Efficient algorithms for matching Attributed Graphs and Function-Described Graphs

Francesc Serratosa
Dept. d'Enginyeria Informàtica
Universitat Rovira i Virgili
Francesc.Serratosa@etse.urv.es

René Alquézar
Dept. Lleng. i Sist. Informàtics
Univ. Politècnica de Catalunya
alquezar@lsi.upc.es

Alberto Sanfeliu
Inst. de Robòtica i Informàtica
Univ. Politècnica de Catalunya
asanfeliu@iri.upc.es

Abstract

Function-Described Graphs (FDGs) have been introduced by the authors as a representation of an ensemble of Attributed Graphs (AGs) for structural pattern recognition alternative to first-order random graphs. In previous works, algorithms for the synthesis of FDGs and a branch-and-bound algorithm for error-tolerant graph matching, which computes a distance measure between AGs and FDG, have been reported. Since the worst-case complexity of that matching algorithm is exponential in the number of nodes, an approximate algorithm to compute a sub-optimal measure is proposed in this paper. Results in 3D-object recognition show that, although the computational time is reduced, there is only a slight decrease of effectiveness while classifying an AG against a set of FDGs.

1. Introduction

Function-Described Graphs (FDGs) were introduced in [11] as a representation of an ensemble of *Attributed Graphs* (AGs) for structural pattern recognition. The synthesis of FDGs was studied in [1] and a branch-and-bound algorithm for error-tolerant graph matching (*etgm*), which computes a distance measure between AGs and FDG, has been reported recently [12]. Here, an efficient approximate algorithm for *etgm* is proposed and tested.

It is well known that *etgm* is an NP-Complete problem [6]. Consequently, no algorithm could be constructed that guarantees to find a distance measure between two graphs in polynomial time. In the past, various approaches to *etgm* have been reported which are based on a tree search by A* algorithm. Nevertheless, the search space of the A* algorithm can be greatly reduced by applying *branch-and-bound* techniques that use some heuristic error estimation function [14,10,13,17]. All these methods are guaranteed to find the optimal solution but they require exponential time in the worst case.

However, if the constraints of graph matching are loosened, then it is possible to find solutions in polynomial time by using sub-optimal or approximate methods, which may fail to find the optimal solution. Methods based on probabilistic relaxation are described in [8,2,15]; the basic idea is that each vertex to vertex mapping is assigned a certain probability that reflects the consistency of the mapping. Other continuous optimisation approaches are based on neural networks (using Hopfield nets [4] or Kohonen maps [18]). An approximate algorithm based on relaxation and simulated annealing is presented in [7]. Finally, genetic algorithms have also been applied to search for the optimal labelling in [3,16]. All these approaches have the drawback that use a heuristic optimisation function or are application dependent.

We propose a method to reduce further the search space of our branch and bound algorithm [12] by initially discarding mappings between vertices, using a function based on the distance between sub-units of the graphs to be matched. The match of a pair of vertices is discarded if the distance between their related sub-units is higher than a threshold. Thus, the graphs are broken down into manageable sub-units that overlap, so that two adjacent sub-units contain information about each other through the mutual graph elements they contain.

Larger the topological structure of the sub-units is, the more effective the scheme is in discarding unacceptable labellings. In these terms, small structural units perform badly and the matching process is impoverished in terms of the contextual information. For instance, in the relaxation methods described in [9,5,2], the sub-units are composed by a pair of vertices and their connecting arcs, and the support function does not consider that the labelling has to be bijective, or that the relative order between arcs has to be preserved. If, on the other hand, the structural sub-units are too large, then those constraints can be fulfilled but the matching process becomes excessively cumbersome in terms of its computational requirements; the limitation stems from the need to explore the space of mappings between these sub-units. As a compromise between representational power and computational requirements, Wilson and Hancock proposed splitting the graphs in sub-units which are structured by a central vertex and all the adjacent vertices connected to it by an arc [15]. For convenience we refer to these sub-units as *expanded vertices* (Figure 1). Their probabilistic approach has the main drawback that the semantic information is only used in the initialisation step while the support function is only based on symbolic information. Our approach takes the distance between expanded vertices with semantic knowledge as an informative heuristic to select a reduced set of configurations prior to the computation of an *approximation to the distance measure* through the branch and bound algorithm. In order to provide a good selection (hopefully including the optimal labelling) and reduce the intrinsically used heuristics, the distance measure between expanded vertices is taken to be the same than the one defined between the whole graphs.

