Pattern Recognition 46 (2013) 551–565

Contents lists available at SciVerse ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Fuzzy multilevel graph embedding

Muhammad Muzzamil Luqman [a,b,*], Jean-Yves Ramel [a], Josep Lladós [b], Thierry Brouard [a]

[a] Laboratoire d'Informatique, Université François Rabelais de Tours, 37200, France
[b] Computer Vision Center, Universitat Autonòma de Barcelona, 08193, Spain

ARTICLE INFO

Article history:
Received 9 December 2011
Received in revised form
10 July 2012
Accepted 31 July 2012
Available online 10 August 2012

Keywords:
Pattern recognition
Graphics recognition
Graph clustering
Graph classification
Explicit graph embedding
Fuzzy logic

ABSTRACT

Structural pattern recognition approaches offer the most expressive, convenient, powerful but computational expensive representations of underlying relational information. To benefit from mature, less expensive and efficient state-of-the-art machine learning models of statistical pattern recognition they must be mapped to a low-dimensional vector space. Our method of explicit graph embedding bridges the gap between structural and statistical pattern recognition. We extract the topological, structural and attribute information from a graph and encode numeric details by fuzzy histograms and symbolic details by crisp histograms. The histograms are concatenated to achieve a simple and straightforward embedding of graph into a low-dimensional numeric feature vector. Experimentation on standard public graph datasets shows that our method outperforms the state-of-the-art methods of graph embedding for richly attributed graphs.

1. Introduction

Pattern recognition has emerged as an important research domain and has supported the development of numerous applications in many different areas of activity. For a general introduction to former we refer the interested reader to [1,2]. The methods for pattern recognition are broadly categorized as statistical, structural or syntactic approaches [3]. In this paper we address the problem of lack of computational tools for structural pattern recognition and propose to exploit the computational efficiency of statistical pattern recognition. This permits a pattern recognition application to benefit from representational power of structural methods and computational efficiency of statistical methods, while avoiding the limitations of both. The next two paragraphs briefly introduce the main advantages and limitations of structural and statistical pattern recognition.

Structural pattern recognition is characterized by the use of symbolic data structures i.e. graphs, strings and trees. Graphs are widely used in structural pattern recognition and can safely be termed as representative of symbolic data structures (strings and trees are special instances of graphs [4]). Graphs provide a convenient and powerful representation of relational information. They are able to represent not only the values of both symbolic and numeric properties of an object, but can also explicitly model the spatial, temporal and conceptual relations that exist between its parts. Moreover, graphs do not suffer from the constraint of fixed dimensionality. For example, the number of nodes and edges in a graph is not limited a priori and depends on the size and the complexity of the actual object to be modeled [5]. And above all, graphs have foundations in strong mathematical formulation and have a mature theory at their basis. However two serious drawbacks of graph based representations are that these representations are sensitive to noise and that the algorithmic tools for performing different operations on them are computational expensive. For instance the much needed operations of graph matching and graph isomorphism are NP-complete. For further reading on structural pattern recognition we refer the interested reader to [4–8].

Statistical pattern recognition is characterized by the use of numeric feature vectors. A very important advantage of these representations is that because of their simple structure, the basic operations that are used in machine learning can easily be executed on them. This makes a large number of mature algorithms for pattern analysis and classification immediately available to statistical pattern recognition. And, as a result of this fact, the statistical pattern recognition offers state-of-the art computational efficient tools of learning, classification and clustering. However, feature vector based representations have associated representational limitations, which arise from their simple structure and the fact that they have same length and structure regardless of the complexity of object to be modeled [9]. For further reading on statistical pattern recognition and classification we refer the interested reader to [10].

* Corresponding author at: Laboratoire d'Informatique, Université François Rabelais de Tours, 37200, France. Tel.: +33 247361443; fax: +33 247361422.
E-mail addresses: mluqman@cvc.uab.es, luqman@univ-tours.fr (M.M. Luqman), ramel@univ-tours.fr (J.-Y. Ramel), josep@cvc.uab.es (J. Lladós), brouard@univ-tours.fr (T. Brouard).

0031-3203/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.patcog.2012.07.029

1.1. Graph embedding

Graph embedding is a natural outcome of parallel advancements in structural and statistical pattern recognition. Over decades, the pattern recognition research community has developed a range of expressive and powerful approaches for diverse problem domains. Graph based structural representations are usually employed for extracting the structure, topology and geometry in addition to the statistical details of underlying data. These representations could not be exploited to their full strength during the next step in processing chain, because of limited availability of computational tools. On the other hand,the efficient and mature computational models offered by statistical approaches work only on vector data and cannot be directly applied to high-dimensional structural representations. Graph embedding acts as a bridge between structural and statistical approaches [3,6,11], and allows a pattern recognition method to benefit from computational efficiency of state-of-the-art statistical models and tools [12] along-with the convenience and representational power of classical symbolic representations [7]. This makes it possible to address the problems of learning, classification and clustering for graphs, which are among the most basic tasks in pattern recognition [13]. Graph embedding has its application to the whole variety of domains which are entertained by pattern recognition and where the use of a relational data structure is mandatory for performing high level tasks. Graph embedding methods are also employed to solve the computationally hard problems geometrically [14]. For further reading on graph embedding we refer the interested reader to [15,16].

The graph embedding methods are formally categorized as implicit graph embedding or explicit graph embedding. The implicit graph embedding methods are based on graph kernels. A graph kernel is a function that can be thought of as a dot product in some implicitly existing vector space. Instead of mapping graphs from graph space to vector space and then computing their dot product, the value of the kernel function is evaluated in graph space. Since it does not explicitly map a graph to a point in vector space, a strict limitation of implicit graph embedding is that it does not permit all the operations that could be defined on vector spaces. For further reading on graph kernels and implicit graph embedding we refer the interested reader to [16,17]. The more useful, explicit graph embedding methods explicitly embed an input graph into a feature vector and enable the use of all the methodologies and techniques devised for vector spaces. The vectors obtained by an explicit graph embedding method can also be employed in a standard dot product for defining an implicit graph embedding function between two graphs [4]. An interesting property of explicit graph embedding is that it embeds graphs in pattern spaces in a manner that similar structures come close to each other and different structures go far away i.e. an implicit clustering is achieved [18]. Another important property of explicit graph embedding is that the graphs of different size and order are embedded into a fixed size feature vector. This means that for constructing the feature vector, an important step is to mark the important details that are available in all the graphs and are applicable to a broad range of graph types. For further reading on explicit graph embedding we refer the interested reader to [16].

Graph embedding is an interesting approximate solution for addressing the problem of in-exact graph matching, which belongs to the class of NP-complete problems. By mapping a high dimensional graph into a point in suitable vector space, graph embedding permits to perform the basic mathematical computations which are required by various statistical pattern recognition techniques, and offers interesting solutions to the problems of graph clustering and classification. However, in our opinion because of the strict limitation of the resulting feature vector of not being capable of preserving the matching between nodes of graphs, graph embedding always lacks the capabilities to address the problem of graph isomorphism (i.e. exact graph matching).

1.2. Proposed method

In this paper we present an unsupervised method for explicit embedding of directed and undirected attributed graphs with many numeric as well as symbolic attributes on both nodes and edges (which represent a very general super class of graphs), into feature vectors. The method is equally applicable to strings and trees as well. We employ fuzzy logic for addressing the noise sensitivity of graph based representations whilst achieving a simple and straightforward embedding of topological, structural and attribute information of a graph into a low-dimensional numeric feature vector. The method has been named as Fuzzy Multilevel Graph Embedding and abbreviated as *FMGE*. It embeds an attributed graph into a feature vector by extracting graph level details, subgraph homogeneity details and elementary level details. The feature vector is constructed by employing a direct encoding of graph level details followed by encoding of the distribution of subgraph homogeneity and elementary level details of the graph. The latter is achieved by constructing fuzzy histograms for numeric information and crisp histograms for symbolic information. The parameters for these histograms are learned during a prior unsupervised learning phase which does not necessarily require any labeled learning set. The work presented in this paper is an evolved version of our previous work [19]. Apart from formalizing our previously proposed graph embedding method to be generally applicable to a wide range of graph representations and the theoretical contributions of our work highlighted in next section, the experimentation has been enlarged by testing on more graph databases and the method is applied to the real problem of graph retrieval and subgraph spotting.

In Section 2 we outline related works on graph embedding. Section 3 presents definitions and formalizes the notation used in this paper. Section 4 introduces an overall global description of FMGE. In Section 5 we present details on the unsupervised learning phase and graph embedding phase of FMGE. Experimental results are presented in Section 6 and are followed by a discussion on the parameters of FMGE in Section 7. The paper concludes by presenting future directions of work in Section 8.

2. Related works

In the literature the problem of graph embedding has been approached by three important families of algorithms. Recent surveys on graph embedding are presented in [4,15,16]. Among these, a first family of graph embedding methods is based on the frequencies of appearance of specific knowledge-dependent substructures in graph. These works are mostly proposed for chemical compounds and molecular structures. Graph representation of molecules are assigned feature vectors whose components are the frequencies of appearance of specific knowledge-dependent substructures in graphs. Interesting works in this family of methods include [20–22]. The methods in this family of graph embedding algorithms are based on finding subgraphs in graph and are capable of exploiting domain knowledge for graph embedding. However, they have the drawback that finding substructures in graphs is computationally challenging.

A second family of graph embedding algorithms is spectral based embedding. Spectral based embedding is a very prominent class of graph embedding methods and is proposed by lots of works in literature. In order to embed graphs into feature vectors, this family of methods extract features from graphs by eigen-decomposition of adjacency and Laplacian matrices and then

apply a dimensionality reduction technique on the eigen-features. Luo et al. in [23] have proposed very interesting work for graph embedding based on eigen-decomposition of adjacency matrix. Another interesting work in this family of algorithms is by Wilson et al. [18]. They employ the spectral matrix of Laplacian of a graph to construct symmetric polynomials whose coefficients are used as graph features. Bai et al. in [24] have used multidimensional scaling characterized by the matrix of the geodesic distances between nodes into a manifold, Torsello et al. in [25] have used the computation of minimum common super-tree and Emms et al. in [26] have employed the encoding of commute time for random walks for graph embedding. The work of Chen et al. [27] on local discriminant embedding and Shaw et al. [14] on structure preserving embedding also propose interesting spectral based embedding methods. Malik et al. in [28] have employed the spectral graph theory for proposing graph embedding methods based on graph cuts. Recently Ren et al. in [29] have proposed an interesting method of spectral based embedding, where they have used the polynomial coefficients of Ihara Zeta function for describing structure and topology of a graph. The spectral family of graph embedding methods is very interesting and provide solid theoretical insight into the meaning and significance of extracted features but a very serious drawback of these methods is that they remain restricted to unattributed graphs. They are also very sensitive to structural errors in graphs i.e. missing nodes and edges.

