Estimation of Distribution Algorithms: A New Evolutionary Computation Approach for Graph Matching Problems

Endika Bengoetxea[1], Pedro Larrañaga[2], Isabelle Bloch[3], and Aymeric Perchant[3]

[1] Department of Computer Architecture and Technology, University of the Basque Country, San Sebastian, Spain
endika@si.ehu.es

[2] Department of Computer Science and Artificial Intelligence, University of the Basque Country, San Sebastian, Spain
pedro@si.ehu.es

[3] Department of Signal and Image Processing
Ecole Nationale Supérieure des Télécommunications, CNRS URA 820, Paris, France
{Isabelle.Bloch,Aymeric.Perchant}@enst.fr

Abstract. The interest of graph matching techniques in the pattern recognition field is increasing due to the versatility of representing knowledge in the form of graphs. However, the size of the graphs as well as the number of attributes they contain can be too high for optimization algorithms. This happens for instance in image recognition, where structures of an image to be recognized need to be matched with a model defined as a graph.

In order to face this complexity problem, graph matching can be regarded as a combinatorial optimization problem with constraints and it therefore it can be solved with evolutionary computation techniques such as Genetic Algorithms (GAs) and Estimation Distribution Algorithms (EDAs).

This work proposes the use of EDAs, both in the discrete and continuous domains, in order to solve the graph matching problem. As an example, a particular inexact graph matching problem applied to recognition of brain structures is shown. This paper compares the performance of these two paradigms for their use in graph matching.

1 Introduction

Many articles about representation of structural information by graphs in domains such as image interpretation and pattern recognition can be found in the literature [1]. In those, graph matching is used for structural recognition of images: the model (which can be an atlas or a map depending on the application) is represented in the form of a graph, where each node contains information for a particular structure and arcs contain information about relationships between structures; a data graph is generated from the images to be analyzed and contains similar information. Graph matching techniques are then used to determine which structure in the model corresponds to each of the structures in a given image.

Most existing problems and methods in the graph matching domain assume graph isomorphism, where both graphs being matched have the same number of nodes and links. In some cases this bijective condition between the two graphs is too strong and it is necessary to weaken it and to express the correspondence as an inexact graph matching problem.

When the generation of the data graph from an original image is done without the aid of an expert, it is difficult to segment accurately the image into meaningful entities, that is why over-segmentation techniques need to be applied [1,2,3]. As a result, the number of nodes in the data graph increases and isomorphism condition between the model and data graphs cannot be assumed. Such problems call for inexact graph matching, and similar examples can be found in other fields.

Several techniques have been applied to inexact graph matching, including combinatorial optimization [4,5,6], relaxation [7,8,9,10,11], EM algorithm [12,13], and evolutionary computation techniques such as Genetic Algorithms (GAs) [14,15].

This work proposes the use of Estimation Distribution Algorithm (EDA) techniques in both the discrete and continuous domains, showing the potential of this new evolutionary computation approach among traditional ones such as GAs.

The outline of this work is as follows: Section 2 is a review of the EDA approach. Section 3 illustrates the inexact graph matching problem and shows how to face it with EDAs. Section 4 describes the experiment carried out and the results obtained. Finally, Section 5 gives the conclusions and suggests further work.

2 Estimation Distribution Algorithms

2.1 Introduction

EDAs [16,17,18] are non-deterministic, stochastic heuristic search strategies that form part of the evolutionary computation approaches, where number of solutions or individuals are created every generation, evolving once and again until a satisfactory solution is achieved. In brief, the characteristic that most differentiates EDAs from other evolutionary search strategies such as GAs is that the evolution from a generation to the next one is done by estimating the probability distribution of the fittest individuals, and afterwards by sampling the induced model. This avoids the use of crossing or mutation operators, and the number of parameters that EDAs require is reduced considerably.

