pi(|V_D|)})$, but for the successive steps there are more values whose probabilities have to be set to 0 and P_{Indiv}^m must be recomputed. Following this method, when instantiating the last variable $X_{\pi(|V_D|)}$, if v is the only value that is missing in the individual, then the probability to take this value v will have its probability set to $\theta^*_{\pi(|V_D|)lv} = 1$, and for the rest of the values $\theta^*_{\pi(|V_D|)lw} = 0 \ \forall w \neq v$. Therefore, the only value that can be assigned to the variable $X_{\pi(|V_D|)}$ will be v.

With this technique the probabilities of the variables are not modified until the condition $|NNO(V_M)^m| = vns^m$ is satisfied. Therefore, the simulation step will behave as in the PLS simulation procedure, without any external manipulation unless the latter condition is not satisfied.

4.3.2.2. All time manipulation (ATM) This second technique is another way of manipulating the probabilities of the values for each variable, but this time the manipulation takes place from the beginning of the generation of the individual. The value of the probabilities remains unaltered only after all the possible values of the variables have al-

ready appeared in the individual (that is, when the condition $NNO(V_M)=\emptyset$ is satisfied).

For this, again the order of sampling the variables $\boldsymbol{\pi}$ will be followed, instantiating them in the same order $(X_{\pi(1)},X_{\pi(2)},\ldots,X_{\pi(|V_D|)})$. At each step the probabilities of a variable will be modified before its instantiation. The required definitions for the mth step (the sampling of the variable $X_{\pi(m)}$) are as follows: let $|V_D|$ be the number of variables of each individual, let also be $NNO(V_M)^m$, vns^m, and $\theta_{\pi(m)lk}$ as defined before. The latter probability will be modified with this method obtaining the new $\theta^*_{\pi(m)lk}$ as follows:

$$\theta^*_{\pi(m)lk}=\begin{cases}\theta_{\pi(m)lk}\cdot\frac{K-P^m_{Indiv}}{K\cdot(1-P^m_{Indiv})} & \text{if } u^i_M\in NNO(V_M)^m \text{ and}\\ & |NNO(V_M)^m|\neq vns^m\\ \frac{\theta_{\pi(m)lk}}{K} & \text{if } u^i_M\notin NNO(V_M)^m \text{ and}\\ & |NNO(V_M)^m|\neq vns^m\\ \theta_{\pi(m)lk}\cdot\frac{1}{1-P^m_{Indiv}} & \text{if } u^i_M\in NNO(V_M)^m \text{ and}\\ & |NNO(V_M)^m|= vns^m\\ 0 & \text{if } u^i_M\notin NNO(V_M)^m \text{ and}\\ & |NNO(V_M)^m|= vns^m,\end{cases}\tag{16}$$

where $K=\lceil\frac{N-vns^m}{vns^m-|NNO(V_M)^m|}\rceil$,

and $P^m_{Indiv}=\sum_{u^k_m\in V_M\setminus NNO(V_M)^m}\theta_{\pi(m)lk}$.

The reason to modify the probabilities in this way is that, at the beginning, the probability for all the values to appear in at least a variable of the individual is higher (as vns^m is usually bigger than $|NNO(V_M)^m|$), and therefore the method does not have to modify the probabilities very much. Only when $|NNO(V_M)^m|$ starts to be very close to vns^m will the effect of the manipulation in the probabilities be stronger, meaning that there are less variables to instantiate and therefore the possibility for all the missing values to appear is also smaller. Finally, when $|NNO(V_M)^m|=vns^m$, only the values not appeared yet have to be selected. For this, the probabilities of the values already appeared are set to 0, and the other ones are modified in the same way as in the previous method.

This second technique modifies the probabilities nearly from the beginning, giving more chance to the values not already appeared, but it also takes into account the probabilities learned by the Bayesian network in the learning step. It does not modify the probabilities in any way when $|NNO(V_M)^m|=0$, that is, when all the values have already appeared in the individual.

