Pattern Recognition 43 (2010) 1642–1655

Contents lists available at ScienceDirect

Pattern Recognition

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

Generalized median graph computation by means of graph embedding in vector spaces

M. Ferrer [a,*], E. Valveny [a], F. Serratosa [b], K. Riesen [c], H. Bunke [c]

[a] Centre de Visió per Computador, Departament de Ciències de la Computació, Universitat Autònoma de Barcelona, 08193 Bellaterra, Spain
[b] Departament d'Enginyeria Informàtica i Matemàtiques, Universitat Rovira i Virgili, 43007 Tarragona, Spain
[c] Institute of Computer Science and Applied Mathematics, University of Bern, Neubrückstrasse, 10, CH-3012 Bern, Switzerland

ARTICLE INFO

Article history:
Received 7 July 2008
Received in revised form
8 October 2009
Accepted 16 October 2009

Keywords:
Graph matching
Weighted mean of graphs
Median graph
Graph embedding
Vector spaces

ABSTRACT

The median graph has been presented as a useful tool to represent a set of graphs. Nevertheless its computation is very complex and the existing algorithms are restricted to use limited amount of data. In this paper we propose a new approach for the computation of the median graph based on graph embedding. Graphs are embedded into a vector space and the median is computed in the vector domain. We have designed a procedure based on the weighted mean of a pair of graphs to go from the vector domain back to the graph domain in order to obtain a final approximation of the median graph. Experiments on three different databases containing large graphs show that we succeed to compute good approximations of the median graph. We have also applied the median graph to perform some basic classification tasks achieving reasonable good results. These experiments on real data open the door to the application of the median graph to a number of more complex machine learning algorithms where a representative of a set of graphs is needed.

1. Introduction

One of the basic objectives of pattern recognition is to develop systems for the analysis or classification of input patterns. A first issue to be addressed in any pattern recognition system is how to represent these input patterns. Feature vectors are one of the most common and widely used data representations. That is, a set of relevant properties, or features, are computed for each pattern and arranged in a vector. The main advantage of this representation is that many operations needed in machine learning can be easily executed on vectors, and a large number of algorithms for pattern analysis and classification become immediately available. However, a disadvantage of feature vectors arises from their simple structure and the fact that they have the same length and structure regardless of the complexity of the object. For a general introduction to pattern recognition and classification the reader is referred to [9,11].

Therefore, for the representation of complex patterns, graphs appear as an appealing alternative. One of the main advantages of graphs over feature vectors is that graphs can explicitly model the relations between the different parts of the object, whereas feature vectors are only able to describe an object as an aggregation of numerical properties. In addition, graphs permit to associate any kind of label (not only numbers) to both edges and nodes. Furthermore, the dimensionality of graphs, that is, the number of nodes and edges, can be different for every object. Thus, the more complex an object, the larger can be the number of nodes and edges used to represent it. An extensive work comparing the representational power of both feature vectors and graphs under the context of web content mining has been presented in [28].

In spite of the strong mathematical foundation underlying graphs and their high power of representation, working with graphs is harder and more challenging than working with feature vectors. They have two main drawbacks when they are intended to be applied in conjunction with classical learning algorithms. On one hand, the computational complexity of the algorithms related to graphs is usually high. For instance, the simple task of comparing two graphs, which is known as graph matching, becomes exponential in the size of graphs [5]. In addition, basic operations that are needed in many pattern recognition methods and that might appear quite simple in the vector domain, such as computing the sum or the mean, turn very difficult or even impossible in the graph domain. Although the existence of labels on both nodes and edges may simplify many problems in graph matching because they restrict the number of possible correspondences between two graphs, many of the available methods, spectral-based methods for instance, are not able to deal with labelled graphs. Therefore, the existence of particular labels

* Corresponding author. Tel.: +34 93 581 23 01; fax.: +34 93 581 16 70.
E-mail addresses: mferrer@cvc.uab.cat, mferrer@iri.upc.edu (M. Ferrer), Ernest.Valveny@uab.es (E. Valveny).

0031-3203/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2009.10.013

makes the problem easier from the theoretical point of view, but it restricts the kind of methods that can be applied.

One of the aforementioned operations is the mean of a set of graphs that, in the graph domain, has been defined using the concept of the median graph. Given a set of graphs, the median graph [16] is defined as a graph that has the minimum sum of distances (SOD) to all graphs in the set. It can be seen as the representative of the set and, therefore, it has a large number of potential applications including many classical algorithms for learning, clustering and classification that are normally used in the vector domain. As a matter of fact, it can be potentially applied to any graph-based algorithm where a representative of a set of graphs is needed. However, the computation of the median graph is exponential both in the number of input graphs and their size [16]. The only exact algorithm proposed up to now [19] is based on an A^* algorithm using a data structure called multimatch. As the computational cost of this algorithm is very high, a set of approximate algorithms have also been presented in the past based on different approaches such as genetic search [16,19], greedy algorithms [14] and spectral graph theory [10,30]. However, all these algorithms can only be applied to restricted sets of graphs, regarding either the type or the size of the graphs.

In this paper we address the computation of the median graph from a new point of view based on three main pillars: the graph embedding in vector spaces [25], the graph edit distance [3], and the weighted mean of a pair of graphs [4].

Graph embedding [15] aims to convert graphs into another structure, such as real vectors, and then operate in the associated space to make easier some typical graph-based tasks, such as matching and clustering [12,7]. To this end, different graph embedding procedures have been proposed in the literature so far. Some of them are based on the spectral graph theory. Others take advantage of typical similarity measures to perform the embedding tasks. For instance, a relatively early approach based on the adjacency matrix of a graph is proposed in [18]. In this work, graphs are converted into a vector representation using some spectral features extracted from the adjacency matrix of a graph. Then, these vectors are embedded into eigenspaces with the use of the eigenvectors of the covariance matrix of the vectors. This approach is then used to perform graph clustering experiments. Another similar approach have been presented in [31]. This work is similar to the previous one, but in this case the authors use the coefficients of some symmetric polynomials constructed from the spectral features of the Laplacian matrix, to represent the graphs into a vectorial form. Finally, in a recent approach [26], the idea is to embed the nodes of a graph into a metric space and view the graph edge set as geodesics between pairs of points in a Riemannian manifold. Then, the problem of matching the nodes of a pair of graphs is viewed as the alignment of the embedded point sets. In this work we will use a new class of graph embedding procedures based on the selection of some prototypes and graph edit distance computation. This approach was first presented in [25], and it is based on the work proposed in [22]. The basic intuition of this work is that the description of the regularities in observations of classes and objects is the basis to perform pattern classification. Thus, based on the selection of concrete prototypes, each point is embedded into a vector space by taking its distance to all these prototypes. Assuming these prototypes have been chosen appropriately, each class will form a compact zone in the vector space.

Using the proposed embedding procedure, we can combine advantages from both domains: we keep the representational power of graphs while being able to operate in a vector space. Thus, all the machinery for statistical pattern recognition can be applied to graphs. In our approach we use a graph embedding based on the computation of the graph edit distance among all graphs in the learning set. The median of the set of vectors obtained with this mapping can be easily computed in the vector space. Then, using the weighted mean of a pair of graphs we have designed a triangulation procedure that permits to go from the vector domain back to the graph domain in order to finally obtain an approximation of the median graph. This new three-step procedure is the main contribution of the paper.

Although this procedure is based on three well-known techniques, it provides some remarkable improvements over other methods for the median graph computation. The use of the edit distance makes this procedure applicable to graphs with any kind of labels. In contrast, some of the existing methods can only deal with labelled graphs, with no symbolic attributes in neither the nodes nor the edges. From this point of view, our contribution is not a simplification, but a generalization of available methods in the sense that we can deal with any type of labels. In addition, this new procedure makes the computation of the median graph extendable to a large number of graphs with large size as we will show later in the experiments. More concretely we have applied the median graph to real classification problems in the context of molecule categorization, web content mining and symbol recognition. The underlying graphs have no constraints regarding the number of nodes and edges. None of the previous existing methods for the median graph computation is able to operate under these hard conditions.

Thus, with this new approach we are able to bring the median graph to the world of real applications in pattern recognition and machine learning, where up to now, all reported works on the median graph could only be evaluated using restricted sets of graphs concerning either the number or the size of the graphs. Therefore the proposed procedure allows us to transfer any machine learning algorithm that uses a median, from the vector to the graph domain.

The rest of this paper is organized as follows. In the next section we define the basic concepts and we introduce the notation we will use later in the paper. Then, in Section 3 we introduce in detail the concept of the median graph. In Section 4 the proposed method for the median computation is described. Section 5 reports a number of experiments and present results achieved with our method. Also a comparison with a reference system is provided. Finally, in Section 6 we draw some conclusions and we point out to possible future work.

2. Basic definitions

This section introduces the basic terminology and notation we will use throughout the paper.

2.1. Graph

Graphs can be defined in many ways. The definition of a graph given below is sufficiently general to include the most important classes of graphs.

Definition. Given L, a finite alphabet of labels for nodes and edges, a *graph* g is defined by the four-tuple $g=(V,E,\mu,\nu)$ where:

- V is a finite set of nodes,
- $E\subseteq V\times V$ is the set of edges,
- μ is the node labelling function ($\mu: V\longrightarrow L$) and
- ν is the edge labelling function ($\nu: V\times V\longrightarrow L$).

