Second-order random graph for modeling sets of attributed graphs and their application to object learning and recognition

Alberto Sanfeliu[1], Francesc Serratosa[2] and René Alquézar[3]

[1] Universitat Politècnica de Catalunya, Inst. de Robòtica i Informàtica Ind., Spain
sanfeliu@iri.upc.es
[2] Universitat Rovira i Virgili, Dept. d'Enginyeria Informàtica i Matemàtiques, Spain
Francesc.Serratosa@etse.urv.es
[3] Universitat Politècnica de Catalunya, Dept. de Llenguatges i Sist. Informàtics, Spain
alquezar@lsi.upc.es

Abstract. The aim of this article is to present a random graph representation, that is based on 2nd order relations between graph elements, for modeling sets of attributed graphs (AGs). We refer to these models as *second-order random graphs* (SORGs). The basic feature of SORGs is that they include both marginal probability functions of graph elements and 2nd-order joint probability functions. This allows a more precise description of both the structural and semantic information contents in a set of AGs and, consequently, an expected improvement in graph matching and object recognition. The article presents a probabilistic formulation of SORGs that includes as particular cases the two previously proposed approaches based on random graphs, namely the first-order random graphs (FORGs) and the function-described graphs (FDGs). We then propose a distance measure derived from the probability of instantiating a SORG into an AG and an incremental procedure to synthesize SORGs from sequences of AGs. Finally, SORGs are shown to improve the performance of FORGs, FDGs and direct AG-to-AG matching in three experimental recognition tasks: one

in which AGs are randomly generated and the other two in which AGs represent multiple views of 3D objects (either synthetic or real) that have been extracted from color images. In the last case, object learning is achieved through the synthesis of SORG models.

1 Introduction

Attributed Graphs (AGs) has been used to solve computer vision problems for decades and in many applications. Some examples include recognition of graphical symbols [13], character recognition [18], shape analysis [5,17], 3D-object recognition [29,25] and video and image database indexing [27]. In these applications, AGs represent both unclassified objects (unknown input patterns) and prototypes. Moreover, these AGs are typically used in the context of nearest-neighbour classification. That is, an unknown input pattern is compared with a number of prototypes stored in the database. The unknown input is then assigned to the same class as the most similar prototype. A number of similarity measures on AGs and related computational procedures have been proposed in this context [3,4,7,10,14,15,20,28].

Nevertheless, the main drawback of representing the data and prototypes by AGs is the computational complexity of comparing two AGs. The time required by any of the optimal algorithms may in the worst case become exponential in the size of the AGs. The approximate algorithms, on the other hand, have only polynomial time complexity, but do not guarantee to

find the optimal solution [2,23]. For some applications, this may not be acceptable. Moreover, in some applications, the classes of objects are represented explicitly by a set of prototypes which means that a huge amount of model AGs must be matched with the input AG and so the conventional error-tolerant graph matching algorithms must be applied to each model-input pair sequentially. As a consequence, the total computational cost is linearly dependent on the size of the database of model graphs and exponential (or polynomial in subgraph methods) with the size of the AGs. For applications dealing with large databases, this may be prohibitive.

To alleviate these problems, some attempts have been made to try to reduce the computational time of matching the unknown input patterns to the whole set of models from the database. Assuming that the AGs that represent a cluster or class are not completely dissimilar in the database, only one structural model is defined from the AGs that represent the cluster, and thus, only one comparison is needed for each cluster.

We distinguish two different methodologies depending on whether they keep probabilistic information in the structure that represent the cluster of AGs (a) or not (b).

(a) In the first methods, the models, which are usually called Random Graph (RG), are described in the most general case through a joint probability space of random variables ranging over graph vertices and arcs. They are

the union of the AGs in the cluster, according to some synthesis process, together with its associated probability distribution. In this manner, a structural pattern can be explicitly represented in the form of an AG and an ensemble of such representations can be considered as a set of outcomes of the RG. In this paper, we briefly recall the two most important probabilistic methods, which are First-Order Random Graphs (FORGs) [30] and Function-Described Graphs (FDGs) [25,26]. The approach presented in the paper by Sengupta *et al.* [21] can be regarded as similar to the FORG approach. Finally, we introduce the Second-Order Random Graphs (SORGs), which can be seen as a generalisation of both of them [24].

(b) In the non-probabilistic methods, we comment four different approximations. The self-organizing map (SOM) is a useful method to cluster sets of objects. It consists of a layer of units (neurons), that adapt themselves to a population of input patterns. SOM was first presented by Kohonen with the limitation that patterns had to be represented in terms of feature vectors only. Afterwards, the same authors presented an extension of this method to strings [10] and then Günter & Bunke presented in [9] a generalisation of the clustering method applied to AGs. Moreover, Seong *et al.* [22], Cordella *et al.* [6] and Jiang *et al.* [12] presented three different methods to cluster sets of AGs. In the first one, a hierarchical model that summarises and organises the input instances incrementally is built up with a succession of AGs. In the second one, the set of AGs is represented by the

maximally general prototype that can be seen as the union of the AGs. And in the third one, the set of AGs is represented by the generalised median of the AGs that belong to the set.

In the following section, we introduce the formal definitions used throughout the paper. In section 3, we recall FORGs and FDGs, which are the two main approximations of the general RG concept proposed in the literature. In section 4, we present SORGs as a quite general formulation for estimating the joint probability of the random elements in a RG synthesised from a set of AGs. In sections 5 and 6, we show respectively that the FORG and FDG approaches can be seen as different simplifications of SORGs. In sections 7 and 8, we propose a distance measure between AGs and SORGs and a method to synthesise SORGs, respectively. Finally, we present a comparative study between SORGs and other probabilistic models presented in the literature. They are applied on AGs randomly generated and on 3D-object recognition. In the last section we provide some discussion about our contribution.

2 Formal Definitions of Random-Graph Representation

Definition 1: Attributed Graph (AG). Let Δ_v and Δ_e denote the domains of possible values for attributed vertices and arcs, respectively. These domains are assumed to include a special value Φ that represents a *null value* of a vertex or arc. An AG G over (Δ_v,Δ_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 $\gamma_v:\Sigma_v\to\Delta_v$ and $\gamma_e:\Sigma_e\to\Delta_e$ assign attribute values to vertices and arcs, respectively.

Definition 2: Random Graph (RG). Let Ω_v and Ω_e be two sets of random variables with values in Δ_v (random vertices) and in Δ_e (random arcs), respectively. A RG R over (Δ_v,Δ_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, $\Sigma_e=\{\varepsilon_{ij} \mid i,j\in\{1,...,n\}, i\neq j\}$ is a set of arcs, the mapping $\gamma_v:\Sigma_v\to\Omega_v$ associates each vertex $\omega_k\in\Sigma_v$ with a random variable $\alpha_k=\gamma_v(\omega_k)$ with values in Δ_v, and $\gamma_e:\Sigma_e\to\Omega_e$ associates each arc $\varepsilon_{ij}\in\Sigma_e$ with a random variable $\beta_k=\gamma_e(\varepsilon_{ij})$ with values in Δ_e. And, finally, P is a joint probability distribution $P(\alpha_1,\ldots,\alpha_n,\beta_1,\ldots,\beta_m)$ of all the random vertices $\{\alpha_i \mid \alpha_i=\gamma_\omega(\omega_i), 1\le i\le n\}$ and random arcs $\{\beta_j \mid \beta_j=\gamma_\varepsilon(\varepsilon_{kl}), 1\le j\le m\}$.