4.3.3. Correction of individuals after the simulation step

This technique is completely different from the ones proposed before, as it is not based on modifying the probabilities generated by the algorithm at all: the idea is to correct the individuals that do not contain an acceptable solution to the problem after they have been completely generated. In order to do this correction, once the individual has been completely generated and has been identified as not correct ($|NNO(V_M)^{|V_D|}|>0$), a variable which contains a value that appears more than once in the individual is chosen randomly and substituted by one of the missing values. This task is performed $|NNO(V_M)^{|V_D|}|$ times, that is, until the individual is correct.

The fact that no modification is done at all in the learned probabilities means that this method does not demerit the learning process, and thus the learning process is respected as when using PLS. As the generation of the individuals is not modified at all with respect to PLS, the only manipulation occurs on the wrong individuals, and the algorithm can be supposed to require less generations to converge to the final solution. Furthermore, this method can also be used with other evolutionary computation techniques such as GAs.

4.3.4. Penalization of wrong individuals

Finally, this last method is not based on modification of the probabilities during the process of the generation of the new individuals either. The idea is completely different and consists in applying a penalization on the fitness value of each individual.

For the experiments explained in Section 5, the penalization has been performed as follows: if $f(\boldsymbol{x})$ is the value obtained by the fitness function for the individual $\boldsymbol{x}=(x_1,\ldots,x_{|V_D|})$, and if $|NNO(V_M)^{|V_D|}|$ is the number of nodes of G_M not present in the individual, the modified fitness value, $f^*(\boldsymbol{x})$ will be changed as follows:

$$f^*(\boldsymbol{x})=\frac{f(\boldsymbol{x})}{|NNO(V_M)^{|V_D|}|+1}.\tag{17}$$

Another important difference with respect to the other methods to control the generation of individuals explained so far, the penalization does allow the generation of incorrect individuals, and therefore these will still appear in the successive generations. This aspect has to be analyzed for every problem, and so it is in this paper. Nevertheless, as these incorrect individuals will be given a lower fitness value it is expected that their number will be reduced in future generations. It is therefore important to ensure that the penalization applied to the problem is strong enough. On the other hand, the existence of these individuals can be regarded as a way to avoid local maxima and to increase the search space, expecting that, starting from them, fittest correct individuals would be found.

5. Description of the experiment

An experiment was carried out in order to test the performance of the three EDAs introduced in Section 4.2 for inexact graph matching. As the main difference between

these three algorithms is the number of dependencies between variables that they take into account, the size of the graphs used influences on parameters such as the best solution obtained after a number of generations, the time to compute the algorithm, and the evolution of the algorithm itself through the search. This section describes the experiments and the results obtained. The three EDA algorithms are also compared to three broadly known GAs: basic (cGA) [15], elitist (eGA) [48] and steady state (ssGA) [49].

Both graphs G_M and G_D were generated at random. G_M contains 30 nodes and 39 arcs, and G_D 100 nodes and 247 arcs. The number of arcs chosen for all these graphs in our experiments was selected knowing that the fitness function will not return a different value depending on $|E_M|$ and $|E_D|$. Following the classification of graphs between sparse and dense introduced in [50], the number of arcs have been chosen to be the median of the sparse graphs of that size. The fitness function selected is the one shown in Eq. (2) in Section 2.2.

The experiment was executed in a two processor Silicon Graphics machine SGI-Origin200 under IRIX OS version 64-Release 6.5 with 500 Mb of RAM.

5.1. The need to obtain correct individuals

All the EDAs and GAs were executed 20 times for the randomly generated graphs without applying any technique to correct the wrong individuals, and this showed that UMDA, MIMIC, EBNA, and the three GAs (cGA, eGA and ssGA) did not contain practically any correct individuals in the last generation (the 100th one). The mean proportion of correct individuals are as follows from a population of 2000 individuals: UMDA, MIMIC and EBNA contained a mean percentage of 9.84%, 8.89% and 9.66%, respectively, whereas for cGA, eGA and ssGA are of 34.06%, 33.49% and 22.04%. From these results we can conclude that some kind of correction or manipulation is required for the problem of inexact graph matching under constraints for both EDAs and GAs.