Finally a third family of graph embedding algorithms is based on dissimilarity of a graph from a set of prototypes. The dissimilarity based graph embedding can handle arbitrary graphs. The dissimilarity based graph embedding methods usually use graph edit distance and exploits domain knowledge. But since graph edit distance is computationally expensive, the dissimilarity based graph embedding methods may become computationally challenging. The work of Riesen and Bunke [4,5] is a very interesting contribution to the existing literature on graph embedding. Their method is applicable to both directed and undirected graphs. It is based on prototype selection for mapping graphs to dissimilarity spaces. The key idea of their approach is to use the distances of an input graph AG to a number of training graphs, termed prototype graphs, as a vectorial description of AG. This method assumes that a dissimilarity function between two graphs has been defined and proposes to use an approximation of graph edit distance, although the method can work with other dissimilarity measures. Given a set of graphs, the method starts by choosing n prototypes that will be used as references for constructing the vectors. The choice of prototype graphs and also their number n, is a critical issue. The method attempts to select the prototypes that best possibly reflect the distribution of the set of graphs. The method showed improvement over traditional graph edit distance based nearest neighbor classifier approach and also allowed the use of sophisticated classifiers which is not possible in original graph space, but we argue that graph edit distance is computationally expensive. As this method has been proposed for pattern recognition, instead of employing graph edit distance as a dissimilarity measure for constructing vectors and then employing a classifier, graph edit distance could have directly been computed between a graph AG and each of the prototype graphs for arriving at a decision.

2.1. Our contribution

Most of the existing works on graph embedding deal only the graphs which are comprised of edges with a single attribute and vertices with either no or only symbolic attributes. These methods are only useful for specific application domains for which they are designed. FMGE does not require any dissimilarity measure between graphs and to the best of our knowledge, FMGE extends the methods in literature by offering the embedding of attributed graphs with many numeric as well as symbolic attributes on both nodes and edges. It is applicable to directed as well as undirected attributed graphs. Many existing solutions for graph embedding offer to utilize the statistical significant details in graphs for embedding them into feature vectors. FMGE exploits the topological, structural and attribute information of the graphs along-with the statistical significant information, for constructing feature vectors of adapted and optimal size. It employs fuzzy overlapping trapezoidal intervals for minimizing the information loss while mapping from continuous graph space to discrete feature vector space. The proposed feature vector is very significant for application domains where the use of graphs is mandatory for representing rich structural and topological information, and an approximate but computational efficient solution is needed. The unsupervised learning abilities of FMGE and the fact that it does not require a labeled graph dataset for learning allows its inexpensive deployment to various application domains.

The extraction of subgraph homogeneity for embedding topological and structural level details and multilevel distribution analysis of graph are the novelty of our work. First of the two important theoretical contributions of this work (w.r.t. our previous work [19]) is unsupervised learning algorithm for constructing fuzzy histograms. This is inspired from a work on deriving fuzzy intervals from crisp intervals [30] and this addresses the problem of noise sensitivity of graphs. The second important contribution which is a novelty to graph embedding methods, is the embedding of information about homogeneity of subgraphs in a graph.

3. Definitions and notations

Definition 1 (*Attributed graph (AG)*). Let A_V and A_E denote the domains of possible values for attributed vertices and edges respectively. These domains are assumed to include a special value that represents a null value of a vertex or an edge. In this paper the term attributed graph is used to refer to an undirected attributed graph, unless explicitly specified. An attributed graph AG over (A_V, A_E) is defined to be a four-tuple:

$$AG = (V, E, \mu^V, \mu^E)$$

where

V is a set of vertices, $E \subseteq V \times V$ is a set of edges,
$\mu^V : V \longrightarrow A_V^k$ is function assigning k attributes to vertices and
$\mu^E : E \longrightarrow A_E^l$ is a function assigning l attributes to edges.

Definition 2 (*Graph order*). The order of a graph $AG = (V, E, \mu^V, \mu^E)$ is given by $|V|$ i.e. the number of vertices in AG.

Let AG_1 and AG_2 be two attributed graphs, then:
AG_1 is smaller than AG_2 iff $|V_1| < |V_2|$
AG_1 and AG_2 are equal ordered iff $|V_1| = |V_2|$
AG_1 is bigger than AG_2 iff $|V_1| > |V_2|$

Definition 3 (*Graph size*). The size of a graph $AG = (V, E, \mu^V, \mu^E)$ is given by $|E|$ i.e. the number of edges in AG.

Let AG_1 and AG_2 be two attributed graphs, then:
AG_1 is thinner than AG_2 iff $|E_1| < |E_2|$
AG_1 and AG_2 are equal sized iff $|E_1| = |E_2|$
AG_1 is thicker than AG_2 iff $|E_1| > |E_2|$

Definition 4 (*Node degree*). The degree of a vertex (or node) V_i in graph $AG = (V, E, \mu^V, \mu^E)$ refers to the number of edges connected to V_i. If AG is a directed graph then each of its nodes has an in-degree

and an out-degree associated to it. The in-degree refers to the number of incoming edges and out-degree refers to the number of outgoing edges for a node. Generally, the terms *densely connected graph* and *sparsely connected graph* are used for abstractly categorizing a graph on the basis of its node degrees.

Definition 5 (*Graph embedding*). Graph embedding is a methodology aimed at representing a whole graph, along with the attributes attached to its nodes and edges, as a point in a suitable vector space.

Definition 6 (*Explicit graph embedding*). Explicit graph embedding maps a graph to a point in suitable vector space. It encodes the graphs by equal size vectors and produces one vector per graph. Mathematically, for a given graph $AG=(V,E,\mu^V,\mu^E)$, explicit graph embedding is a function ϕ, which maps graph AG from graph space G to a point $(f_1,f_2,\ldots f_n)$ in n dimensional vector space $\mathbb{R}^n$. It is given as

$$\phi : G \rightarrow \mathbb{R}^n$$

$$AG \mapsto \phi(AG) = (f_1, f_2, \ldots, f_n)$$

4. Overview of fuzzy multilevel graph embedding (FMGE)

A block diagram of FMGE is presented in Fig. 1. It accepts a collection of m attributed graphs as input and encodes their topological, structural and attribute details into m equal size feature vectors. The feature vector of FMGE is named as *Fuzzy Structural Multilevel Feature Vector* and abbreviated as *FSMFV*.

4.1. Input of FMGE

As input FMGE accepts a collection of m attributed graphs $\{AG_1, AG_2, \ldots, AG_e, \ldots, AG_m\}$, where the eth graph is denoted by $AG_e=(V_e,E_e,\mu^{V_e},\mu^{E_e})$.

4.2. Output of FMGE

As output FMGE produces a collection of m same size feature vectors, given by $\{FSMFV_1, FSMFV_2, \ldots, FSMFV_e, \ldots, FSMFV_m\}$. The eth input graph AG_e is embedded into feature vector $FSMFV_e$: $AG_e \mapsto \phi(AG_e) = FSMFV_e$. Where $FSMFV_e$ is a point in n dimensional vector space $\mathbb{R}^n : FSMFV_e = (f_{e_1}, f_{e_2}, \ldots, f_{e_n})$.

4.3. Description of feature vector of FMGE

Fuzzy Structural Multilevel Feature Vector (*FSMFV*) contains features extracted from three levels of information in graph: (i) graph level information i.e. graph order and graph size, (ii) the structural level information extracted from node degrees and subgraph homogeneity and (iii) the elementary level information extracted from node and edge attributes. FSMFV is a vector in n dimensional vector space $\mathbb{R}^n$, given as $FSMFV=(f_1,f_2,\ldots,f_n)$. The overall structure of FSMFV is presented in Fig. 2.

4.3.1. Embedding of graph level information

The graph level information in FSMFV is embedded by two numeric features, encoding the order and the size of graph.

Graph order: A graph vertex is an abstract representation of the primitive components of underlying content. The order of a graph provides very important discriminatory topological information on the graph. Graph order ($|V|$) allows to discriminate between a small graph in Fig. 4(a) and bigger graphs in Fig. 4(b) and (c) ($|V|=3$ vs $|V|=4$). At the same time it permits to define a similarity between two equal ordered graphs in Fig. 4(b) and (c) ($|V|=4$).

Graph size: An edge is an abstract representation of the relationship between the primitive components of underlying content. Graph size also provides important discriminatory information on the topological details of graph. Two equal ordered graphs in Fig. 4(b) and (c) are differentiated by graph size ($|E|$) i.e. a thin graph in Fig. 4(b) is differentiated from a thicker graph in Fig. 4(c) ($|E|=3$ vs $|E|=4$). At the same time it permits to define a similarity between two equal sized graphs in Fig. 4(a) and (b) ($|E|=3$).

4.3.2. Embedding of structural level information

The embedding of structural level information is a novelty and the most critical part of FMGE. No existing work on graph embedding clearly uses this information. We use node degrees information and subgraph homogeneity measure embedded by histograms of node attributes resemblance and edge attributes resemblance, for embedding structural level information. Fig. 3 outlines this part of FSMFV.

Node degrees: The degrees of nodes represent the distribution of edges in graph and provide complementary discriminatory information on the structure and topology of graph. It permits to discriminate between densely connected graphs and sparsely connected graphs. Node degrees information is encoded by a histogram of s_i fuzzy intervals. The fuzzy intervals are learned during a prior learning phase, which employs degrees of all the nodes of all graphs in dataset. Node degrees features (h^d in Fig. 3), for an attributed graph, embeds the histogram of its nodes for the s_i fuzzy intervals. In Fig. 3, the histogram h^d is a fuzzy histogram as node degrees is a numeric information. For directed graphs this feature is represented by two sub-features of in-degree and out-degree i.e. a fuzzy histogram for encoding the distribution of in-degrees and an other fuzzy histogram for encoding the distribution of out-degrees of nodes. The node degree information for the graphs in Fig. 4(a)–(c) permits to increase the precision of the

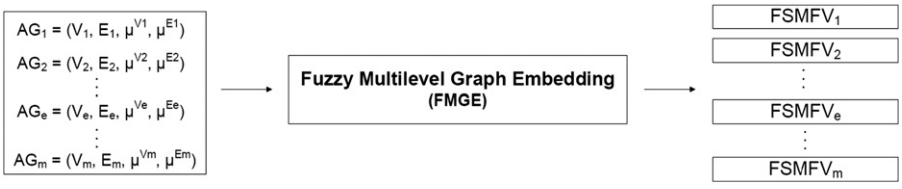

Fig. 1. Overview of Fuzzy Multilevel Graph Embedding (FMGE).

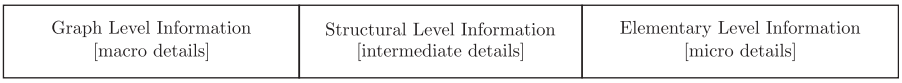

Fig. 2. The Fuzzy Structural Multilevel Feature Vector (FSMFV).

Histogram of node degrees h^d	Histograms of node attributes resemblance h_d^{nr} and $h_1^{nr}, h_2^{nr}, \ldots, h_k^{nr}$	Histograms of edge attributes resemblance $h_1^{er}, h_2^{er}, \ldots, h_l^{er}$

Fig. 3. Embedding of structural level information.