Definition 3: Outcome graph. An Outcome graph is any AG obtained by instantiating all random vertices and random arcs of a RG in a way that satisfies all the structural relations. Such instantiation is associated with a structural isomorphism $\mu:G'\to R$, where G' is the extension of G to the order of R (in which null-attribute vertices and arcs have been added to form a complete AG [26]). Hence, a RG represents the set of all possible AGs that can be outcome graphs of it, according to an associated probability distribution.

Definition 4: Probability of an outcome graph. For each outcome graph *G* of a RG *R*, the joint probability of random vertices and arcs is defined over an instantiation that produces *G*. Let *G* be oriented with respect to *R* by the structurally coherent isomorphism μ; for each vertex ω_i in *R*, let $\mathbf{a}_i = \gamma_v\left(\mu^{-1}(\omega_i)\right)$ be the corresponding attribute value in *G'*, and similarly, for each arc ε_{kl} in *R* (associated with random variable β_j) let $\mathbf{b}_j = \gamma_e\left(\mu^{-1}(\varepsilon_{kl})\right)$ be the corresponding attribute value in *G'*. Then the *probability of G according to* (or given by) *the orientation* μ, denoted by $P(G|\mu)$, is defined as

$$P(G|\mu) = \Pr\left(\wedge_{i=1}^{n}\left(\alpha_i = \mathbf{a}_i\right) \wedge \wedge_{j=1}^{m}\left(\beta_j = \mathbf{b}_j\right)\right) = p(\mathbf{a}_1,\ldots,\mathbf{a}_n,\mathbf{b}_1,\ldots,\mathbf{b}_m) \quad (1)$$

3 Approximating Probability Distributions in the Literature

If we want to represent the cluster of AGs by a probability distribution it is impractical to consider the high order probability distribution $P(\alpha_1,\ldots,\alpha_n,\beta_1,\ldots,\beta_m)$ where all components and their relations in the structural patterns are taken jointly (eq. 1). For this reason, some other more practical approaches have been presented that propose different approximations [25,26,30]. 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. In the

following sections, we summarise the two main such approaches, FORGs and FDGs.

3.1 First-Order Random Graphs (FORGs)

Wong and You [30] proposed the First-Order Random Graphs (FORGs), in which strong simplifications are made so that RGs can be used in practice. They introduced three suppositions about the probabilistic independence between vertices and arcs:

1) The random vertices are mutually independent;

2) The random arcs are independent given values for the random vertices;

3) The arcs are independent of the vertices except for the vertices that they connect.

Definition 5: First-Order Random Graphs (FORGs). A FORG *R* is a RG that satisfies the assumptions 1, 2, 3 shown above.

Based on these assumptions, for a FORG *R*, the probability $P(G|\mu)$ becomes

$$P(G|\mu) = \prod_{i=1}^{n} p_i(\mathbf{a}_i) \prod_{j=1}^{m} q_j(\mathbf{b}_j \,|\, \mathbf{a}_{j1}, \mathbf{a}_{j2}) \qquad (2)$$

where $p_i(\mathbf{a}) \hat{=} \Pr(\alpha_i = \mathbf{a}),\ 1 \le i \le n,$ are the marginal probability density functions for vertices and $q_j(\mathbf{b} \,|\, \mathbf{a}_{j1}, \mathbf{a}_{j2}) \hat{=} \Pr(\beta_j = \mathbf{b} \,|\, \alpha_{j1} = \mathbf{a}_{j1}, \alpha_{j2} = \mathbf{a}_{j2}),\ \ 1 \le j \le m,$ are the conditional probability functions for the arcs, where α_{j1}, α_{j2} refer to the random vertices for the endpoints of the random arc β_j.

The storage space of FORGs is $O(nN + mMN^2)$ where N and M are the number of elements of the domains Δ_v and Δ_e.

3.2 Function-Described Graphs (FDGs)

Serratosa *et al*. [1,23,25,26] proposed the Function-Described Graphs (FDGs), which lead to another approximation of the joint probability P of the random elements. On one hand, some independence assumptions (a) are considered, but on the other hand, some useful 2^{nd}-order functions (b) are included to constrain the generalisation of the structure.

(a) Independence assumptions in the FDGs

1) The attributes in the vertices are independent of the other vertices and of the arcs.

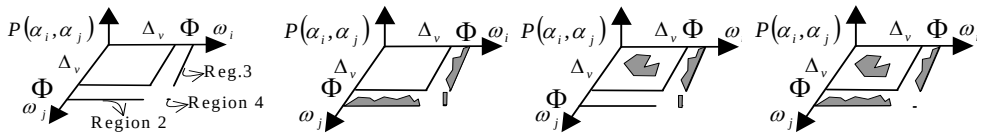
2) The attributes in the arcs are independent of the other arcs and also of the vertices. However, it is mandatory that all non-null arcs be linked to a non-null vertex at each extreme in every AG covered by an FDG. In other words, any outcome AG of the FDG has to be structurally consistent [26].

(b) 2^{nd}-order functions in the FDGs

In order to tackle the problem of the over-generalisation of the sample, the *antagonism*, *occurrence* and *existence* relations are introduced in FDGs, which apply to pairs of vertices or arcs. In this way, random vertices and arcs are not assumed to be mutually independent, at least with regards to the structural information, since the above relations represent a *qualitative*

information of the 2nd-order joint probability functions of a pair of vertices or arcs.

To illustrate the meaning of the FDG 2nd-order relations it is convenient to split the domain of the joint probabilities in four regions (see Figure 1.a).



(a) The four regions (b) Antagonism (c) Occurrence (d) Existence

Figure 1. 2nd-order probability of two FDG vertices

The first one is composed by the points that belong to the Cartesian product of the domains of actual attributes of the two vertices, Δ_v, corresponding to the cases where both elements are not null. The second and third regions are one-dimensional (shown as straight lines) in which only one of the vertices has the null value. Finally, the fourth region is the single point where both vertices are null. The 2nd-order relations are defined as follows:

Antagonism relations: w_i and w_j are *antagonistic* if the probabilities in the first region are all zero (figure 1.b), $A_\omega(\omega_i,\omega_j)=1\Leftrightarrow \Pr(\alpha_i\neq\Phi\wedge\alpha_j\neq\Phi)=0$. In the 3D-object modelling case, two faces are antagonistic if it is not possible to see both in a same view.