5.2. Combining correction methods and algorithms

Once proved the need to control the generation of the individuals in each population, the four methods described in Section 4.3.1 were combined with the three EDA algorithms. In the case of the GAs, the last two methods described in the same section were used for cGA eGA and ssGA, as the ones based on the modification of the probability in the simulation step do not apply in GAs which do not perform such a step.

All the programs were designed to finish the search when all the populations contained the same individuals or when a maximum of 100 generations was reached. None of EDAs finished before the 100th generation. GAs were programmed to generate the same number of individuals as with EDAs, and therefore 100 generations were executed for all the algorithms. The ssGA algorithm was also programmed in order to generate the same number of individuals by allowing the appropriated number of iterations. The initial population for all the algorithms was generated using the same random generation procedure based on a uniform distribution for all the possible values.

In EDAs, the following parameters were used: a population of 2000 individuals ($N = 2000$), from which the best 1000 are selected ($Se = 1000$) to estimate the probability, and the elitist approach was selected (that is, always the best individual is included for the next population and 1999 individuals are simulated). In GAs a population of 2000 individuals was also selected, with a mutation probability of $1.0/|V_D|$ and a crossover probability of 1.

5.3. Experimental results

The results obtained are shown in Fig. 3 and Tables 1 and 2. Fig. 3 shows the mean evolution of 20 executions for all the algorithms. The reader is reminded that in the case of *PLS only* (when applying only the PLS simulation method alone) and *penalization* (when applying the penalization technique) methods do not ensure a population of only correct individuals.

Tables 1 and 2 show the evaluation of the best individual at the last generation, the number of generations to reach the final solution, and the computation time. The null hypothesis of the same distribution densities was also tested for each of the different algorithms and for each of the correction methods to control the generation of new individuals. The non-parametric tests of Kruskal–Wallis (for more than two populations) and Mann–Whitney (for two populations) were used. This task was carried out with the statistical package S.P.S.S. release 9.00. The results of applying the Kruskal–Wallis test to the different parameters (fitness value and execution time) are also shown in both the tables. Similarly the Kruskal–Wallis test was also applied to the correction methods between EDAs alone, with a result of $p = 0.164$ for fitness values and $p < 0.001$ for the time, to the correction between GAs alone, $p < 0.001$ was obtained for both fitness and time values, to the penalization method between EDAs only, $p = 0.471$ for the fitness value and $p < 0.001$ for time, and to penalization between GAs only, again $p < 0.001$ for both fitness and time values.

The computation time is given in CPU time of the process, and therefore it is not dependent on the multiprogramming level of the instant of the execution. This computation time is presented as a measure to illustrate the different computation complexity of all the algorithms.

The penalization method itself showed to require a stronger penalization when using graphs of sizes such as the ones for our experiments. Therefore, the penalization introduced in Section 4.3.4 should be stronger by reducing even more the fitness value in case the condition $|NNO(V_M)^{|V_D|}| > 0$ is satisfied.

At the light of the results we can conclude that from the three GAs used, ssGA appears clearly as the one that

