Soft Comput (2014) 18:393–412
DOI 10.1007/s00500-013-1065-z

METHODOLOGIES AND APPLICATION

A Granular Computing approach to the design of optimized graph classification systems

Filippo Maria Bianchi · Lorenzo Livi · Antonello Rizzi · Alireza Sadeghian

Published online: 12 June 2013

Abstract Research on Graph-based pattern recognition and Soft Computing systems has attracted many scientists and engineers in several different contexts. This fact is motivated by the reason that graphs are general structures able to encode both topological and semantic information in data. While the data modeling properties of graphs are of indisputable power, there are still different concerns about the best way to compute similarity functions in an effective and efficient manner. To this end, suited transformation procedures are usually conceived to address the well-known Inexact Graph Matching problem in an explicit embedding space. In this paper, we propose two graph embedding algorithms based on the Granular Computing paradigm, which are engineered as key procedures of a general-purpose graph classification system. Tests have been conducted on benchmarking datasets relying on both synthetic and real-world data, achieving competitive results in terms of test set classification accuracy.

Communicated by W. Pedrycz.

F. M. Bianchi · L. Livi (✉) · A. Rizzi
Department of Information Engineering, Electronics, and Telecommunications, SAPIENZA University of Rome, Via Eudossiana 18, 00184 Rome, Italy
e-mail: livi@diet.uniroma1.it

F. M. Bianchi
e-mail: bianchi@diet.uniroma1.it

A. Rizzi
e-mail: antonello.rizzi@uniroma1.it

A. Sadeghian
Department of Computer Science, Ryerson University, 350 Victoria Street, Toronto, ON M5B 2K3, Canada
e-mail: asadeghi@ryerson.ca

Keywords Graph-based pattern recognition · Granular Computing · Granular modeling · Inexact Graph Matching · Graph embedding

1 Introduction

Research on inductive modeling has defined many automatic systems able to cope with patterns defined on $\mathbb{R}^n$ (Theodoridis and Koutroumbas 2006). However, many recognition problems coming from interesting practical applications deal directly with *structured patterns*, such as images (Neuhaus and Bunke 2007; Del Vescovo and Rizzi 2007a, b), audio/video signals (Rizzi and Del Vescovo 2006), biochemical compounds (Borgwardt et al. 2005), and metabolic networks (Tun et al. 2006), for instance. Usually, in order to take advantage of the existing data-driven modeling systems, each pattern of a structured domain $\mathcal{X}$ is transformed to an $\mathbb{R}^m$ feature vector by adopting a suitable *explicit preprocessing* function $\phi : \mathcal{X} \to \mathbb{R}^m$. The design of these functions is a challenging problem, mainly due to the implicit *semantic and informative gap* between $\mathcal{X}$ and $\mathbb{R}^m$. A key element to design an automatic system dealing with classification problems on structured domains is the *information granulation and compression* of the input set $\mathcal{X}$, achieved through the definition of suited *information granules* (Bargiela and Pedrycz 2003). Another approach is the one of kernel-based learning machines (Schölkopf and Smola 2002), where the representation of the input data in a high-dimensional embedding space is performed *implicitly*, defining a suitable *valid* kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$.

Labeled graphs are general and flexible structures able to model both topological and semantic information in data. Consequently, the graph-based representation has

been adopted extensively in different contexts. A labeled graph is a tuple $G = (\mathcal{V}, \mathcal{E}, \mu, \nu)$, where $\mathcal{V}$ is the (finite) set of vertices (also referred as nodes), $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges, $\mu : \mathcal{V} \rightarrow \mathcal{L}_{\mathcal{V}}$ is the vertex labeling (total) function with $\mathcal{L}_{\mathcal{V}}$ denoting the vertex-labels set, and $\nu : \mathcal{E} \rightarrow \mathcal{L}_{\mathcal{E}}$ is the edge (total) labeling function with $\mathcal{L}_{\mathcal{E}}$ denoting the edge-labels set. The generality of both $\mathcal{L}_{\mathcal{E}}$ and $\mathcal{L}_{\mathcal{V}}$ permits to represent a broad set of patterns. Each inductive modeling engine that has to deal with labeled graphs as input patterns must be able to calculate effectively, and efficiently, both structural and labels-related commonalities. For this purpose, a suited *graph matching* procedure (Gao et al. 2010; Livi and Rizzi 2012a, b) must be defined, able to act as the basic matching measure for any given pair of graphs of $\mathcal{G}$. Of great interest are Inexact Graph Matching (IGM) procedures that can be defined, from a very high level of abstraction, as nonnegative functions of the form $f : \mathcal{G} \times \mathcal{G} \rightarrow \mathbb{R}^+$. Such a function can be constructed by means of an explicit embedding, $\phi : \mathcal{G} \rightarrow \mathbb{R}^m$, such that the matching between graphs is actually performed through vector distance computations (e.g., Euclidean distance). When the IGM algorithm is conceived in such a way, we talk about *graph embedding* (Livi and Rizzi 2012a, b).