Occurrence relations: There is an *occurrence relation* if the joint probability function equals zero in the second region (figure 1.c), $O_\omega(\omega_i,\omega_j)=1\Leftrightarrow \Pr(\alpha_i\neq\Phi\wedge\alpha_j=\Phi)=0$. The case of the third region is

analogous to the second one with the only difference of swapping the elements. In the 3D-object modelling case, a face is "occurrent" with respect to another if always that the former is visible, the latter is visible too.

Existence relations: Finally, there is an *existence* relation between two vertices if the joint probability function equals zero in the fourth region (figure 1.d), $E_{\omega}(\omega_i,\omega_j)=1 \Leftrightarrow \Pr(\alpha_i=\Phi \wedge \alpha_j=\Phi)=0$. In the 3D-object modelling case, there is an existence relation between two faces if one of them or both appear in all the views used to synthesise the model of the object.

Definition 6: Function-Described Graphs (FDGs). An FDG *F* is a RG that satisfies the assumptions 1 and 2 shown above and contains the information of the 2nd-order relations of *antagonism*, *occurrence* and *existence* between pairs of vertices or arcs.

Based on these assumptions, for an FDG *F*, $P(G|\mu)$ becomes

$$P(G|\mu)=\prod_{i=1}^{n} p_i(\mathbf{a}_i)\prod_{j=1}^{m} q_j(\mathbf{b}_j) \qquad (3)$$

where $p_i(\mathbf{a})$ is defined as in FORGs and $q_j(\mathbf{b}) \triangleq \Pr(\beta_j=\mathbf{b}\,|\,\alpha_{j1}\neq\Phi,\alpha_{j2}\neq\Phi)$.

However, the isomorphism μ not only has to be structurally coherent but also has to fulfil the 2nd-order constraints (antagonism, exitence and occurrence) [25,26]. Otherwise, $P(G|\mu)$ is considered to be zero. The basic idea of these constraints is the satisfaction by an AG to be matched of the antagonism, occurrence and existence relations inferred from the set of AGs used to synthesise the FDG.

The storage space of FDGs is $O\left(nN+mM+n^2+m^2\right)$ where N and M are the number of elements of the domains Δ_v and Δ_e, respectively.

4 Second-order Random-Graph Representation

We show next that the joint probability of an instantiation of the random elements in a RG can be approximated as follows:

$$P\left(G\middle|\mu\right)=p\left(\mathbf{d}_1,,,\mathbf{d}_s\right)\approx\prod_{i=1}^{s}p_i\left(\mathbf{d}_i\right)\prod_{i=1}^{s-1}\prod_{j=i+1}^{s}r_{ij}\left(\mathbf{d}_i,\mathbf{d}_j\right) \qquad (4)$$

where $p_i(\mathbf{d}_i)$ are the marginal probabilities of the random elements γ_i, (vertices or arcs) and r_{ij} are the Peleg compatibility coefficients [16] that take into account both the marginal and 2nd-order joint probabilities,

$$r_{ij}\left(\mathbf{d}_i,\mathbf{d}_j\right)=\frac{\Pr\left(\gamma_i=\mathbf{d}_i\wedge\gamma_j=\mathbf{d}_j\right)}{p_i\left(\mathbf{d}_i\right)p_j\left(\mathbf{d}_j\right)} \qquad (5)$$

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 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.

Eq. (4) is obtained by assuming independence in the conditional probabilities (section 4.1) and rearranging the joint probability expression using Bayes rule (section 4.2)

4.1 Conditional Probabilities

The conditional density probability $p(\gamma_i/(\gamma_{i+1},...,\gamma_s))$ of a random element γ_i is used to compute the joint density probability $p(\gamma_1,...,\gamma_s)$. Applying the Bayes rule to the conditional probability, the following expression holds,

$$p(\gamma_i/(\gamma_{i+1},...,\gamma_s)) = \frac{p(\gamma_i)\cdot p((\gamma_{i+1},...,\gamma_s)/\gamma_i)}{p(\gamma_{i+1},...,\gamma_s)} \qquad (6)$$

Due to the fact that this *(s+1-i)*-order probability can not be stored in practice, we have to suppose at this point that the conditioning random variables γ_{i+1} to γ_s are independent to each other. In that case, an estimate is given by

$$p(\gamma_i/(\gamma_{i+1},...,\gamma_s)) = p(\gamma_i)\cdot\prod_{j=i+1}^{s}\frac{p(\gamma_j/\gamma_i)}{p(\gamma_j)} = p(\gamma_i)\cdot\prod_{j=i+1}^{s}\frac{p(\gamma_j,\gamma_i)}{p(\gamma_j)\cdot p(\gamma_i)} \qquad (7)$$

Thus, if we use the Peleg compatibility coefficients then the conditional probability is,

$$prob(\gamma_i=\mathbf{d}_i/(\gamma_{i+1}=\mathbf{d}_{i+1},...,\gamma_s=\mathbf{d}_s)) = p_i(\mathbf{d}_i)\cdot\prod_{j=i+1}^{s} r_{ij}(\mathbf{d}_i,\mathbf{d}_j) \qquad (8)$$

4.2 Joint Probability

Using the Bayes theorem, the joint probability density function $p(\gamma_1,,,\gamma_s)$ can be split into the product of another joint probability function and a conditional one,

$$p(\gamma_1,,,\gamma_s) = p(\gamma_2,,,\gamma_s)\cdot p(\gamma_1/(\gamma_2,,,\gamma_s)) \qquad (9)$$

and applying *n*-1 times the same theorem on the remaining joint probability,

$$p(\gamma_1,,,\gamma_s) = p(\gamma_s)\cdot\prod_{i=1}^{s-1} p(\gamma_i /(\gamma_{i+1},,,\gamma_s)) \qquad (10)$$

If we use eq. (8) to estimate the conditional probabilities, then the joint probability $p(\mathbf{d}_1,,,\mathbf{d}_s)$ can be estimated as $p^*(\mathbf{d}_1,,,\mathbf{d}_s)$ where,

$$P(G|\mu) \approx p^*(\mathbf{d}_1,,,\mathbf{d}_s) = p_s(\mathbf{d}_s)\cdot\prod_{i=1}^{s-1}\left[p_i(\mathbf{d}_i)\cdot\prod_{j=i+1}^{s} r_{ij}(\mathbf{d}_i,\mathbf{d}_j)\right] \qquad (11)$$

and introducing the first factor into the product, we have

$$P(G|\mu) \approx p^*(\mathbf{d}_1,,,\mathbf{d}_s) = \prod_{i=1}^{s} p_i(\mathbf{d}_i)\prod_{i=1}^{s-1}\prod_{j=i+1}^{s} r_{ij}(\mathbf{d}_i,\mathbf{d}_j) \qquad (12)$$

