Matching Attributed Graphs:  
2nd-Order Probabilities for Pruning the Search Tree

Francesc Serratos\(^1\) and Alberto Sanfeliu\(^2\)

\(^1\) Universitat Rovira i Virgili, Dept. d'Enginyeria Informàtica i Matemàtiques, Spain  
Francesc.Serratos@urv.net  
http://www.etse.urv.es/~fserrato

\(^2\) Universitat Politècnica de Catalunya, Institut de Robòtica i Informàtica Industrial, Spain  
sanfeliu@iri.upc.es

Abstract. A branch-and-bound algorithm for matching Attributed Graphs (AGs) with Second-Order Random Graphs (SORGs) is presented. We show that the search space explored by this algorithm is drastically reduced by using the information of the 2\(^{nd}\)-order joint probabilities of vertices of the SORGs. A SORG is a model graph, described elsewhere, that contains 1\(^{st}\) and 2\(^{nd}\)-order probabilities of attribute relations between elements for representing a set of AGs compactly. In this work, we have applied SORGs and the reported algorithm to the recognition of real-life objects on images and the results show that the use of 2\(^{nd}\)-order relations between vertices is not only useful to decrease the run time but also to increase the correct classification ratio.

1 Introduction

A Second Order Random Graph (SORG) is a model graph introduced by the authors that contains 1\(^{st}\)-order and 2\(^{nd}\)-order probabilities of attributes to describe a set of Attributed Graphs (AGs) [1,2]. Let us consider, as an example, the 3D-object modelling and recognition problem. The basic idea is that only a single SORG is synthesised from the AGs that represent several views of a 3D-object. Therefore, in the recognition process, only one comparison is needed between each object model represented by a SORG and the unclassified object (view of a 3D-object) represented by an AG.

SORGs can be seen as a generalisation of FDGs [3,4] and First-Order Random Graphs [5,6]. Moreover, Bunke [7] presented a model of sets of graphs, called network of models, in which all the graphs are pre-processed generating a symbolic data structure. In the SORGs, to deal with the 1\(^{st}\)-order and 2\(^{nd}\)-order probabilities, there is a random variable \(\alpha_i\) (or \(\beta_i\)) associated with each vertex \(\omega_i\) (or arc \(\epsilon_i\), respectively), which represents the attribute information of the corresponding graph elements in the set of AGs. A random variable has a probability density function \(p_i\) defined over the same attribute domain of the AGs, including a null value \(\Phi\) that denotes the non-instantiation of an SORG graph element in an AG.

The distance measure between an AG and a SORG was proposed in [2] for error-tolerant graph matching. Here, we present a branch-and-bound algorithm which computes exactly this distance measure. This algorithm uses the 2\(^{nd}\)-order probabilities in the SORG to prune the search tree.
2 Formal Definitions of Random-Graph Representation

**Definition 1: Attributed Graph (AG).** Let $\Delta_v$ and $\Delta_e$ denote the domains of possible values for attributed vertices and arcs, respectively. These domains are assumed to include a special value $\Phi$ that represents a null value of a vertex or arc. An AG $G$ over $(\Delta_v, \Delta_e)$ is defined to be a four-tuple $G=(\Sigma_v, \Sigma_e, \gamma_v, \gamma_e)$, where $\Sigma_v = \{v_k \mid k = 1, ..., n\}$ is a set of vertices (or nodes), $\Sigma_e = \{e_{ij} \mid i, j \in [1, ..., n], i \neq j\}$ is a set of arcs (or edges), and the mappings $\gamma_v : \Sigma_v \rightarrow \Delta_v$ and $\gamma_e : \Sigma_e \rightarrow \Delta_e$ assign attribute values to vertices and arcs.

**Definition 2: Random Graph (RG).** Let $\Omega_v$ and $\Omega_e$ be two sets of random variables with values in $\Delta_v$ (random vertices) and in $\Delta_e$ (random arcs), respectively. A random-graph structure $R$ over $(\Delta_v, \Delta_e)$ is defined to be a tuple $(\Sigma_v, \Sigma_e, \gamma_v, \gamma_e, P)$, where $\Sigma_v = \{\omega_k \mid k = 1, ..., n\}$ is a set of vertices, $\Sigma_e = \{e_{ij} \mid i, j \in [1, ..., n], i \neq j\}$ is a set of arcs, the mapping $\gamma_v : \Sigma_v \rightarrow \Omega_v$ associates each vertex $\omega_k \in \Sigma_v$ with a random variable $\alpha_k = \gamma_v(\omega_k)$ with values in $\Delta_v$, and $\gamma_e : \Sigma_e \rightarrow \Omega_e$ associates each arc $e_{ij} \in \Sigma_e$ with a random variable $\beta_{ij} = \gamma_e(e_{ij})$ with values in $\Delta_e$. And, finally, $P$ is a joint probability distribution $P(\alpha_1, ..., \alpha_n, \beta_1, ..., \beta_m)$ of all the random vertices $\{\alpha_i \mid \alpha_i = \gamma_v(\omega_i), 1 \leq i \leq n\}$ and random arcs $\{\beta_{ij} \mid \beta_{ij} = \gamma_e(e_{ij}), 1 \leq j \leq m\}$.