Notice that in this definition there is not any restriction about the nature of the node and edge labels. That is, the alphabet of labels is not constrained in any way. For example, L can be defined as a

vector space (i.e. $L=\mathbb{R}^n$) or simply as a set of discrete labels (i.e. $L=\{\Delta,\Sigma,\Psi,\ldots\}$). Edges are defined as ordered pairs of nodes, that is, an edge is defined by (u,v) where $u,v\in V$. The edges are directed in the sense that if the edge is defined as (u,v) then $u\in V$ is the source node and $v\in V$ is the target node. Note that an edge may connect a node $u\in V$ with itself. A graph is called *undirected* if for each edge $(u,v)\in E$ there is an edge $(v,u)\in E$ such that $v(u,v)=v(v,u)$.

2.2. Graph distance

The process of evaluating the structural similarity of two graphs is commonly referred to as graph matching. This issue has been addressed by a large number of works. For an extensive review of different graph matching methods and applications, the reader is referred to [5]. In this paper, we will use the graph edit distance [3,27], one of the most widely used methods to compute the dissimilarity between two graphs.

The basic idea behind the graph edit distance is to define the dissimilarity of two graphs as the minimum amount of distortion required to transform one graph into the other [3]. To this end, a number of distortion or edit operations e, consisting of the insertion, deletion and substitution of both nodes and edges are defined. Given these edit operations, for every pair of graphs, g_1 and g_2, there exists a sequence of edit operations, or edit path $p(g_1,g_2)=(e_1,\ldots,e_k)$ (where each e_i denotes an edit operation) that transforms g_1 into g_2. In Fig. 1 an example of an edit path between two given graphs g_1 and g_2 is given. This edit path consists of one edge deletion, one node substitution, one node insertion and two edge insertions. In general, several edit paths exist between two given graphs. This set of edit paths is denoted by $\wp(g_1,g_2)$. In order to quantitatively evaluate which edit path is the best, edit costs are introduced. The basic idea is to assign a penalty cost c to each edit operation according to the amount of distortion it introduces in the transformation. The edit distance d between two graphs g_1 and g_2, denoted by $d(g_1,g_2)$, is the cost of the edit path with minimum cost that transforms one graph into the other.

Definition. Given two graphs $g_1=(V_1,E_1,\mu_1,\nu_1)$ and $g_2=(V_2,E_2,\mu_2,\nu_2)$, the *graph edit distance* between g_1 and g_2 is defined by

$$d(g_1,g_2)=\min_{(e_1,\ldots,e_k)\in\wp(g_1,g_2)}\sum_{i=1}^{k}c(e_i) \tag{1}$$

where $\wp(g_1,g_2)$ denotes the set of edit paths that transform g_1 into g_2 and $c(e)$ denotes the cost of an edit operation e.

A number of optimal and approximate algorithms for the computation of the graph edit distance have been proposed. Optimal algorithms are usually based on combinatorial search procedures that explore all the possible mappings of nodes and edges of one graph to the nodes and edges of the second graph [27]. The major drawback of such an approach is its computational complexity, which is exponential in the number of nodes of the involved graphs. Consequently, its application is restricted to graphs of rather small size in practice. As an alternative, a number of suboptimal methods have been proposed to make the graph edit distance computation less computationally demanding. Some of these methods are based on local optimization [20]. A linear programming method to compute the graph edit distance with unlabelled edges is presented in [17]. Such method can be used to obtain lower and upper edit distance bounds in polynomial time. In [21] simple variants of the standard method are proposed to derive two fast suboptimal algorithms for graph edit distance, which make the computation substantially faster. Finally, in [24], a new efficient algorithm is presented based on a fast suboptimal bipartite optimization procedure.

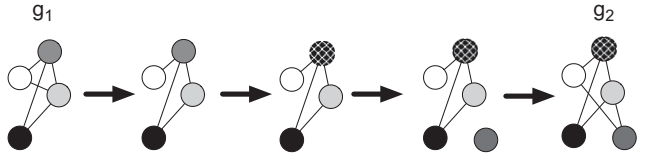

Fig. 1. A possible edit path between two graphs g_1 and g_2. Not that node labels are indicated by different colors.

3. Generalized median graph

The generalized median graph has been proposed to represent a set of graphs.

Definition. Let U be the set of graphs that can be constructed using labels from L. Given $S=\{g_1,g_2,\ldots,g_n\}\subseteq U$, the *generalized median graph* $\overline{g}$ of S is defined as

$$\overline{g}=\operatorname*{argmin}_{g\in U}\sum_{g_i\in S}d(g,g_i) \tag{2}$$

That is, the generalized median graph $\overline{g}$ of S is a graph $g\in U$ that minimizes the sum of distances (SOD) to all the graphs in S. Notice that $\overline{g}$ is usually not a member of S, and in general more than one generalized median graph may exist for a given set S.

As shown in Eq. (2) some distance measure $d(g,g_i)$ between the candidate median g and every graph $g_i\in S$ must be computed. However, since the computation of the graph edit distance is a well-known NP-complete problem, the computation of the generalized median graph can only be done in exponential time, both in the number of graphs in S and their size (even in the special case of strings, the time required is exponential in the number of input strings [6]). As a consequence, in real applications we are forced to use suboptimal methods in order to obtain approximate solutions for the generalized median graph in reasonable time. Such approximate methods [10,14,16,19,30] apply some heuristics in order to reduce the complexity of the graph edit distance computation and the size of the search space.

Another alternative to reduce the computation time is to use the set median graph instead of the generalized median graph. The difference between the two concepts is only the search space where the median is looked for. As it is shown in Eq. (2), the search space for the generalized median graph is U, that is, the whole universe of graphs. In contrast, the search space for the set median graph is simply S, that is, the set of graphs in the given set. It makes the computation of set median graph exponential in the size of the graphs, due to the complexity of graph edit distance, but polynomial with respect to the number of graphs in S, since it is only necessary to compute pairwise distances between the graphs in the set. The set median graph is usually not the best representative of a set of graphs, but it is often a good starting point towards the search of the generalized median graph.

4. New approximate algorithm based on graph embedding in vector spaces

In the last section we have shown that the computation of the generalized median graph is a rather complex task. In this section we present a novel approach for the computation of the median graph that is faster and more accurate than previous approximate algorithms. It is based on graph embedding in a vector space and it consists of three main steps.

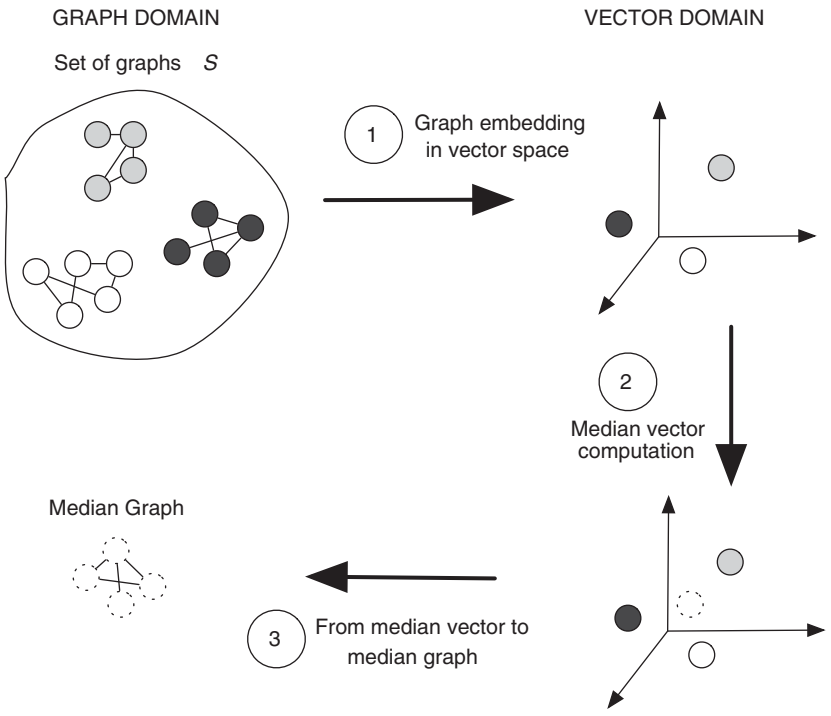

Fig. 2. Overview of the approximate procedure for median graph computation.

Given a set $S=\{g_1,g_2,\ldots,g_n\}$ of n graphs, the first step is to embed every graph in S into the n-dimensional space of real numbers, i.e. each graph becomes a point in $\mathbb{R}^n$. The second step consists of computing the median using the vectors obtained in the previous step. Finally, the go from the vector space back to the graph domain, converting the median vector into a graph. The resulting graph is taken as the median graph of S. These three steps are depicted in Fig. 2. In the following subsections, these three main steps will be further explained.

4.1. Graph embedding in vector spaces

The embedding procedure we use in this paper follows the procedure proposed in [25]. For the sake of completeness, we briefly describe this approach in the following.