In EDAs, the individuals are not said to contain genes, but variables which dependencies have to be analyzed. 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
  $D_0 \leftarrow$ 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
    $\rho_l(\boldsymbol{x}) = \rho(\boldsymbol{x}|D_{l-1}^{Se}) \leftarrow$ Estimate the probability distribution
          of an individual being among the selected individuals
    $D_l \leftarrow$ Sample $N$ individuals (the new population) from $\rho_l(\boldsymbol{x})$
```

Fig. 1. Pseudocode for EDA approach.

EDAs the interrelations are expressed explicitly through the joint probability distribution associated with the individuals selected at each iteration. 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, as this requires the adaptation of methods to learn models from data developed in the domain of probabilistic graphical models.

Figure 1 shows the pseudocode of EDA, in which we distinguish four main steps in this approach:

1. At the beginning, the first population D_0 of N individuals is generated, usually by assuming an uniform distribution (either discrete or continuous) on each variable, and evaluating each of the individuals.
2. Secondly, a number Se ($Se \leq N$) of individuals are selected, usually the fittest ones.
3. Thirdly, the n–dimensional probabilistic model that better expresses the interdependencies between the n variables is induced.
4. Next, the new population of N new individuals is obtained by simulating the probability distribution learned in the previous step.

Steps 2, 3 and 4 are repeated until a stopping condition is verified. The most important step of this new paradigm is to find the interdependencies between the variables (step 3). This task will be done using techniques from the field of probabilistic graphical models.

Next, some notation is introduced. 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 i^{th} component of $\boldsymbol{X}$. Let $\boldsymbol{y} = (x_i)_{X_i \in \boldsymbol{Y}}$ be a value of $\boldsymbol{Y} \subseteq \boldsymbol{X}$. 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 independences between 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, the structure of which is given by S. The structure S for $\boldsymbol{X}$ assumes that X_i and its non descendants are independent given $\boldsymbol{Pa}_i^S$, $i = 2, \ldots , n$. Therefore, the factorization can be written as follows:

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

Furthermore, regarding the local generalized probability densities associated with the probabilistic graphical model, these are precisely the ones appearing in Equation 1.

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

$$\rho(\boldsymbol{x} \mid \boldsymbol{\theta}_S) = \prod_{i=1}^{n} \rho(x_i \mid \boldsymbol{pa}_i^S, \boldsymbol{\theta}_i) \tag{2}$$

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

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

2.2 EDAs in Discrete Domains

In the particular case where every variable $X_i \in \boldsymbol{X}$ is discrete, the probabilistic graphical model is called *Bayesian network* [19]. If the variable X_i has r_i possible values, $x_i^1, \ldots , x_i^{r_i}$, the local distribution, $p(x_i \mid \boldsymbol{pa}_i^{j,S}, \boldsymbol{\theta}_i)$ is:

$$p({x_i}^k \mid \boldsymbol{pa}_i^{j,S}, \boldsymbol{\theta}_i) = \theta_{x_i^k \mid \boldsymbol{pa}_i^j} \equiv \theta_{ijk} \tag{3}$$

where $\boldsymbol{pa}_i^{1,S}, \ldots , \boldsymbol{pa}_i^{q_i,S}$ denotes the values of $\boldsymbol{Pa}_i^S$, that is the set of parents of the variable X_i in the structure S; 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 that variable X_i takes its k^{th} value, knowing that the set of its parent variables take its j^{th} value. We assume that every θ_{ijk} is greater than zero.

All the EDAs 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).

Without Interdependencies. The Univariate Marginal Distribution Algorithm (UMDA) [20] is a representative example of this category, which can be written as:

$$p_l(\boldsymbol{x};\boldsymbol{\theta}^l) = \prod_{i=1}^{n} p_l(x_i;\boldsymbol{\theta}_{\boldsymbol{i}}^{\boldsymbol{j}}) \tag{4}$$

where $\boldsymbol{\theta}^l = \left\{\theta^l_{ijk}\right\}$ is recalculated every generation by its maximum likelihood estimation, i.e. $\widehat{\theta}^l_{ijk} = \frac{N^{l-1}_{ijk}}{N^{l-1}_{ij}}$. N^l_{ijk} is the number of cases on which the variable X_i takes the value x_i^k when its parents are on their j^{th} combination of values for the l^{th} generation, and $N^{l-1}_{ij} = \sum_k N^{l-1}_{ijk}$.

Pairwise Dependencies. An example of this second category is the greedy algorithm called MIMIC (Mutual Information Maximization for Input Clustering) [21]. The main idea in MIMIC is to describe the true mass joint probability as closely as possible by using only one univariate marginal probability and $n-1$ pairwise conditional probability functions.