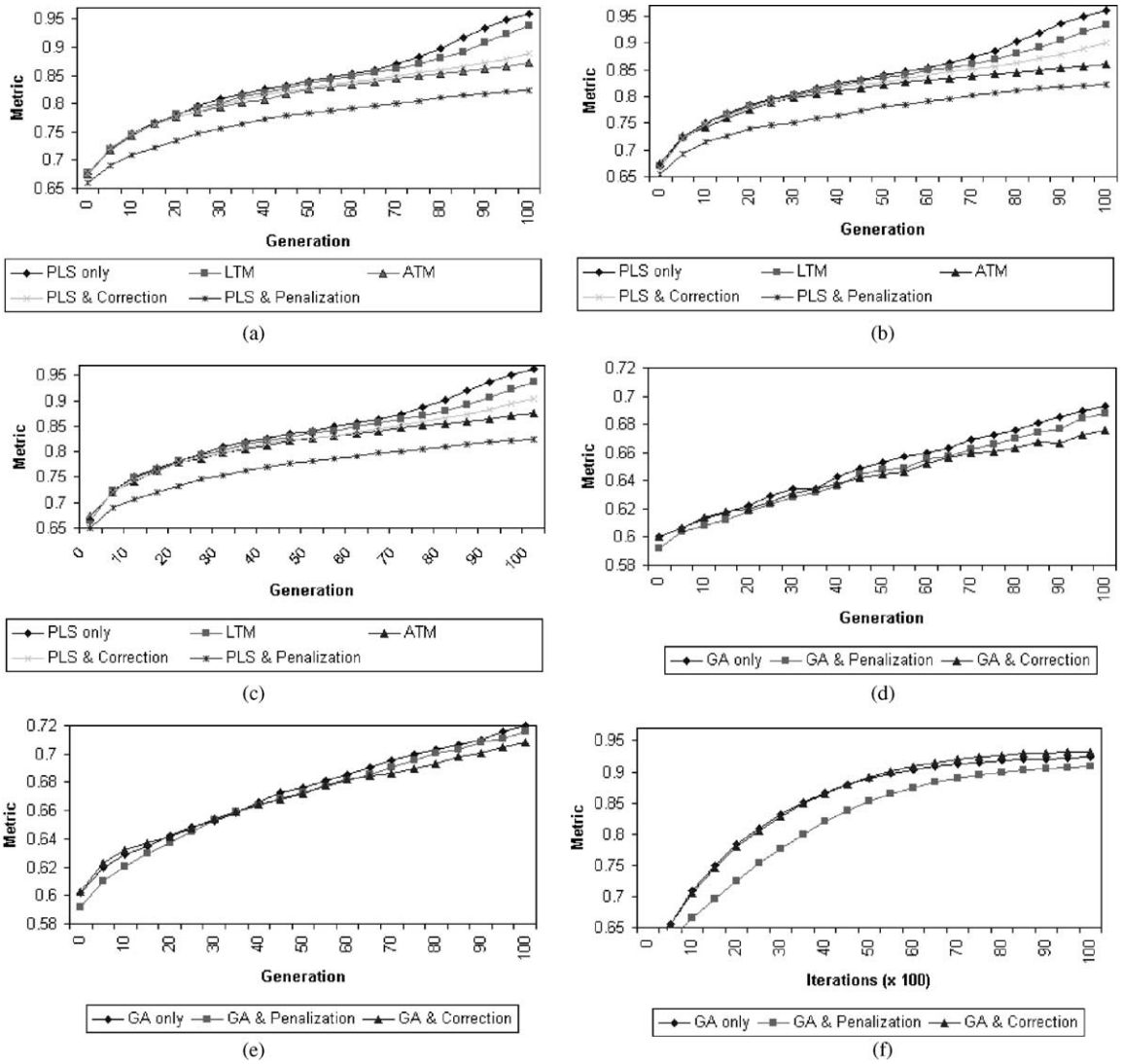

Fig. 3. Graphs showing the best individual at each generation of the searching process for the algorithms UMDA, MIMIC, EBNA, cGA, eGA, and ssGA for the case of the 30 & 100 node graphs. Note the different scales in axis y between the algorithms. (a) UMDA; (b) MIMIC; (c) EBNA; (d) cGA; (e) eGA; (f) ssGA.

Table 1
Mean fitness value results of 20 executions for the experiment

	LTM	ATM	Correction	Penalization	Statistical significance
UMDA	0.940502	0.874684	0.892806	0.825850	$p < 0.001$
MIMIC	0.936400	0.859538	0.898960	0.824063	$p < 0.001$
EBNA	0.936739	0.875429	0.905114	0.823836	$p < 0.001$
cGA	—	—	0.674490	0.687297	$p = 0.004$
eGA	—	—	0.706609	0.712994	$p = 0.160$
ssGA	—	—	0.932318	0.911038	$p < 0.001$
Statistical significance	$p = 0.773$	$p < 0.001$	$p < 0.001$	$p < 0.001$	

Table 2
Mean time to compute each of the 20 executions of the experiment (hh:mm:ss)