Assume we have a set of training graphs $T=\{g_1,g_2,\ldots,g_n\}$ and a graph dissimilarity measure $d(g_i,g_j)$ $(i,j=1,\ldots,n;\ g_i,g_j\in T)$. Then, a set $P=\{p_1,\ldots,p_m\}\subseteq T$ of m prototypes is selected from T (with $m\leq n$). After that, the dissimilarity between a given graph of $g\in T$ and every prototype $p\in P$ is computed. This leads to m dissimilarity values, $d_1,\ldots,d_m$ where $d_k=d(g,p_k)$. These dissimilarities can be arranged in a vector $(d_1,\ldots,d_m)$. In this way, we can transform any graph of the training set T into an m-dimensional vector using the prototype set P. More formally this embedding procedure can be defined as follows:

Definition. Given a set of training graphs $T=\{g_1,g_2,\ldots,g_n\}$ and a set of prototypes $P=\{p_1,\ldots,p_m\}\subseteq T$, the *graph embedding*:

$$\psi : T\longrightarrow \mathbb{R}^m \tag{3}$$

is defined as the function:

$$\psi(g)\longrightarrow(d(g,p_1),d(g,p_2),\ldots,d(g,p_m)) \tag{4}$$

where $g\in T$, and $d(g,p_i)$, is a graph dissimilarity measure (in this paper the graph edit distance between the graph g and the i-th prototype p_i).

We perform the graph embedding step according to this definition, but let the training set T and the prototype set P be the same, i.e. the set S for which the median graph is to be computed. So, we compute the graph edit distance between every pair of graphs in the set S. These distances are arranged in a distance matrix.[1] Each row/column of the matrix can be seen as an n-dimensional vector. Since each row/column of the distance matrix is assigned to one graph, such an n-dimensional vector is the vectorial representation of the corresponding graph. Fig. 3 illustrates this procedure.

4.2. Computation of the median vector

Once all the graphs have been embedded in the vector space, the median vector is computed. To this end we use the concept of *Euclidean Median*.

Given a set $X=\{x_1,x_2,\ldots,x_m\}$ of m points with $x_i\in\mathbb{R}^n$ for $i=1\ldots m$, the geometric median is defined as

$$\textit{Geometric median}=\arg\min_{y\in\mathbb{R}^n}\sum_{i=1}^{m}\|x_i-y\|$$

where $\|x_i-y\|$ denotes the Euclidean distance between the points $x_i,y\in\mathbb{R}^n$.

That is, the *Euclidean Median* is a point $y\in\mathbb{R}^n$ that minimizes the sum of the Euclidean distances between itself and all the points in X. It corresponds with the definition of the median graph, but in the vector domain. The Euclidean median cannot be calculated in a straightforward way. The exact location of the Euclidean median cannot be found when the number of elements in X is greater than 5 [2]. No algorithm in polynomial time is known, nor has the problem been shown to be NP-hard [13]. In this work we will use the most common approximate algorithm for the computation of the Euclidean median, that is Weiszfeld's algorithm [29]. It is an iterative procedure that converges to the Euclidean median. To this end, the algorithm first chooses an initial estimate solution y (this initial solution is often chosen

[1] Actually, only $n(n-1)/2$ distances have to be computed, since the distance between two graphs is symmetric.

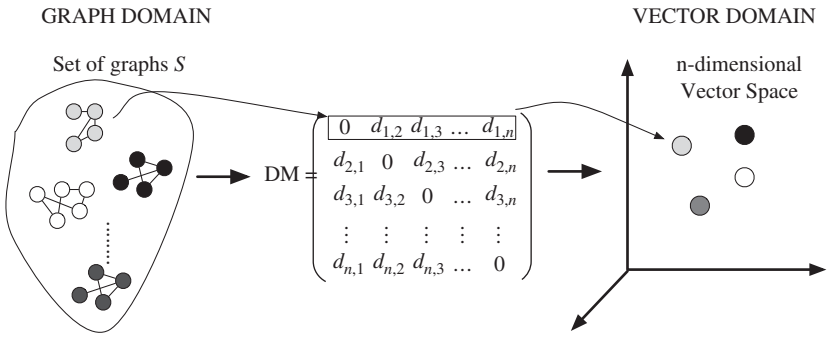

Fig. 3. Step 1. Graph embedding.

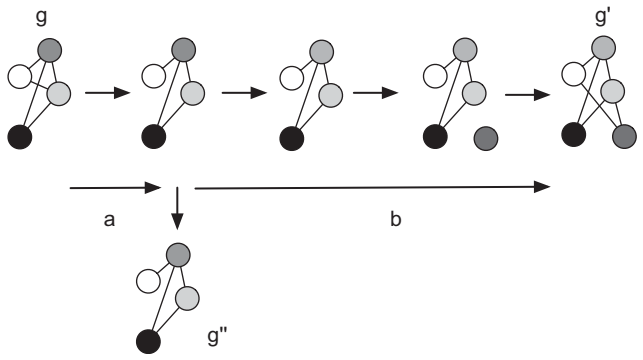

Fig. 4. Weighted mean of a pair of graphs.

randomly). Then, the algorithm defines a set of weights that are inversely proportional to the distances from the current estimate to the samples, and creates a new estimate that is the weighted average of the samples according to these weights. The algorithm may finish either when a predefined number of iterations is reached, or under some other criteria, for instance, when the difference between the current estimate and the previous one is less than a established threshold.

4.3. Back to graph domain

The last step in order to obtain the median graph is to transform the Euclidean median into a graph. Such a graph will be considered as an approximation of the median graph of the set S. To this end we will use a triangulation procedure based on the weighted mean of a pair of graphs [4] and the edit path between two given graphs. For the sake of completeness the definition of the weighted mean of a pair of graphs is included here.

Definition. Let g and g' be graphs. The *weighted mean* of g and g' is a graph g'' such that

$$d(g,g'') = a$$

$$d(g,g') = a + d(g'',g')$$

That is, the graph g'' is a graph in between the graphs g and g' along the edit path between them. Furthermore, if the distance between g and g'' is a and the distance between g'' and g' is b, then the distance between g and g' is $a+b$. Fig. 4 illustrates this idea.

The weighted mean of a pair of graphs is used in the procedure that transforms the Euclidean median vector into a graph. This procedure, illustrated in Fig. 5(a), is based on a triangulation procedure among points in the vector space as follows. Given the n-dimensional points representing every graph in S (represented as white dots in Fig. 5(a)), and the Euclidean Median vector v_m (represented as a grey dot in Fig. 5(a)), we first select the three closest points to the Euclidean median ($v_1 - v_3$ in Fig. 5(a)). Notice that we know the corresponding graph of each of these points (in Fig. 5(a) we have indicated this fact by labelling them with the pair v_j, g_j with $j = 1,\ldots,3$). Then, we compute the median vector v'_m of these three points (represented as a black dot in Fig. 5(a)). Notice that v'_m is in the plane formed by v_1, v_2 and v_3. With $v_1 - v_3$ and v'_m at hand (Fig. 5(b)), we arbitrarily choose two out of these three points (without loss of generality we can assume that we select v_1 and v_2) and we project the remaining point (v_3) onto the line joining v_1 and v_2. In this way, we obtain a point v_i in between v_1 and v_2 (Fig. 5(c)). With this point at hand, we can compute the percentage of the distance in between v_1 and v_2 where v_i is located(Fig. 5(d)). As we know the corresponding graphs of the points v_1 and v_2 we can obtain the graph g_i corresponding to v_i by applying the weighted mean procedure explained before (Fig. 5(e)). Once g_i is known, then we can obtain the percentage of distance in between v_i and v_3 where v'_m is located and obtain g'_m applying again the weighted mean procedure (Fig. 5(f)). Finally, g'_m is chosen as the approximation for the generalized median of the set S.

4.4. Discussion on the approximations

This approximate embedding procedure is composed of three steps: the graph embedding into a vector space, the median vector computation and the return to the graph domain. Each of these steps introduces some kind of approximation to the final solution. In the first step, in order to deal with large graphs an approximate edit distance algorithm is normally used. Thus, each vector representing a graph includes small errors in its coordinates with respect to the optimal distance between two graphs. Nevertheless, the two approximate methods for the edit distance computation we used [21,24] provide correlation scatter plots showing a high accuracy with respect the exact distance computation. Also the median vector computation introduces a certain amount of error, since the Weiszfeld method obtains approximations for the median vector. This factor may lead to choose three points that might not be the best points to go back to the graph domain. In addition, small errors may be introduced when choosing v'_m instead of directly v_m to perform the weighted mean of a pair of graphs. Finally, when the weighted mean between two points is computed, the graph edit path is composed of a set of discrete steps, each of them with its own cost. In the return to the graph domain, the percentage of distance needed to obtain the weighted mean of a pair of graphs may fall in between two of this edit operations. Since we choose only one of them, small errors may also be introduced in this step.