Multiple Interdependencies. We will use EBNA (Estimation of Bayesian Network Algorithm) [22] as an example of this category. The EBNA approach was introduced for the first time in [23], where the authors use the Bayesian Information Criterion (BIC) [24] as the score to evaluate the goodness of each structure found during the search. Following this criterion, the corresponding BIC score –$BIC(S,D)$– for a Bayesian network structure S constructed from a database D and containing N cases can be proved to be as follows:

$$BIC(S,D) = \sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} N_{ijk}\log\frac{N_{ijk}}{N_{ij}} - \frac{\log N}{2}\sum_{i=1}^{n}(r_i-1)q_i \tag{5}$$

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 j^{th} value, and $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$.

Unfortunately, to obtain the best model all possible structures must be searched through, which has been proved to be NP-hard [25]. Even if promising results have been obtained through global search techniques [26,27,28], 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 [29], which is a greedy search heuristic that begins with an arc-less structure and adds iteratively the arcs that produce maximum improvement according to the BIC approximation –but other measures can also be applied. The algorithm stops when adding another arc would not increase the score of the structure.

Local search strategies are another way of obtaining good models. These begin with 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 strong dependence on the initial structure. Nevertheless, since it has been shown in [30] 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 structure.

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

Simulation in Bayesian Networks. In EDAs, 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 work is the *Probabilistic Logic Sampling* (PLS) proposed in [31]. Following this method, the instantiations 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.

2.3 EDAs in Continuous Domains

In this section we introduce an example of the probabilistic graphical model paradigm that assumes the joint density function to be a multivariate Gaussian density.

The local density function for the i^{th} variable is computed as the linear-regression model

$$f(x_i \mid \boldsymbol{pa}_i^S, \boldsymbol{\theta}_i) \equiv \mathcal{N}(x_i; m_i + \sum_{x_j \in \boldsymbol{pa}_i} b_{ji}(x_j - m_j), v_i) \tag{6}$$

where $\mathcal{N}(x; \mu, \sigma^2)$ is a univariate normal distribution with mean μ and variance σ^2.

Local parameters are given by $\boldsymbol{\theta}_i = (m_i, \boldsymbol{b}_i, v_i)$, where $\boldsymbol{b}_i = (b_{1i}, \ldots, b_{i-1i})^t$ is a column vector. Local parameters are as follows: m_i is the unconditional mean of X_i, v_i is the conditional variance of X_i given $\boldsymbol{Pa}_i$, and b_{ji} is a linear coefficient that measures the strength of the relationship between X_j and X_i. A probabilistic graphical model built from these local density functions is known as a *Gaussian network* [32]. Gaussian networks are of interest in continuous EDAs because the number of parameters needed to specify a multivariate Gaussian density is smaller.

Next, an analogous classification of continuous EDAs as for the discrete domain is done, in which these continuous EDAs are also classified depending on the number of dependencies they take into account.

Without Dependencies. In this case, the joint density function is assumed to follow a n–dimensional normal distribution, and thus it is factorized as a product of n unidimensional and independent normal densities. Using the mathematical notation $\boldsymbol{X} \equiv \boldsymbol{\mathcal{N}}(\boldsymbol{x}; \boldsymbol{\mu}, \sum)$, this assumption can be expressed as:

$$f_{\boldsymbol{\mathcal{N}}}(\boldsymbol{x}; \boldsymbol{\mu}, \textstyle\sum) = \displaystyle\prod_{i=1}^{n} f_{\mathcal{N}}(x_i; \mu_i, \sigma_i) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{1}{2}(\frac{x_i - \mu_i}{\sigma_i})^2}. \tag{7}$$

An example of continuous EDAs in this category is UMDA$_c$ [33].

Bivariate Dependencies. An example of this category is MIMIC$_c^G$ [33], which is basically an adaptation of the MIMIC algorithm [21] to the continuous domain.

Multiple Dependencies. Algorithms in this section are approaches of EDAs for continuous domains in which there is no restriction in the learning of the density function every generation. An example of this category is EGNA$_{BGe}$ (Estimation of Gaussian Network Algorithm) [33]. The method used to find the Gaussian network structure is a Bayesian score+search. In EGNA$_{BGe}$ a local search is used to search for good structures.

Simulation of Gaussian Networks. A general approach for sampling from multivariate normal distributions is known as the conditioning method, which generates instances of $\boldsymbol{X}$ by sampling X_1, then X_2 conditionally to X_1, and so on. The simulation of a univariate normal distribution can be done with a simple method based on the sum of 12 uniform variables.