2. Attributed Graphs (AGs)

Let $H=(\Sigma_v,\Sigma_e)$ be a directed graph structure of order *n* where $\Sigma_v=\{v_k \mid k=1,...,n\}$ is a set of vertices (or nodes) and $\Sigma_e=\{e_{ij} \mid i,j\in\{1,...,n\}, i\neq j\}$ is a set of edges (or arcs). We use the term *graph element* to refer to either a vertex or an edge. Let Δ_v and Δ_e be the global domains of possible values for non-null attributed vertices and arcs, respectively. A *null value* of a graph element is represented by Φ.

An *attributed graph G* over (Δ_v,Δ_e) with an underlying graph structure $H=(\Sigma_v,\Sigma_e)$ is defined to be a pair (V,A) where $V=(\Sigma_v,\gamma_v)$ is an *attributed vertex set* and $A=(\Sigma_e,\gamma_e)$ is an *attributed arc set*. The mappings $\gamma_v:\Sigma_v\to\Delta_\omega$ and $\gamma_e:\Sigma_e\to\Delta_\varepsilon$ assign attribute values to vertices and arcs, respectively, where $\Delta_\varepsilon=\Delta_e\cup\{\Phi\}$ and $\Delta_\omega=\Delta_v\cup\{\Phi\}$.

A *complete AG* is an AG with a complete graph structure *H* (but possibly including null elements). An attributed graph $G=(V,A)$ of order *n* can be *extended* to form a complete AG $G'=(V',A')$ of order $k, k\geq n$, by adding vertices and arcs with null attribute values Φ. We call *G'* the *k-extension* of *G*.

3. Function-Described Graphs (FDGs)

A *function-described graph F* over (Δ_v,Δ_e) with an underlying graph structure $H=(\Sigma_\omega,\Sigma_\varepsilon)$ is defined to be a tuple (W,B,P,R) such that

1. $W=(\Sigma_\omega,\gamma_\omega)$ is a *random vertex set* and $\gamma_\omega:\Sigma_\omega\to\Omega_\omega$ is a mapping that associates each vertex $\omega_i\in\Sigma_\omega$ with a random variable $\alpha_i=\gamma_\omega(\omega_i)$ with values in Δ_ω.

2. $B=(\Sigma_\varepsilon,\gamma_\varepsilon)$ is a *random arc set* and $\gamma_\varepsilon:\Sigma_\varepsilon\to\Omega_\varepsilon$ is a mapping that associates each arc $\varepsilon_{kl}\in\Sigma_\varepsilon$ with a random variable $\beta_j=\gamma_\varepsilon(\varepsilon_{kl})$ with values in Δ_ε.

3. $P=(P_\omega,P_\varepsilon)$ are two sets of marginal (or first-order) probability density functions for random vertices and edges, respectively. This is, $P_\omega=\{p_i(\mathbf{a}), 1\leq i\leq n\}$ and $P_\varepsilon=\{q_j(\mathbf{b}), 1\leq j\leq m\}$ (being *m* the number of edges), where $p_i(\mathbf{a})\equiv\Pr(\alpha_i=\mathbf{a})$ for all $\mathbf{a}\in\Delta_\omega$ and $q_j(\mathbf{b})\equiv\Pr(\beta_j=\mathbf{b} \mid \alpha_{j1}\neq\Phi\wedge\alpha_{j2}\neq\Phi)$ for all $\mathbf{b}\in\Delta_\varepsilon$ such that α_{j1},α_{j2} refer to the random variables for the endpoints of the random arc associated with β_j. By definition, $\Pr(\beta_j=\Phi \mid \alpha_{j1}=\Phi\vee\alpha_{j2}=\Phi)=1$.