Although all these approximations may seem too severe to obtain good medians, in the next section we will show that this method is able to obtain reasonable good approximations for the

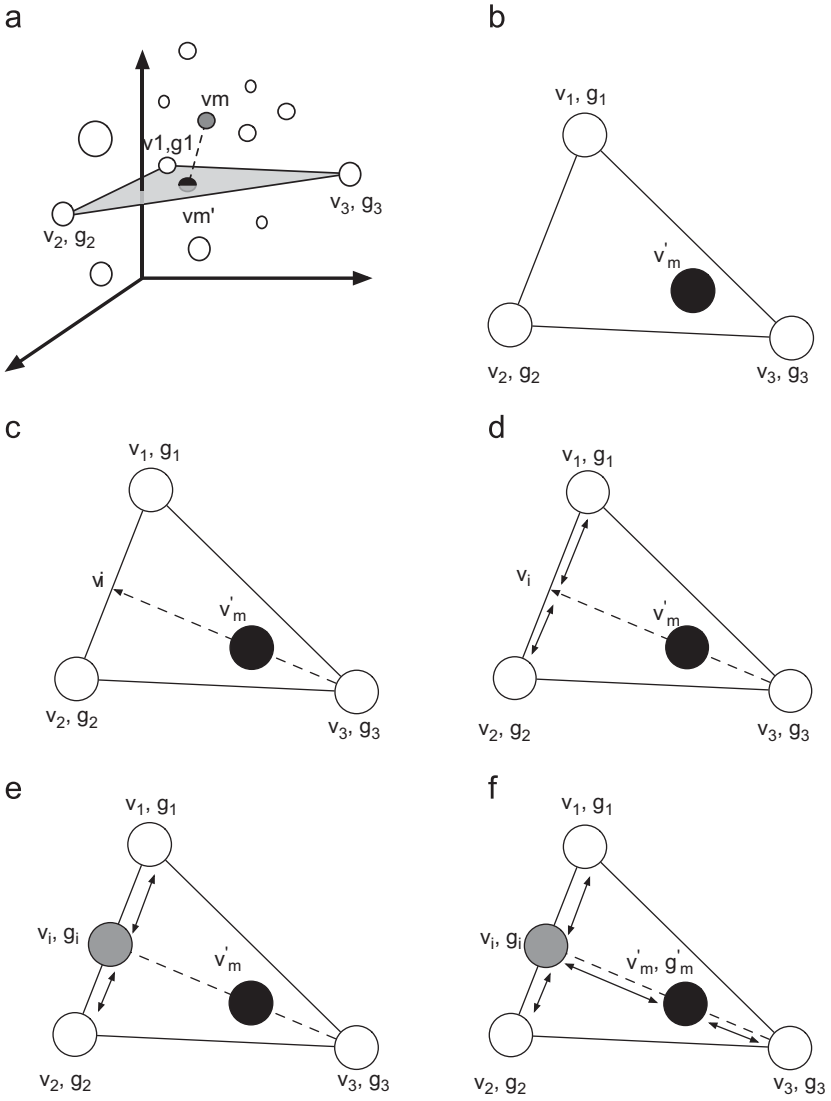

Fig. 5. Illustration of the triangulation procedure.

median graph. But, at the same time, these well defined entry points of approximation can be used to improve the medians obtained. That is, each of these three steps can be further studied and other options for each of them can be proposed in order to minimize the error introduced and obtain better medians, as we will comment on the proposals for future work.

5. Experimental set-up

In this section we provide the results of an experimental evaluation of the proposed algorithm. To this end three completely different real databases have been used, namely a molecule database containing 2000 instances of molecules from two classes, active and inactive; a database containing 2340 graph-based representations of web-pages belonging to six different classes and the GREC symbol database containing 1100 instances of graphical symbols from 22 different classes. The experiments consisted of two different phases. The first step was to compute several medians for each database in order to evaluate whether we obtain good approximations of the median graph. Then, in a second stage, we used these medians to perform some classification tasks. In all these experiments the edit distance between graphs was computed using the approach introduced in [21] for the GREC dataset and the procedure presented in [24] for the Molecule and the webpage dataset. In the following, we will firstly explain these three datasets in detail. Then, the results of the experiments will be presented and analysed.

5.1. Datasets

In order to make the paper self-contained we briefly explain the used datasets in the following, but the reader is referred to [23] for a detailed explanation of each of them.

Molecule dataset: The molecule database consists of graphs representing molecular compounds. These graphs have been extracted from the AIDS Antiviral Screen Database of Active Compounds [1]. The database consists of two different classes of molecules: active and inactive, depending on whether they show activity against HIV or not. The molecules are converted into graphs in a straightforward way, representing atoms as nodes and covalent bonds as edges. The nodes are labelled with the number of the corresponding chemical symbol, while the edges are labelled with the valence of the linkage. Some examples of each class are shown in Fig. 6. In order to simplify the representation,

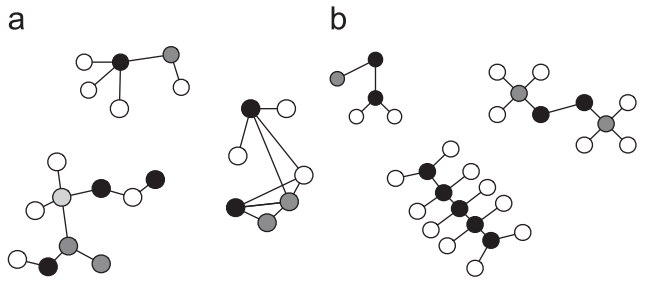

Fig. 6. Examples of active compounds (a) and inactive compounds (b).

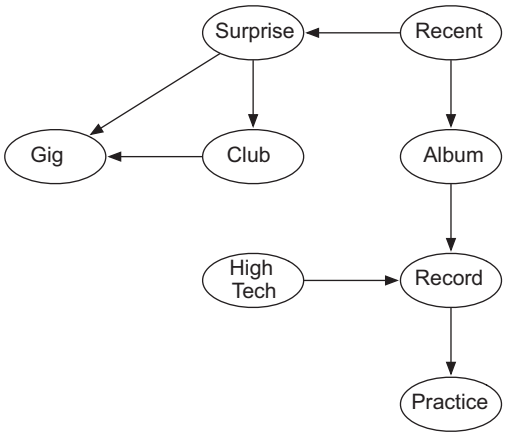

Fig. 7. Example of a webgraph.

different chemical symbols are represented using different gray levels.

In this database there is a total number of 2000 graphs, 400 corresponding to active molecules and 1600 to inactive ones. In order to create the training set we have randomly chosen 200 graphs from each class and included the 1600 remaining ones in the test set.

Webpage dataset: In [28] several methods to create graph-based representations of webpages are introduced. In our case, graphs representing webpages are constructed as follows. First, all words appearing in the web document are converted into nodes in the web graph, except for stop words which contain little information. The nodes are attributed with the corresponding word and its frequency. That is, even if a word appears more than once in the web document, only one node with this word label is added to the graph, and the frequency of the word is used as an additional attribute. Then, if a word w_i immediately precedes another word w_j in the document, a directed edge from the node corresponding to the word w_i to the node corresponding to the word w_j is added to the graph. In order to keep the essential information of the document, only the most frequently used words (nodes) are kept in the graph and the terms are combined to the most frequently occurring form (Fig. 7).

The dataset is composed of 2340 documents belonging to 20 different categories (Business, Health, Politics, Sports, Technology, Entertainment, Art, Cable, Culture, Film, Industry, Media, Multimedia, Music, Online, People, Review, Stage, Television and Variety). The last 14 categories are sub-categories of Entertainment. These web documents were originally hosted at Yahoo as news pages (http://www.yahoo.com). For simplicity, from now on, the six main classes will be referred as B, H, P, S, T and E for Business, Health, Politics, Sports, Technology and Entertainment, respectively.

Note that not all the classes have the same number of graphs. Table 1 shows the number of graphs in each class.

Table 1
Number of graphs in each class.

	Class					
	B	E	H	P	S	T
Number of graphs	142	1389	494	114	141	60

We have divided this dataset into a training and a test set. As one can see in Table 1, the class with the smallest number of graphs is class T, with 60 graphs. For this reason, we have randomly chosen 30 graphs of every class to form the training set (that is, the training set is composed of 180 graphs) and the remaining 2160 graphs are used as the test set.

GREC dataset: The GREC database used in our experiments corresponds to a subset of the symbol database of GREC 2005 contest [8]. The whole database is composed of a set of 150 symbols from architecture, electronics and other technical fields. We have used a subset of 22 different symbols, those which are composed only of straight lines. In order to work with a large dataset, with arbitrarily strong distortions, we have generated several instances of each symbol applying different distortion operators to the original images. Such distortion operators include moving the junction points between two lines within a predefined radius r, splitting junction points and deleting some lines. These images are converted into graphs by assigning a node to each junction or terminal point and an edge to each line. The labels for the nodes are coordinates in a 2-dimensional space corresponding to the location of the point. The labels for the edges are simply binary numbers indicating whether a line exists between two given nodes or not. Fig. 8 shows an example of two symbols and different distorted instances of them.

For each of the 22 symbols (classes) in the dataset we have generated 50 distorted instances. So, totally we have 1100 images. The complete dataset is split into a training set of 440 (20 graphs for each class) and a test set of 660 elements (30 graphs per class).

Table 2 summarizes some important parameters of each dataset including the number of classes, total number of elements in the dataset, elements in the training and test set and the maximum, minimum and average size of the graphs in the dataset.