In the approximations of the joint probability in the FDG and FORG approaches, random vertices and random arcs are treated separately, for this reason the above expression can be split considering vertices and arcs separately as follows

$$P(G|\mu) \approx p^*(\mathbf{a}_1,,\mathbf{a}_n,\mathbf{b}_1,,\mathbf{b}_m) = \prod_{i=1}^{n} p_i(\mathbf{a}_i)\prod_{i=1}^{m} p_i(\mathbf{b}_i)\prod_{i=1}^{n-1}\prod_{j=i+1}^{n} r_{ij}(\mathbf{a}_i,\mathbf{a}_j)\prod_{i=1}^{n}\prod_{j=1}^{m} r_{ij}(\mathbf{a}_i,\mathbf{b}_j)\prod_{i=1}^{m-1}\prod_{j=i+1}^{m} r_{ij}(\mathbf{b}_i,\mathbf{b}_j) \qquad (13)$$

5 Approximation of the joint probability by FORGs

In the FORG approach, the Peleg coefficients between vertices and between arcs do not influence on the computation of the joint probability. That is, by assumption 1 and 2 (section 3.1), $r_{ij}(\mathbf{a}_i,\mathbf{a}_j)=1$ and $r_{ij}(\mathbf{b}_i,\mathbf{b}_j)=1$ for all the vertices and arcs, respectively. On the contrary, assumption 3 (sec 3.1) makes that the probability on the arcs be conditioned on the values of the vertices that the arc connects, $q_j(\mathbf{b}_j \,|\, \mathbf{a}_{j1},\mathbf{a}_{j2})$. In a similar deduction to that of

section 4.3, and considering assumption 1, we arrive at the equivalence: $q_j(\mathbf{b}_j \mid \mathbf{a}_{j1}, \mathbf{a}_{j2}) = p_j(\mathbf{b}_j) r_{j_1 j}(\mathbf{a}_{j_1}, \mathbf{b}_j) r_{j_2 j}(\mathbf{a}_{j_2}, \mathbf{b}_j)$. Thus,

$$P(G|\mu) = \prod_{i=1}^{n} p_i(\mathbf{a}_i) \prod_{j=1}^{m} p_j(\mathbf{b}_j) \prod_{j=1}^{m} \prod_{i=j_1, j_2} r_{ij}(\mathbf{a}_i, \mathbf{b}_j) \qquad (14)$$

6 Approximation of the joint probability by FDGs

In the FDG approach, the 2nd-order probabilities between vertices can be estimated from the marginal probabilities and the 2nd-order relations as follows (a similar expression is obtained for the arcs, see [25]),

$$\begin{aligned} &\Pr(\alpha_i = \mathbf{a}_i \wedge \alpha_j = \mathbf{a}_j) = 0 \; if \; Condition \; Q_{2nd} \\ &\Pr(\alpha_i = \mathbf{a}_i \wedge \alpha_j = \mathbf{a}_j) \approx p_i(\mathbf{a}_i) p_j(\mathbf{a}_j) \; otherwise \end{aligned} \qquad (15)$$

where the Condition Q_{2nd} is

$$Q_{2nd} : \begin{cases} (A_\omega(\omega_i, \omega_j) \wedge \mathbf{a}_i \neq \Phi \wedge \mathbf{a}_j \neq \Phi) \vee (O_\omega(\omega_i, \omega_j) \wedge \mathbf{a}_i \neq \Phi \wedge \mathbf{a}_j = \Phi) \vee \\ (O_\omega(\omega_j, \omega_i) \wedge \mathbf{a}_i = \Phi \wedge \mathbf{a}_j \neq \Phi) \vee (E_\omega(\omega_i, \omega_j) \wedge \mathbf{a}_i = \Phi \wedge \mathbf{a}_j = \Phi) \end{cases} \qquad (16)$$

Note that, in the first case, it can be assured that the joint probability is null, but in the second case, we assume that the random elements are independent and the probability is estimated as a product of the marginal ones.

Thus, the Peleg coefficients are simplified as r_{ij}', using eq. (15),

$$r_{ij}'(\mathbf{a}_i, \mathbf{a}_j) = \begin{cases} 0 \; if \; Q_{2nd} \wedge p_i(\mathbf{a}_i) \neq 0 \wedge p_j(\mathbf{a}_j) \neq 0 \\ 1 \; otherwise \end{cases} \qquad (17)$$

Moreover, due to the independence assumption 2 (sec 3.2), it is not possible to have a non-null arc and a null vertex as one of its endpoints in an outcome graph. Thus, we have $p(\alpha_{j1} = \Phi \wedge \beta_j \neq \Phi) = 0$ and $p(\alpha_{j2} = \Phi \wedge \beta_j \neq \Phi) = 0$. In the other cases, by assumption 1, they are assumed to be independent and

so computed as the product of the marginal ones. The Peleg coefficients between vertices and arcs are simplified as

$$r_{ij}''(\mathbf{a}_i,\mathbf{b}_j)=\begin{cases}0 \; if \; (i=j_1 \vee j_2)\wedge \mathbf{a}_i=\Phi \wedge \mathbf{b}_j \neq \Phi \\ 1 \; otherwise\end{cases} \qquad (18)$$

The final expression of the joint probability of an outcome AG with respect to an FDG is

$$P(G|\mu)=\prod_{i=1}^{n} p_i(\mathbf{a}_i)\prod_{j=1}^{m} p_j(\mathbf{b}_j)\prod_{i=1}^{n}\prod_{j=1}^{n} r_{ij}'(\mathbf{a}_i,\mathbf{a}_j)\prod_{i=1}^{m}\prod_{j=1}^{m} r_{ij}'(\mathbf{b}_i,\mathbf{b}_j)\prod_{j=1}^{m}\prod_{i=j_1,j_2} r_{ij}''(\mathbf{a}_i,\mathbf{b}_j) \qquad (19)$$

7 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 [1]. It is related to the probability of G according to the labelling function $\mu: G \to S$, denoted $P(G|\mu)$ in eq. (4). We may attempt to minimise a *global cost* measure *C* of the morphism μ 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, $C = f(P(G|\mu))$. For instance, $C = -\ln(P(G|\mu))$ would be a possible choice. Thus, considering eq. (4),

$$C(G|\mu)=-\ln\left(\prod_{i=1}^{s} p_i(\mathbf{d}_i)\prod_{i=1}^{s-1}\prod_{j=i+1}^{s} r_{ij}(\mathbf{d}_i,\mathbf{d}_j)\right) \qquad (20)$$

However, only that one graph element had a probability of zero, the joint probability would be zero and *C* would be infinite. Since this may happen

due to the noisy presence of an unexpected element or the absence of a model's element, only that one graph element were not properly mapped, the involved graphs would be wrongly considered to be completely different. We must therefore admit the possibility of both extraneous and missing elements in the data graphs, since the data extracted from the information sources (e.g. images) will usually be noisy, incomplete or uncertain. As a consequence, the matches for which $P(G|\mu)=0$ should not be discarded since they could be the result of a noisy feature extraction and graph formation. In addition, a model (SORG) should match to a certain degree not only the objects (AGs) in its learning set but also the ones that are "near".