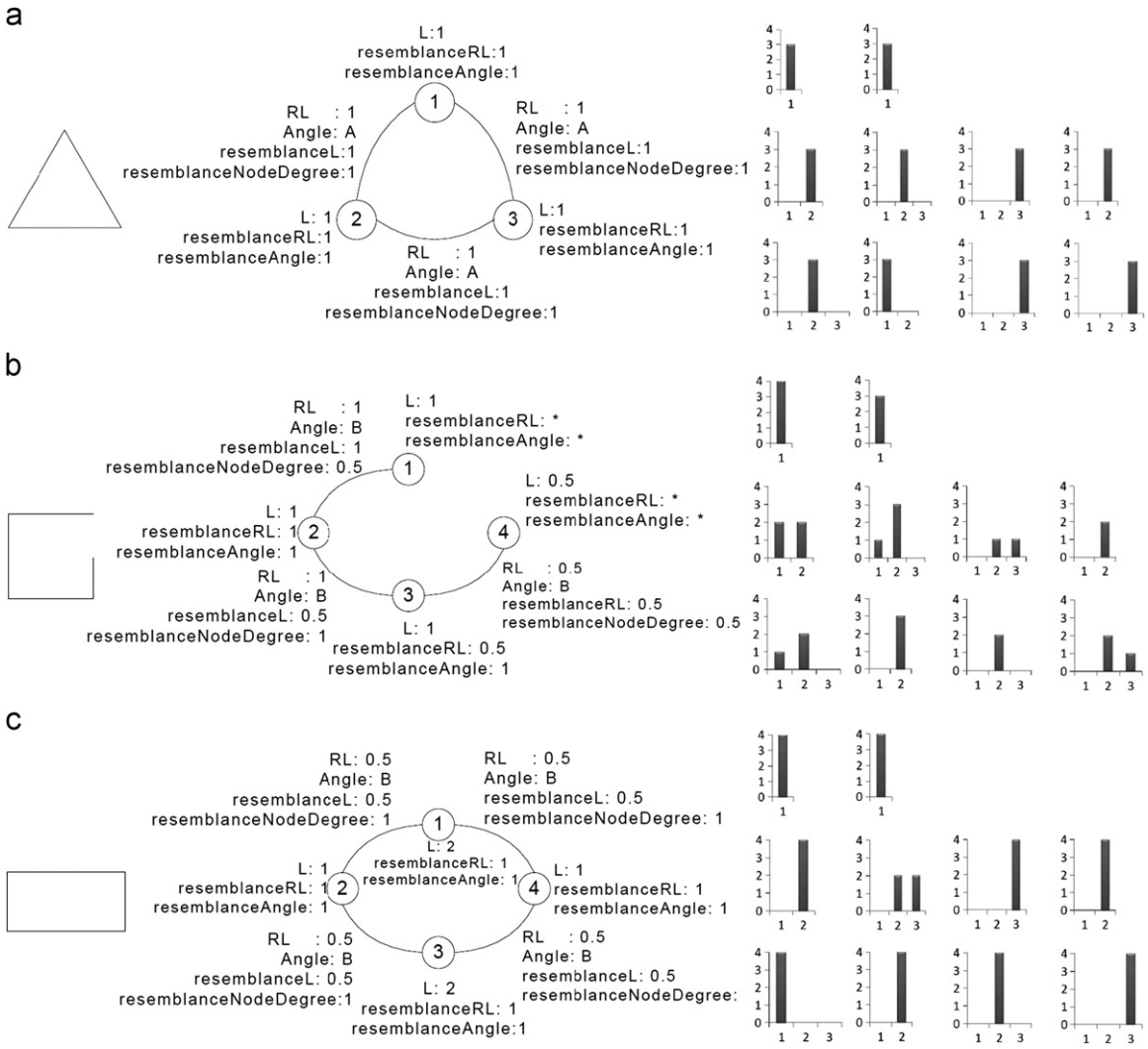

Fig. 4. An example for illustration of the extraction of discriminatory information, for basic geometric shapes of unit length, by FSMFV. A vertex in graph is an abstract representation of a primitive line in underlying content, with an attribute of length *L*. An edge in graph represents the connectivity relationship between two primitive lines in underlying content, with their relative length *RL* and angle *Angle* between them as the edge attributes. The respective resemblance attributes are shown on the nodes and edges. The histograms in first row for each graph present the histograms for graph order and graph size (left to right). The center row presents histograms for node degree, node attributes *L*, *resemblanceRL* and *resemblanceAngle* respectively from left to right. The last row presents the histograms for edge attributes *RL*, *Angle*, *resemblanceL* and *resemblanceNodeDegree* respectively from left to right. (a) $|V|=3$, $|E|=3$, $edgeAtt=\{\{1,1,1\},\{A,A,A\},\{1,1,1\},\{1,1,1\}\}$, $nodeDegree=\{2,2,2\}$, $nodeAtt=\{\{1,1,1\},\{1,1,1\},\{0,0,0\}\}$ and $FSMFV=3,3,0,3,0,3,0,3,0,0,0,3,0,3,0,3,0,3,0,3,0,0,0,3,0,0,3$. (b) $|V|=4$, $|E|=3$, $edgeAtt=\{\{1,1,0.5\},\{B,B,B\},\{1,0.5,0.5\},\{0.5,1,0.5\}\}$, $nodeDegree=\{1,2,2,1\}$, $nodeAtt=\{\{1,1,1,0.5\},\{*,1,0.5,*\},\{*,1,1,*\}\}$ and $FSMFV=4,3,2,2,1,3,0,0,1,1,0,2,1,2,0,0,3,0,2,0,0,2,1$. (c) $|V|=4$, $|E|=4$, $edgeAtt=\{\{.5,.5,.5,.5\},\{B,B,B,B\},\{.5,.5,.5,.5\},\{1,1,1,1\}\}$, $nodeDegree=\{2,2,2,2\}$, $nodeAtt=\{\{2,1,2,1\},\{1,1,1,1\},\{1,1,1,1\}\}$ and $FSMFV=4,4,0,4,0,2,2,0,0,4,0,4,4,0,0,0,4,0,4,0,0,0,4$.

similarity between graphs. For example, the graphs in Fig. 4(a) and (b) represent two different topologies ($|V|=3$ and $|V|=4$) and a relatively similar geometry ($|E|=3$). On the other hand, the graphs in Fig. 4(b) and (c) represent two different geometries ($|E|=3$ and $|E|=4$) and a quite similar topology ($|V|=4$). Furthermore, the graphs in Fig. 4(a) and (c) represent two different topologies ($|V|=3$ and $|V|=4$) and geometries ($|E|=3$ and $|E|=4$). By incorporating the node degree information, it is very straightforward to conclude that graphs in Fig. 4(a) and (c) are more similar to each other than the graph in Fig. 4(a).

Node attributes resemblance for edges: The resemblance between two primitive components that have a relationship between them is a supplementary information available in the graph. The node attributes resemblance for an edge encodes structural information for the respective node-couple. To compute resemblance information for an edge, the node degrees of its two nodes and the list of node attributes as given by μ^V are employed for extracting additional information. This additional information is represented as new edge attributes and is processed like other edge attributes. Given an edge between two nodes, say $node_1$ and $node_2$ in a graph, the resemblance between a

numeric node attribute a is computed by the following first equation and the resemblance between a symbolic node attribute b is computed by the following second equation:

$$\text{resemblance}(a_1,a_2) = \min(|a_1|,|a_2|)/\max(|a_1|,|a_2|) \quad (1)$$

where

$a \in \{\text{node degree}, \mu^V\}$,
a_1 is the value of the attribute a for $node_1$ and
a_2 is the value of the same attribute a for $node_2$.

$$\text{resemblance}(b_1,b_2) = \begin{vmatrix} 1 & b_1 = b_2 \\ 0 & \text{otherwise} \end{vmatrix} \quad (2)$$

where

$b \in \mu^V$,
b_1 is the value of the attribute b for $node_1$ and
b_2 is the value of the same attribute b for $node_2$.

For each symbolic node attribute we represent the resemblance attribute nr by exactly two numeric features. The resemblance for symbolic attributes can either be 0 or 1 (Eq. (2)). The cardinalities of the two resemblance values in input graph, are encoded by a crisp histogram which is used as features in FSMFV. For each numeric node attribute, the resemblance attribute nr is represented by s_{nr} features in FSMFV. This resemblance information is encoded by a fuzzy histogram of s_{nr} fuzzy intervals. The fuzzy intervals are learned during a prior learning phase, which employs resemblance attribute nr of all the edges of all graphs in dataset. In Fig. 3 the histogram h_d^{nr} represents the features for encoding resemblance attribute for node degrees. Whereas, the histograms $h_1^{nr},h_2^{nr},\ldots,h_k^{nr}$ represent the features for encoding resemblance attributes for k node attributes μ^V. The histogram h_d^{nr} is a fuzzy histogram since node degree is a numeric information. Each of the histograms $h_1^{nr},h_2^{nr},\ldots,h_k^{nr}$, is a crisp histogram if its corresponding attribute is symbolic and is a fuzzy histogram if the attribute that it is encoding is a numeric attribute. Node attributes resemblance for edges of example graphs in Fig. 4 provides more precision for discriminating between the graphs.

Edge attributes resemblance for nodes: The resemblance among the relationships associated to a primitive component is a supplementary information available in the graph. The edge attributes resemblance for a node encodes the structural information for the respective edges of node and brings more topological information to FSMFV. To compute resemblance information for a node, each attribute of its edges (as given by μ^E) is employed for extracting additional information. This additional information is represented as new node attributes and is processed like other node attributes. Given a node, say *node* in a graph, the resemblance for the edges connected to *node* is computed as the mean of the resemblances between all the pair of edges connected to *node*. For a pair of edges, say $edge_1$ and $edge_2$ connected to *node*, the resemblance for a numeric attribute c is computed by the following first equation and the resemblance between a symbolic edge attribute d is computed by the following second equation:

$$\text{resemblance}(c_1,c_2) = \min(|c_1|,|c_2|)/\max(|c_1|,|c_2|) \quad (3)$$

where

$c \in \mu^E$,
c_1 is the value of the attribute c for $edge_1$ and
c_2 is the value of the same attribute c for $edge_2$.

$$\text{resemblance}(d_1,d_2) = \begin{vmatrix} 1 & d_1 = d_2 \\ 0 & \text{otherwise} \end{vmatrix} \quad (4)$$

where

$d \in \mu^E$,
d_1 is the value of the attribute d for $edge_1$ and
d_2 is the value of the same attribute d for $edge_2$.

For each symbolic edge attribute, the resulting resemblance attribute er is treated as a numeric attribute and is embedded by a fuzzy histogram. The resemblance for symbolic edge attributes is computed as mean of the resemblances between all the pair of edges connected to a node. Although the resemblance for a pair of edges is always either 0 or 1 but mean resemblance for all the pair of edges of a node can be any numeric value (this is different from the symbolic node attributes resemblance which is always either 0 or 1). Therefore, for each symbolic and numeric edge attribute, the resemblance attribute er is represented by s_{er} features in FSMFV. This resemblance information is encoded by a fuzzy histogram of s_{er} fuzzy intervals. The fuzzy intervals are learned during a prior learning phase, which employs resemblance attribute er of all the nodes of all graphs in dataset. In Fig. 3 the histograms $h_1^{er},h_2^{er},\ldots,h_l^{er}$ represent the features for encoding resemblance attributes for l edge attributes μ^E. Each of the histograms $h_1^{er},h_2^{er},\ldots,h_l^{er}$, is a fuzzy histogram. The edge attributes resemblance for nodes of example graphs in Fig. 4 compliments the node resemblance attributes for differentiating the graphs.

4.3.3. Embedding of elementary level information

Embedding of elementary level information allows FMGE to extract discriminatory information from individual nodes and edges of the graph. The symbolic attributes are encoded by crisp histograms and the numeric attributes by fuzzy histograms. FMGE can embed attributed graphs with many symbolic and numeric attributes on both nodes and edges. For every symbolic node attribute, each modality that can be taken by this attribute is represented by exactly one numeric feature in FSMFV. This feature encodes the cardinality of this modality in an input graph. Each numeric node attribute is encoded by a fuzzy histogram of its s_i fuzzy intervals. The fuzzy intervals are learned for each of the numeric node attributes in the input graph dataset during a prior learning phase. The features for a numeric attribute of an input graph, embeds the histogram for these s_i fuzzy intervals. Fig. 5 outlines this part of FSMFV.