4. $R=(A_\omega,A_\varepsilon,O_\omega,O_\varepsilon,E_\omega,E_\varepsilon)$ is a collection of boolean functions defined over pairs of graph elements (i.e. relations on the sets of vertices and arcs) that allow the incorporation of qualitative second-order probability information. A_ω and A_ε are the *vertex antagonism* and *arc antagonism functions*, respectively, where $A_\omega:\Sigma_\omega\times\Sigma_\omega\to\{0,1\}$ is defined by $A_\omega(\omega_i,\omega_j)=1\Leftrightarrow\Pr(\alpha_i\neq\Phi\wedge\alpha_j\neq\Phi)=0$, and similarly, $A_\varepsilon:\Sigma_\varepsilon\times\Sigma_\varepsilon\to\{0,1\}$ is defined by $A_\varepsilon(\varepsilon_{kl},\varepsilon_{pq})=1\Leftrightarrow\Pr(\beta_i\neq\Phi\wedge\beta_j\neq\Phi)=0$, where $\beta_i=\gamma_\varepsilon(\varepsilon_{kl})$ and $\beta_j=\gamma_\varepsilon(\varepsilon_{pq})$. In addition, O_ω and O_ε are the *vertex occurrence* and *arc occurrence functions*, where $O_\omega:\Sigma_\omega\times\Sigma_\omega\to\{0,1\}$ is defined by $O_\omega(\omega_i,\omega_j)=1\Leftrightarrow\Pr(\alpha_i\neq\Phi\wedge\alpha_j=\Phi)=0$, and $O_\varepsilon:\Sigma_\varepsilon\times\Sigma_\varepsilon\to\{0,1\}$ is defined by $O_\varepsilon(\varepsilon_{kl},\varepsilon_{pq})=1\Leftrightarrow\Pr(\beta_i\neq\Phi\wedge\beta_j=\Phi)=0$. We say that two graph elements (of the same type) are *co-occurrent* if and only if the occurrence relation applies to them in both directions. Finally, E_ω and E_ε are the *vertex existence* and *arc existence functions*, where $E_\omega:\Sigma_\omega\times\Sigma_\omega\to\{0,1\}$ is defined by $E_\omega(\omega_i,\omega_j)=1\Leftrightarrow\Pr(\alpha_i=\Phi\wedge\alpha_j=\Phi)=0$, and $E_\varepsilon:\Sigma_\varepsilon\times\Sigma_\varepsilon\to\{0,1\}$ is defined by $E_\varepsilon(\varepsilon_{kl},\varepsilon_{pq})=1\Leftrightarrow\Pr(\beta_i=\Phi\wedge\beta_j=\Phi)=0$.

A random element δ of an FDG is a *null random element* if its probability of instantiation to the null value is one, $\Pr(\delta=\Phi)=1$. A *complete FDG* is an FDG with a complete graph structure *H*. An FDG $F=(W,B,P,R)$ of order *n* can be *extended* to form a complete FDG $F'=(W',B',P',R')$ of order $k, k\geq n$, by adding null vertices and null arcs and extending appropriately both the set of probability density functions and the boolean functions that relate graph elements. We call *F'* the *k-extension* of *F*.

4. Distance measure between AGs and FDGs

Let *G'* be the *k*-extension of an AG *G* and *F'* be the *k*-extension of an FDG *F*, both defined over a common attribute domain (Δ_v,Δ_e), where *k* is the sum of the number of vertices in *G* and *F*. Let $f:G'\to F'$ be a mapping that labels graph elements, which is actually defined as a pair of morphisms $f = (f_v, f_e)$, where $f_v : \Sigma_v \rightarrow \Sigma_w$ and $f_e : \Sigma_e \rightarrow \Sigma_\varepsilon$ apply on vertices and arcs, respectively.

A reasonable choice for the set *H* of valid morphisms *f* is to require a one-to-one mapping between vertices of the extended graphs and to determine the arc mapping from the vertex mapping assuming structural coherence. In addition, the relative order between arcs must be preserved when dealing with planar graphs. A *global cost* C_f^R can be associated with each *f* in *H*, and the distance measure between *G* and *F* is defined as the minimum of all such costs:

$$d_f^R = \min_{f \in H} \left\{ C_f^R \right\} \qquad (1)$$

Now, the global cost C_f^R is defined with two terms that depend on the first-order probability information and three more terms that depend on the second-order constraints stored in the FDG,

$$C_f^R = \begin{cases} K_1 * \sum_{\forall v_i \in \Sigma_v \text{ of } G'} C_{f_v}\left(v_i, f_v(v_i)\right) + K_2 * \sum_{\forall e_{ij} \in \Sigma_e \text{ of } G'} C_{f_e}\left(e_{ij}, f_e(e_{ij})\right) + \\ K_3 * \sum_{\forall v_i, v_j \in \Sigma_v \text{ of } G'} C_{A_\omega}\left(v_i, v_j, f_v(v_i), f_v(v_j)\right) + \\ K_4 * \sum_{\forall v_i, v_j \in \Sigma_v \text{ of } G'} C_{O_\omega}\left(v_i, v_j, f_v(v_i), f_v(v_j)\right) + \\ K_5 * \sum_{\forall v_i, v_j \in \Sigma_v \text{ of } G'} C_{E_\omega}\left(v_i, v_j, f_v(v_i), f_v(v_j)\right) \end{cases} \qquad (2)$$

These five terms are weighted by non-negative constants K_1 to K_5, to compensate for the different number of elements in the additions as well as to balance the influence of second-order costs with respect to first-order costs in the overall value.