Hence, it is more appropriate for practical purposes to decompose the global cost C into the sum of some *bounded* individual costs, one for each of the graph element matches (first-order costs) and one for each Peleg compatibility coefficient or pair of element matches (second-order costs)

$$C(G|\mu)=\sum_{i=1}^{s}C_p\left(p_i(\mathbf{d}_i)\right)+\sum_{i=1}^{s-1}\sum_{j=i+1}^{s}C_r\left(r_{i,j}(\mathbf{d}_i,\mathbf{d}_j)\right) \tag{21}$$

where first- and second-order costs are given respectively by

$$C_p\left(p_i(\mathbf{d}_i)\right)=Cost\left(p_i(\mathbf{d}_i)\right) \tag{22}$$

$$C_r\left(r_{i,j}(\mathbf{d}_i,\mathbf{d}_j)\right)=\begin{cases}0 \quad \text{if } p_i(\mathbf{d}_i)<K_{\mathrm{Pr}} \vee p_j(\mathbf{d}_j)<K_{\mathrm{Pr}} \\ Cost(p_{i,j}(\mathbf{d}_i,\mathbf{d}_j))-Cost(p_i(\mathbf{d}_i))-Cost(p_j(\mathbf{d}_j)) \quad otherwise\end{cases} \tag{23}$$

and the function $Cost(\Pr)$ yields a bounded normalised 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,

$$Cost(\Pr) = \begin{cases} \dfrac{-\ln(\Pr)}{-\ln(K_{\Pr})} & \text{if } \Pr \geq K_{\Pr} \\ \\ 1 & \text{otherwise} \end{cases} \qquad (24)$$

In the first case of equation (23), both the joint probability and at least one of the marginal probabilities are practically zero, and as commented before, the indecisiveness *0/0* is solved as 1 for the Peleg coefficient, yielding a null second-order cost, since ln(1)=0. Note that the global cost given by equation (21) is not an edit operation cost. Moreover, second-order costs may be positive or negative, thus correcting (if necessary) the sum of first-order costs and using, to this end, the information stored in the second-order joint probability functions.

Once a cost measure *C* is defined, a distance measure between an AG and a SORG and the optimal labelling μ^* are defined respectively as

$$d = \min_{\mu \in H} \left\{ C\left(G|\mu\right) \right\} \qquad \text{and} \qquad \mu^* = \arg\min_{\mu \in H} \left\{ C\left(G|\mu\right) \right\} \qquad (25)$$

The algorithm we use to calculate *d* and μ^* is a classical recursive tree search procedure, where the search space is reduced by a *branch and bound* technique (not described here due to lack of space).

8 Synthesis of Second-Order Random Graphs

Below, we present the *Incremental-synthesis-of-SORGs* method (**Algorithm 1**) to synthesise an SORG from a sequence of AGs. The algorithm uses two procedures: *SORG-synthesis-from-labelled-AGs,* to transform an AG into an equivalent SORG, and *SORG-synthesis-from-labelled-SORGs* to build a SORG from two SORGs with a given labelling. The synthesis method is parameterised by a matching algorithm $M(G,F)$ that is supposed to return an optimal (or a "good" suboptimal) labelling between an AG *G* and an SORG *F*, according to an appropriate distance measure. In practice, we use as algorithm $M(G,F)$ the *branch-and-bound* method aforementioned that calculates the distance measure described in the previous section.

Algorithm 1: Incremental-synthesis-of-SORGs

Inputs: A sequence of AGs $G_1,...G_m$, $m \geq 1$, over a common domain.

A matching algorithm $M(G,F)$ between an AG and an SORG that finds an optimal or sub-optimal labelling

Output: An SORG F that represents the given set of AGs.

Begin

F := SORG-synthesis-from-labelled-AGs(G_1) { build the first SORG from G_1 }

for $i := 2$ **to** m **do**

let $d : G_i, F \to \Re$ and $\mu : G_i \to F$ be the distance and labelling found by $M(G_i, F)$

(G'_i, F', μ'):=Extend-labelling-AG-SORG (G_i, F, μ) { It extends the AG and the SORG with null elements to make them structurally isomorphic and also extends the given labelling accordingly }

H := SORG-synthesis-from-labelled-AGs(G') { build an auxiliary SORG H from G'}

let $\gamma : G' \rightarrow H$ be a bijective mapping used in the previous synthesis

let $\varphi : H \rightarrow F'$ be the bijective mapping determined by the

composition $\mu' = \varphi \circ \gamma$

F := SORG-synthesis-from-labelled-SORGs$\left(H, F', \varphi\right)$ { build F using synthesis from 2 SORGs }

endfor

end-algorithm

The algorithm presented above is similar to the one described in [26] for the incremental synthesis of FDGs. The only difference with the case of synthesising FDGs is that, instead of inferring the FDG second-order constraints, second-order joint probability density functions must be estimated now. To this end, it is enough to modify as follows the procedures that carry out the synthesis of a new model (now an SORG) from a set of AGs (*SORG-synthesis-from-labelled-AGs*) or from a set of previous models (*SORG-synthesis-from-labelled-SORGs*) when a common labelling scheme is given [26].

Let $D = \left\{G^g \mid 1 \le g \le z\right\}$ be a set of AGs defined over a common attribute domain. Assuming that a common labelling between all the AGs in *D* is given, let v_i^g be the node labelled *i* in the AG G'^g (the extension of G^g to a minimum common order).

The second-order joint probability density functions of pairs of vertices $P_{\omega\omega} = \{p_{ij}(\mathbf{a_1}, \mathbf{a_2}), 1 \le i \le n, 1 \le j \le n, i \ne j\}$ can be estimated in the maximum likelihood sense using frequencies of attributes and null values in D as

$$p_{ij}(\mathbf{a_1}, \mathbf{a_2}) = \Pr(\alpha_i = \mathbf{a_1} \wedge \alpha_j = \mathbf{a_2}) = \frac{\#g:\ 1 \le g \le z:\ \gamma_v^g(v_i^g) = \mathbf{a_1} \wedge \gamma_v^g(v_j^g) = \mathbf{a_2}}{z} \quad (26)$$

If the SORG synthesis is from a set of previous SORGs $\{F^k \mid 1 \le k \le h\}$ with a given common labelling, let t^k be a weight for each F^k given by

$$t^k = z^k \Big/ \sum_{g=1}^{h} z^g, \quad 1 \le k \le h, \quad (27)$$

where z^k is the (stored) number of AGs that was used to synthesise the SORG F^k. Then $P_{\omega\omega} = \{p_{ij}(\mathbf{a_1}, \mathbf{a_2}), 1 \le i \le n, 1 \le j \le n, i \ne j\}$ can be estimated from the corresponding probability density functions in the previous SORGs as

$$p_{ij}(\mathbf{a_1}, \mathbf{a_2}) = \Pr(\alpha_i = \mathbf{a_1} \wedge \alpha_j = \mathbf{a_2}) = \sum_{k=1}^{h} t^k * p_{ij}^k(\mathbf{a_1}, \mathbf{a_2}) \quad (28)$$

The joint probability functions of pairs of vertices and edges $P_{\omega\varepsilon} = \{\rho_{ij}(\mathbf{a}, \mathbf{b}), 1 \le i \le n, 1 \le j \le m\}$ and the joint probability functions of pairs of edges $P_{\varepsilon\varepsilon} = \{q_{ij}(\mathbf{b_1}, \mathbf{b_2}), 1 \le i \le m, 1 \le j \le m, i \ne j\}$ can be estimated in both cases (set of AGs or set of SORGs) similarly.