Node attributes: The node attributes provide additional details on primitive components of underlying content and aid FSMFV to discriminate between two similar structured graphs. In Fig. 5 the histograms $h_1^n,h_2^n,\ldots,h_k^n$ represent the features for encoding k node attributes μ^V. Each of the histograms $h_1^n,h_2^n,\ldots,h_k^n$, is a crisp histogram if its corresponding attribute is symbolic and is a fuzzy histogram if the attribute that it is encoding is a numeric attribute. The node attributes are very important information for discriminating between two equal ordered and equal sized graphs, which are representing quite similar structure as well (e.g. the graph of a small square can be differentiated from that of a big square by using a length attribute on the nodes). The length attribute L of the graph nodes in Fig. 4(c) provides additional details on the graph and clearly discriminates it from the graph in Fig. 4(b).

Histograms of node attributes $h_1^n, h_2^n, \ldots, h_k^n$	Histograms of edge attributes $h_1^e, h_2^e, \ldots, h_l^e$

Fig. 5. Embedding of elementary level information.

Edge attributes: The edge attributes provide supplementary details on the relationships between the primitive components of the underlying content and aid FSMFV to discriminate between two similar structured graphs. In Fig. 5 the histograms $h_1^e, h_2^e, \ldots, h_l^e$ represent the features for encoding l edge attributes μ^E. Each of the histograms $h_1^e, h_2^e, \ldots, h_l^e$ is a crisp histogram if its corresponding attribute is symbolic and is a fuzzy histogram if the attribute that it is encoding is a numeric attribute. The edge attributes are very important information for discriminating between two equal ordered and equal sized graphs, which are representing quite similar structure as well (e.g. the graph of a square can be differentiated from that of a rhombus by using angle attribute on the edges). The relative length *RL* and angle *A* attribute of the graph edges in Fig. 4(b) provide additional details on the graph and clearly discriminates it from the graph in Fig. 4(a).

5. Framework of fuzzy multilevel graph embedding (FMGE)

In FMGE framework, the mapping of input collection of graphs to appropriate points in a suitable vector space $\mathbb{R}^n$ is achieved in two phases i.e. the off-line unsupervised learning phase and the on-line graph embedding phase. The unsupervised learning phase learns a set of fuzzy intervals for features linked to distribution analysis of the input graphs i.e. features for node degree, numeric node and edge attributes and the corresponding resemblance attributes. We refer each of them as an attribute *i*. For symbolic node and edge attributes and the corresponding resemblance attributes, the unsupervised learning phase employs the modalities taken by this attribute and treat them as crisp intervals. The graph embedding phase employs these intervals for computing different bins of the respective fuzzy or crisp histograms.

5.1. Unsupervised learning phase

The off-line unsupervised learning phase of FMGE is outlined in Fig. 6. It learns a set of fuzzy intervals for encoding numeric information and crisp intervals for encoding symbolic information in graphs.

5.1.1. Input

The input to unsupervised learning phase of FMGE is the collection of m attributed graphs, given by $\{AG_1, AG_2, \ldots, AG_e, \ldots, AG_m\}$. Where the eth graph is denoted by $AG_e = (V_e, E_e, \mu^{V_e}, \mu^{E_e})$. A second input to this phase is for each feature the desired number of fuzzy intervals. This is referred by s_i for an attribute *i*.

5.1.2. Output

As output the unsupervised learning phase of FMGE produces s_i fuzzy overlapping trapezoidal intervals for an attribute i in input graphs (example in Fig. 8).

5.1.3. Description

The main steps for learning of fuzzy intervals for an attribute i are outlined in Fig. 7. The first step is the computation of crisp intervals from the list of values of attribute i for all the graphs in input collection of graphs. This is straightforward and is achieved by any standard data discretization technique. A survey of popular discretization techniques is presented by Liu et al. [31]. We propose to use equally spaced bins for obtaining an initial set of crisp intervals, as they avoid over-fitting and offers FMGE a better generalization capability to unseen graphs during graph embedding phase. Algorithm 5.1 outlines the pseudo-code for computing an initial set of crisp intervals for an attribute *i*. It uses a pseudo-call '*GetEqualSpacBin*' for getting an initial set of equally spaced bins. This pseudo-call refers to an appropriate data discretization function (available in underlying implementation platform). The initial set of equally spaced bins are used to construct a data structure, which stores the crisp intervals. This data structure is employed for computing s_i fuzzy intervals for attribute *i*. After computing the initial set of crisp intervals, in next step, these crisp intervals are arranged in an overlapping fashion to get fuzzy overlapping intervals. Normally trapezoidal, triangular and Gaussian arrangements are popular choices for fuzzy membership functions [30]. We propose to use the trapezoidal membership function, which is the generally used fuzzy membership function. It allows a range of instances to lie under full membership and assigns partial membership to the boundary instances. Fig. 8 outlines a trapezoidal interval defined over crisp intervals. A trapezoidal interval is defined by four points; as is given by points *a*,*b*,*c*,*d* in Fig. 8. It is very important to highlight here that the first fuzzy overlapping trapezoidal interval covers all values till $-\infty$ and the last fuzzy overlapping trapezoidal interval is limited by ∞. This makes sure that during the graph embedding phase every attribute instance falls under the range of fuzzy overlapping trapezoidal intervals and it further strengthens the generalization abilities of the method to unseen graphs. A fuzzy interval defined in trapezoidal fashion assigns a membership weight of 1 (full membership) between points *b* and *c*. The membership weight gradually approaches 0 as we move from *b* to *a* and from *c* to *d*. This trapezoidal behavior allows to assign full membership, partial membership and no membership to attribute instances. This is important to highlight here that the total membership assigned to an instance is always exactly equal to 1 i.e. either one full membership or two partial memberships are assigned to each attribute instance.

Algorithm 5.2 outlines the pseudo-code for computing fuzzy overlapping trapezoidal intervals from an initial set of crisp intervals for an attribute i in input collection of graphs. It first computes the initial set of crisp intervals using Algorithm 5.1 and then arranges them in an overlapping trapezoidal fashion for obtaining a set of fuzzy overlapping trapezoidal intervals for attribute *i*. The number of fuzzy intervals for attribute i depends upon the number of features desired for attribute i in FSMFV, and

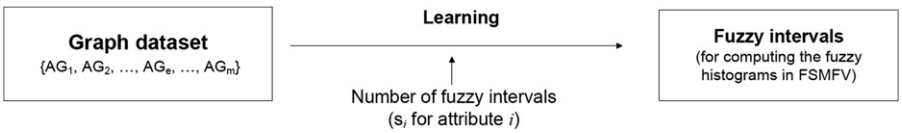

Fig. 6. The unsupervised learning phase of FMGE.

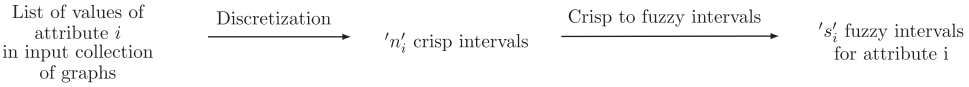

Fig. 7. Learning fuzzy intervals for an attribute *i*.

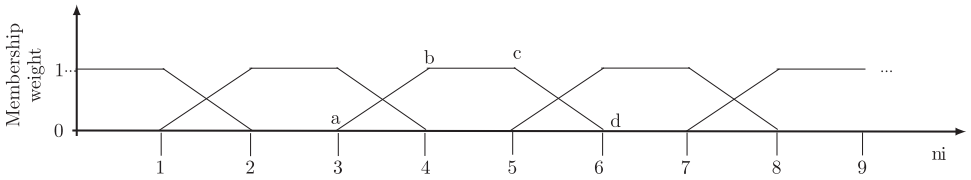

Fig. 8. Five fuzzy overlapping trapezoidal intervals (s_i) defined over nine equally spaced crisp intervals (n_i).

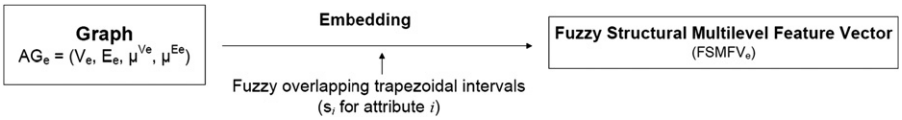

Fig. 9. The graph embedding phase of FMGE.

is controlled by parameter s_i which can either be manually specified, automatically learned by using an equal frequency based discretization or empirically learned and optimized on validation set. The number of crisp intervals n_i for desired number of fuzzy intervals s_i is computed by Eq. (5). For the sake of continuity we have used the terms fuzzy intervals, fuzzy overlapping intervals and fuzzy overlapping trapezoidal intervals interchangeably

$$n_i = 2 \times s_i - 1 \tag{5}$$

5.2. Graph embedding phase

The graph embedding phase of FMGE is outlined in Fig. 9. It employs the crisp intervals and fuzzy intervals (from unsupervised learning phase) to compute respective histograms for embedding an input attributed graph into a feature vector. This achieves the mapping of the input graph to appropriate points in a suitable vector space $\mathbb{R}^n$.

5.2.1. Input

The input to graph embedding phase of FMGE is an attributed graph AG_e to be embedded, as given by $AG_e = (V_e, E_e, \mu^{V_e}, \mu^{E_e})$. A second input to this phase is the s_i fuzzy overlapping trapezoidal intervals for the attribute i (from learning phase).

5.2.2. Output

The graph embedding phase of FMGE produces a feature vector $FSMFV_e$ for input graph AG_e. The length of this feature vector is uniform for all graphs in an input collection and is given by the following equation:

$$\text{Length of FSMFV} = 2 + \sum s_i + \sum c_i \tag{6}$$

where

- 2 refers to the features for graph order and graph size,
- $\sum s_i$ refers to the sum of number of bins in fuzzy interval encoded histograms for numeric information in graph (i.e. attribute i),
- $\sum c_i$ refers to the sum of number of bins in crisp interval encoded histograms for symbolic information in graph.