	LTM	ATM	Correction	Penalization	Statistical significance
UMDA	00:11:50	00:12:56	00:11:59	00:13:05	$p < 0.001$
MIMIC	00:17:19	00:18:24	00:17:29	00:18:50	$p < 0.001$
EBNA	03:18:06	03:19:06	03:18:13	03:19:16	$p < 0.001$
cGA	—	—	00:09:07	00:09:08	$p < 0.001$
eGA	—	—	00:09:07	00:09:07	$p = 1.000$
ssGA	—	—	00:09:03	00:09:04	$p = 0.317$
Statistical significance	$p < 0.001$	$p < 0.001$	$p < 0.001$	$p < 0.001$	

obtains the best results. Furthermore, the computation time to generate the final solution is also less than the one required by the other two GAs.

The best individuals obtained using the different EDAs are very similar: even if with some correction methods EBNA obtains the best results, in some cases such as in LTM and penalization UMDA performs better. As explained before, EBNA is expected to return better results due to its ability to estimate more accurately the probability distribution every generation, in spite of a higher computational cost. Nevertheless, the small differences between EDAs do not appear to be significant for this example and no significant results are obtained. This effect can be explained by the fact that both graphs have been created at random and that they should not reflect any dependence between variables, and as a result EBNA cannot find more dependencies than other simpler EDAs. On the other hand, when comparing EDAs and GAs, it appears clearly that EDAs obtain better results using any of the correction methods applied to the individuals. It is important to note however that only ssGA obtains nearly as good results as EDAs.

When penalization was used, the proportion of incorrect individuals for this method for UMDA, MIMIC, EBNA, cGA, eGA and ssGA were of 33.66%, 33.69%, 33.69%, 66.81%, 68.32% and 100%, respectively. A stronger penalization could improve these values, but it would never ensure a 100% of correct individuals in the population. Note that the higher percentage in GAs does not imply obtaining better results.

6. Conclusions and further work

This paper introduces EDA algorithms as a new approach to inexact graph matching. Its foundations are based on an evolutionary computation paradigm that applies learning and simulation of Bayesian networks as an important part of the search process. Two adaptations of the PLS simulation schema on Bayesian networks have been introduced for the first time, allowing EDAs to take into account the constraints of the problem.

In experiments with simulated graphs the robustness of this approach has been proved against searches based on GAs. It remains to the future to test the same algorithms for graphs generated from real data (e.g. images). Additionally, other fitness functions should be tested.

Looking at the time required to execute algorithms that make use of complex structures of Bayesian networks such as EBNA (which required more than 3 h), it appears clearly that in the future parallelism techniques should be applied in order to obtain shorter execution times. Some parallel algorithms have already been proposed for similar purposes [51–53].

Acknowledgements

This article has been partially supported by the University of the Basque Country, Department of Education, University and Research of The Basque Government, the Spanish Ministry for Science and Education, and the French Ministry for Education, Research and Technology with the projects 9/UPV/EHU 00140.226-12084/2000, PI 1999-40, HF1999-0107, and Picasso-00773TE, respectively. The authors would also like to thank Ramon Etxeberria, Iñaki Inza and Jose A. Lozano for their useful advise and contribution to this paper.

References

[1] A.D.J. Cross, R.C. Wilson, E.R. Hancock, Inexact graph matching using genetic search, Pattern Recognition 30 (6) (1997) 953–970.
[2] M.Singh, A. Chatterjeeand, S. Chaudhury, Matching structural shape descriptions using genetic algorithms, Pattern Recognition 30 (9) (1997) 1451–1462.
[3] S. Gold, A. Rangarajan, A graduated assignment algorithm for graph matching, IEEE Trans. Pattern Anal. Mach. Intell. 18 (4) (1996) 377–388.
[4] R.C. Wilson, E.R. Hancock, Structural matching by discrete relaxation, IEEE Trans. Pattern Anal. Mach. Intell. 15 (6) (1996) 634–698.