9 Results

We carried out three different types of experiments to assess the usefulness of our new representation and to compare it with some other representations presented in the literature. In the first experiments, the AGs were synthetically generated varying some parameters such as the number of vertices or the distance between the AGs in their clusters. In the second

experiments, we used 3D-objects artificially created by a CAD program. In the last experiments, we used a real application in which AGs represent coloured 3D objects. They were extracted and recognised from some 2D images. We present these two applications on 3D-objects due to the fact that in the first one, the 3D objects and the images are less complex and there is no segmentation process that distorts the obtained AGs and the run time needed to compute the classification. Thus, the first experiments are useful to study our representation from the theoretical point of view, the second ones are useful to apply our methods on a 3D-object non-noisy representation and the third ones are useful to apply the representation on noisy, real and complex images.

We present the experiments in the following three sections. In each experiment, we compare SORGs with three other methods: FDGs, FORGs and AG-to-AG matching. First, we show some information of the AGs and the structures obtained in the synthesis process and then we show the run time and ratio of correctness of the classification processes for each method. SORGs, FDGs and FORGs were synthesised using the *dynamic clustering* in which the models are incrementally updated from a sequence of AGs that represent the same cluster or 3D-object [26] (We used the order of presentation of AGs that obtained the best results). In the SORG method, AGs were classified using the distance measure described in this paper. In the FDG method, the AGs were classified applying the *distance measure*

between AGs and FDGs relaxing second-order constraints (moderate costs on the antagonisms, existences and occurrences), without the *efficient module*, presented in [23,25]. FORGs were compared using the methods presented in [25]. Finally, in the direct AG-to-AG matching method, we used the *edit-operations distance* between AGs presented in [20]. The algorithms presented here were implemented in visual C++ and run on a Pentium IV (1.6Ghz).

9.1 Experiments with randomly generated AGs

The AGs used in this section were generated by the random graph generator process shown in figure 2 (this graph generator was also used and explained in depth in [26]). We first generated 10 initial AGs randomly, one for each model, that had 15 vertices and 5 arcs per vertex. From these AGs, the reference and test sets were derived in the following way. For each initial AG, a reference and a test set of 10 AGs was built by randomly deleting 3 vertices and replacing the attribute of the other vertices by adding gaussian noise with variance V to the attribute values. Then, from each set of 10 reference AGs, an FDG was synthesised.

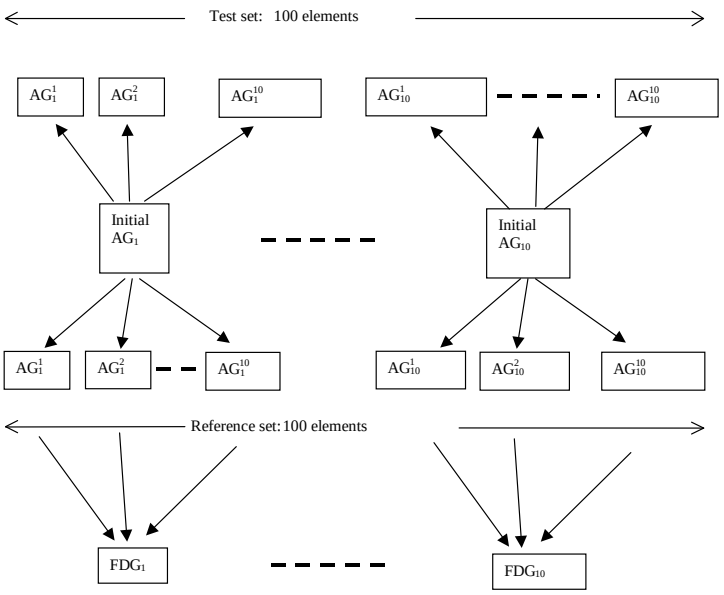


Figure 2. Random generation of reference and test sets and FDG synthesis.

Figure 3 shows in (a) the ratio of recognition correctness and in (b) the time in seconds spent to compute an AG classification in average applying 4 different classification methods: SORGs, FDGs, FORGs and direct AG-AG matching. We have seen that the second-order knowledge kept in the SORGs is higher than in the FDGs and than in the FORGs. We see, through the results, that this knowledge is useful to represent the cluster of AGs and so to increase the recognition ratio. The direct AG-AG matching methods have similar results than SORGs and FDGS only when there is few noise in the test set. When the variance of the noise increases, the AGs in the tests set are very different from the AGs in the reference sets and then the ratio of classification decreases. While considering the run time, we see that the higher differences appear when the variance of the noise is large. FDGs is the fastest method since the antagonisms are useful to prune the search tree

(see [25] for more details). Nevertheless, the Peleg coefficients computed in the distance between AGs and SORGs are also useful to prune the search tree. For this reason, SORGs obtain better results than FORGs. Finally, the direct AG to AG matching is the slowest method when the variance is bigger than 0.6. This is due to the fact that the AGs in the test set are very different to those in the reference set and so the *branch and bound* algorithm can scarcelly prune the search tree.

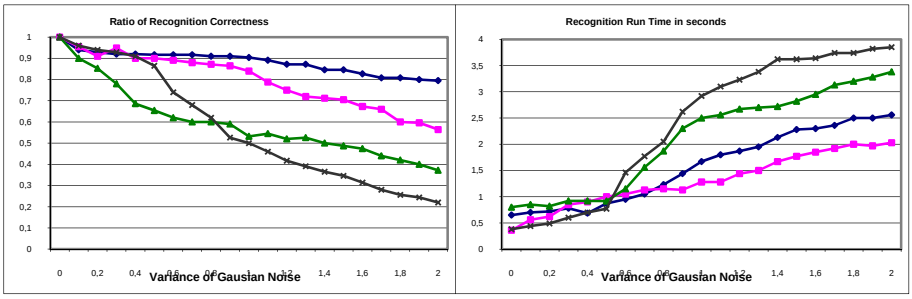


Figure 3. (a) Ratio of recognition correctness (b) run time spent in the classification. SORG: ; FDG: ; FORG: ; AG-AG:

9.2 **Experimental validation using synthetic 3D objects**

In the second experimental validation of our representation, we designed five objects by a CAD program (Figure 4) and then, we took all the topologically different views from these objects (21 views from the first and second object and 12, 24 and 23 from the other three; in total, 101views). Furthermore, we built an AG from each view in which the vertices represent the planar faces (their attribute value is the actual face area) and the arcs represent the edges between faces (their attribute value is the edge length).