5.2.3. Description

The values of attribute i in input graph AG_e are fuzzified by employing its s_i fuzzy intervals (learned during unsupervised learning phase) and trapezoidal membership function. Mathematically, the membership function α defined over a trapezoidal interval x is given by Eq. (7). In Eq. (7), x refers to an instance of attribute i to be fuzzified and a,b,c,d refers to the limits of a trapezoidal fuzzy interval for attribute i. Function $\alpha(x)$ computes the degree of membership of an instance x with the trapezoidal interval defined by a,b,c,d. The possible memberships can be *a full membership* if x is between b and c, *a partial membership* if x is between a and b or is between c and d, or it can be *a no membership* if x is outside the interval a,b,c,d. We represent the fuzzy histogram of attribute i as h_i^{num}, which actually refers to the fuzzy histogram for node degrees (h^d in Fig. 3), the fuzzy histograms for numeric node attributes resemblance (h_d^{nr} and $h_1^{nr}, h_2^{nr}, \ldots, h_k^{nr}$ in Fig. 3), the fuzzy histograms for numeric and symbolic edge attributes resemblance ($h_1^{er}, h_2^{er}, \ldots, h_l^{er}$ in Fig. 3), the fuzzy histograms for numeric node attributes ($h_1^n, h_2^n, \ldots, h_k^n$ in Fig. 5) and the fuzzy histograms for numeric edge attributes ($h_1^e, h_2^e, \ldots, h_l^e$ in Fig. 5). The fuzzy histogram of an attribute i represents the embedding of attribute i for input graph AG_e. For an input graph AG_e, the fuzzy histogram h_i^{num} for attribute i is constructed by first computing the degree-of-memberships of all instances of attribute i in AG_e for the s_i fuzzy intervals. And then summing the memberships for each of the s_i fuzzy interval:

$$\alpha(x) = \begin{vmatrix} (x-a)/(b-a) & a \le x < b \\ 1 & b \le x \le c \\ (x-d)/(c-d) & c < x \le d \\ 0 & \text{otherwise} \end{vmatrix} \tag{7}$$

Each symbolic attribute j in input graph AG_k is encoded by a histogram of all its possible modalities (or labels). This histogram encodes the number of instances for each possible label of the symbolic attribute. We call this histogram as a crisp histogram (in-contrary to fuzzy histogram for numeric attributes). We call a crisp histogram for a symbolic attribute j as h_j^{sym}, which actually referrers to the crisp histograms for symbolic node attributes resemblance ($h_1^{nr}, h_2^{nr}, \ldots, h_k^{nr}$ in Fig. 3), the crisp histograms for symbolic node attributes ($h_1^n, h_2^n, \ldots, h_k^n$ in Fig. 5) and the crisp histograms for symbolic edge attributes ($h_1^e, h_2^e, \ldots, h_l^e$ in Fig. 5).

After constructing the fuzzy interval encoded histograms for each numeric attribute i and crisp histogram for each symbolic attribute j, the $FSMFV_e$ for input graph AG_e is constructed from the value of graph order, the value of graph size and fuzzy interval encoded histograms h_i^{num} of its node degree, numeric node and edge attributes appended by the crisp histograms h_j^{sym} for symbolic node and edge attributes. This gives an embedding of the input graph AG_e into a feature vector $FSMFV_e$. The histogram illustration of the graph

embedding phase and the resulting feature vectors for the example graphs are given in Fig. 4. *Two* fuzzy trapezoidal intervals are used for embedding node degree ([−∞,−∞,1,2] and [1,2,∞,∞]) whereas *three* fuzzy trapezoidal intervals are used for node attribute 'L' and edge attribute 'RL' ([−∞,−∞,0.5,1], [0.5, 1, 1.5, 2] and [1.5,2,∞,∞]) and *three* fuzzy intervals are used for embedding numeric resemblance attributes ([−∞,−∞,0.25,0.5], [0.25, 0.5, 0.75, 1.0], [0.75,1.0,∞,∞]). *Two* fuzzy intervals are used for embedding resemblance attribute for 'Angle' ([−∞,−∞,0,1.0], [0,1.0,∞,∞]). The symbolic edge attribute 'Angle' has two possible labels. Thus, in total the FSMFV for these graphs is comprised of 23 features (1 for graph order, 1 for graph size, 2 for node degree, 3 for node attribute 'L', 3 for resemblance on edge attribute 'RL', 2 for resemblance on edge attribute 'Angle', 3 for edge attribute 'RL', 2 for edge attribute 'Angle', 3 for resemblance on 'L', 3 for resemblance on 'nodeDegree').

Algorithm 5.1. GetInitCrispInterval ($listvaluesAttribute_i$, n_i).

comment: Computes equally spaced crisp intervals.
comment: Requires: List of values of an attribute i
comment: Requires: Number of crisp intervals for attribute i ($n_i \geq 2$)
comment: Returns: 'n_i' crisp intervals for attribute i
$equallySpacedBins \leftarrow$ GETEqualpacBin($listvaluesAttribute_i$, n_i)
$crispIntervals \leftarrow empty$
$st \leftarrow -\infty$
$en \leftarrow equallySpacedBins[1]$
$j \leftarrow 1$
repeat
  $crispIntervals[j].start \leftarrow st$
  $crispIntervals[j].end \leftarrow en$
  $st \leftarrow en$
  $en \leftarrow equallySpacedBins[j+1]$
  $j \leftarrow j+1$
until $j > n_i$
return (crispIntervals)

Algorithm 5.2. GetFuzzyOvlapTrapzInterval ($listvaluesAttribute_i$, s_i).

comment: Computes fuzzy intervals for an attribute i.
comment: Requires: List of values of an attribute i
comment: Requires: Number of fuzzy intervals for attribute i ($s_i \geq 2$)
comment: Returns: 's_i' fuzzy intervals for attribute i
$n_i \leftarrow 2 * s_i - 1$
$crispIntervals \leftarrow$ GetInitCrispInterval ($listvaluesAttribute_i$,n_i)
$fuzzyIntervals \leftarrow empty$
$fuzzyIntervals[1].a \leftarrow -\infty$
$fuzzyIntervals[1].b \leftarrow -\infty$
$fuzzyIntervals[1].c \leftarrow crispIntervals[1].end$
$fuzzyIntervals[1].d \leftarrow crispIntervals[2].end$
$j \leftarrow 1$
$jcrisp \leftarrow 0$
repeat
  $j \leftarrow j+1$
  $jcrisp \leftarrow jcrisp+2$
  $fuzzyIntervals[j].a \leftarrow fuzzyIntervals[j-1].c$
  $fuzzyIntervals[j].b \leftarrow fuzzyIntervals[j-1].d$
  if $(jcrisp+1 \geq n_i)$
    then
      $fuzzyIntervals[j].c \leftarrow \infty$
      $fuzzyIntervals[j].d \leftarrow \infty$
      break
  $fuzzyIntervals[j].c \leftarrow crispIntervals[jcrisp+1].end$
  $fuzzyIntervals[j].d \leftarrow crispIntervals[jcrisp+2].end$
until $j \geq s_i$
return (*fuzzyIntervals*)

6. Experimentations

The experimentation has been performed to confirm that FMGE and the subsequent classification and clustering in real valued vector space are not only applicable to different graph classification and clustering problems, but also that in certain cases it outperform the classical techniques for the latter. We have employed various standard datasets from the fields of graphics recognition, object recognition and document image analysis for addressing the problems of recognition of graphic primitives, object classification and query by example (QBE) and focused retrieval in graphic document images.

6.1. Graph datasets

We have employed graph datasets from IAM graph database repository [32] for document image analysis and graphics recognition experimentation and GEPR graph database [16] for object recognition experiments. For application of FMGE to hard problems of query by example (QBE) and focused retrieval in graphic document image repositories, we have constructed a graph repository by extracting graphs from images of architectural floor plans and electronic diagrams from SYSED image dataset [33] and have made it publicly available[1] for academia and research purposes. Tables 1–3 summarize characteristic properties of the graph datasets.

IAM Letter, GREC, Fingerprint and Mutagenicity graphs: Letter graph dataset is comprised of graphs extracted from drawings of 15 capital letters of Roman alphabet that consists of straight lines only i.e. A, E, F, H, I, K, L, M, N, T, V, W, X, Y and Z. The line drawings are converted into graphs by representing lines by undirected edges and ending points of lines by nodes. Edges are unlabeled while each node is labeled with a two-dimensional attribute giving its position relative to a reference coordinate system. In order to test classifiers under different conditions, distortions are applied on the graphs with three different levels of strength (low, medium and high).

GREC graph dataset is composed of graphs representing 22 symbols from architectural and electronic drawings. Graphs are extracted from denoised images by tracing the lines from end to end and detecting intersections and corners. Ending points, corners, intersections and circles are represented by nodes and labeled with a two-dimensional attribute giving their position. The nodes are connected by undirected edges which are labeled as line or arc. An additional attribute specifies the angle with respect to the horizontal direction or the diameter in case of arcs. For an adequately sized set, the graphs are distorted by translations and scalings of the graphs in a certain range, and random deletions and insertions of both nodes and edges.

Fingerprints are converted into graphs by representing ending points and bifurcation points of the skeletonized regions in filtered and denoised images by nodes. Undirected edges are inserted to link nodes that are directly connected through a ridge in the skeleton. Each node is labeled with a two-dimensional attribute giving its position. Edges are attributed with an angle denoting their orientation with respect to horizontal direction.

Mutagenicity is the ability of a chemical compound to cause mutations in DNA and is a property that hampers a compound to

[1] http://www.rfai.li.univ-tours.fr/PagesPerso/mmluqman/public/SESYD_graphs.zip

become a marketable drug [34]. The molecules are converted into attributed graphs in a straightforward manner by representing atoms as nodes and the covalent bonds as edges. Nodes are labeled with the number of the corresponding chemical symbol and edges by the valence of the linkage.

GEPR graphs: The GEPR graph database consists of graphs extracted from three large publicly available image databases, i.e. the Amsterdam Library of Object Images (ALOI), the Columbia Object Image Library (COIL) and the Object Databank by Carnegie-Mellon University (ODBK). The images are smoothed using a Gaussian filter, they are segmented using a Pyramidal segmentation algorithm and a Region Adjacency Graph (RAG) is constructed. The nodes of graph have the relative size and average color components of the corresponding regions as attributes and the edges are unlabeled [16].

SESYD graphs: The SESYD image dataset contains synthetically generated and degraded line drawing document images of architectural floor plans and electronic diagrams [33]. The query images simulate contextual noise which occurs in cropped regions of graphic document images. The images are represented by attributed graphs by using the work of Qureshi et al. [35]. The topological and geometric details about structure of graphic content are extracted and are represented by an attributed relational graph (ARG). In first step, the graphic content is vectorized and is represented by a set of primitives. In next step, these primitives become nodes and topological relations between them become arcs in ARG. As attributes nodes have *relative-length* (normalized between 0 and 1), *primitive-type* (Vector for filled regions of shape and Quadrilateral for thin regions) and a two-dimensional position attribute; whereas arcs of the graph have *connection-type* (one of L, X, T, P, S) and *relative-angle* (normalized between 0° and 90°) as attributes.

Table 1
IAM graph database. Number of attributes is given as a pair "numeric;symbolic".

Dataset	Size			Classes	Avg		Max		Attributes	
	Train	Valid	Test		$\|V\|$	$\|E\|$	$\|V\|$	$\|E\|$	V	E
Letter LOW	750	750	750	15	4.7	3.1	8	6	2;0	0;0
Letter MED	750	750	750	15	4.7	3.2	9	7	2;0	0;0
Letter HIGH	750	750	750	15	4.7	4.5	9	9	2;0	0;0
GREC	836	836	1628	22	11.5	12.2	25	30	2;1	1;1
Fingerprint	500	300	2000	4	5.4	4.4	26	25	2;0	1;0
Mutagenicity	500	500	1500	2	30.3	30.8	417	112	0;1	1;0

Table 2
GEPR graph database (number of attributes given as pair "numeric;symbolic").

Dataset	Size	Classes	Avg		Max		Attributes	
			$\|V\|$	$\|E\|$	$\|V\|$	$\|E\|$	V	E
ALOI	1800	25	18.37	17.25	134	156	2;0	0;0
COIL	1800	25	34.88	32.33	100	92	2;0	0;0
ODBK	1248	104	56.91	54.37	528	519	2;0	0;0

Table 3
SESYD graph dataset details.