3 Graph Matching as a Combinatorial Optimization Problem with Constraints

3.1 Traditional Representation of Individuals

The choice of an adequate individual representation is a very important step in any problem to be solved with heuristics that will determine the behavior of the search. An individual represents a point in the search space that has to be evaluated, and therefore is a solution. For a graph matching problem, each solution represents a match between the nodes of a data graph G_2 and those of model graph G_1.

A possible representation that has already been used either in GAs or discrete EDAs [34] consists of individuals with $|V_2|$ variables, where each variable can take any value between 1 and $|V_1|$. More formally, the individual as well as the solution it represents could be defined as follows: for $1 \leq k \leq |V_1|$ and $1 \leq i \leq |V_2|$, $X_i = k$ means that the i^{th} node of G_2 is matched with the k^{th} node of G_1.

3.2 Representing a Matching as a Permutation

Permutation-based representations have been typically applied to problems such as the Travelling Salesman Problem (TSP), but they can also be used for inexact graph matching. In this case the meaning of the individual is completely different, as an individual does not show directly which node of G_2 is matched with each node of G_1. In fact, what we obtain from each individual is the order in which nodes will be analyzed and treated so as to compute the matching solution that it is representing.

For the individuals to contain a permutation, the individuals will have the same size as the *traditional* ones described in Section 3.1 (i.e. $|V_2|$ variables long). However, the number of values that each variable can take will be $|V_2|$, and not $|V_1|$ as in that representation. In fact, it is important to note that a permutation is a list of numbers in which all the values from 1 to n have to appear in an individual of size n. In other words, our new representation of individuals needs to satisfy a strong constraint in order to be considered as correct, that is, they all have to contain every value from 1 to n, where $n = |V_2|$.

More formally, for $1 \leq k \leq |V_2|$ and $1 \leq i \leq |V_2|$, $X_i = k$ means that the k^{th} node of G_2 will be the i^{th} node that is analyzed for its most appropriate match.

Now it is important to define a procedure to obtain the solution that each permutation symbolizes. As this procedure will be done for each individual, it is important that this translation is performed by a fast and simple algorithm. A way of doing this is introduced next.

A solution for the inexact graph matching problem can be calculated by comparing the nodes to each other and deciding which is more similar to which using a similarity function $\varpi(i, j)$ defined to compute the similarity between nodes i and j. The similarity measures used so far in the literature have been applied to two nodes, one from each graph, and their aim was to help in the computation of the fitness of a solution, that is, the final value of a fitness function. However, the similarity measure $\varpi(i, j)$ proposed in this work is quite different, as these two nodes to be evaluated are both in the data graph $(i, j \in V_2)$ –see Section 4.3 for more details. With these new similarity values we will identify for each particular node of G_2 which other nodes in the data graph are most similar to it, and try to group it with the best set of already matched nodes.

Given an individual $\boldsymbol{x}= (x_1, \ldots, x_{|V_1|}, x_{|V_1|+1}, \ldots, x_{|V_2|})$, the procedure to do the translation is performed in two phases as follows:

1. The first $|V_1|$ values $(x_1, \ldots, x_{|V_1|})$ that directly represent nodes of V_2 will be matched to nodes $1, 2 \ldots, |V_1|$ (that is, the node $x_1 \in V_2$ is matched with the node $1 \in V_1$, the node $x_2 \in V_2$ is matched with the node $2 \in V_1$, and so on, until the node $x_{|V_1|} \in V_2$ is matched with the node $|V_1| \in V_1$).
2. For each of the following values of the individual, $(x_{|V_1|+1}, \ldots, x_{|V_2|})$, and following their order of appearance in the individual, the most similar node will be chosen from all the previous values in the individual by means of the similarity measure ϖ. For each of these nodes of G_2, we assign the matched node of G_1 that is matched to the most similar node of G_2.

The first phase is very important in the generation of the individual, as this is also the one that ensures the correctness of the solution represented by the permutation: all the values of V_1 are assigned from the beginning, and as we assumed $|V_2| > |V_1|$, we conclude that all the nodes of G_1 will have at least a occurrence in the solution represented by any permutation.

Looking for correct individuals