5.2. Assessment of the median quality

To evaluate the quality of the obtained median graphs, we compare their SOD with the SOD of the set median graph. We do not use other approximations of the median graph as a reference for two reasons. First, the existing methods are not able to compute the median graph with these large graphs and datasets we are dealing with. Second, as the set median graph is the graph belonging to the training set with minimum SOD, it is a good reference to evaluate the generalized median graph quality. Therefore, for a given dataset, the same edit distance algorithm is used to compute the set median and the approximate median. In this sense, since they are computed using the same method, the same amount of error or distortion (due to the approximation) is introduced to both of them. Thus, we can say that they are fairly comparable. For this experiment we have randomly chosen an increasing number of graphs from the training set, and we have computed the median graph of each of these sets (that corresponds to the set S in the definition of the median graph). The different number of graphs chosen from the training set for

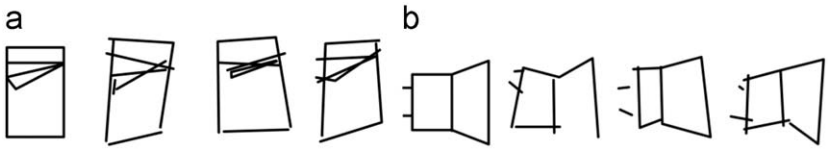

Fig. 8. Examples of GREC symbols with some distortions: (a) architectural symbol and (b) electrical symbol.

Table 2
Some characteristics of each dataset.

Property	Molecule	Webpage	GREC
Number of classes	2	6	22
Total number of elements	2000	2340	1100
Number of elements in the training set	400	180	440
Number of elements in the test set	1600	2160	660
Max. number of nodes in a graph	95	834	25
Min. number of nodes in a graph	1	43	3
Mean number of nodes	15.7	186.1	11.5

Table 3
Number of graphs in S for each database.

Class	Number of graphs in S
Molecules	$10, 20, 30, \ldots, 100$
Webpages	$5, 10, 15, \ldots, 30$
GREC	5, 10, 15, 20

each database is shown in Table 3. We repeated the experiment 10 times for each size of the set S in order to generalize the results.

Results for the molecule, webpage and GREC datasets are shown in Figs. 9–11, respectively. In these figures, the mean value over all classes and repetitions of the SOD of the set median (continuous line) and the SOD of the computed median (dashed line) are displayed. In addition, for each size of the set (x-axis), bars showing the standard deviation of the SODs of each method (set median in black and computed median in grey) are plotted.

Discussion: The results of Figs. 9–11 show that the performance of both the set median and the generalized median graph are comparable to each other. In this sense, we can observe that in two of the databases (the molecule and the webpage datasets), the results for the SOD are slightly better for the generalized median graph. However, due to the large standard deviation and the similarity between the results for both methods), we cannot conclude that these results are statistically significant.

Nevertheless, with these results at hand, we can conclude that our method achieves, in general, good approximations of the generalized median graph, independently of the size of the training set. That means that the generalized median adapts well to the increasing variability and distortion as the number of graphs in the training set increases.

5.3. Classification

In this section we will use the previously computed medians to perform some basic classification experiments. For each dataset we classified each element in the test set according to three different schemes. The first one is a 1NN classifier using the whole training set. The other two use the generalized median and the set median graphs respectively. Each element in the test set is classified into the class of the most similar median. It is important to note that, again, for each size of S we repeated the experiment 10 times. Thus the classification rates shown in the following are the average results over these 10 repetitions. In addition, we show the standard deviation of the classification rate in order to show the stability of our method against different repetitions of the experiment.

Molecule dataset: Table 4 shows the results of classification for the three mentioned methods on the molecule dataset. The values are the mean values over the two classes.

Although the 1NN approach performs better, it is important to notice that both kinds of medians achieve quite good levels. But at this point, it is relevant to mention that the recognition rate of the computed median is, in general, slightly better than the one obtained for the set median. In addition the median approach shows more stability in the results than the set median. For instance, as we can see in the table, for a size of S equal to 90 there is a negative peak in the recognition rate for the set median, while the median approach shows more stable results. This means that even for a large number of graphs the median we compute is able to keep the basic information of the class. We can remark the direct correlation of these results with those obtained in the SOD evolution. That is, we obtain better recognition rates with the median as the median has better SOD evolution. Only for a large number of graphs in S (from 70 on), it seem that the set median performs slightly better than the median approach (except for the size 90).

In spite of the loss in the recognition rate of the median-based methods, it should be remarked the difference in the number of comparisons required by both methods. While the number of comparisons is 640,000 for the 1NN classifier, in the median-based classifiers the number of comparisons is only 3200. This reduction may play an important role in graph-based applications, where the time needed for comparing two graphs is sometimes quite high and the time requirements are also important.

Webpages dataset: Table 5 shows the results of classification for the webpage dataset. Here, the classification rates are the mean over the six classes. The results show that the classification rates for both medians are reasonably close to those of the 1NN classifier. For instance, the minimum difference between the 1NN classifier and the median is lower than a 4% (for a size of S equal to 10). The recognition rates for both medians are very close to each other, being slightly better in the set median (except for the case when the size of S is equal to 10). In addition, the standard deviation of the set median is lower than that of the generalized median.

Although the 1NN classifier performs better, it is important to notice again the difference between the number of comparisons needed for both classifiers. While the 1NN classifier needs 388,800 comparisons, the use of the median graph reduces such quantity to 12,960 comparisons.

With the fact that the median achieves classification rates quite close to the 1NN classifier, we can think of using the median to filter out the number of possible classes before applying the 1NN approach. With this approach we aim to improve the performance of the 1NN classifier and reduce the number of comparisons needed.

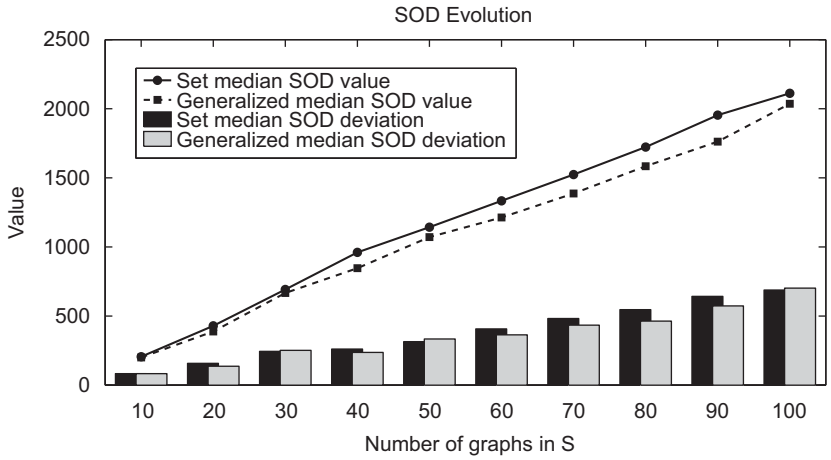

Fig. 9. SOD evolution on molecule dataset.

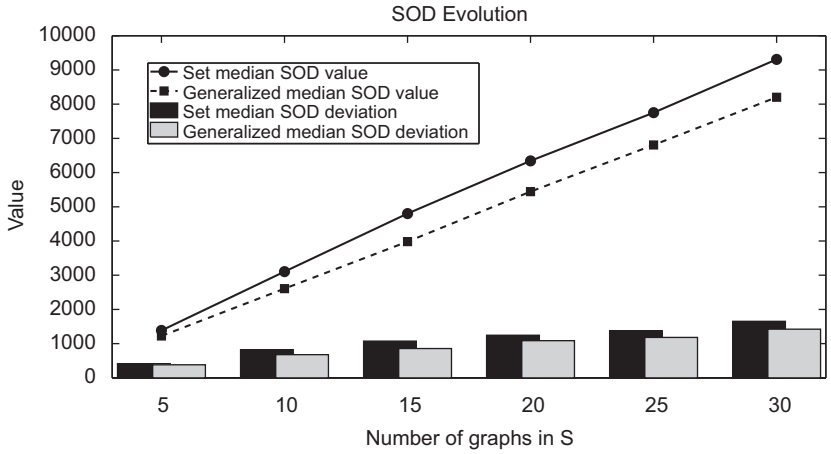

Fig. 10. SOD evolution on webpage dataset.

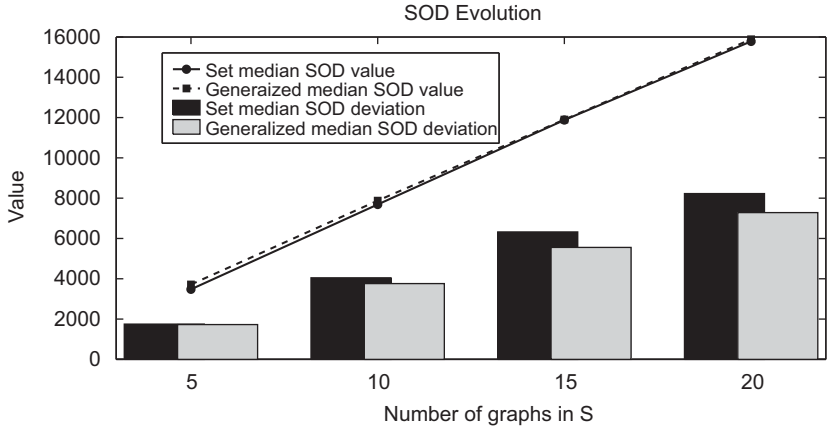

Fig. 11. SOD evolution on GREC dataset.