	Image		Attributed graph	
Electronic diagrams	Backgrounds	8	Avg. order	212
	Models	21	Avg. size	363
	Symbols	9600	Node attribs.	4
			Edge attribs.	2
	Documents	800	Graphs	800
	Queries	1000	Graphs	1000
Architectural floor plans	Backgrounds	2	Avg. order	359
	Models	16	Avg. size	733
	Symbols	4216	Node attribs.	4
			Edge attribs.	2
	Documents	200	Graphs	200
	Queries	1000	Graphs	1000

6.2. Graph classification

The graph classification experimentation has been performed on IAM Letter, GREC, Fingerprint and Mutagenicity graphs (details in Table 1). The initial crisp intervals for graph classification experimentation are obtained by employing an equal spaced discretization technique. This permits FMGE to ignore the shape of distribution of an attribute i for learning s_i fuzzy intervals for it. The shape of this distribution is not important and keeping FMGE independent of it allows to learn on a set of graphs and to generalize to unseen graphs in diverse test sets.

Table 4 presents classification rates with an insight in the FSMFV construction process, for different datasets in IAM graph database. The classification rates have been obtained by employing a 1-NN classifier. We have used equal spaced discretization technique and three intervals for embedding each numeric attribute in graph. Also, average resemblance of the couple of edges is used for defining edge attributes resemblance for nodes. The results in Table 4 show that for all datasets the structural and elementary level features obtain better classification rates than graph level features. This is partly because of the fact that the number of features for each of these categories is more than the number of graph level features, and partly because of the fact that structure and attribute details in a graph provide critically important discriminatory information about the graph. The feature vector construction validation experimentation clearly shows that FMGE gets most of its discriminatory power from structural and elementary features.

The graph classification experimentation was repeated for 2–30 intervals for each attribute i in graph and the number of intervals maximizing the classification rate on validation set were used for evaluating the performance on test set. The length of

Table 4
Validating the construction of Fuzzy Structural Multilevel Feature Vector of FMGE.

Dataset	Recognition rate [1-NN classifier]			
	Graph level features	Structural level features	Elementary level features	Graph + Structural + Elementary features
Letter LOW	27.3	92.5	81.3	94.1
Letter MED	22.4	63.6	48.7	73.6
Letter HIGH	21.1	46.1	47.3	56.8
GREC	52.5	93.9	96.6	97.3
Fingerprint	47.6	64.5	75.9	73.7
Mutagenicity	58.4	65.8	68.5	68.5

Table 5
Experimental results for graph classification on IAM graph database repository.

Dataset	**Graph edit distance based ref. system** [*k*-NN classifier]	**Dissimilarity based embed.** Bunke et al. [4] [SVM classifier]	**FMGE** resemblance:AVG [k-NN classifier]	**FMGE** resemblance:STD [k-NN classifier]	**FMGE** rresemblance:AVG [SVM classifier]	**FMGE** resemblance:STD [SVM classifier]
Letter LOW	99.3	99.3	97.1	97.1	98.2	98.2
Letter MED	94.4	94.9	75.7	75.7	83.1	83.1
Letter HIGH	89.1	92.9	66.5	66.5	70.0	70.0
GREC	82.2	92.4	97.5	97.5	99.4	99.4
Fingerprint	79.1		79.9	79.7	87.6	87.1
Mutagenicity	66.9		69.1	68.2	76.5	76.5

obtained feature vectors varied from 50 to 150. Table 5 presents the classification results for IAM graphs and compares them with a reference system and the best state-of-the-art results from dissimilarity based graph embedding technique of Bunke et al. [4]. We employed a nearest neighbor classifier with Euclidean distance for comparing the graph classification results of FMGE with reference system. We optimized the parameter k (over 1 to 5 nearest neighbors) along-with the number of intervals, on validation set for every dataset. For evaluating the performance on test set we used the parameter values which maximized the classification rate on validation set. Our choice of the nearest neighbor classifier (k-NN) with Euclidean distance, is motivated from the reference system on IAM graphs. The reference system employs a graph edit distance based nearest neighbor classifier, because of the fact that there is a lack of general classification algorithms that can be applied to graphs. One of the few classifiers directly applicable to arbitrary graphs is the k-nearest-neighbor classifier (k-NN). Given a labeled set of training graphs, an unknown graph is assigned to the class that occurs most frequently among the k nearest graphs (in terms of edit distance) from the training set. The decision boundary of this classifier is a piecewise linear function which makes it very flexible. Contrary to our choice of nearest neighbor classifier, Ref. [4] has employed an SVM classifier. To compare our results with [4] we replicated their classifier configuration for our experimentation. We employed an SVM classifier with RBF-kernel. The SVM parameters were optimized on validation set for every dataset and the values maximizing the classification rate on validation set were used for evaluating the performance on test set. Experimental results are presented for two setups of FMGE. This refers to the way in which edge attributes resemblance is computed for nodes. Columns 3 and 5 show the classification results, when the average resemblance (AVG) of the couple of edges is used for defining edge attributes resemblance for nodes. Columns 4 and 6 show the classification results, when the standard deviation of the resemblance (STD) of the couple of edges is used for defining edge attributes resemblance for nodes. Both setups give the same classification rates except for fingerprint and mutagenicity graphs. This illustrates that for IAM graphs the resemblance between the couple of edges for the nodes remains stable in both setups.

IAM graphs are comprised of graphs with only two numeric node attributes. These attributes give (x,y) position of the corresponding primitive in underlying image. Only GREC graphs have an additional symbolic attribute on nodes. GREC, Fingerprint and Mutagenicity graphs have attributes on edges. The structural and elementary level information that FMGE extracts, is directly or indirectly linked to the node and edge attributes. In our opinion, the attributes on graphs in IAM dataset are not enough for FMGE for extracting a lot of discriminatory information. However, even then the results are comparable to the best state-of-the-art results. The attributes on node and edges of GREC and Mutagenicity graphs permits FMGE to obtain the highest classification results for these graphs. For Letter LOW and Fingerprint graphs FMGE results are comparable to state-of-the-art results and reference system. For letter MED and letter HIGH graphs FMGE results are not very good because of very high level of distortions in graphs, which totally change the topology of graphs and the underlying letter becomes unrecognizable even for a human observer [34]. The reason that Bunke et al. [4] have better results on some of the datasets is actually direct outcome of manual selection of the prototype graphs for their method. As FMGE extracts most of the information from node and edge attributes, so more information on the nodes and edges of graphs will result into extraction and eventually embedding of more discriminatory information by FMGE and the classification performance will be improved.

6.3. Graph clustering

The graph clustering experimentation has been performed on IAM Letter, GREC, Fingerprint and Mutagenicity graphs (details Table 1) and the GEPR graphs (details Table 2). Clustering is generally performed in an unsupervised manner and normally for clustering tasks no separate learning set is available. This scenario is very well simulated by our experimentation. During a first pass on the graph dataset to be clustered FMGE learned fuzzy intervals for various numeric attributes in graphs and then during second pass on the same graph dataset it employed these fuzzy intervals for graph embedding. A more sophisticated and advanced discretization technique has been employed for obtaining the initial set of crisp intervals for graph clustering experimentation. This technique is originally proposed by [36] for discretization of continuous data and is based on use of Akaike Information Criterion (AIC). It starts with an initial histogram of data and finds optimal number of bins for underlying data by iteratively merging the adjacent bins using an AIC-based cost function. This gives an optimal set of crisp intervals for underlying data which could be safely termed as equal frequency intervals. The resulting fuzzy intervals, which are calculated from the spread of the respective attribute's values, are true representative of the shape of distribution and are very interesting for FMGE.

IAM Letter, GREC, Fingerprint and Mutagenicity graphs: The graphs in training, validation and test sets of the respective graph datasets were merged to construct datasets for graph clustering experimentation. We have employed the well known and popular k-means clustering paradigm with Euclidean distance and random non-deterministic initialization for our experiments. Fig. 10 presents the Silhouette metric for analyzing the quality of clustering. The Silhouette metric [37] is a standard cluster fitness validation metric. It measures the standardized difference between separation of clusters and the average spread of clusters. The average Silhouette width over all clusters is a value in the range $[-1, 1]$ – the closer this value is to 1 the better is the cluster quality. The curves in Fig. 10 present the Silhouette metric for 2–25 clusters. As an example to understand these results, consider

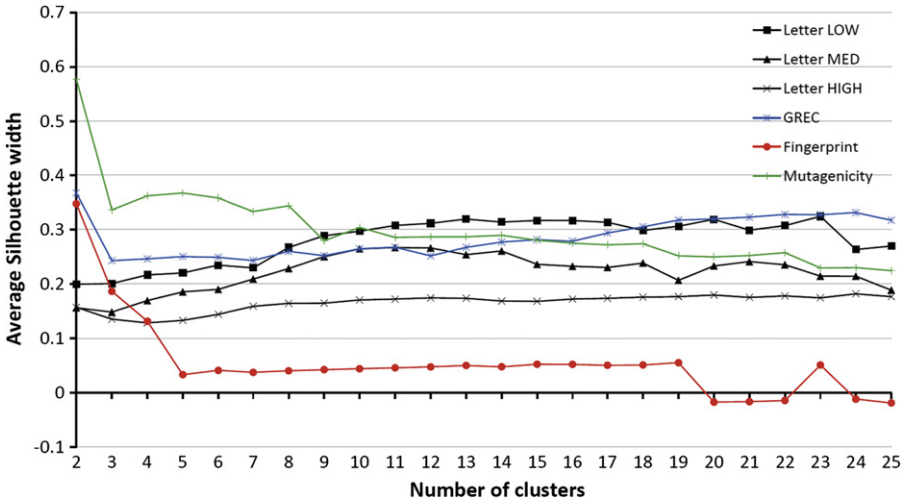

Fig. 10. Number of clusters versus average Silhouette width for *k*-means clustering.

Table 6
Quality of *k*-means clustering in FSMFV feature space.

Dataset	FSMFV feature vector space correctly clustered graphs (%)
Letter LOW	89
Letter MED	60
Letter HIGH	41
GREC	82
Fingerprint	57
Mutagenicity	82

Table 7
Performance indexes obtained for GEPR graph datasets.

Dataset	Performance index
ALOI	0.379
COIL	0.377
ODBK	0.355

that the mutagenicity graphs originally have two classes (Table 1) and in Fig. 10 the Silhouette metric for Mutagenicity graphs for two classes is 0.58. The obtained Silhouette metric for actual number of clusters in the datasets, is always near or superior to 0.2. The latter is a generally used threshold for compact, better separable and good structured clusters.

Together with Silhouette metric we have evaluated the quality of FMGE graph clustering by the percentage of correctly clustered graphs. Table 6 presents the results. We have used the actual number of classes in dataset (Table 1) as the number of clusters to be found by *k*-means clustering. The presented results show the quality of *k*-means clustering in the FSMFV feature space, which is calculated as the ratio of correctly clustered graphs to total number of graphs in dataset. For example, a 89% clustering quality for Letter LOW graphs means that 89% of the graphs are assigned to correct clusters. These results demonstrate that the proposed methodology of first embedding graphs into feature vectors and then applying a clustering technique has the significant potential to turn an impossible operation in original graph space into a realizable operation with an acceptable accuracy in resulting feature vector space. The results demonstrate graph clustering abilities of FMGE and highlight its unsupervised learning capabilities. The percentage of correctly clustered graphs for Letter LOW graphs, GREC graphs and Mutagenecity graphs is very good ($\geq 82\%$). However, for Letter MED graphs and Fingerprint graphs its not so good. The low clustering quality for letter HIGH graphs is because of lack of discreminatory attributes on nodes and edges and the fact that high level of distortions entirely changes the topology of the graph. This effects the FSMFV of the graphs in this dataset and results into too much overlap between clusters.