As explained in Section 2.2, the simulation process is PLS [31]. But a simple PLS algorithm will not take into account any restriction the individuals must have for a particular problem. The interested reader can find a more exhaustive review of this topic in [34], where the authors propose different methods to obtain only correct individuals that satisfy the particular constraints of the problem.

3.3 Obtaining a Permutation with Continuous EDAs

Continuous EDAs provide the search with other types of EDA algorithms that can be more suitable for some problems. But again, the main goal is to find a representation of individuals and a procedure to obtain an univocal solution to the matching from each of the possible permutations.

In this case we propose a strategy based on the previous section, trying to translate the individual in the continuous domain to a correct permutation in the discrete one, evaluating it as explained in Section 3.2. This procedure has to be performed for each individual in order to be evaluated. Again, this process has to be fast enough in order to reduce computation time.

With all these aspects in mind, individuals of the same size ($n = |V_2|$) will be defined, where each of the variables of the individual can take any value following a Gaussian distribution. This new representation of individuals is a continuous value in $\mathbb{R}^n$ that does not provide directly the solution it symbolizes: the values for each of the variables only show the way to translate from the continuous world to a permutation, and it does not contain similarity values between nodes of both graphs. This new type of representation can also be regarded as a way to focus the search from the continuous world, where the techniques that can be applied to the estimation of densities are completely different.

In order to obtain a translation to a discrete permutation individual, we propose to order the continuous values of the individual, and to set its corresponding discrete values by assigning to each $x_i \in \{1, \ldots, |V_2|\}$ the respective order in the continuous individual. The procedure described in this section is further described in [35].

4 Experimental Results. The Human Brain Example

4.1 Overview of the Human Brain Example

The example chosen to test the performance of the different EDAs for permutation-based representations in inexact graph matching is a problem of recognition of regions in 3D Magnetic Resonance Images (MRI) of the brain. The data graph $G_2 = (V_2, E_2)$ is generated after over-segmenting an image and contains a node for each segmented region (subset of a brain structure). The model graph

$G_1 = (V_1, E_1)$ contains a node for each of the brain regions to be recognized. The experiments carried out in this chapter are focused on this type of graphs, but could similarly be adapted to any other inexact graph matching problem.

More specifically, the model graph was obtained from the main structures of the the inner part of the brain (the brainstem). This example is a reduced version of the brain images recognition problem in [1]. In our case the number of nodes of G_2 (number of structures of the image to be recognized) is 94, and contains 2868 arcs. The model graph contains 13 nodes and 84 arcs.

4.2 Description of the Experiment

This section compares EDA algorithms each other and to a broadly known GA, the GENITOR [36], which is a steady state type algorithm (ssGA).

Both EDAs and GENITOR were implemented in ANSI C++ language, and the experiment was executed on a two processor Ultra 80 Sun computer under Solaris version 7 with 1 GByte of RAM.

The initial population for all the algorithms was created using the same random generation procedure based on a uniform distribution. The fitness function used is described later in Section 4.4.

In the discrete case, all the algorithms were set to finish the search when a maximum of 100 generations or when uniformity in the population was reached. GENITOR, as it is a ssGA algorithm, only generates two individuals at each iteration, but it was also programmed in order to generate the same number of individuals as in discrete EDAs by allowing more iterations (201900 individuals). In the continuous case, the ending criterion was to reach 301850 evaluations (i.e. number of individuals generated).

In EDAs, the following parameters were used: a population of 2000 individuals ($N = 2000$), from which a subset of the best 1000 are selected ($S_e = 1000$) to estimate the probability, and the elitist approach was chosen (that is, always the best individual is included for the next population and 1999 individuals are simulated). In GENITOR a population of 2000 individuals was also set, with a mutation rate of $p_m = \frac{1}{|V_2|}$ and a crossover probability of $p_c = 1$. The operators used in GENITOR where CX [37] and EM [38].

4.3 Definition of the Similarity Function