The individual costs caused by vertices are defined using the probabilities stored in the FDG as

$$C_{f_v}(v_i, \omega_q) = \begin{cases} \dfrac{-\ln\left(p_q(\mathbf{a}_i)\right)}{-\ln(K_{\mathrm{Pr}})} & \text{if } p_q(\mathbf{a}_i) \geq K_{\mathrm{Pr}} \\ 1 & \text{otherwise} \end{cases} \qquad (3)$$

where K_{Pr} is a threshold on low probabilities that is introduced to avoid the case $\ln(0)$. In the case of the arcs, the individual costs are defined using the arc conditional probabilities as follows. Let $\gamma_e(e_{ij}) = \mathbf{b}_m$ in the AG arc and let $\gamma_\varepsilon(\varepsilon_{ab}) = \beta_n$ in the matched FDG arc. Then, in general,

$$C_{f_e}(e_{ij}, \varepsilon_{ab}) = \begin{cases} \dfrac{-\ln\left(q_n(\mathbf{b}_m)\right)}{-\ln(K_{\mathrm{Pr}})} & \text{if } q_n(\mathbf{b}_m) \geq K_{\mathrm{Pr}} \\ 1 & \text{otherwise} \end{cases} \qquad (4)$$

However, if either v_i or v_j is a null extended vertex in the AG, then the conditional probability $q_n(\mathbf{b}_m)$ is not applicable and must be replaced by the conditional probability $\Pr\left(\beta_n = \mathbf{b}_m \middle| \alpha_a = \Phi \vee \alpha_b = \Phi\right)$, which is 1 if $\mathbf{b}_m = \Phi$ and 0 otherwise.

Some second-order costs are added to the global cost of the labelling when second-order constraints (antagonism, occurrence, existence) are broken. Equations (5) to (7) show the second-order costs, which can be only 1 or 0, associated with the three relations of antagonism, occurrence and existence between pairs of vertices.

$$C_{A_\omega}(v_i, v_j, \omega_p, \omega_q) = \begin{cases} A_\omega(\omega_p, \omega_q) & \text{if } \mathbf{a}_i \neq \Phi \wedge \mathbf{a}_j \neq \Phi \wedge p_p(\Phi) \neq 1 \wedge p_q(\Phi) \neq 1 \\ 0 & \text{otherwise} \end{cases} \qquad (5)$$

$$C_{O_\omega}(v_i, v_j, \omega_p, \omega_q) = \begin{cases} O_\omega(\omega_p, \omega_q) & \text{if } \mathbf{a}_i \neq \Phi \wedge \mathbf{a}_j = \Phi \wedge p_p(\Phi) \neq 1 \\ 0 & \text{otherwise} \end{cases} \qquad (6)$$

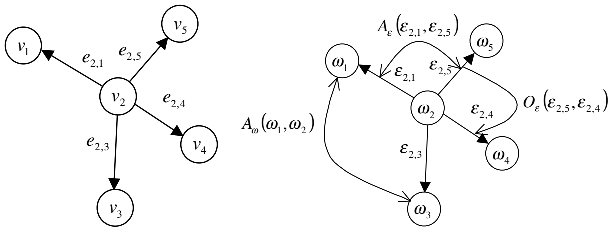
$$C_{E_\omega}(v_i, v_j, \omega_p, \omega_q) = \begin{cases} E_\omega(\omega_p, \omega_q) & \text{if } \mathbf{a}_i = \Phi \wedge \mathbf{a}_j = \Phi \wedge p_p(\Phi) \neq 1 \wedge p_q(\Phi) \neq 1 \\ 0 & \text{otherwise} \end{cases} \qquad (7)$$