To this end, we propose first to measure the appearance frequency of the correct class within a predefined number of retrieved classes. That is, for every input pattern we will rank all classes in an increasing order based on the distance of the input pattern to the median of each class. Then, we will set a depth (the predefined number of classes) and we will see if the correct class appears in this set. In this experiment, we have used a depth of three classes. The computation was repeated 10 times in order to better generalize the results.

In Fig. 12, we can see the results of the frequency of the correct class appearance and its deviation depending on the number of graphs in *S*. Results show that in general, the appearance frequency of the correct class using the computed median is quite similar to the appearance frequency using the set median,

being even better in the computed median when the number of classes is equal to 30. Only when the number of classes is equal to 20, the difference between them is greater than 2%. It is important to note that, in general, both medians achieve more than 95% of appearance frequency, which is higher than the classification rate using the 1NN approach. The low value of the deviation when the number of graphs increase (25 and 30 graphs) suggests that the results (and so, the median) get more stable with a large training set.

Table 4
Classification rate for molecule database with an increasing number of graphs in the training set.

$\|S\|$	Rate (%)			Std Dev.		
	1NN	Median	Set median	1NN	Median	Set median
10	92.11	89.52	84.02	2.10	5.36	3.75
20	95.38	89.95	87.79	1.60	4.24	5.86
30	96.40	88.87	86.70	1.32	4.58	3.89
40	96.42	89.22	84.77	1.16	5.11	4.14
50	96.72	90.40	86.90	0.82	4.78	3.96
60	97.45	87.70	86.75	0.74	4.03	4.67
70	97.81	87.45	88.12	0.51	4.43	4.75
80	97.99	86.77	89.05	0.24	4.31	4.11
90	97.45	86.72	77.32	0.24	8.90	14.42
100	98.54	85.00	91.25	0.87	4.57	4.95

Table 5
Classification rate for webpage database with an increasing number of graphs in the training set.

$\|S\|$	Rate (%)			Std Dev.		
	1NN	Median	Set median	1NN	Median	Set median
5	78.56	65.40	76.34	5.37	11.48	5.71
10	87.06	83.68	79.82	3.23	5.51	3.01
15	88.01	78.59	81.59	2.07	7.53	3.04
20	90.06	78.39	82.77	1.12	8.20	2.58
25	91.18	80.44	83.05	0.67	9.65	1.11
30	91.48	82.77	83.42	2.49	8.29	2.98

Finally, with these results at hand, we performed a modified classification experiment, mixing the median-based methods with the classic 1NN classifier. The objective of this experiment is to investigate whether it is possible to increase the recognition rate of the 1NN classifier using a smaller number of comparisons. The basic idea is as follows. First, we compare each element of the test set against the medians, and rank the classes from the most similar to the least similar median. After that, the same element in the test set is compared against the training set but using only a reduced number of classes (3 for instance) instead of using all classes as in the 1NN classical approach. The input query is assigned to the class of the most similar element within the reduced number of classes. It is clear in this experiment that if the number of classes is set to 1, then the results are the same as the classification using the median, and if the number of classes is the total number of classes (in this case 6) then the results are the same as in the 1NN classifier. Again, we repeated the experiment 10 times.

Fig. 13 shows the results for medians computed with 30 elements and for a number of classes ranging from 1 (the minimum number of classes) to 6 (all the possible classes), and also the deviation achieved in the classification rate.

If the number of classes is equal to 1 the set median outperforms the median approach. However, for the rest of the cases this changes completely. This result reinforces the hypothesis that the computed median keeps better the basic information of the class. But what is important to note is that even for a number of classes equal to 2 the recognition rate for the median is better than the 1NN classifier. The maximum recognition rate is achieved for the number of classes being equal to 4. This means that we can obtain better results than the 1NN classifier with less than half the number of comparisons needed by the 1NN classifier (in the case where the number of classes is equal to 2). Here, the gain achieved by the set median approach is significantly lower, maintaining practically the same recognition rates as in the 1NN classifier. These results are supported by the low values of the deviation achieved by both methods for the number of classes being equal to 2 or more.

These results show a correspondence with the SOD evolution experiments. That is, better medians in terms of the SOD achieve better recognition rates here. Thus, the median can help us to

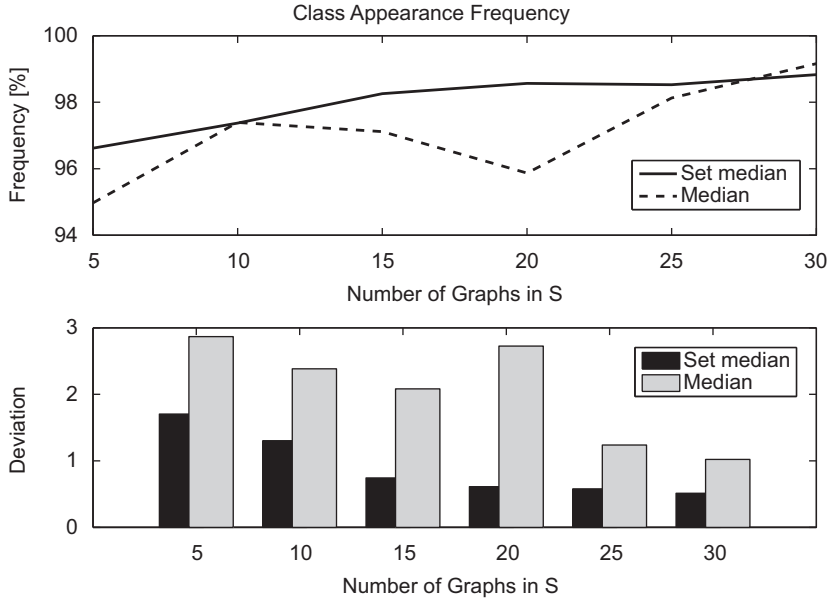

Fig. 12. Appearance frequency of the correct class.

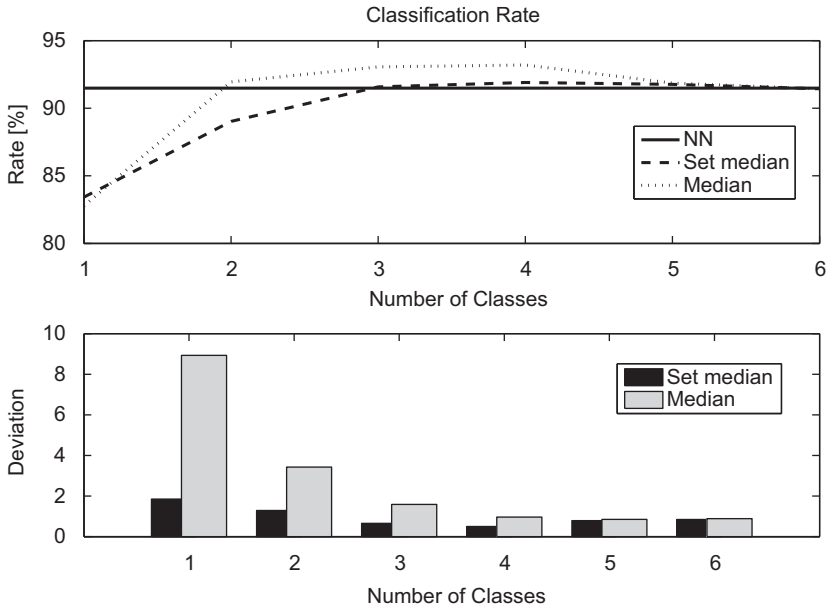

Fig. 13. Mixed-classification rate for the webpage database with an increasing number of classes used for filtering the original set of classes.

Table 6
Classification rate for GREC database with an increasing number of graphs in the training set.

$\|S\|$	Rate (%)			Std Dev.		
	1NN	Median	Set median	1NN	Median	Set median
5	93.27	79.07	78.33	1.31	2.30	1.56
10	96.59	78.33	77.78	0.77	2.78	2.46
15	97.95	78.53	77.31	0.66	1.93	1.71
20	98.63	78.25	76.68	0.68	1.49	1.60

achieve better classification rates using fewer comparisons than the 1NN classifier.

GREC dataset: Table 6 shows the results of classification rate for the GREC database. The values are the mean values over all the 22 classes and repetitions.

The results show that the 1NN classifier outperforms in all cases the recognition rate for both the set and the computed median. Nevertheless, it is important to notice that in this case, the recognition rates for the computed medians slightly outperform the results for the set medians. This could mean that our method is able to obtain better representatives than the set median. In spite of the loss in the recognition rate in the median-based methods, we should point out again the difference in the number of comparisons needed in the classification experiments. While the 1NN approach took 290,400 comparisons, this number was reduced to 14,520 in the median-based approaches.

Again, in order to extend the classification results, a frequency appearance experiment (repeated 10 times) has been conducted. In this case the results for both the class appearance frequency and the deviation are shown in Fig. 14 as a function of the number of graphs used to compute the medians. In this case we defined the number of classes equal to 8. The results show that the set median performs better, both in the class appearance frequency and the deviation, for values of the number of graphs in S up to 10, while for larger numbers, the computed median obtain better results. Nevertheless, it is important to note that for a number of classes equal to 10 the percentage of times that the correct class appears is greater than 99%.