Speaking about the similarity concept, we have used only a similarity measure based on the grey level distribution, so that the function ϖ returns a higher value for two nodes when the grey level distribution over two segments of the data image is more similar. In addition, no clustering process is performed, and therefore the similarity measure ϖ is kept constant during the generation of individuals. These decisions have been made knowing the nature and properties of an MRI image. More formally, the function ϖ can be defined as the set of functions that measure the correspondence between the two nodes of the data graph G_2: $\varpi = \{\rho_\sigma^{u_2} : V_2 \to [0,1], u_2 \in V_2\}$.

4.4 Definition of the Fitness Function

We have chosen a function proposed in [1] as an example. Following this function, an individual $\boldsymbol{x}= (x_1, \ldots, x_{|V_2|})$ will be evaluated as follows:

$$f(\boldsymbol{x}; \rho_\sigma, \rho_\mu, \alpha) = \alpha \left[\frac{1}{|V_2||V_1|} \sum_{i=1}^{|V_2|} \sum_{j=1}^{|V_1|} \left(1 - |c_{ij} - \rho_\sigma^{u_1^i}(u_2^j)|\right) \right] +$$

$$(1-\alpha) \left[\frac{1}{|E_2||E_1|} \sum_{e_1^l=(u_1^i, v_1^{i'}) \in E_1} \quad \sum_{e_2^k=(u_2^j, v_2^{j'}) \in E_2} \left(1 - |c_{ij} c_{i'j'} - \rho_\mu^{e_1^l}(e_2^k)|\right) \right] \qquad (8)$$

where

$$c_{ij} = \begin{cases} 1 \text{ if } X_i = j \\ 0 \text{ otherwise,} \end{cases}$$

α is a parameter used to adapt the weight of node and arc correspondences in f. For each $u_1^i \in V_1$, $\rho_\sigma^{u_1^i}$ is a function from V_2 into $[0,1]$ that measures the correspondence between u_1^i and each node of V_2. Similarly, for each $e_1 \in E_1$, ρ_μ is the set of functions from E_2 into $[0,1]$ that measure the correspondence between the arcs of both graphs G_1 and G_2. The value of f associated for each variable returns the goodness of the matching. Typically ρ_σ and ρ_μ are related to the similarities between node and arc properties respectively.

Node properties are described as attributes on grey level and size, while edge properties correspond to spatial relationships between nodes.

4.5 Experimental Results

Results such as the best individual obtained, the computation time, and the number of evaluations to reach the final solution were recorded for each of the experiments. The computation time obtained is the CPU time of the process for each execution, and therefore it is not dependent on the load of the system. The latter is given as a measure to illustrate the different computation complexity of all the algorithms.

Each algorithm was executed 10 times. The non-parametric tests of Kruskal-Wallis and Mann-Whitney were used to test the null hypothesis of the same distribution densities for all –or some– of them. This task was done with the statistical package S.P.S.S. release 9.00. The results for the tests applied to all the algorithms are shown in Table 1. The study of particular algorithms gives the following results:

- Between algorithms of similar complexity only:
  - UMDA vs. UMDA$_c$. Fitness value: $p < 0.001$; CPU time: $p < 0.001$; Evaluations: $p < 0.001$.

Table 1. Mean values of experimental results after 10 executions for each algorithm of the inexact graph matching problem of the Human Brain example.

	Best fitness value	Execution time	Number of evaluations
UMDA	0.718623	00:53:29	85958
UMDA$_c$	0.745036	03:01:05	301850
MIMIC	0.702707	00:57:30	83179
MIMIC$_c$	0.747970	03:01:07	301850
EBNA	0.716723	01:50:39	85958
EGNA	0.746893	04:13:39	301850
ssGA	0.693575	07:31:26	201900
	$p < 0.001$	$p < 0.001$	$p < 0.001$

- MIMIC vs. MIMIC$_c$. Fitness value: $p < 0.001$; CPU time: $p < 0.001$; Evaluations: $p < 0.001$.
- EBNA vs. EGNA. Fitness value: $p < 0.001$; CPU time: $p < 0.001$; Evaluations: $p < 0.001$.

These results show that the differences between EDAs in the discrete and continuous domains are significant in all the cases analyzed, meaning that the behavior of selecting a discrete learning algorithm or its equivalent in the continuous domain is very different. It is important to note that the number of evaluations was expected to be different, as the ending criteria for the discrete and continuous domains were also different. In all the cases, continuous EDAs obtained a fitter individual, but the CPU time and number of individuals created was also bigger.

– Between discrete algorithms only:
- Fitness value: $p < 0.001$. CPU time: $p < 0.001$. Evaluations: $p < 0.001$.

In this case significant results are also obtained in fitness value, CPU time, and number of evaluations. The discrete algorithm that obtained the best result was UMDA, closely followed by EBNA. The differences in the CPU time are also according to the complexity of the learning algorithm they apply. Finally, the results show that MIMIC required significantly less individuals to converge (to reach the uniformity in the population), whereas the other two EDA algorithms require nearly the same number of evaluations to converge. The genetic algorithm GENITOR is far behind the performance of EDAs. The computation time is also a factor to consider: the fact that GENITOR requires about 7 hours for each execution shows the complexity of the graph matching problem.