5. Expanded Vertices

An AG expanded vertex (Fig. 1.a) or an FDG expanded vertex (Fig 1.b) are an AG or an FDG, respectively, with a specific structure formed by a *central vertex* and all the adjacent vertices, called *external vertices*, which are connected to it by an outgoing arc. More formally, an *AG expanded vertex* EV_i over (Δ_v, Δ_e) with an underlying graph structure $H = (\Sigma_v, \Sigma_e)$ is defined to be an AG $EV_i = (V, A)$ where the vertices belong to a finite set $\Sigma_v = \{v_1, \ldots, v_i, \ldots v_n\}$ that contains a given vertex v_i and the arcs belong to a finite set $\Sigma_e = \{e_{i,1}, \ldots, e_{i,i-1}, e_{i,i+1}, \ldots e_{i,n}\}$ formed by all the arcs departing from v_i in the AG. The vertex v_i is called the *central vertex* and the vertices $v_j : i \neq j, 1 \leq j \leq n$ are called the *external vertices*.

An AG *G* of order *k* over the domain (Δ_v, Δ_e) with structure $H = (\Sigma_v, \Sigma_e)$ can be represented as a set of expanded vertices $\{EV_i\}$, such that $G = \bigcup_{i=1..k} EV_i$, where v_i is the central vertex of $EV_i, \forall v_i \in \Sigma_v$.

An *FDG expanded vertex* EW_i over (Δ_v, Δ_e) with an underlying graph structure $H = (\Sigma_\omega, \Sigma_\varepsilon)$ is defined to be an FDG $EW_i = (W, B, P, R)$ with a finite set of vertices $\Sigma_\omega = \{\omega_1, \ldots, \omega_i, \ldots \omega_m\}$ that contains a given vertex ω_i and a finite set of arcs $\Sigma_\varepsilon = \{\varepsilon_{i,1}, \ldots, \varepsilon_{i,i-1}, \varepsilon_{i,i+1}, \ldots \varepsilon_{i,n}\}$ formed by all the arcs departing from ω_i in the FDG. Again, the vertex ω_i is called the *central vertex* and the vertices $\omega_j : i \neq j, 1 \leq j \leq m$ are called the *external vertices*.



(a) (b)

Figure 1. Expanded vertex of an AG (a) and an FDG (b).

An FDG *F* can be partially represented in distributed form as the set of its expanded vertices $\{EW_i\}$, where ω_i is the central vertex of $EW_i, \forall \omega_i \in \Sigma_\omega$. We note that the whole underlying structure of an FDG is included in this new representation; the arcs appear once but vertices possibly are included more than once. However, the antagonism, occurrence and existence relations of the original FDG cannot be represented entirely: those that involve two graph elements belonging to the same expanded vertex are included, but this is not possible when the related graph elements belong to different expanded vertices.

6. Distance between expanded vertices

The distance between an AG expanded vertex and an FDG expanded vertex is the same than the distance defined between an AG and an FDG. That is possible because expanded vertices can be seen as graphs with a specific structure. However, when both expanded vertices are non-null, the mapping *f* between graph elements of expanded vertices EV_i and EW_j is constrained to satisfy the obvious requirement $f(v_i) = \omega_j$, i.e. the central vertices are mapped one to the other.

With this consideration, and not including the second-order costs, the maximum distance d_{max} obtained between two non-null expanded vertices is

$$d_{max} = \begin{cases} 2n-1 & \text{if } n \geq m \\ n+m-1 & \text{if } n < m \end{cases} \tag{8}$$

where *n* and *m* are the number of vertices of the AG expanded vertex and the FDG expanded vertex, respectively. If $n \geq m$ then the maximum distance value is the cost of substituting $m-1$ arcs and *m* vertices plus the cost of inserting $n-m$ arcs and vertices. On the contrary, if $n < m$, the maximum value is the cost of substituting $n-1$ arcs and *n* vertices plus the cost of deleting $m-n$ AG vertices. The cost of deleting the AG arcs is zero when the external vertices have been deleted.

7. A fast algorithm to compute the distance between expanded vertices

Algorithm 1 calculates the distance between expanded vertices with a computational cost $O(n^2 \cdot m)$. It is based on the idea that if the labelling function has to be coherent structurally and the order between arcs has to be maintained, then the external vertices and their arcs can be regarded as a cyclic string where each element is a pair (arc, external vertex). In the case of an AG expanded vertex EV_i, the string is defined as $S_i = \{(e_{i(1)}, v_{i(1)}), \ldots, (e_{i(n')}, v_{i(n')})\}$ where $\{v_{i(1)}, \ldots, v_{i(n')}\}$ is an ordered set of external vertices of EV_i, being $e_{i(j)} = e_{ik} \Leftrightarrow v_{i(j)} = v_k$ and *n'*=*n*-1. Similarly, the string related with an FDG expanded vertex EW_j is defined as $S_j = \{(\varepsilon_{i(1)}, \omega_{i(1)}), \ldots, (\varepsilon_{i(m')}, \omega_{i(m')})\}$ where $\{\omega_{i(1)}, \ldots, \omega_{i(m')}\}$ is an ordered set of external vertices of EW_j, being $\varepsilon_{i(j)} = \varepsilon_{ik} \Leftrightarrow \omega_{i(j)} = \omega_k$ and *m'*=*m*-1.