**Definition 3: Probability of a RG instantiation.** Given an AG $G$ and a RG $R$, the joint probability of random vertices and arcs is defined over an instantiation that produces $G$, and such instantiation is associated with a structural isomorphism $\mu : G' \rightarrow R$, where $G'$ is the extension of $G$ to the order of $R$. $G'$ represents the same object than $G$ but some vertices or arcs have been added with the null value $\Phi$ to be $\mu$ bijective. Let $G$ be oriented with respect to $R$ by the structurally coherent isomorphism $\mu$; for each vertex $\omega_i$ in $R$, let $a_i = \gamma_v(\mu^{-1}(\omega_i))$ be the corresponding attribute value in $G'$, and similarly, for each arc $e_{kl}$ in $R$ (associated with random variable $\beta_{ij}$) let $b_{ij} = \gamma_e(\mu^{-1}(e_{kl}))$ be the corresponding attribute value in $G'$. Then the probability of $G$ according to (or given by) the orientation $\mu$, denoted by $P_R(G|\mu)$, is defined as

$$P_R(G|\mu) = \Pr\left(\land_{i=1}^n (\alpha_i = a_i) \land \land_{j=1}^m (\beta_{ij} = b_{ij})\right) = p(a_1, ..., a_n, b_1, ..., b_m)$$  \hspace{1cm} (1)

We define $d_i$ to represent a vertex or arc attribute value ($a_i$ or $b_i$). Thus, if $s$ is the number of vertices and arcs, $s=m+n$, eq. (1) can be rewritten as,

$$P_R(G|\mu) = p(d_1, ..., d_s)$$  \hspace{1cm} (2)

3 Second-Order Random-Graph Representation

If we want to represent the cluster of AGs by a RG, it is impractical to consider the high order probability distribution defined in the RGs $P(\alpha_1, ..., \alpha_n, \beta_1, ..., \beta_m)$ (defini-
tion 2), where all components and their relations in the structural patterns are taken jointly due to time and space costs. For this reason, some other more practical approaches have been presented that propose different approximations [1,4,5,6]. All of them take into account in some manner the incidence relations between attributed vertices and arcs, i.e., assume some sort of dependence of an arc on its connecting vertices. Also, a common ordering (or labelling) scheme is needed that relates vertices and arcs of all the involved AGs, which is obtained through an optimal graph mapping process called synthesis of the random graph representation. We showed in [1] that all the approximations in the literature of the joint probability of an instantiation of the random elements in a RG (eq. 1) can be described in a general form as follows:

\[ P_r(G|\mu) = \prod_{i=1}^{s} p_i(a_i) \prod_{i=1}^{m} \prod_{j=i+1}^{m} r_{ij}(a_i, a_j) \prod_{i=1}^{n} \prod_{j=i+1}^{n} r_{ij}(b_i, b_j) \]  

(3)

where \( p_i \) are the marginal probabilities of the \( s \) random elements \( \gamma_i \) (vertices or arcs) and \( r_{ij} \) are the Peleg compatibility coefficients [9] that take into account both the marginal and 2nd-order joint probabilities of random vertices and arcs.

According to eq. (2), we can generalise the joint probability as,

\[ P_r(G|\mu) = p(d_1, \ldots, d_s) = \prod_{i=1}^{s} p_i(d_i) \prod_{i=1}^{s} \prod_{j=i+1}^{s} r_{ij}(d_i, d_j) \]  

(4)

and define the Peleg coefficient,

\[ r_{ij}(d_i, d_j) = \frac{p_{ij}(d_i, d_j)}{p_i(d_i)p_j(d_j)} \]  

(5)

The Peleg coefficient, with a non-negative range, is related to the “degree” of dependence between two random variables. If they are independent, the joint probability, \( p_{ij} \), is defined as the product of the marginal ones, thus, \( r_{ij} = 1 \) (or a value close to 1 if the probability functions are estimated). If one of the marginal probabilities is null, the joint probability is also null. In this case, the indecisiveness 0/0 is solved as 1, since this do not affect the global joint probability, which is null.