– Between continuous algorithms only:
- Fitness value: $p = 0.342$. CPU time: $p < 0.001$. Evaluations: $p = 1.000$.

Differences between all the continuous EDAs appear to be not significant. As expected, the CPU time required for each of them is according to the complexity of the learning algorithm. On the other hand, the fact of having the same number of evaluations is due to the same ending criterion. Speaking about the differences in computation time between discrete and continuous EDA algorithms, it is important to note that the latter ones require all the

300000 individuals to be generated before they finish the search. The computation time for the continuous algorithms is also longer than the discrete equivalents as a result of several factors: firstly, due to the higher number of evaluations they perform each execution, secondly because of the longer individual-to-solution translation procedure that has to be done for each of the individuals generated, and lastly, as a result of the longer time required to learn the model in continuous spaces.

We can conclude from the results that generally speaking continuous algorithms perform better than discrete ones, either when comparing all of them in general or only with algorithms of equivalent complexity.

5 Conclusions and Further Work

This work describes the application of the EDA approach to graph matching. Different individual representations have been shown in order to allow the use of discrete and continuous representation and algorithms.

In an experiment with real data a comparison of the performance of this new approach between the discrete and continuous domains has been done, and continuous EDAs have shown a better performance looking at the fittest individual obtained, however a longer execution time and more evaluations were required. Additionally, other fitness functions should be tested with this new approach. Techniques such as [39,40] could also help to introduce better similarity measures and therefore improve the results obtained considerably.

For the near future there are several tasks to be done. The most important is to perform more experiments with more data images (more data graphs) in order to evaluate the effectiveness of the proposed matching heuristic with more examples. In addition, a deeper study on the influence of node and arc correspondences requires also to be done. These new experiments are expected to highlight the importance of the structural aspects (the edges) as appreciated in our recent work.

Acknowledgments

This work has been partially supported by the University of the Basque Country, 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, HF1999-0107, and Picasso-00773TE respectively. The authors would also like to thank Ramon Etxeberria, Iñaki Inza and Jose A. Lozano for their useful advice and contributions to this work.

References

1. A. Perchant and 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, pages 1801–1806, Venice, Italy, May 1999.
2. A. Perchant, C. Boeres, I. Bloch, M. Roux, and 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*, pages 61–70, Castle of Haindorf, Austria, 1999.
3. Aymeric Perchant. *Morphism of graphs with fuzzy attributes for the recognition of structural scenes (In French).* PhD thesis, Ecole Nationale Supérieure des Télécommunications, Paris, France, September 2000.
4. A. D. J. Cross and E. R. Hancock. Convergence of a hill climbing genetic algorithm for graph matching. In *Lecture notes in Computer Science 1654*, pages 220–236, York, UK, 1999. E. R. Hancock, M. Pelillo (Eds.).
5. A. D. J. Cross, R. C. Wilson, and E. R. Hancock. Inexact graph matching using genetic search. *Pattern Recognition*, 30(6):953–70, 1997.
6. M. Singh and A. Chatterjeeand S. Chaudhury. Matching structural shape descriptions using genetic algorithms. *Pattern Recognition*, 30(9):1451–62, 1997.
7. A. W. Finch, R. C. Wilson, and E. R. Hancock. Matching Delaunay graphs. *Pattern Recognition*, 30(1):123–40, 1997.
8. S. Gold and A. Rangarajan. A graduated assignment algorithm for graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(4):377–88, 1996.
9. E. R. Hancock and J. Kittler. Edge-labeling using dictionary-based relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(2):165–181, 1990.
10. R. C. Wilson and E. R. Hancock. Bayesian compatibility model for graph matching. *Pattern Recognition Letters*, 17:263–276, 1996.
11. R. C. Wilson and E. R. Hancock. Structural matching by discrete relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):634–648, 1997.
12. A. D. J. Cross and E. R. Hancock. Graph matching with a dual-step EM algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1236–53, 1998.
13. A. W. Finch, R. C. Wilson, and E. R. Hancock. Symbolic graph matching with the EM algorithm. *Pattern Recognition*, 31(11):1777–90, 1998.
14. C. Boeres, A. Perchant, I. Bloch, and M. Roux. A genetic algorithm for brain image recognition using graph non-bijective correspondence. Unpublished manuscript, 1999.