Figure 5 shows the average number of vertices of the AGs. From each set of AGs that represent one 3D-object, we synthesised an FDG, thus, 5 FDGs were built. To built the AGs that composed the test set, we modified the attribute values of the vertices and arcs of the initial 101 AGs by adding some Gaussian noise with variance *V*. The advantage of this controlled experiment is that the generated structures represent the 3D-objects without the uncertainty of the segmentation process. For instance, in the FDG case, an antagonism relation between two vertices appears when these elements have never seen together in the same view. And also, an occurrence relation appears when a vertex is visible in all the views in which another one is visible too. See [26] for more details of the synthetic data used. Moreover, there is no time spent on the segmentation process.

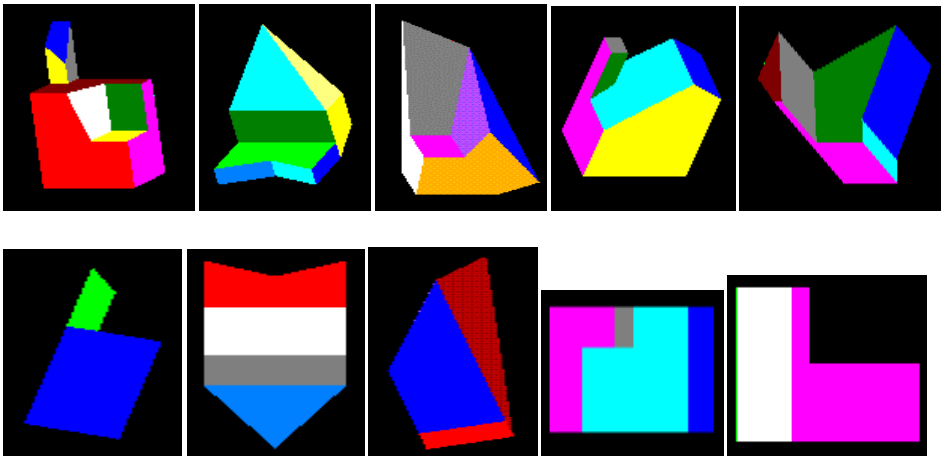


Figure 4. 10 different views extracted from the 5 objects created by a CAD program. Each object is represented by the 2 images of a column. The first line of views are the more representative of the 3D-object and the second line are the simplest views.

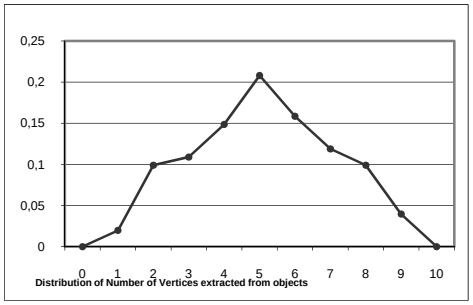


Figure 5. Ratio of the number of vertices in average of the AGs extracted from the 5 objects. AGs have from 1 to 9 vertices and the average is 5.

The results obtained on the ratio of classification and run time are similar than the ones obtained in the previous section (figure 6). In this case, we have shown that SORGs and FDGs are useful methods to represent 3D-objects although the extracted AGs have lost part of the three-dimensional information of the objects. Only when the variance is higher than 1.0, the classification ratio decreases drastically.

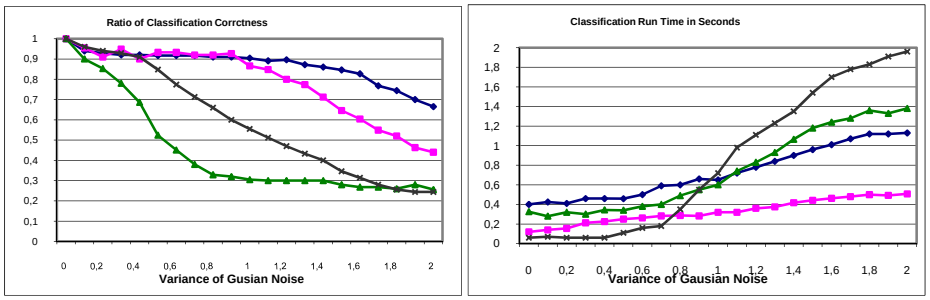


Figure 6. (a) Ratio of recognition correctness (b) run time spent in the classification. SORG: ; FDG: ; FORG: ; AG-AG:

9.3 Application of graph structures to 3D object recognition

Finally, 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). AGs are obtained by the segmentation process presented in [8]. AG nodes represent regions and their attribute value is their average hue and arcs represent adjacent regions and their attribute value is the distance between average hues. Figure 7 shows the 20 objects at angle 100 and their segmented images with the AGs. These AGs have from 6 to 18 vertices and the average number is 10 (figure 8). 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 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 similar method was used for the other experiments with 3, 4, 6 and 9 clusters per 3D-object.

Figure 7. The 20 selected objects at angle 100 and the segmented images with the AGs.

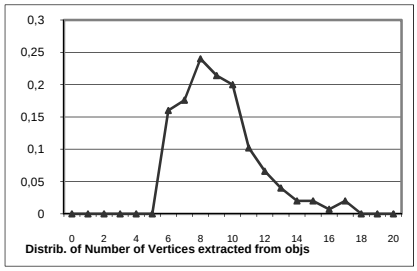


Figure 8. Ratio of the number of vertices in average of the AGs.

Figure 9.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 degree views of the 3D-object and 4 AGs per cluster, there is poor probabilistic knowledge and therefore there is a lack of discrimination between objects. Figure 9.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.

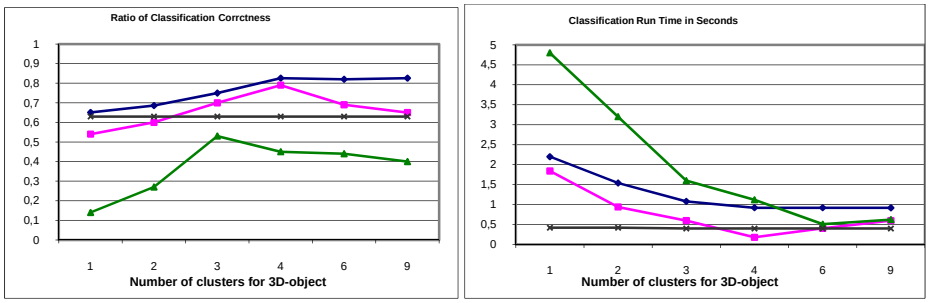


Figure 9. (a) Ratio of recognition correctness (b) run time spent in the classification. SORG: ; FDG: ; FORG: ; AG-AG:

10 Conclusions and future work

We have presented SORGs as a general formulation of an approximation of the joint probability of random elements in a RG, that describes a set of AGs, based on 2^{nd}-order probabilities and marginal ones. We have seen that the FORG and FDG approaches are two specific cases of SORGs. In both cases, the marginal probabilities of the random vertices and arcs are

considered, but the difference between them is in how are the 2^{nd}-order relations between vertices or arcs estimated. FORGs keep only the 2^{nd}-order probability between arcs and their extreme vertices, since the other joint probabilities are estimated as a product of the marginal ones. On the contrary, FDGs keep only a qualitative and structural information of the 2^{nd}-order probabilities between all the vertices and arcs. If we compare both methods, FORGs have local (arc and endpoint vertex) 2^{nd}-order semantic knowledge of the set of AGs but do not use any 2^{nd}-order structural information of the set of the AGs. FDGs do not keep any 2^{nd}-order semantic information but include the 2^{nd}-order structural information of all the set of AGs. For this reason, the storage space of FORGs increases to the square on the size of the random-element domain but the FDGs increases to the square on the number of vertices and arcs.