4 Distance Measure Between AGs and SORGs

The distance measure presented in this section provides a quantitative value of the match between an AG \( G \) (data graph) and a SORG \( S \) (model graph) similar to the one presented in [2]. It is related to the probability of \( G \) according to the labelling function \( \mu : G \rightarrow S \), denoted \( P(G|\mu) \) in eq. (4). We may attempt to minimise a global cost measure \( C \) of the morphism \( \mu \) in the set \( H \) of allowable configurations, by taking the cost as a monotonic decreasing function of the conditional probability of the data graph given the labelling function, \( C = f(P(G|\mu)) \). With some steps depicted in [2] we arrive to the final expression

\[ C(G|\mu) = -(s-2) \sum_{i=1}^{s} C_i^2(d_i) + \sum_{i=1}^{s-1} \sum_{j=i+1}^{s} C_{ij}^2(d_i, d_j) \]  

(6)
where first-order and second order costs are given by
\[
C^1_i(d_i) = \text{Cost}(p_i(d_i)) \quad C^2_{i,j}(d_i, d_j) = \text{Cost}(p_{i,j}(d_i, d_j))
\]
and the function \(\text{Cost}(Pr)\) yields a bounded normalized cost value between 0 and 1 depending on the negative logarithm of a given probability \(Pr\) and parameterised by a positive constant \(K_{pr} \in [0,1]\), which is a threshold on low probabilities that is introduced to avoid the case \(\ln(0)\), which would give negative infinity. This is,
\[
\text{Cost}(Pr) = \begin{cases} 
-\ln(Pr) & \text{if } Pr \geq K_{pr} \\
-\ln(K_{pr}) & \text{otherwise} 
\end{cases}
\]

Once a cost measure \(C\) is defined, a distance measure between an AG and a SORG and the optimal labelling \(\mu^*\) are defined respectively as
\[
d = \min_{\mu \in H} \left\{ C(G|\mu) \right\} \quad \text{and} \quad \mu^* = \text{arg} \min_{\mu \in H} \left\{ C(G|\mu) \right\}
\]

5 Algorithm for Computing the Distance Measure

The distance and the optimal morphism between an AG \(G\) and an SORG \(F\) are calculated by an algorithm for error-tolerant graph matching. Our approach is based on a tree search by A* algorithm, where the search space is reduced by a branch and bound technique. The algorithm searches a tree where the nodes represent possible mappings between vertices of both graphs and branches represent combinations of pairs of graph vertices that satisfy the labelling constraints. Hence, the paths from the root to the leaves represent allowed labellings \(f\).

The distance measure has been theoretically defined such that both graphs are extended with null elements to have the same number of elements and to be complete. Nevertheless, in practice, our algorithm only needs the SORG to be extended with one null vertex, because the different permutations of the null vertices are regarded as equivalent labellings. Thus, the AG spurious vertices are possibly matched with this unique null SORG vertex \((v_0, e)\) and hence the mapping is not forced to be injective. On the other hand, the SORG graph elements that remain unmatched when arriving at a leaf are considered to be matched with null AG vertices \(v_\emptyset\) or null AG arcs \(e_\emptyset\). Consequently, a final cost of deleting these elements may be added to the cost of the labelling in the leaves of the search tree. Nevertheless, if a sub-graph isomorphism from an AG to a SORG is looked for, then it is not needed to match all the SORG vertices with an AG vertex (null or not) and this deleting cost has not to be computed. This is the case of our application in Section 5.

In general, solving a branch and bound problem requires a branch evaluation function and a global evaluation function. The former assigns a cost to the branch incident to a node \(N\) of the tree, which is the cost of the new match (or pair) appended. The latter is used to guide the search at a node \(N\) and refers to the cost of the best complete path through \(N\) (i.e. including the pairs of vertices already matched when arriving at \(N\)). The cost of a labelling \(f\) is given by the value of the global evaluation function in a leaf of the search tree.
Each node \( N \) of the search tree at level \( p>0 \) is described by a collection of pairs of vertices of the graphs, \( N = \{(v_i, \omega_q)\} \), where \( i=1,2,...,p \) correspond to the indices of the vertices \( v_i \) in the AG and \( q_i \) are the distinct indices of the vertices \( \omega_{q_i} \) in the SORG such that \( f(v_i) = \omega_{q_i} \). Moreover, we define the sets \( N_v = \{v_1, v_2, ..., v_p\} \) and \( N_\omega = \{\omega_1, \omega_2, ..., \omega_p\} \) of vertices that have already been matched between both graphs, and also, the sets \( M_v = \{v_{p+1}, v_{p+2}, ..., v_n\} \) and \( M_\omega = \{\omega_{q_{p+1}}, \omega_{q_{p+2}}, ..., \omega_{q_n}\} \) of vertices that have not been matched yet. Assume that \( N = \{(v_i, \omega_{q_i})| v_i \neq v_j \in M_v \} \) indicates the unique path from the root to a tree node \( N \) and \( T = \{(v_i, \omega_{q_i})| v_i \neq v_j \in M_\omega \} \) indicates the unique path from the root to a leaf \( T \). The vertices of \( M_\omega \) in each node \( N \) are explored using the order imposed by the costs \( C^1(v_p, \omega_j) \) being \( \omega_j \in M_\omega \).