For each possible orientation of one of the expanded vertices (subscript *s* in the algorithm), the distance between the strings S_i and S_j is computed by a method reminiscent on the Levenshtein algorithm, with the difference that the cost of inserting or deleting string elements is obtained by substituting these elements with null vertices and arcs. Moreover, the cost to substitute the central vertices is initially applied independently of the string alignment cost caused by arcs and external vertices.

Algorithm 1: Distance-between-expanded-vertices
Inputs: EV_i: AG expanded vertex; EW_j: FDG expanded vertex.
Output: The distance d_f between EV_i and EW_j.
Begin
Let S_i and S_j be the cyclic strings of EV_i and EW_j, respectively.
$d_f := \infty$;
$D(0,0) := C_{f_v}(v_i, \omega_j)$; { $D(0,0) := 0$ in the Levenshtein algorithm}
For s:=0 **to** n'-1 **do** {orientation of the AG expanded vertex}
  For l:=1 **to** n' **do**
    $D(l,0) := D(l-1,0) + C_{f_v}(v_{i(l+s)}, \omega_\Phi) + C_{f_e}(e_{i,i(l+s)}, \varepsilon_\Phi)$
  end-for
  For k:=1 **to** m' **do**
    $D(0,k) := D(0,k-1) + C_{f_v}(v_\Phi, \omega_{j(k)}) + C_{f_e}(e_\Phi, \varepsilon_{j,j(k)})$
  end-for
  For l:=1 **to** n' **do**
   For k:=1 **to** m' **do**
     $m_1 := D(l-1,k-1) + C_{f_v}(v_{i(l+s)}, \omega_{j(k)}) + C_{f_e}(e_{i,i(l+s)}, \varepsilon_{j,j(k)})$
     {Substitution in the Lev. algorithm}
     $m_2 := D(l-1,k) + C_{f_v}(v_{i(l+s)}, \omega_\Phi) + C_{f_e}(e_{i,i(l+s)}, \varepsilon_\Phi)$
     {Insertion in the Levenshtein algorithm}
     $m_3 := D(l,k-1) + C_{f_v}(v_\Phi, \omega_{j(k)}) + C_{f_e}(e_\Phi, \varepsilon_{j,j(k)})$
     {Deletion in the Levenshtein algorithm}
     $D(l,k) := \min(m_1, m_2, m_3)$
   end-for
  end-for
  if $D(n',m') < d_f$ **then** $d_f := D(n',m')$ **end-if**
end-for
end-algorithm

8. A sub-optimal method to compute the distance between AGs and FDGs

Figure 2 shows the basic scheme of the method proposed to compute a sub-optimal approximation of $d_f^R(G,F)$ within a set of labelling functions H, denoted $d_f^{R\wedge}(G,F)$, and also, the best labelling reached $f\wedge \in H$. Obviously, $d_f^{R\wedge}(G,F) \geq d_f^R(G,F)$, since some labelling functions, which belong to H, are discarded due to the constraints derived from the distances between expanded vertices. The method consists of three main modules. The first one computes all the distances between expanded vertices using the distance measure without second order costs, $d_f(EV_i, EW_j)$. The thresholding module decides which matches of vertices are discarded depending on an externally imposed threshold τ, $\tau \in [0,1]$. If the normalised distance between expanded vertices is higher than τ the mapping is considered unlikely and it is forbidden.

$$Forbid(v_i, \omega_j) = \begin{cases} True & \text{if } \dfrac{d_f(EV_i, EW_j)}{2n_i - 1} > \tau \wedge n_i \geq m_j \\ True & \text{if } \dfrac{d_f(EV_i, EW_j)}{n_i + m_j - 1} > \tau \wedge n_i < m_j \\ False & \text{otherwise} \end{cases} \quad (9)$$

where n_i and m_j represent the number of vertices of the expanded vertices EV_i and EW_j, respectively.