[5] A.D.J. Cross, E.R. Hancock, Graph matching with a dual-step EM algorithm, IEEE Trans. Pattern Anal. Mach. Intell. 20 (11) (1998) 1236–1253.
[6] A.W. Finch, R.C. Wilson, E.R. Hancock, Symbolic graph matching with the EM algorithm, Pattern Recognition 31 (11) (1998) 1777–1790.
[7] E. Bengoetxea, P. Larrañaga, I. Bloch, A. Perchant, Estimation of distribution algorithms: a new evolutionary computation approach for graph matching problems, in: M. Figueiredo, J. Zerubia, A.K. Jain (Eds), Lecture Notes in Computer Science, Vol. 2134, Sophia Antipolis, France, 2001, pp. 454–468, Third International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR-2001).
[8] E. Bengoetxea, P. Larrañaga, I. Bloch, A. Perchant, Solving graph matching with EDAs using a permutation-based representation, in: P. Larrañaga, J.A. Lozano (Eds.), Estimation of Distribution Algorithms, A New tool for Evolutionary Computation, Kluwer Academic Publishers, 2001, pp. 243–265.
[9] E. Bengoetxea, P. Larrañaga, I. Bloch, A. Perchant, C.Boeres, Inexact graph matching using learning and simulation of Bayesian networks, An empirical comparison between different approaches with synthetic data, in: Proceedings of CaNew workshop, ECAI 2000 Conference, ECCAI, Berlin, August 2000.
[10] A. Perchant, Morphism of graphs with fuzzy attributes for the recognition of structural scenes, Ph.D. Thesis, Ecole Nationale Supérieure des Télécommunications, Paris, France, September 2000 (In French).
[11] A. Perchant, I. Bloch, A new definition for fuzzy attributed graph homomorphism with application to structural shape recognition in brain imaging, in: IMTC'99, 16th IEEE Instrumentation and Measurement Technology Conference, Venice, Italy, May 1999, pp. 1801–1806.
[12] A. Perchant, I. Bloch, graph fuzzy homomorphism interpreted as fuzzy association graphs, in: Proceedings of the International Conference on Pattern Recognition, ICPR 2000, Vol. 2, Barcelona, Spain, 2000, pp. 1046–1049.
[13] A. Perchant, C. Boeres, I. Bloch, M. Roux, C. Ribeiro, Model-based Scene Recognition Using Graph Fuzzy Homomorphism Solved by Genetic Algorithm, in: GbR'99 2nd International Workshop on Graph-Based Representations in Pattern Recognition, Castle of Haindorf, Austria, 1999, pp. 61–70.
[14] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison/Wesley, Reading, MA, 1989.
[15] J.H. Holland, Adaptation in Natural and Artificial Systems, The University of Michigan Press, Michigan, 1979.
[16] J.J. Grefenstette, Optimization of control parameters for genetic algorithms, IEEE Trans. Systems, Man Cybernet. 16 (1) (1986) 122–128.
[17] P. Larrañaga, R. Etxeberria, J.A. Lozano, J.M. Peña, Combinatorial optimization by learning and simulation of Bayesian networks, in: Proceedings of the Conference in Uncertainty in Artificial Intelligence, UAI 2000, Stanford, CA, USA, 2000, pp. 343–352.
[18] P. Larrañaga, J.A. Lozano, Estimation of Distribution Algorithms, A New Tool for Evolutionary Computation, Kluwer Academic Publishers, Dordrecht, 2001.
[19] H. Mühlenbein, G. Paaß, From recombination of genes to the estimation of distributions i. Binary parameters, in: Parallel Problem Solving from Nature - PPSN IV, Lecture Notes in Computer Science, Vol. 1411, 1996, pp. 178–187.
[20] R. Howard, J. Matheson, Influence diagrams, in: R. Howard, J. Matheson (Eds.), Readings on the Principles and Applications of Decision Analysis, Vol. 2, Strategic Decision Group, Menlo Park CA, 1981.
[21] S.L. Lauritzen, Graphical Models, Oxford University Press, Oxford, 1996, pp. 721–764.
[22] J. Pearl, Probabilistic Reasoning in Intelligent Systems, Morgan Kaufmann, Palo Alto, CA, 1988.
[23] D. Heckerman, M.P. Wellman, Bayesian networks, Commun. ACM 38 (1995) 27–30.
[24] A.P. Dawid, Conditional independence in statistical theory, J. Roy. Statist. Soc. Ser. B 41 (1979) 1–31.
[25] G. Syswerda, Simulated crossover in genetic algorithms, in: Foundations of Genetic Algorithms, Vol. 2, Morgan Kaufmann, San Mateo, CA, 1993, pp. 239–255.
[26] S. Baluja, Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning, Technical Report, Carnegie Mellon Report, CMU-CS-94-163, 1994.
[27] G. Harik, F.G. Lobo, D.E. Golberg, The compact genetic algorithm, in: Proceedings of the IEEE Conference on Evolutionary Computation, 1998, pp. 523–528.
[28] H. Mühlenbein, The equation for response to selection and its use for prediction, Evolut. Comput. 5 (1998) 303–346.
[29] J.S. DeBonet, C.L. Isbell, P. Viola, MIMIC: Finding optima by estimating probability densities, in: M. Mozer, M. Jordan, Th. Petsche (Eds.), Advances in Neural Information Processing Systems, Vol. 9, 1997.
[30] S. Baluja, S. Davies, Using optimal dependency-trees for combinatorial optimization: learning the structure of the search space, Technical Report, Carnegie Mellon Report, CMU-CS-97-107, 1997.
[31] M. Pelikan, H. Mühlenbein, The bivariate marginal distribution algorithm, in: R. Roy, T. Furuhashi, P.K. Chandhory (Eds.), Advances in Soft Computing-Engineering Design and Manufacturing, Springer, Berlin, London, 1999, pp. 521–535.
[32] H. Mühlenbein, T. Mahning, A. Ochoa, Schemata, distributions and graphical models in evolutionary optimization, J. Heurist. 5 (1999) 215–247.
[33] R. Etxeberria, P. Larrañaga, Global optimization with Bayesian networks, in: Special Session on Distributions and Evolutionary Optimization, II Symposium on Artificial Intelligence, CIMAF99, 1999, pp. 332–339.
[34] M. Pelikan, D.E. Goldberg, E. Cantú-Paz, BOA: The Bayesian optimization algorithm, in: W. Banzhaf, J. Daida, A.E. Eiben, M.H. Garzon, V. Honavar, M. Jakiela, R.E. Smith (Eds.), Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99, Orlando, FL, Vol. 1, Morgan Kaufmann Publishers, San Francisco, CA, 1999, pp. 525–532.
[35] H. Mühlenbein, T. Mahning, FDA—a scalable evolutionary algorithm for the optimization of additively decomposed functions, Evolut. Comput. 7 (4) (1999) 353–376.
[36] G. Harik, Linkage learning in via probabilistic modeling in the EcGA, Technical Report, IlliGAL Technical Report, No. 99010, 1999.
[37] H. Akaike, New look at the statistical model identification, IEEE Trans. Automat. Control 19 (6) (1974) 716–723.