The branch evaluation function \( K \) depends on the cost of the new match between vertices, the cost of all the arcs related to these two vertices that involve vertices from \( N_v \) and \( N_\omega \) and the 2nd-order costs referring to these same vertices. Thus, the cost assigned to the branch incident to \( N \) is given by

\[
K(v_i, \omega_j) = -(p-3)
\left(
C(v_i, \omega_j) + \sum_{k=1}^{n} C(e_{v_i, \omega_{q_k}}) + C(e_{v_i, \omega_{q_{p+k}}}) + \sum_{q=1}^{q_n} C(v_j, \omega_q, \omega_{q_j})
\right)
\]

(10)

The global evaluation function \( l^*(N) \) at a node \( N \) of level \( p \) is defined as the cost \( g^*(N) \) of an optimal path from the root to the node \( N \) plus the cost \( h^*(N) \) of an optimal path from the node \( N \) to a leaf \( T = \{(v_i, \omega_{q_i})| i = 1,2,...,n \} \) constrained to be reached through the node \( N \):

\[
l^*(N) = g^*(N) + h^*(N) \quad \text{where} \quad g^*(N) = \sum_{i=1}^{p} K(v_i, \omega_j) \quad \text{and} \quad h^*(N) = \min_{i=1}^{n} h^*(v_i, \omega_{q_i})
\]

(11)

where \( t \) denotes a feasible path from \( N \) to \( T \).

On the other hand, the global evaluation function \( l^*(N) \) is unknown in an inner node \( N \), since \( h^*(N) \) is unknown and can only be approximated by a consistent lower-bounded estimate.

For that purpose, let \( K^*(v_i, \omega_j) \) be the cost of adding a pair of vertices to \( N \), where \( v_i \in M_v \) and \( \omega_j \in M_\omega \), defined as

\[
K^*(v_i, \omega_j) = -(p-3)
\left(
C(v_i, \omega_j) + \sum_{k=1}^{n} C(e_{v_i, \omega_{q_k}}) + C(e_{v_i, \omega_{q_{p+k}}}) + \sum_{q=1}^{q_n} C(v_j, \omega_q, \omega_{q_j})
\right)
\]

(12)

Then, for each unmatched vertex \( v_i \in M_v \), a corresponding vertex \( \omega_j \in M_\omega \) can be associated such that the cost \( K^*(v_i, \omega_j) \) is minimised.

Finally, the heuristic function \( l(N) \) that estimates \( l^*(N) \) in a node \( N \) is given by

\[
l(N) = g^*(N) + h(N) \quad \text{where} \quad h(N) = \sum_{i=p+1}^{n} \min_{\forall \omega_j \in M_\omega} \{K^*(v_i, \omega_j)\}
\]

is a consistent lower bounded estimate of \( h^*(N) \).
The algorithm to compute the distance measure \( d \) and the corresponding optimal labelling \( f_{opt} \) between a given AG and a given FDG only invokes the following recursive procedure TreeSearch at the root node.

**Procedure** TreeSearch\((G,F,f,g^*,v_i,d_j,f_{opt})\)

**Input parameters:** \( G \) and \( F \): An AG and a SORG

\( f \): Optimal path (labelling) from the root to the current node

\( g^* \): Minimum value from the root to the current node

\( v_i \): Current AG vertex to be matched

**Let** \( W \) be a sequence of \( w_i \) of \( F \) ordered by \( C'(v_i,w_i) \)

**For** each vertex \( w_i \) in \( W \) not used yet in \( f \) or \( w_o \) do

\( K:= \) Branch-Evaluation-Function\((G,F,f,v_i,w_i)\)

\( h:= \) Bound-Estimate-Function\((G,F,fU(f(v_i)=w_i))\)