Finally, we performed the same modified recognition experiment as with the webpage dataset. Results for both the classification rate and deviation are shown in Fig. 15, for a number of classes ranging from 1 to 22 and for a number of graphs in S equal to 15. In this case, the results obtained with the medians are in the best case equal to those for the 1NN classifier. Although both median approaches perform quite similar, the results show that for a small number of classes (up to 5), the median approach slightly outperforms the set median. This means that in general, the median approach is able to better represent the best class among the first classes in the ranking. But, what is important to notice is that using less than 10 classes we achieve almost the same classification rates as with the 1NN classifier but using less than half of the number of comparisons. Thus, the use of the median is fully justified for large databases. Fig. 15 also suggests that most probably if we are able to obtain better medians we can improve these results and achieve better results than the 1NN classifier. It is important to note that in this case, the values of the deviation for the median are, in general, lower than those of the set median.

Again, a correspondence between the SOD evolution results and this experiment can be seen. As the medians have similar SOD, they also have similar recognition rates here. It suggests that if we are able to improve the medians in terms of the SOD, we will also be able to improve the classification rates.

6. Discussion and conclusions

Representation of the objects in pattern recognition is often carried out by either feature vectors or graphs. The main advantage of feature vectors is that many operations needed in machine learning can be easily executed on vectors, and a huge number of algorithms for pattern analysis and classification become immediately available. But they have a limited representational power, making them not suitable to represent structured objects. Differently, graphs have a high representational power, but have two main drawbacks. The computational complexity of the algorithms related to graphs is usually high and some basic operations that are needed in many pattern recognition methods, such as computing the sum or the mean, that might appear quite

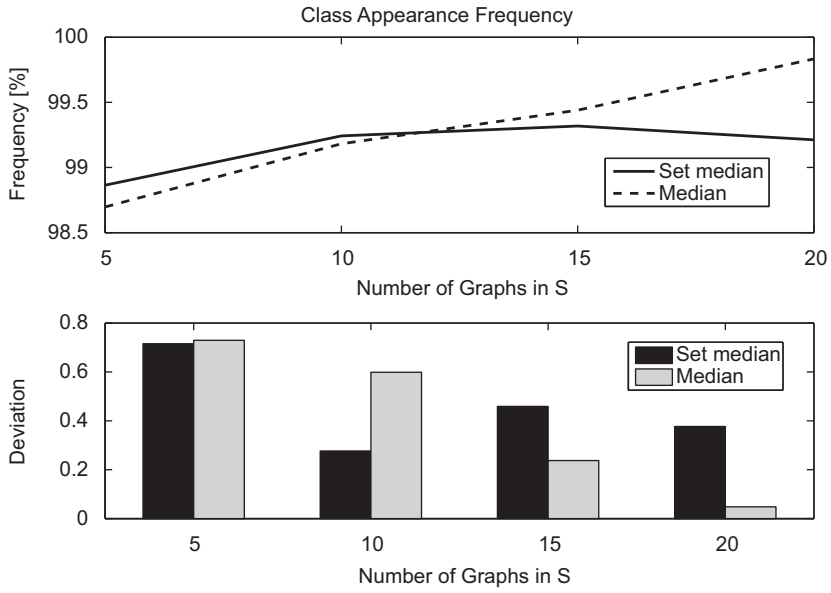

Fig. 14. Appearance frequency of the correct class.

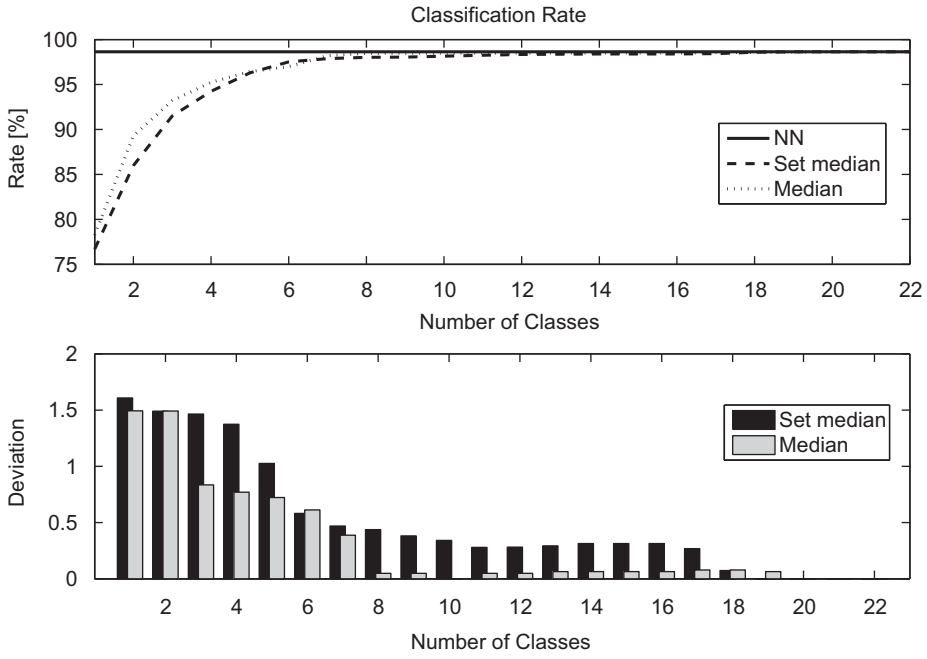

Fig. 15. Mixed-classification rate for the GREC database with an increasing number of classes used for filtering the original set of classes.

simple in the vector domain, turn very difficult or even impossible in the graph domain. One of these operations is the median of a given set, that in the graph domain, is called the median graph. It can be seen as the representative of the set and, therefore, it has a large number of potential applications including many classical algorithms for learning, clustering and classification that are normally used in the vector domain. However, the computation of the median graph is a highly complex task.

In the present paper we have proposed a novel technique to obtain approximate solutions for the median graph. This new approach is based on embedding of graphs into vector spaces. First, the graphs are turned into points of n-dimensional vector spaces using the graph edit distance paradigm. Then, the crucial point of obtaining the median of the set is carried out in the vector space, not in the graph domain, which simplifies dramatically this operation. Finally, using the graph edit distance again we can transform the obtained median vector to a graph by means of the weighted mean of a pair of graphs and a triangulation procedure. This embedding approach allows us to get the main advantages of both the vector and graph representations. That is, we compute the more complex parts in real vector spaces but keeping the representational power of graphs. This permits to apply the median graph computation to real data and to extend the application of the median graph to real classification problems. In particular we have applied it to three different databases containing a high number of graphs with high number of nodes each. Under these strong conditions, the generalized median could not be computed before, due to the large computational resources needed for the existing methods.

In a first experiment we compare the obtained SOD with the SOD of the set median graph. The results over these three datasets demonstrated that our method is able to obtain good approximations of the median graph. In addition we show that there is a direct correlation between the quality of the medians and the classification accuracy we can achieve with them.

Classification experiments show that with the use of the median graph we can achieve similar results as in the 1NN classifiers but with less number of comparisons. This result is useful in large databases, where the number of comparisons in the 1NN classifier may be unfeasible. Furthermore, we have proven that in some cases, the median approach can slightly outperform the set median approach, which tells us that the median can potentially be a better representative of a set. In addition, we have shown that a mixed solution in between the median and the 1NN classifier is able to obtain better classification accuracy than the 1NN classifier with still a reduced number of comparisons. These last classification results are similar of those obtained in [25]. This makes the graph embedding in vector spaces procedure a very promising way to improve the classical leaning algorithms with the power of graphs.

Nevertheless, there are still a number of issues to be investigated in the future. One of the points that introduces some distortion in the median computation is the triangulation procedure. In this way it would be interesting to try to find more accurate triangulation procedures in order to improve the computed medians. It is also of interest the application of these medians to other learning algorithms. For instance, up to now, the graph-based *k*-means algorithm for graph clustering has been mainly used with the set median. The new approach we present will permit to perform some clustering tasks using the generalized median instead of the set median, probably finding more accurate centres and improving the response of this algorithm.

Acknowledgements

This work has been partially supported by the CICYT Project TIN2006-15694-C02-02 and by the Spanish research programme Consolider Ingenio 2010: MIPRCV (CSD2007-00018). Kaspar Riesen and Horst Bunke like to acknowledge support from the Swiss National Science Foundation (Project 200021-113198/1).