[38] G. Schwarz, Estimating the dimension of a model, Ann. Statist. 7 (2) (1978) 461–464.
[39] G.F. Cooper, E.A. Herskovits, A Bayesian method for the induction of probabilistic networks from data, Mach. Learning 9 (1992) 309–347.
[40] D.M. Chickering, D. Geiger, D. Heckerman, Learning Bayesian networks is NP-hard. Technical Report, Technical Report MSR-TR-94-17, Microsoft Research, Redmond, WA, 1994.
[41] R. Etxeberria, P. Larrañaga, J.M. Picaza, Analysis of the behaviour of genetic algorithms when searching Bayesian networks from data, Pattern Recognition Lett. 18 (11–13) (1997) 1269–1273.
[42] P. Larrañaga, C.M.H. Kuijpers, R.H. Murga, Y. Yurramendi, Searching for the best ordering in the structure learning of Bayesian networks, IEEE Trans. Systems, Man Cybernet. 41 (4) (1996) 487–493.
[43] P. Larrañaga, M. Poza, Y. Yurramendi, R.H. Murga, C.M.H. Kuijpers, Structure learning of Bayesian networks by genetic algorithms, A performance analysis of control parameters, IEEE Trans. Pattern Anal. Mach. Intell. 18 (9) (1996) 912–926.
[44] W. Buntine, Theory refinement in Bayesian networks, in: Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence, 1991, pp. 52–60.
[45] D.M. Chickering, D. Geiger, D. Heckerman, Learning Bayesian networks: Search methods and experimental results, in: Preliminary Papers of the Fifth International Workshop on Artificial Intelligence and Statistics, 1995, pp. 112–128.
[46] M. Henrion, Propagating uncertainty in Bayesian networks by probabilistic logic sampling, in: J.F. Lemmer, L.N. Kanal (Eds.), Uncertainty in Artificial Intelligence, Vol. 2, North-Holland, Amsterdam, 1988, pp. 149–163.
[47] R. Santana, A. Ochoa, Dealing with constraints with estimation of distribution algorithms: The univariate case, in: Second Symposium on Artificial Intelligence, Adaptive Systems, CIMAF 99, La Habana, 1999, pp. 378–384.
[48] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer, Berlin, Heidelberg, 1992.
[49] D. Whitley, J. Kauth, GENITOR: A different genetic algorithm, in: Proceedings of the Rocky Mountain Conference on Artificial Intelligence, Vol. II, 1988, pp. 118–130.
[50] P. Larrañaga, C.M.H. Kuijpers, M. Poza, R. Murga, Decomposing Bayesian networks: triangulation of the moral graph with genetic algorithms, Stat. Comput. 7 (1997) 19–34.
[51] J.A. Lozano, R. Sagarna, P. Larrañaga, Parallel estimation of distribution algorithms, in: P. Larrañaga, J.A. Lozano (Eds.), Estimation of Distribution Algorithms, A New Tool for Evolutionary Computation, Kluwer Academic Publishers, Dordrecht, 2001, pp. 129–145.
[52] R. Sangüesa, U. Cortés, A. Gisolfi, A parallel algorithm for building possibilistic causal networks, Int. J. Approx. Reasoning 18 (1998) 251–270.
[53] Y. Xiang, T. Chu, Parallel learning of belief networks in large and difficult domains, Data Mining Knowledge Discovery 3 (1999) 315–339.