\( l:=g^*+K+h \); \{Heuristic function of \( l^* \)\}

If \( l<d_j \) then \{partial cost < best distance\}

If \( I<n \) then \{some vertex still not matched\}

TreeSearch\((G,F,fU(f(v_i)=w_j),g^*+K,v_i,d_j,f_{opt})\)

Else \{all AG vertices have been matched\}

\( d_i:=l_j \); \( f_{opt}:=fU(f(v_i)=w_j) \)

**End-procedure**

6 Recognition of Real-Life Objects on Images

We present a real application to recognise coloured objects using 2D images. Images were extracted from the database COIL-100 from Columbia University (www.cs.columbia.edu/CAVE/research/softlib/coil-100.html). It is composed by 100 isolated objects and for each object there are 72 views (one view each 5 degrees). Figure 1 shows some objects at angle 100 and their segmented images with the adjacency graphs. These graphs have from 6 to 18 vertices and the average number is 10. The test set was composed by 36 views per object (taken at the angles 0, 10, 20 and so on), whereas the reference set was composed by the 36 remaining views (taken at the angles 5, 15, 25 and so on). We compared SORGs to 3 other classifiers. The probabilistic models First-Order Random Graphs (FORGs) [6], Function-Described Graphs (FDGs) [4] and the Nearest-Neighbour classifier (AG-AG) with the edit-operation distance between graphs as measure of similarity.

We made 6 different experiments in which the number of clusters that represents each 3D-object varied. If the 3D-object was represented by only one cluster, the 36 AGs from the reference set that represent the 3D-object were used to synthesise the SORGs, FORGs or FDGs. If it was represented by 2 clusters, the 18 first and consecutive AGs from the reference set were used to synthesise one of the SORGs, FORGs or FDGs and the other 18 AGs were used to synthesise the other ones. A

![Fig. 1. Some objects at angle 100 and the segmented images with the AGs](image-url)
similar method was used for the other experiments with 3, 4, 6 and 9 clusters per 3D-object.

![Ratio of classification](image1.png) ![Run time per comparison](image2.png)

**Fig. 2.** (a) Ratio of recognition correctness (b) run time spent in the classification. SORG: ---; FDG: ---; FORG: ---; AG-AG: ---

Figure 2.a shows the ratio of correctness of the four classifiers varying the number of clusters per each object. When objects are represented by only 1 or 2 clusters, there are too much spurious regions (produced in the segmentation process) to keep the structural and semantic knowledge of the object. For this reason, different regions or faces (or vertices in the AGs) of different views (that is, AGs) are considered to be the same face (or vertex in the AGs). The best result appears when each object is represented by 3 or 4 clusters, that is, each cluster represents 90 degrees of the 3D-object. When objects are represented by 9 clusters, each cluster represents 40 degrees of the 3D-object and 4 AGs per cluster, there is poor probabilistic knowledge and therefore the distance costs on the vertices and arcs are coarse. Figure 2.b shows the average run time spent to compute the classification. When the number of clusters per object decreases, the number of total comparisons also decreases but the time spent to compute the distance increases since the structures that represent the clusters (SORGs, FORGs or FDGs) are bigger.

When the best classification is reached, FDGs have less run-time than SORGs but lower recognition ratio. This is due to the fact that the algorithm to compute the distance in the FDG classifier prunes harder the search tree than the SORGs since it uses a qualitative information of the 2\textsuperscript{nd}-order relation [3]. Therefore, the time spent to search the best labelling decreases but the optimal one may not be found.

### 7 Conclusions and Future Work

We have presented an algorithm to compute the distance measure between AGs and SORGs. It is based on a well known algorithm that uses the branch-and-bound technique and the distance between vertices and arcs as the *heuristic function* to prune the search tree. Due to the fact that SORGs keep 2\textsuperscript{nd}-order probabilities between vertices, we incorporate this knowledge into the *heuristic function* to reduce harder the space explored by the algorithm.

The experimental results show that, in the FDG classifier, the use of the antagonism relations between vertices, is useful not only to decrease the run time of the matching algorithm but also to increase the recognition ratio (thanks to a better modelling of the set of objects). Nevertheless, if the 2\textsuperscript{nd}-order probabilities are kept in the
model as in the case of the SORGs, the classification ratio increases but the run time also increases.

We are defining a matching algorithm that computes a sub-optimal distance between AGs and SORGs in polynomial cost. It is based on the distance between cliques that uses 2nd-order probabilities between the external vertices of both cliques. This distance will be useful to select some SORG candidates. Thus, the distance algorithm explained in this article with exponential cost will be only applied to this few candidates.

References