In this paper, we describe a general purpose graph-based classification system which relies on two new graph embedding methods. We focus on a specific methodology to design the explicit embedding function $\phi(\cdot)$ following the Granular Computing paradigm (Bargiela and Pedrycz 2003; Pedrycz 2010). To prove the effectiveness of the proposed system, we provide a wide experimental evaluation over synthetic and real-world benchmarking datasets of labeled graphs, focusing the comparison with other state-of-the-art systems mainly on the test set classification results.

The paper is organized as follows. Section 2 briefly introduces Granular Computing as a data analysis paradigm. In Sect. 3 we introduce some state-of-the-art approaches to the IGM problem (Sect. 3.1), focusing in Sect. 3.2 on particular IGM algorithms belonging to the graph embedding family. Throughout Sect. 4 we discuss the details of the proposed Granular Computing based graph classification system. The experimental evaluation on synthetic and well-known benchmarking datasets is carried out in Sect. 5. Finally, in Sect. 6 we draw our conclusions, delineating the future directions.

2 Brief introduction to Granular Computing and modeling

Granular Computing (GrC) is a novel paradigm in the broad domain of information processing. The analysis of complex data is usually characterized by the need of different levels of representation of the underlying system or process. The identification of those representation layers is guided by suited objectives, aimed at the recognition of peculiar regularities that can characterize the data at hand. The low level entities are often described in terms of features that can be either given a priori or extracted from the system. The GrC approach to complex data modeling is driven by the aim of representing compactly such entities that are *indistinguishable* at the current level of abstraction of the system: these groups of low level entities are called *information granules* (Bargiela and Pedrycz 2003; Bello et al. 2008; Pedrycz 2010). The indistinguishability property of the low level entities has also other implications that point beyond the pure *dissimilarity* based aggregation. In fact, information granules represent aggregated data conveying a proper and homogeneous *semantic* interpretation of system/process (Pedrycz 2010). Data can be observed with different levels of granularity in the same way an image can be viewed at different resolutions. In a data analysis performed at highly detailed level, small features become relevant, while in a lower resolution analysis it is possible to find more aggregated features that characterize the data. The proper granulation level depends on the type of data, but also on the type of problem and analysis to be faced. For this very reason, information granules with different aggregation levels can be extracted from the same input data.

In complex system modeling, the possibility to analyze data samples (and thus the system itself) at different granulation levels is a key point. Systems models can be expressed (and synthesized) at different granulation levels, relying on atomic elements (symbols) expressed in the corresponding semantic level. In fact, complex input-output relations can be difficult to discover with a wrong information aggregation procedure, while they can be easily expressed in terms of the right set of symbols. As instance, let us consider a data mining problem in a bioinformatics context, where we want to discover the causal relation between some complex functions in cell membranes with respect to its structure. Depending on the nature of the underlying process to be modeled by a data-driven procedure, we can identify at least three different levels of granulation. The first one relies on single atoms, for example considering hydrogen atoms H to be a salient feature in the database. The second one considers chemical groups, such as the *methylene* (CH_2), as the fundamental elements to be used in system description and modeling. The third one defines cell membrane structure description in terms of macromolecules, such as *phospholipids*, *glycolipids*, and *cholesterols* (see Fig. 1). Depending on the complexity of the process to be modeled, in some cases it is useful to define a set of symbols, to be used as basic