GEPR graphs: The graphs were embedded into feature vectors by FMGE and the scripts in GEPR graphs repository were used to evaluate the performance index of the clustering for obtained FSMFVs. These scripts employ a C index based clustering validation index to evaluate the separation between classes [16]. The smaller the value, the better is the separation of the classes; the index value is in the interval [0, 1] reaching 0 in the ideal case in which all the inter-class distances are smaller than all the intra-class distances [16]. Table 7 provides the cluster validation index for GEPR graphs. GEPR graphs have only two numeric attributes on nodes and no attributes on edges. The lack of attributes on edges and less number of attributes on nodes is the main reason for adequate clustering quality. For graphs extracted from object images a very important attribute that can be extracted from images is the relations between neighboring parts of the object. In a region adjacency graph this could be represented by the relationship between adjacent regions of the image. The addition of more attributes on nodes and edges of the graphs will enable FMGE to extract and eventually embed more information on graphs. This will result into more discriminant feature vectors and higher quality clustering.

6.4. Graph retrieval

The graph retrieval experimentation on SESYD graphs presents an application of FMGE to the hard problems of query by example (QBE) and focused retrieval in graphic document image repositories. The growing size of document image repositories has resulted in an increasing demand from users to have an efficient browsing mechanism for graphic content. The format of these documents restricts the use of classical keyword based indexing mechanisms and a very interesting research problem is to investigate into mechanisms of automatically indexing the content of graphic document images. The fact that graph based representations are widely used for graphic document images [4,7,38,39], turns content spotting for graphic document into the problems of graph retrieval and subgraph spotting. The system outlined in this section is reproduced from our recent work in [40]. We achieve graph retrieval and subgraph spotting through

explicit graph embedding. A graph repository is automatically indexed during an off-line learning phase; where we (i) break the graphs into 2-node subgraphs (a.k.a. cliques of order 2), which are primitive building-blocks of a graph, (ii) embed the 2-node subgraphs into feature vectors by employing our recently proposed explicit graph embedding technique, (iii) cluster the feature vectors in classes by employing a classic agglomerative clustering technique, (iv) build an index for the graph repository and (v) learn a Bayesian network classifier. The subgraph spotting is achieved during the on-line querying phase; where we (i) break the query graph into 2-node subgraphs, (ii) embed them into feature vectors, (iii) employ the Bayesian network classifier for classifying the query 2-node subgraphs and (iv) retrieve the respective graphs by looking-up in the index of the graph repository. The graphs containing all query 2-node subgraphs form the set of result graphs (AG_{result}) for the query (AG_{query}). For spotting the query graph AG_{query} in a result graph AG_{result}, we employ the adjacency matrix of graph AG_{result} along with a score function. The adjacency matrix of graph AG_{result} has a value of "0" if there is no edge between "$node_1$" and "$node_2$" in the original graph AG_{result}, a value of "1" if there is an edge between "$node_1$" and "$node_2$" in the original graph AG_{result} and a value of "2" if one of the query 2-node subgraphs is classified (by Bayesian network) as belonging to the cluster of this 2-node subgraph (which compose of edge between "$node_1$" and "$node_2$"). The query graph AG_{query} is finally spotted in the result graph AG_{result}, by looking up in the neighborhood of each 2-node subgraph of AG_{result} which is in the result i.e. each "$AG_{result}(i,j)=2$" in the adjacency matrix of result graph AG_{result}. We explore "o" connected neighbors of each "$AG_{result}(i,j)=2$". The parameter "o" is proportional to the graph order of query graph $AG_{query}(|V_{query}|)$. We compute a score for each "$AG_{result}(i,j)=2$" using Eq. (8). The computed score of "$AG_{result}(i,j)=2$" also gives a confidence value for subgraph spotting of query graph AG_{query} in result graph AG_{result}:

$$score = \sum_{z=0}^{2} \left(z \times \frac{|z|}{o} \right) \tag{8}$$

where

- z is a value in the adjacency matrix (either 0,1 or 2),
- $|z|$ is number of times the value z occurs in neighborhood and
- o is number of connected neighbors that are looked-up.

In our experimentation, the electronic diagrams and architectural floor plans graphs (Table 3), were treated independently of each other for learning and querying phases of the system. During the automatic indexation phase of graph repositories, a total of 516,714 2-node subgraphs were extracted for electronic diagrams and 305,824 2-node subgraphs for architectural floor plans. These 2-node subgraphs were clustered into 455 classes for electronic diagrams and 211 classes for architectural floor plans. The indexation of electronic diagrams graph repository took 17 h on a standard PC with 2 GB of RAM and the subgraph spotting was performed in real-time. Fig. 11 presents the precision and recall plot for the central tendency of the queries of respective attributed graph datasets. It is important to highlight here that the results in Fig. 11 shows the retrieval performance of the system. The focused retrieval performance of the system was evaluated manually for a subset of queries. The graph retrieval experimentation results clearly demonstrate that FMGE is capable of automatically indexing graph repositories and is capable of maintaining a high precision rate for sufficiently large graph repositories. The performance of the system strongly depends upon the graph extraction phase. In our case, we have used a graph extraction method, which is based on a prior vectorization phase. The employed vectorization technique approximates the circles and arcs by a set of straight lines. A small distortion in the shape of underlying content results into a change in topology of the graph and it effects the graph embedding phase of the system. Because of the latter and the fact that architectural floor plans are comprised of many symbols with arcs and circles, the system has a low performance for this dataset. However, generally (for the given datasets) the system is able to maintain a high precision value for the series of different recall values (Fig. 11). This provides a very interesting solution for the automatic indexation of graph repositories for retrieval and (specially for) browsing of content in graph representation of graphic document image repositories. Our proposed method does not rely on any domain specific details and offers a very general solution to the problem of subgraph spotting; indeed equally applicable to a wide range of application domains where the use of graph as a data structure is mandatory. The system does not impose any strict restrictions on the size of query subgraph. The fact that the system does not require any labeled training-set enables its less expensive and fast deployment to a wide range of application domains.

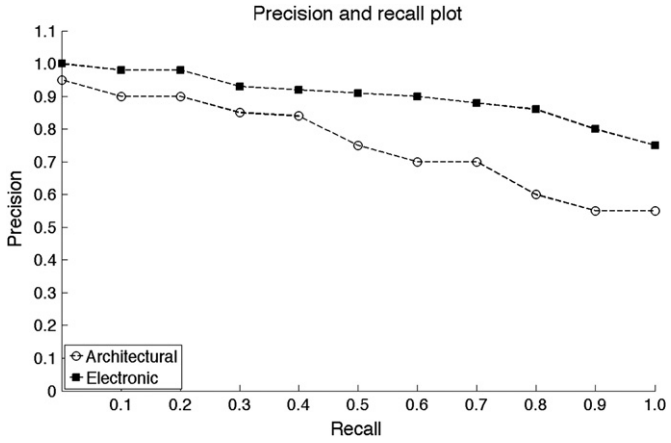

Fig. 11. Precision and recall plot.

7. Discussion

We have outlined to use two basic discretization techniques in this paper. However FMGE is fully capable of employing sophisticated state-of-the-art discretization methods. Also, our proposed framework employs trapezoidal membership function from fuzzy logic but FMGE is fully capable of utilizing any of the available membership functions from fuzzy logic. In light of domain knowledge, appropriate choices could be made for discretization technique and fuzzy membership function.

An important parameter of FMGE is the number of fuzzy intervals to be associated to each attribute. The unsupervised learning phase of FMGE learns s_i fuzzy intervals for an attribute i in input collection of graphs. The parameter s_i for attribute i is independent of other attributes. It is also important to highlight here that this parameter is not necessarily same for all attributes. This parameter can be learned and tested on a validation dataset, if one is available. This is demonstrated by our experimentation on graph classification. If no validation dataset is available the number of fuzzy intervals s_i has to be specified manually. A third possibility is that if the training set can be considered to be a true representative of the test set, an equal frequency discretization technique could be used for automatically finding the appropriate number of fuzzy intervals. This is demonstrated by our graph clustering experimentation. As a general rule of thumb, using three fuzzy intervals for the attributes is a safe choice. However, a

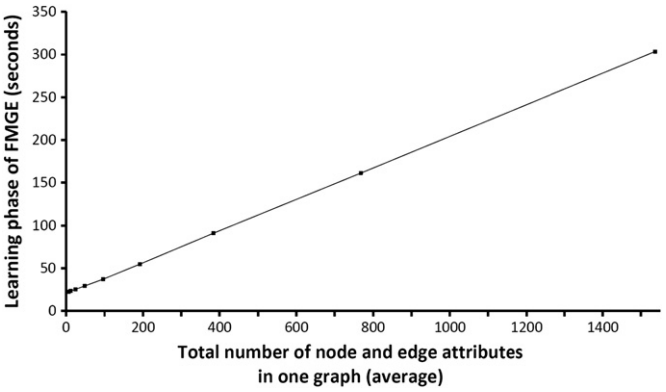

Fig. 12. Time complexity of unsupervised learning phase of FMGE for synthetically generated graphs from IAM GREC dataset. All parameters except number of attributes are kept constant.

more intelligent choice could be made in the light of domain knowledge.

For embedding edge attributes resemblance for nodes (structural level information), we have outlined the use of a measure of central tendency (i.e. mean) and the spread of attribute's resemblance (i.e. the standard deviation) on the attribute resemblance of all couple of edges for a node.

We have used a simple non-parametric classifier for graph classification experimentation and a basic clustering technique for graph clustering experimentation for demonstration purposes. However, all sophisticated classifiers and clustering methods are fully applicable to the resulting feature vectors of FMGE.

7.1. Complexity

The graph embedding phase of FMGE requires only a fraction of a second and can be performed in real-time on standard PC. The unsupervised learning phase of FMGE can be computational intensive depending on the size of dataset and the graphs. However, this phase is performed off-line and has linear time complexity. This is illustrated in Fig. 12, where we present time complexity of the learning phase of FMGE.

8. Conclusion and perspectives

We have presented a method of explicit graph embedding, with an aim to bridge the gap between structural and statistical approaches of pattern recognition. Our work proposes a straightforward, simple and computational efficient solution for facilitating the use of graph based powerful representations together with learning and computational strengths of state-of-the-art machine learning, classification and clustering. The proposed method exploits multilevel analysis of graph for embedding it into a feature vector. The feature vector contains graph level features (graph order and graph size), along-with structural level features (node degree, node attributes resemblance for edges and edge attributes resemblance for nodes) and elementary level features (node attributes and edge attributes). We have minimized the information loss, while mapping from continuous graph space to discrete vector space, by employing fuzzy overlapping trapezoidal intervals. These intervals are learned during an unsupervised learning phase. The intervals are employed during graph embedding phase for constructing fuzzy interval encoded histograms for structural and elementary level features. The method is linear to order and size of graphs and the number of node and edge attributes. The method could be deployed in an unsupervised fashion for graph clustering problem even if no separate learning set is available as it has built-in capability to implicitly learn from the input collection of graphs. Its unsupervised learning capabilities also allows it to generalize to unseen graphs i.e. to be deployed in a supervised fashion where it learns on a graph dataset and embeds unseen graphs. The experimental results are encouraging and demonstrate the applicability of our method to graph clustering and classification problems.