References

[1] Development Therapeutics Program DTP. AIDS Antiviral Screen, 2004 〈http://dtp.nci.nih.gov/docs/aids/aids_data.html〉.
[2] C. Bajaj, The algebraic degree of geometric optimization problems, Discrete Comput. Geom. 3 (2) (1988) 177–191.
[3] H. Bunke, G. Allerman, Inexact graph matching for structural pattern recognition, Pattern Recognition Lett. 1 (4) (1983) 245–253.
[4] H. Bunke, S. Günter, Weighted mean of a pair of graphs, Computing 67 (3) (2001) 209–224.
[5] D. Conte, P. Foggia, C. Sansone, M. Vento, Thirty years of graph matching in pattern recognition, Int. J. Pattern Recognition Artif. Intell. 18 (3) (2004) 265–298.
[6] C. de la Higuera, F. Casacuberta, Topology of strings: median string is NP-complete, Theor. Comput. Sci. 230 (1–2) (2000) 39–48.
[7] M.F. Demirci, A. Shokoufandeh, Y. Keselman, L. Bretzner, S.J. Dickinson, Object recognition as many-to-many feature matching, Int. J. Comput. Vision 69 (2) (2006) 203–222.
[8] P. Dosch, E. Valveny, Report on the second symbol recognition contest, in: W. Liu, J. Lladós (Eds.), GREC, Lecture Notes in Computer Science, vol. 3926, Springer, Berlin, 2005.
[9] R. Duda, P. Hart, D. Stork, Pattern Classification, second ed., Wiley Interscience, 2000.
[10] M. Ferrer, F. Serratosa, A. Sanfeliu, Synthesis of median spectral graph, Second Iberian Conference of Pattern Recognition and Image Analysis, Lecture Notes in Computer Science, vol. 3523, Springer, Berlin, 2005.
[11] M. Friedman, A. Kandel, Introduction to Pattern Recognition, World Scientific, Singapore, 1999.
[12] K. Grauman, T. Darrell, Fast contour matching using approximate earth mover's distance, in: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004.
[13] S.L. Hakimi, Location Theory, CRC Press, Boca Raton, FL, 2000.
[14] A. Hlaoui, S. Wang, Median graph computation for graph clustering, Soft Comput. 10 (1) (2006) 47–53.
[15] P. Indyk, Algorithmic applications of low-distortion geometric embeddings, in: IEEE Symposium on Foundations of Computer Science, 2001.
[16] X. Jiang, A. Münger, H. Bunke, On median graphs: properties, algorithms, and applications, IEEE Trans. Pattern Anal. Mach. Intell. 23 (10) (2001) 1144–1151.
[17] D. Justice, A.O. Hero, A binary linear programming formulation of the graph edit distance, IEEE Trans. Pattern Anal. Mach. Intell. 28 (8) (2006) 1200–1214.
[18] B. Luo, R.C. Wilson, E.R. Hancock, Spectral embedding of graphs, Pattern Recognition 36 (10) (2003) 2213–2230.
[19] A. Münger, Synthesis of prototype graphs from sample graphs, Diploma Thesis, University of Bern, 1998 (in German).
[20] M. Neuhaus, H. Bunke, An error-tolerant approximate matching algorithm for attributed planar graphs and its application to fingerprint classification, in: A.L.N. Fred, T. Caelli, R.P.W. Duin, A.C. Campilho, D. de Ridder (Eds.), SSPR/SPR, Lecture Notes in Computer Science, vol. 3138, Springer, Berlin, 2004.
[21] M. Neuhaus, K. Riesen, H. Bunke, Fast suboptimal algorithms for the computation of graph edit distance, Joint IAPR International Workshops, SSPR and SPR 2006, Lecture Notes in Computer Science, vol. 4109, Springer, Berlin, 2006.
[22] E. Pekalska, R.P.W. Duin, P. Paclík, Prototype selection for dissimilarity-based classifiers, Pattern Recognition 39 (2) (2006) 189–208.
[23] K. Riesen, H. Bunke, IAM graph database repository for graph based pattern recognition and machine learning, in: SSPR/SPR, 2008.
[24] K. Riesen, H. Bunke, Approximate graph edit distance computation by means of bipartite graph matching, Image Vision Comput. 27 (7) (2009) 950–959.
[25] K. Riesen, M. Neuhaus, H. Bunke, Graph embedding in vector spaces by means of prototype selection, Sixth IAPR-TC-15 International Workshop, GbRPR 2007, Lecture Notes in Computer Science, vol. 4538, Springer, Berlin, 2007.
[26] A. Robles-Kelly, E.R. Hancock, A Riemannian approach to graph embedding, Pattern Recognition 40 (3) (2007) 1042–1056.
[27] A. Sanfeliu, K. Fu, A distance measure between attributed relational graphs for pattern recognition, IEEE Trans. Syst. Man Cybern. 13 (3) (1983) 353–362.
[28] A. Schenker, H. Bunke, M. Last, A. Kandel, Graph-Theoretic Techniques for Web Content Mining, World Scientific Publishing Co., Inc., USA, 2005.
[29] E. Weiszfeld, Sur le point pour lequel la somme des distances de *n* points donnés est minimum, Tohoku Math. J. 43 (1937) 355–386.
[30] D. White, R.C. Wilson, Mixing spectral representations of graphs, in: 18th International Conference on Pattern Recognition (ICPR 2006), Hong Kong, China, IEEE Computer Society, 20–24 August 2006.
[31] R.C. Wilson, E.R. Hancock, B. Luo, Pattern vectors from algebraic graph theory, IEEE Trans. Pattern Anal. Mach. Intell. 27 (7) (2005) 1112–1124.

About the Author—M. FERRER was born in Terrassa, 24 October 1975. He received his telecommunications engineering degree from the Universitat Ramon Llull, La Salle (Barcelona) in 2003. In 2004 he joined the Departament d'Informática i Matemàtiques, Universitat Rovira i Virgili. In 2005 he moved to the Computer Vision Center, Departament de Ciéncies de la Computació, Universitat Autónoma de Barcelona, where he obtained his Ph.D. in June 2008.

About the Author—E. VALVENY is an associate professor at the Computer Science Department of the Universitat Autònoma de Barcelona (UAB), where he obtained his Ph.D. degree in 1999. He is also member of the Computer Vision Center (CVC) at UAB. His research work has mainly focused on symbol recognition in graphic documents. Other areas of interest are in the field of computer vision and pattern recognition, more specifically in the domain of document analysis, including shape representation, character recognition, document indexing and layout analysis. He is currently a member of the IAPR TC-10, the Technical Committee on Graphics Recognition, and IAPR-TC-5 on Benchmarking and Software. He has been co-chair of the two editions of the International Contest on Symbol Recognition, supported by IAPR-TC10. He has worked in several industrial projects developed in the CVC and published several papers in national and international conferences and journals.

About the Author—F. SERRATOSA was born in Barcelona, 24 May 1967. He received his computer science engineering degree from the Universitat Politècnica de Catalunya (Barcelona) in 1993. Since then, he has been active in research in the areas of computer vision, robotics and structural pattern recognition. He received his PhD from the

same university in 2000. He is currently associate professor of computer science at the Universitat Rovira i Virgili (Tarragona). He has published more than 40 papers and he is an active reviewer in some congresses and journals. He is the author of two patents on the computer vision field.

About the Author—K. RIESEN received his M.S. degree in computer science from the University of Bern, Switzerland, in 2006. Currently he is a Ph.D. student and lecture assistant in the research group of computer visions and artificial intelligence at the University of Bern, Switzerland. His research interest is mainly focussed on structural pattern recognition, particularly graph based pattern classification and clustering. He has 18 publications, including two journal papers.

About the Author—H. BUNKE received his M.S. and Ph.D. degrees in computer science from the University of Erlangen, Germany. In 1984, he joined the University of Bern, Switzerland, where he is a professor in the Computer Science Department. He was department chairman from 1992 to 1996, dean of the faculty of science from 1997 to 1998 and a member of the Executive Committee of the Faculty of Science from 2001 to 2003. From 1998 to 2000 he served as 1st vice-president of the International Association for Pattern Recognition (IAPR). In 2000 he also was acting president of this organization. He is a Fellow of the IAPR, former editor-in-charge of the International Journal of Pattern Recognition and Artificial Intelligence, editor-in-chief of the journal Electronic Letters of Computer Vision and Image Analysis, editor-in-chief of the book series on Machine Perception and Artificial Intelligence by World Scientific Publ. Co., advisory editor of Pattern Recognition, associate editor of Acta Cybernetica and Frontiers of Computer Science in China, and former associate editor of the International Journal of Document Analysis and Recognition, and Pattern Analysis and Applications. He received an honorary doctor degree from the University of Szeged, Hungary, and held visiting positions at the IBM Los Angeles Scientific Center (1989), the University of Szeged, Hungary (1991), the University of South Florida at Tampa (1991, 1996, 1998–2007), the University of Nevada at Las Vegas (1994), Kagawa University, Takamatsu, Japan (1995), Curtin University, Perth, Australia (1999) and Australian National University, Canberra (2005). He served as a co-chair of the Fourth International Conference on Document Analysis and Recognition held in Ulm, Germany, 1997 and as a track co-chair of the 16th and 17th International Conference on Pattern Recognition held in Quebec City, Canada, and Cambridge, UK, in 2002 and 2004, respectively. Also he was chairman of the IAPR TC2 Workshop on Syntactic and Structural Pattern Recognition held in Bern 1992, a co-chair of the Seventh IAPR Workshop on Document Analysis Systems held in Nelson, NZ, 2006, and a co-chair of the 10th International Workshop on Frontiers in Handwriting Recognition, held in La Baule, France, 2006. He was on the program and organization committee of many other conferences and served as a referee for numerous journals and scientific organizations. He is on the Scientific Advisory Board of the German Research Center for Artificial Intelligence (DFKI). He has more than 550 publications, including 40 authored, co-authored, edited or co-edited books and special editions of journals.