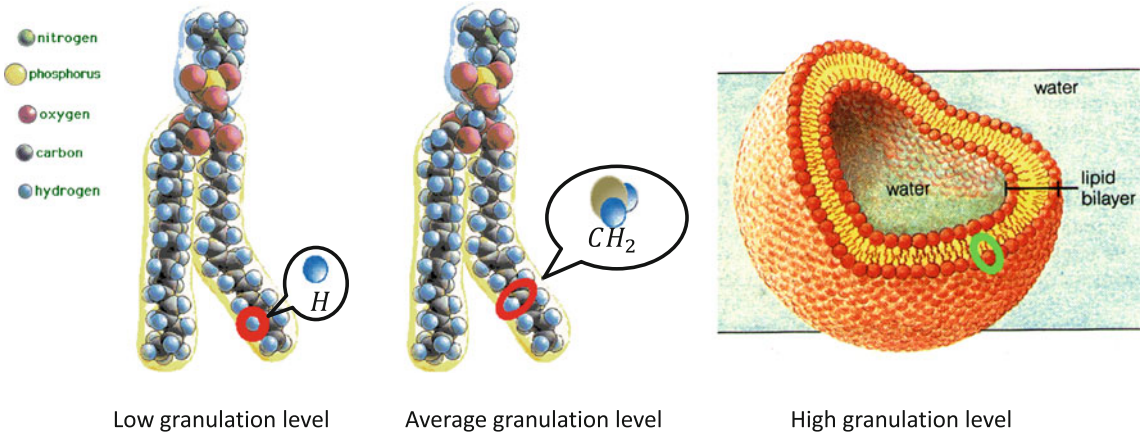


Fig. 1 Complex correlations between some functional property of the cellular membrane and its inner chemical structure can be better discovered and described when representing data at hand in the most suited granulation level. Depending on the modeling task at hand, best atomic information granules can be found at atomic level (*left*), chemical compounds level (*center*), or at a higher level involving more complex macromolecules (*right*)

modeling elements, belonging to (a few) different granulation levels. The availability of an automatic procedure able to determine the best granulation level is essential in data-driven complex systems modeling.

3 Graph-based recognition algorithms and systems

3.1 State-of-the-art approaches of IGM

The aim of IGM algorithms is to determine the matching degree of two input labeled graphs considering both structural and semantic information, i.e., the content of the labels. The challenge is clear, yet very difficult, and consists in obtaining a good evaluation of *how much* the two graphs are similar. In the technical literature, the problem is coped by confronting the graphs directly on their domain $\mathcal{G}$, or producing a new representation of them, that is, an embedding into a suitable space (Livi and Rizzi 2012a, b). These matching algorithms are also called graphs *dissimilarity* or *similarity* measures, depending on the semantic of the specific method. The definition of a (dis)similarity measure between graphs permits to perform recognition and learning tasks with standard tools, such as the k-NN classifier, (Fuzzy) Neural Networks, or Kernel Machines (Theodoridis and Koutroumbas 2006).

Livi and Rizzi (2012a, b) distinguish three mainstream approaches for the IGM problem: Graph Edit Distance (GED), Graph kernels, and Graph embedding. GED-based algorithms (Fankhauser et al. 2011; Gao et al. 2008; Xiao et al. 2008; Neuhaus and Bunke 2007; Neuhaus et al. 2006) search for what is called the minimum cost *edit path* among two input graphs, i.e., a sequence of basic *edit operations* on both vertices and edges, taking into account also the labels. Usually, these approaches are very flexible and adaptable to a wide range of contexts, requiring only the definition of suited problem-dependent dissimilarity measures in both the vertex end edge label spaces. Graph kernels functions (Livi et al. 2012b; Gärtner 2008; Borgwardt et al. 2005; Kashima et al. 2003) are conceived to exploit the famous *kernel trick* property of *positive definite* (pd) kernels. This property permits to employ the family of kernel machines (e.g., the well-known Support Vector Machines) on the domain of graphs $\mathcal{G}$ (Schölkopf and Smola 2002). A recent interesting development in this field is the establishment of the so-called *information-theoretic kernels* (Carli et al. 2012; Príncipe 2010; Martins et al. 2009), aimed at the definition of pd kernel functions on probability distributions. Finally, graph embedding algorithms (Livi et al. 2012a; Gibert et al. 2011; Riesen and Bunke 2010; Del Vescovo and Rizzi 2007a, b) are in some sense a generalization of graph kernels. Indeed they explicitly develop an *embedding space* $\mathcal{D}$, enabling the possibility to inspect and modify the processed data with further analysis. Moreover, we will see that they are usually hybridized formulations, based on a *core* matching procedure operating directly on $\mathcal{G}$. In this scenario of explicit embeddings, techniques based on the *dissimilarity representation* of the input set have found wide application in different contexts (Carli et al. 2010; Riesen and Bunke 2010; Batista et al. 2010; Pekalska and Duin 2005).