However, the most important implication of the given general formulation of the 2^{nd}-order random graph representation is that it opens the door to the development of other probabilistic graph approaches, either full 2^{nd}-order or not. In addition, it is interesting to study empirically the relation between the amount of data to be kept in the model and the recognition ratio and run time in several applications. That is, to know in which applications is worthwhile to use explicitly the 2^{nd}-order probabilities or is enough to estimate them by other ways less costly in space requirements, such as FORGs and FDGs.

References

1. R. Alquézar, F. Serratosa, A. Sanfeliu, "Distance between Attributed Graphs and Function-Described Graphs relaxing 2nd order restrictions". *Proc. SSPR'2000 and SPR'2000,* Alicante, Spain, Springer LNCS-1876, pp. 277-286, 2000.
2. S.Berretti, A.Del Bimbo & E.Vicario, "Efficient matching and indexing of graph models in content-based retrieval", *IEEE Trans. on PAMI*, Vol. 23, No. 10, pp: 1089-1105, 2001.
3. H. Bunke, "Error-tolerant graph matching: a formal framework and algorithms". *Proc. Workshops SSPR'98 & SPR'98,* Sydney, Australia, Springer LNCS-1451, pp.1-14, 1998.
4. H.Bunke & G.Allerman, "Inexact graph matching for structural pattern recognition", *Pattern Recognition Letters*, 1 (4), pp: 245-253, 1983.
5. V.Cantoni *et al*., "2D object recognition by multiscale tree matching", *Pattern Recognition*, 31, pp: 1443-1455, 1994.
6. L.P.Cordella, P.Foggia, C.Sansone & M.Vento, "Learning structural shape descriptions from examples", *Pattern Recognition Letters*, Vol. 23, pp: 1427-1437, 2002.
7. W.J. Christmas, J. Kittler and M. Petrou, "Structural matching in computer vision using probabilistic relaxation", *IEEE Transactions on PAMI*, vol. 17, pp. 749-764, 1995.

8. P.F. Felzenswalb and D.P. Huttenlocher, “Image Segmentation Using Local Variation”, *Proc. of the IEEE Computer Soc. Conf. on Computer Vision and Pattern Recognition*, pp. 98-104, 1998.
9. S.Günter & H.Bunke, “Self-organizing map for clustering in the graph domain”, *Pattern Recognition Letters*, Vol. 23, pp: 405-417, 2002.
10. B.Huet & E.R.Hancock, “Relational object recognition from large structural libraries”, *Pattern Recognition*, Vol. 35, pp: 1895-1915, 2002.
11. T. Kohonen & P.Somervuo, “Self-Organising Map on symbol strings”, Neurocomputing 21, pp: 19-30,1998.
12. X.Jiang, A.Münger and H. Bunke, “On median graphs: Properties, algorithms and applications”, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 23, No. 10, pp: 1144-1151, 2001.
13. J. Lladós, E. Martí, J.J. Villanueva, “Symbol Recognition by Error-Tolerant Subgraph Matching between Region Adjacency Graphs”, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 23, No. 10, pp: 1137-1143, 2001.
14. B.T. Messmer & H. Bunke, “A new algorithm for error-tolerant subgraph isomorphism detection”, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 20, pp: 493-504, 1998.
15. R.Myers, R. Wilson & E.Hancock, “Bayesian graph edit distance”, *IEEE Trans. on PAMI*, Vol. 22, pp: 628-635, 2000.

16. S. Peleg and A. Rosenfeld, “Determining compatibility coefficients for curve enchancement relaxation processes”, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 8, pp. 548-555, 1978.
17. M.Pelillo, K.Siddiqi & S.Zucker, “Matching hierarchical structures using associated graphs”, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 21, pp: 1105-1120, 1999.
18. J.Rocha & T.Pavlidis, “A shape analysis model with applications to a character recognition system”, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pp: 393-404, 1994.
19. A. Sanfeliu, R. Alquézar, J. Andrade, J.Climent, F. Serratosa & J.Vergés, “Graph-based Representations and Techniques for Image Processing and Image Analysis”, *Pattern Recognition*, vol. 35, pp: 639-650, 2002.
20. 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.
21. K.Sengupta & K.L.Boyer, “Organizing large structural model bases,” *IEEE Trans. on PAMI*, Vol. 17, No. 4, pp: 321-331, 1995.
22. D.S. Seong, H.S. Kim & K.H. Park, “Incremental Clustering of Attributed Graphs”, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 23, pp. 1399-1411, 1993.

23. F. Serratosa, R. Alquezar & A. Sanfeliu, “Efficient algorithms for matching attributed graphs and function-described graphs”, in *Proceedings ICPR’2000, 15th Int. Conf. on Pattern Recognition*, Barcelona, Spain, Vol.2, pp. 871-876, 2000.
24. F. Serratosa, R. Alquézar & A. Sanfeliu, “Estimating the Joint Probability Distribution of Random Vertices and Arcs by means of Second-order Random Graphs”, *Proc. Syntactic and Structural Pattern Recognition, SSPR’2002, LNCS 2396*, pp: 252-262, 2002.
25. F. Serratosa, R. Alquézar & A. Sanfeliu, “Function-described graphs for modeling objects represented by attributed graphs”, *Pattern Recognition*, 36 (3), pp. 781-798, 2003.
26. F. Serratosa, R. Alquézar & A. Sanfeliu, “Synthesis of Function-Described Graphs and clustering of Attributed Graphs”, *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 16, No. 6, pp. 621-655, 2002.
27. K.Shearer, S.Venkatesh & H.Bunke, “Video indexing and similarity retrieval by largest common subgraph detection using decision trees”, *Pattern Recognition*, 34 (5), pp: 1075-1091, 2001.
28. W.D.Wallis, P.Shoubridge, M.Kraetz & D.Ray, “Graph distances using graph union”, *Pattern Recognition Letters*, Vol. 22, pp: 701-704, 2001.
29. E.K. Wong, “Model matching in robot vision by subgraph isomorphism” *Pattern Recognition*, Vol. 25, pp: 287-304, 1994.

30. A.K.C. Wong & M. You, “Entropy and distance of random graphs with application to structural pattern recognition”, *IEEE Trans. on PAMI*, vol. 7, pp. 599-609, 1985.