15. R. Myers and E.R. Hancock. Least committment graph matching with genetic algorithms. *Pattern Recognition*, 34:375–394, 2001.
16. P. Larrañaga and J. A. Lozano. *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation.* Kluwer Academic Publishers, 2001.
17. I. Inza, P. Larrañaga, R. Etxeberria, and B. Sierra. Feature subset selection by Bayesian networks based optimization. *Artificial Intelligence*, 123 (1–2):157–184, 2000.
18. I. Inza, M. Merino, P. Larrañaga, J. Quiroga, B. Sierra, and M. Girala. Feature subset selection by genetic algorithms and estimation of distribution algorithms. a case study in the survival of cirrhotic patients treated with TIPS. *Artificial Intelligence in Medicine*, Accepted for publication, 2001.
19. J. Pearl. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann, Palo Alto, CA, 1988.
20. H. Mühlenbein. The equation for response to selection and its use for prediction. *Evolutionary Computation*, 5:303–346, 1998.

21. J. S. De Bonet, C. L. Isbell, and P. Viola. MIMIC: Finding optima by estimating probability densities. In *Advances in Neural Information Processing Systems*, volume 9. M. Mozer, M. Jordan and Th. Petsche eds., 1997.
22. P. Larrañaga, R. Etxeberria, J. A. Lozano, and J. M. Peña. Combinatorial optimization by learning and simulation of Bayesian networks. In *Proceedings of the Conference in Uncertainty in Artificial Intelligence, UAI 2000*, pages 343–352, Stanford, CA, USA, 2000.
23. R. Etxeberria and P. Larrañaga. Global optimization with Bayesian networks. In *Special Session on Distributions and Evolutionary Optimization*, pages 332–339. II Symposium on Artificial Intelligence, CIMAF99, 1999.
24. G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 7(2):461–464, 1978.
25. D. M. Chickering, D. Geiger, and D. Heckerman. Learning Bayesian networks is NP–hard. Technical report, Technical Report MSR-TR-94-17, Microsoft Research, Redmond, WA, 1994.
26. R. Etxeberria, P. Larrañaga, and J. M. Picaza. Analysis of the behaviour of the genetic algorithms when searching Bayesian networks from data. *Pattern Recognition Letters*, 18(11–13):1269–1273, 1997.
27. P. Larrañaga, C. M. H. Kuijpers, R. H. Murga, and Y. Yurramendi. Searching for the best ordering in the structure learning of Bayesian networks. *IEEE Transactions on Systems, Man and Cybernetics*, 41(4):487–493, 1996.
28. P. Larrañaga, M. Poza, Y. Yurramendi, R. H. Murga, and C. M. H. Kuijpers. Structure learning of Bayesian networks by genetic algorithms. A performance analysis of control parameters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(9):912–926, 1996.
29. W. Buntine. Theory refinement in Bayesian networks. In *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, pages 52–60, 1991.
30. D. M. Chickering, D. Geiger, and D. Heckerman. Learning Bayesian networks: Search methods and experimental results. In *Preliminary Papers of the Fifth International Workshop on Artificial Intelligence and Statistics*, pages 112–128, 1995.
31. M. Henrion. Propagating uncertainty in Bayesian networks by probabilistic logic sampling. *Uncertainty in Artificial Intelligence*, 2:149–163, 1988. J.F. Lemmer and L.N. Kanal eds., North-Holland, Amsterdam.
32. R. Shachter and C. Kenley. Gaussian influence diagrams. *Management Science*, 35:527–550, 1989.
33. P. Larrañaga, R. Etxeberria, J.A. Lozano, and J.M. Peña. Optimization in continuous domains by learning and simulation of Gaussian networks. In *Proceedings of the Workshop in Optimization by Building and using Probabilistic Models. A Workshop within the 2000 Genetic and Evolutionary Computation Conference, GECCO 2000*, pages 201–204, Las Vegas, Nevada, USA, 2000.
34. E. Bengoetxea, P. Larrañaga, I. Bloch, A. Perchant, and 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, aug 2000.
35. E. Bengoetxea, P. Larrañaga, I. Bloch, and A. Perchant. Solving graph matching with EDAs using a permutation-based representation. In P. Larrañaga and J. A. Lozano, editors, *Estimation of Distribution Algorithms. A new tool for Evolutionary Computation*. Kluwer Academic Publishers, 2001.
36. D. Whitley and J. Kauth. GENITOR: A different genetic algorithm. In *Proceedings of the Rocky Mountain Conference on Artificial Intelligence*, volume 2, pages 118–130, 1988.

37. J.M. Oliver, D.J. Smith, and J.R.C. Holland. A study of permutation crossover operators on the TSP. In Lawrence Erlbaum, editor, *Genetic Algorithms and their applications: Proceedings of the Second International Conference*, pages 224–230, Hillsdale, New Jersey, 1987. Grefenstette, J.J. (Ed.).
38. W. Banzhaf. The molecular traveling salesman. *Biological Cybernetics*, 64:7–14, 1990.
39. I. Bloch. On fuzzy distances and their use in image processing under imprecision. *Pattern Recognition*, 32:1873–1895, 1999.
40. I. Bloch. Fuzzy relative position between objects in image processing: a morphological approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(7):657–664, 1999.