3.2 Graph embedding approaches

A graph embedding algorithm consists in defining explicitly a mapping function $\phi : \mathcal{G} \rightarrow \mathcal{D}$, where $\mathcal{D}$ is a kind of geometric space, such as the usual Euclidean space $\mathcal{D} \subseteq \mathcal{R}^n$. Different generalized representations have been discussed

by Pekalska and Duin (2005), in the so-called theory of *dissimilarity representations*. This powerful approach consists in deriving a dissimilarity matrix $\mathbf{D}$ of the input set $\mathcal{X}$, where $D_{ij} = d(x_i, x_j)$, with $x_i, x_j \in \mathcal{X}$, and $d(\cdot,\cdot)$ is a suited dissimilarity function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$. Then, the embedding space $\mathcal{D}$ is derived elaborating the matrix $\mathbf{D}$ by means of transformation, projection, and normalization techniques. Other approaches embed a graph into a *Riemannian manifold*, using metric properties of differential geometry operators to obtain a distance measure (Robles-Kelly and Hancock 2007; Escolano et al. 2011).

In the following, we describe three different state-of-the-art explicit embedding techniques for labeled graphs, which are closely related to the contribution of this paper.

3.2.1 GED-based dissimilarity embedding

The approach, widely described by Riesen and Bunke (2010), consists in producing a dissimilarity representation Pekalska and Duin (2005) for the input graphs $\mathcal{G}$ using a GED as core dissimilarity algorithm. Given a set of labeled graphs $\mathcal{G} = \{G_1, \ldots, G_t\}$, a core GED-based dissimilarity function $d : \mathcal{G} \times \mathcal{G} \to \mathbb{R}^+$, a prototypes set $\mathcal{P} = \{P_1, \ldots, P_n\}, \mathcal{P} \subseteq \mathcal{G}$, the embedding vector of each graph G is defined as

$$\phi^{\mathcal{P}}(G) = [d(G, P_1), \ldots, d(G, P_n)]^T, \quad \forall G \in \mathcal{G}, \tag{1}$$

where the superscript $\mathcal{P}$ remarks that the embedding is relative to the chosen set of prototypes $\mathcal{P}$. Consequently, the whole input set $\mathcal{G}$ is mapped into a dissimilarity space $\mathcal{D}$. In this scenario, prototypes selection strategies play a crucial role (Riesen and Bunke 2010).

The matching value $d(G_1, G_2)$ of two given input graphs is computed executing exactly $2|\mathcal{P}|$ IGM computations using the direct IGM algorithm $d(\cdot,\cdot)$, needed to produce the respective embedding vectors.

3.2.2 Embedding of sequenced graphs

Livi et al. (2012a) described a novel graph embedding method employed in a graph-based classification system. The embedding method can be schematized by means of two *mapping* functions. The first one, say $f_1 : \mathcal{G} \to \mathcal{S}$, maps a graph $G \in \mathcal{G}$ to a sequence of vertex labels $s \in \mathcal{S}$. The second one, say $f_2 : \mathcal{S} \to \mathcal{D}$, maps each sequence $s \in \mathcal{S}$ to a numeric vector $\underline{\mathbf{h}} \in \mathcal{D}$, where usually $\mathcal{D} \subseteq \mathbb{R}^n$. The first transformation, i.e., the mapping $f_1(\cdot)$, is performed applying a seriation algorithm to the input graphs, such as an eigenvectors-based algorithm (Robles-Kelly and Hancock 2005). The second transformation is performed by the means of the GRADIS (GRanular computing Approach for DIscrete Sequences) procedure that performs mining and embedding operations on the sequenced graphs set $\mathcal{S}$ (Livi et al. 2012a).

It is worth noting that the nature of the set $\mathcal{S}$ is directly defined by the vertices labels set $\mathcal{L}_{\mathcal{V}}$. Practically, it is possible to process any type of sequence (i.e., multidimensional time series, sequence of complex events, and so on) for which it is possible to define a suited dissimilarity function of the type $d : \mathcal{L}_{\mathcal{V}} \times \mathcal{L}_{\mathcal{V}} \to \mathbb{R}^+$. It is important to underline that once obtained $\mathcal{S}$, all useful information stored in the edge labels is definitively unavailable to subsequent processing stages. To this end, this information should be used appropriately in the seriation stage through the definition of a meaningful norm on the specific edge labels set (i.e., the set $\mathcal{L}_{\mathcal{E}}$).