About the Author—ENDIKA BENGOETXEA is lecturer at the Computer Architecture and Technology Department at the University of the Basque Country. He obtained his B.Sc. in Computer Science from the University of the Basque Country and Brighton University in 1994, and his M.Sc. in Medical Imaging from the University of Aberdeen in 1999. His research interests reside in evolutionary algorithms, graph matching and their application to medical images.

About the Author—PEDRO LARRAÑAGA is Associate Professor at the Department of Computer Science and Artificial Intelligence at the University of the Basque Country. His current research interests are in the fields of Bayesian networks, combinatorial optimization and data analysis with applications to medicine, molecular biology and finance.

About the Author—ISABELLE BLOCH is Professor at the Ecole Nationale Supérieure des Télécommunications (ENST) Paris (Department of Signal and Image Processing). She graduated from *Ecole des Mines de Paris* in 1986, received Ph.D. from ENST Paris in 1990 and the *Habilitation à Diriger des Recherches* from University Paris 5 in 1995. Her research interests include 3D image and object processing, fuzzy mathematical morphology, decision theory, data fusion in image processing, fuzzy set theory, evidence theory, medical imaging as well as aerial and satellite imaging.

About the Author—AYMERIC PERCHANT is graduated from ENST, Paris, France. He received his Ph.D. in morphism of graphs with fuzzy attributes for the recognition of structural scenes in 2000.

About the Author—CLAUDIA BOERES is a Ph.D. student in the postgraduation program in Production Engineering (COPPE) at the Universidade Federal of Rio de Janeiro (UFRJ) and a lecturer in the Department of Informatics at the Universidade Federal do Espirito Santo (UFES). She is currently interested in studying the fuzzy attributed graphs correspondence problem, treating it as an optimization problem, and solving it using metaheuristics.