The last module computes the sub-optimal distance $d_f^{R\wedge}(G,F)$ using the branch and bound algorithm presented in [12] but only on the set of vertex mappings allowed by the thresholding module, the ones for which $Forbid(v_i^G, \omega_j^F) = False$.

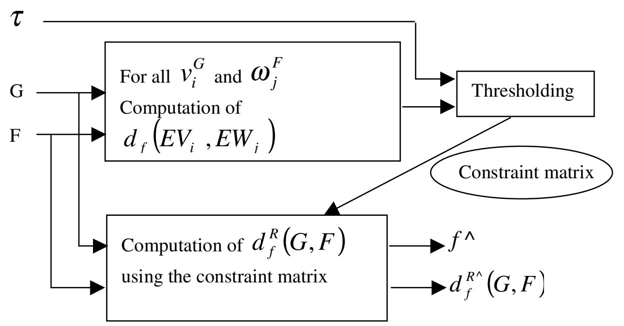


Figure 2. Basic scheme of the method for sub-optimal distance computation.

Note that, given a set of distances between expanded vertices, the lower is τ, the more restrictions are imposed. Then, the branch and bound algorithm has to explore a less number of possible morphisms and therefore, more approximated is the measure and faster is the process. If $\tau = 1$, then the combinatorial algorithm explores the whole set of possible morphisms and the output is exactly $d_f^R(G,F)$. On the contrary, if $\tau = 0$, only the matches of totally isomorphic subunits are not discarded, and the combinatorial algorithm explores few (or any) morphisms, being $d_f^{R\wedge}(G,F)$ "very sub-optimal".

9. Results

The contribution of FDGs to structural pattern recognition is illustrated by the three-dimensional object recognition problem. The original data is composed by 101 AGs, which represent the semantic and structural information of the views taken from five objects. Vertices in the AGs represent the faces, with one attribute, which is the area of the face. Arcs represent the edges between faces, with one attribute, which is the length of the edge. Five FDGs were built from the AGs that represent their views using a supervised synthesis method. An antagonism relation between two graph elements appears when these elements have never seen together in the same view. On the other hand, an occurrence relation appears when a graph element is visible in all the views in which another one is visible too. See [12] for more details.

The aim of the tests performed was to study the balance between the effectiveness and efficiency in the classification process. Table 1 shows the recognition ratio applying different levels of semantic (standard deviation) and structural noise (number of vertices deleted or inserted) on the test set and different thresholds on the sub-optimal distance.

# vertices	**0**	**0**	**0**	**0**	**1**	**2**	**1**
St. Dev.	**0.0**	**4.0**	**8.0**	**12.0**	**0.0**	**0.0**	**8.0**
$\tau = 1$	100	97.1	95.0	92.5	89.2	85.2	83.6
$\tau = 0.4$	100	96.5	94.9	92.6	87.5	82.4	80.3
$\tau = 0.3$	100	80.3	75.6	71.0	81.5	77.7	72.1
$\tau = 0.2$	100	73.7	70.0	56.2	72.3	65.3	63.2
$\tau = 0.1$	100	46.3	26.9	25.2	56.2	50.1	45.3

Table 1. Recognition ratio (% of correct classification).

# vertices	**0**	**0**	**0**	**0**	**1**	**2**	**1**
St. Dev.	**0.0**	**4.0**	**8.0**	**12.0**	**0.0**	**0.0**	**8.0**
$\tau = 1$	**1.00**	1.51	3.03	3.75	2.07	3.13	4.21
$\tau = 0.4$	0.99	1.21	1.59	2.79	1.95	2.97	3.54
$\tau = 0.3$	0.84	1.01	1.00	1.08	1.12	1.03	2.06
$\tau = 0.2$	0.57	0.77	0.86	0.75	1.15	0.78	0.89
$\tau = 0.1$	0.09	0.56	0.49	0.35	0.89	0.35	0.47

Table 2. Computational costs (normalised run times).

Table 2 shows the computational costs associated with the results of Table 1. All the values are normalised by the time spent to compute the classification according to the optimal distance, that is $\tau = 1$, and without noise.

In the cases where $\tau = 1$, all the mappings are allowed and the first and second steps are not computed, thus saving their run time. For this reason it would be possible to have a run time with $\tau < 1$ higher than with $\tau = 1$. There is not a high gain in efficiency in this application without a significant loss of classification correctness. We believe it is basically for two reasons. The former, some combinations that the first step forbids would be possibly removed by the branch and bound technique. The latter, the AGs are small (five vertices as average), and then the cost to compute the first step is equivalent to the cost of the third, considering the branch and bound technique.