Our current research findings are encouraging and ongoing research is in progress to take this work forward for improving the quality of embedding achieved by FMGE. Important directions of future research include application of dimensionality reduction and feature selection methods to the resulting feature vectors, the detection of outliers for cleaning leaning set and exploiting graph paths and Morgan's index [41] for including more information on the topology of graph.

Acknowledgments

This work was partially supported by the Spanish projects TIN2008-04998, TIN2009-14633-C03-03 & CSD2007-00018 and partially by the PhD grant PD-2007- 1/Overseas/FR/HEC/222 from Higher Education Commission of Pakistan.

References

[1] J. De Sa, Pattern Recognition: Concepts, Methods, and Applications, Springer Verlag, 2001.
[2] M. Friedman, A. Kandel, Introduction to Pattern Recognition, World Scientific, 1999.
[3] H. Bunke, S. Gunter, X. Jiang, Towards bridging the gap between statistical and structural pattern recognition: two new concepts in graph matching, in: International Conference on Advances in Pattern Recognition, Springer, 2001, pp. 1–11.
[4] H. Bunke, K. Riesen, Recent advances in graph-based pattern recognition with applications in document analysis, Pattern Recognition 44 (5) (2011) 1057–1067.
[5] K. Riesen, H. Bunke, Graph classification based on vector space embedding, International Journal of Pattern Recognition and Artificial Intelligence 23 (6) (2009) 1053–1081.
[6] H. Bunke, C. Irniger, M. Neuhaus, Graph matching – challenges and potential solutions, in: International Conference on Image Analysis and Processing, 2005, pp. 1–10.
[7] D. Conte, P. Foggia, C. Sansone, M. Vento, Thirty years of graph matching in pattern recognition, International Journal of Pattern Recognition and Artificial Intelligence 18 (3) (2004) 265–298.
[8] A. Shokoufandeh, D. Macrini, S. Dickinson, K. Siddiqi, S. Zucker, Indexing hierarchical structures using graph spectra, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (7) (2005) 1125–1140.
[9] M. Ferrer, E. Valveny, F. Serratosa, K. Riesen, H. Bunke, Generalized median graph computation by means of graph embedding in vector spaces, Pattern Recognition 43 (4) (2010) 1642–1655.
[10] R. Duda, P. Hart, D. Stork, Pattern Classification, vol. 2, Wiley Interscience, 2000.
[11] V. Roth, J. Laub, M. Kawanabe, J. Buhmann, Optimal cluster preserving embedding of nonmetric proximity data, IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (12) (2003) 1540–1551.
[12] T. Chen, Q. Yang, X. Tang, Directed graph embedding, in: International Joint Conference on Artificial Intelligence, 2007, pp. 2707–2712.
[13] E.L. Nathan Linial, Y. Rabinovich, The geometry of graphs and some of its algorithmic applications, Combinatorica 15 (2) (1995) 215–245.
[14] B. Shaw, T. Jebara, Structure preserving embedding, in: International Conference on Machine Learning, 2009, pp. 1–8.
[15] G. Lee, A. Madabhushi, Semi-supervised graph embedding scheme with active learning (SSGEAL): classifying high dimensional biomedical data, in: Pattern Recognition in Bioinformatics, Lecture Notes in Computer Science, vol. 6282, Springer, 2010, pp. 207–218.
[16] P. Foggia, M. Vento, Graph embedding for pattern recognition, in: D. Ünay, Z. Çataltepe, S. Aksoy (Eds.), Recognizing Patterns in Signals, Speech, Images and Videos, Lecture Notes in Computer Science, vol. 6388, Springer, 2010, pp. 75–82.
[17] K. Riesen, H. Bunke, Graph Classification and Clustering Based on Vector Space Embedding, World Scientific Publishing Co., Inc., 2010.
[18] R.C. Wilson, E.R. Hancock, B. Luo, Pattern vectors from algebraic graph theory, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (7) (2005) 1112–1124.

[19] M. Luqman, J. Lladós, J.-Y. Ramel, T. Brouard, A fuzzy-interval based approach for explicit graph embedding, in: Recognizing Patterns in Signals, Speech, Images and Videos, vol. 6388, 2010, pp. 93–98.
[20] S. Kramer, luc de Raedt, Feature construction with version spaces for biochemical application, in: 18th International Conference on Machine Learning, 2001, pp. 258–265.
[21] A. Inokuchi, T. Washio, H. Motoda, An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data, Lecture Notes in Computer Science, vol. 1910, 2000, pp. 13–23.
[22] N. Sidère, P. Héroux, J.-Y. Ramel, Vector representation of graphs: application to the classification of symbols and Letters, in: International Conference on Document Analysis and Recognition, 2009, pp. 681–685.
[23] B. Luo, Spectral embedding of graphs, Pattern Recognition 36 (10) (2003) 2213–2230.
[24] X. Bai, H. Yu, E. Hancock, Graph matching using spectral embedding and alignment, in: International Conference on Advances in Pattern Recognition, vol. 3, IEEE, 2004, pp. 398–401.
[25] a. Torsello, E. Hancock, Graph embedding using tree edit-union, Pattern Recognition 40 (5) (2007) 1393–1405.
[26] D. Emms, R. Wilson, E. Hancock, Graph embedding using quantum commute times, in: International Conference on Graph-based Representations in Pattern Recognition, Springer-Verlag, 2007, pp. 371–382.
[27] H. Chen, H. Chang, T. Liu, Local discriminant embedding and its variants, in: Computer Vision and Pattern Recognition, IEEE, 2005, pp. 846–853.
[28] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Transactions on Pattern Analysis and Machine Learning 22 (8) (2002) 888–905.
[29] P. Ren, R. Wilson, E. Hancock, Graph Characterization via Ihara coefficients, IEEE Transactions on Neural Networks 22 (2011) 233–245.
[30] H. Ishibuchi, T. Yamamoto, Deriving fuzzy discretization from interval discretization, in: International Conference on Fuzzy Systems, IEEE, 2003, pp. 749–754.
[31] H. Liu, F. Hussain, C. Tan, M. Dash, Discretization: an enabling technique, Data Mining and Knowledge (2002) 393–423.
[32] K. Riesen, H. Bunke, IAM graph database repository for graph based pattern recognition and machine learning, in: Structural, Syntactic, and Statistical Pattern Recognition, Springer, 2010, pp. 287–297.
[33] M. Delalandre, E. Valveny, T. Pridmore, D. Karatzas, Generation of synthetic documents for performance evaluation of symbol recognition & spotting systems, International Journal on Document Analysis and Recognition (2010) 1–21.
[34] K. Riesen, Classification and Clustering of Vector Space Embedded Graphs, Ph.D. Thesis, University of Bern, January 2010.
[35] R. J. Qureshi, J.-Y. Ramel, H. Cardot, P. Mukherji, Combination of Symbolic and Statistical Features for Symbols Recognition, in: International Conference on Signal Processing, Communications and Networking, 2007, pp. 477–482.
[36] O. Colot, P. Courtellemont, A. El-Matouat, Information criteria and abrupt changes in probability laws, in: Signal Processing VII: Theories and Applications, 1994, pp. 1855–1858.
[37] L. Kaufman, P.J. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis, John Wiley & Sons, Ltd., 1990.
[38] J. Lladós, E. Valveny, G. Sánchez, E. Martí, Symbol Recognition: Current Advances and Perspectives, in: Graphics Recognition Algorithms and Applications, vol. 2390, (2002) pp. 104–128.
[39] K. Tombre, S. Tabbone, P. Dosch, Musings on symbol recognition, in: Graphics Recognition. Ten Years Review and Future Perspectives, (2006) pp. 23–34.
[40] M. M. Luqman, J.-Y. Ramel, J. Lladós, T. Brouard, Subgraph Spotting through Explicit Graph Embedding : An Application to Content Spotting in Graphic Document Images, in: Eleventh International Conference on Document Analysis and Recognition (ICDAR), (2011) pp. 870–874.
[41] H.L. Morgan, The generation of a unique machine description for chemical structures - a technique developed at chemical abstracts service, Journal of Chemical Documentation 5 (2) (1965) 107–113.

Muhammad Muzzamil Luqman received his Ph.D. degree in Computer Science from François-Rabelais University of Tours, France and Autonoma University of Barcelona, Spain in 2012. He got his masters from François-Rabelais University of Tours, France in 2008. In 2004 he was awarded gold medal and academic roll of honor by Government College University Lahore, Pakistan for achieving distinction in Bachelors of Computer Science (honors). Currently Luqman is a teaching and research assistant at François-Rabelais University of Tours, France. His main research interests include Structural Pattern Recognition, Machine Learning, Document Image Analysis/ Recognition and Graphics Recognition.

Jean-Yves Ramel received his Ph.D. degree in Computer Sciences in 1996 from the National Institute of Applied Sciences of Lyon (INSA Lyon France). After being an Assistant Professor at the Industrial Engineering Department of the National Institute of Applied Sciences of Lyon from 1998 to 2002; Jean-Yves Ramel is currently a Full Professor at the Computer Science Department of PolytechTours (French engineering school). He is also a staff researcher of the Computer Science Laboratory of Tours in the Image Analysis and Pattern Recognition Group (EA 2101). His current research fields are image analysis, document image indexation and structural pattern recognition. He has been the head of a number of Image Analysis and Indexation R+D projects and published several papers in national and international conferences and journals. Jean-Yves RAMEL is an active member of the Pattern Recognition for Image Understanding French Association (AFRIF), a member society of the IAPR (TC-10, TC-11 and TC-15) and also a PC member of a number of international conferences. His team has developed several open source software dealing with Document Image Analysis. Jean-Yves RAMEL was the recipient of a Google Digital Humanities Award in 2010 and has also experience in technological transfer and patent registration.

Josep Lladós received the degree in Computer Sciences in 1991 from the Universitat Politècnica de Catalunya and the Ph.D. degree in Computer Sciences in 1997 from the Universitat Autònoma de Barcelona (Spain) and the Université Paris 8 (France). Currently he is an Associate Professor at the Computer Science Department of the Universitat Autònoma de Barcelona and is a staff researcher of the Computer Vision Center, where he is also the director. He is the head of the Pattern Recognition and Document Analysis Group (2005SGR-00472). His current research fields are document analysis, graphics recognition and structural and syntactic pattern recognition. He has been the head of a number of Computer Vision R+D projects and published several papers in national and international conferences and journals. J. Lladós is an active member of the Image Analysis and Pattern Recognition Spanish Association (AERFAI), a member society of the IAPR. He is currently the chairman of the IAPR-ILC (Industrial Liaison Committee). Formerly he served as chairman of the IAPR TC-10, the Technical Committee on Graphics Recognition, and also he is a member of the IAPR TC-11 (reading Systems) and IAPR TC-15 (Graph based Representations). He serves on the Editorial Board of the ELCVIA (Electronic Letters on Computer Vision and Image Analysis) and the IJDAR (International Journal in Document Analysis and Recognition), and also a PC member of a number of international conferences. He was the recipient of the IAPR-ICDAR Young Investigator Award in 2007. Josep Lladós has also experience in technological transfer and in 2002 he created the company ICAR Vision Systems, a spin-off of the Computer Vision Center working on Document Image Analysis, after win the entrepreneurs award from the Catalonia Government on business projects on Information Society Technologies in 2000.