10. Conclusions

An approximate method for the computation of a sub-optimal distance measure between AGs and FDGs has been presented. The aim of this method is to reduce the number of possible vertex mappings before the branch and bound algorithm is run to compute the distance measure. Results show that the computational time is lower with a moderate loss of effectiveness for classifying small-size AGs. Nevertheless, it is the authors believe that a much higher improvement would be obtained for larger AGs.

Acknowledgements

This work has been partially supported by the Spanish "Comisión Interministerial de Ciencia y Tecnología", under project TAP98-0473.

References

[1] R. Alquezar, A. Sanfeliu and F. Serratosa, "Synthesis of Function-Described Graphs". *Advances in Pattern Recognition, Proc. Joint IAPR Int. Workshops SSPR'98 and SPR'98,* Sydney, Australia, Springer LNCS-1451, pp. 112-121, 1998.

[2] W.J. Christmas, J. Kittler and M. Petrou, "Structural matching in computer vision using probabilistic relaxation", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 17, pp. 749-764, 1995.

[3] A. Cross, R. Wilson and E.Hancock, "Genetic search for structural matching", *Computer Vision- FCCV'96, Lecture notes in computer sicence 1064, Springer Verlag*, pp. 514-525, 1996.

[4] J. Feng, M. Laumy and M. Dhome, "Inexact matching using neural networks", *Pattern recognition in practice IV: Multiple paradigms, comparative studies and hybrid systems*, pp. 177-184, 1994.

[5] O.D. Faugeras and K.E. Price, "Semantic labelling of Aerial Images using Stochastic Relaxation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, pp. 633-642, 1981.

[6] M.R. Garey and D.S. Johnson, "Computers and intractability: A guide to the theory of NP-Completeness", *Freeman and Company*, 1979.

[7] L. Herault, R. Horaud, F. Veillon and J.J. Niez, "Symbolic image matching by simulated annealing", *British machine vision conference*, Oxford, pp. 319-324, 1990.

[8] J. Kittler, W.J. Christmas and M. Petrou, "Probabilistic relaxation for matching problems in machine vision", *Proceedings 4th International Conference on Computer Vision*, pp. 666-674, 1993.

[9] A. Rosenfeld, R.A. Hummel and S.W. Zucker, "Scene labeling by relaxation operators", *IEEE Transactions on Systems, Man and Cybernetics*, vol. 6, pp. 420-443, 1976.

[10] A. Sanfeliu and K. Fu, "A distance measure between attributed relational graphs for pattern recognition", *IEEE Transactions on Systems, Man and Cybernetics*, vol. 13, pp. 353-362, 1983.

[11] F. Serratosa and A. Sanfeliu, "Function-Described Graphs applied to 3D object recognition". In *Proceedings ICIAP'97, 9th Int. Conf. Image Analysis and Processing,* Firenze, Italy, Vol. I, pp. 701-708, 1997.

[12] F. Serratosa, R. Alquézar and A. Sanfeliu, "Function-Described Graphs for structural pattern recognition", Submitted to *Transactions on Pattern Analysis and Machine Intelligence*, 1999.

[13] L. Shapiro and R.M. Haralick, "A metric for comparing relational descriptions", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol 7, pp. 90-94, 1985.

[14] W.H. Tsai and K.S. Fu, "Error-correcting isomorphism of attributed relational graphs for pattern analysis", *IEEE Transactions on Systems, Man and Cybernetics*, vol. 9, pp. 757-768, 1979.

[15] R.C. Wilson and E.R. Hancock, "Structural matching by discrete relaxation", *Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, pp. 634-648, 1997.

[16] Y.K. Wang, K.C. Fan and J.T. Horng, "Genetic-based search for error-correcting graph isomorphism", *IEEE Transactions on Systems Man and Cybernetics*, vol. 27, pp. 588-597, 1997.

[17] A.K.C. Wong, M. You and S.C. Chan, "An algorithm for graph optimal monomorphism", *IEEE Transactions on Systems, Man and Cybernetics*, vol. 20, pp. 628-636, 1990.

[18] L. Xu and E. Oja, "Improved Simulated annealing, Botzmann machine and attributed graph matching", *Lec. Notes in Comp. Sc.*, vol. 412, pp. 151-161, 1990.