Notwithstanding the potentialities of an explicit embedding approach, like the one provided by the GRADIS procedure, once the seriation stage is performed and the set of sequences $\mathcal{S}$ is derived, it is possible to apply also a more direct sequence matching scheme, such as a classifier based on the k-NN rule, equipped with the generalized global alignment *Dynamic Time Warping* (DTW) algorithm (Sakoe 1978), or an *SVM* classifier equipped with a suited *string kernel* (Yu and Hancock 2006), tailored to the specific sequence type.

3.2.3 GrC based symbolic histograms representation

A graph embedding approach based on *symbolic histograms* (Del Vescovo and Rizzi 2007a, b; Rizzi and Del Vescovo 2006) consists in identifying a set of *frequent subgraphs* $\mathcal{A} = \{g_1, \ldots, g_m\}$ of the input set $\mathcal{G}$. Let $G_1 = (\mathcal{V}_1, \mathcal{E}_1, \mu_1, \nu_1)$ and $G_2 = (\mathcal{V}_2, \mathcal{E}_2, \mu_2, \nu_2)$ be two labeled graphs. Graph G_1 is a subgraph of G_2, written also as $G_1 \subseteq G_2$, if these conditions hold:

- $\mathcal{V}_1 \subseteq \mathcal{V}_2$,
- $\mathcal{E}_1 \subseteq \mathcal{E}_2$,
- $\mu_1(v) = \mu_2(v), \quad \forall v \in \mathcal{V}_1$, and
- $\nu_1(e) = \nu_2(e), \quad \forall e \in \mathcal{E}_1$.

The computation of $\mathcal{A}$ is performed by means of a *clustering ensemble* procedure applied on an appropriate set of subgraphs extracted from the training set. A suited adaptive filtering of the obtained partitions is performed with the aim of selecting only compact and populated clusters of subgraphs. The set of subgraphs defining $\mathcal{A}$ is eventually derived by compressing the information of the corresponding filtered clusters considering the representative subgraphs only; the representative subgraph of a cluster can be conveniently computed by means of the well-known Minimum Sum Of Distances (MinSOD) technique (Del Vescovo et al. 2011). Once obtained $\mathcal{A}$, called *symbols alphabet*, the embedding consists in a

mapping function $\phi^{\mathcal{A}} : \mathcal{G} \rightarrow \mathbb{R}^m$ that assigns to each graph G_i an integer-valued vector $\underline{\mathbf{h}}_i$ called *symbolic histogram*, which is defined as follows:

$$\underline{\mathbf{h}}_i = \phi^{\mathcal{A}}(G_i) = [\text{occ}(g_1), \ldots, \text{occ}(g_m)]^T, \quad \forall G_i \in \mathcal{G}. \qquad (2)$$

The function $\text{occ}(\cdot)$ counts the *occurrences* of each representative subgraphs $g_j \in \mathcal{A}$ in the input graphs. The occurrence of a subgraph g_j into a graph G_i is evaluated using a weighted GED-based *core* IGM procedure $d(\cdot, \cdot)$ (Del Vescovo and Rizzi 2007a, b). If the matching score reaches the symbol-dependent threshold τ_j, the occurrence is considered.

Each symbol $g_j \in \mathcal{A}$ forms thus a granule of information (Bargiela and Pedrycz 2003), containing both metric (i.e., information related to the intra-cluster dissimilarities) and semantic information about the aggregated data it represents (the MinSOD element is a compressed cluster representation directly interpretable by field experts). The embedding space $\mathcal{D}$ is defined as the vector space containing all the symbolic histogram representations $\{\underline{\mathbf{h}}_i\}_{i=1}^n \subset \mathcal{D} \subseteq \mathbb{R}^m$. The procedure for extracting $\mathcal{A}$ from the training set can be seen as an unsupervised feature extraction algorithm, since during the computation of the alphabet $\mathcal{A}$ no information about the classes is considered.

4 Proposed GrC based graph classification system

In this section we describe the graph classification system that we called GRALG (GRaunlar computing Approach for Labeled Graphs), which belongs to the family of the GrC-based graph embedding techniques, introduced in Sect. 3.2.3. In fact, it relies on the computation of the symbols alphabet $\mathcal{A}$, extracting from the input (training) set suited information granules expressed in terms of frequent subgraphs, which are identified as the representatives of compact and populated clusters. Such clusters are generated trough a clustering ensemble procedure, which in our algorithm has been implemented elaborating on the well-known Basic Sequential Algorithmic Scheme (BSAS) (Theodoridis and Koutroumbas 2006). GRALG performs an optimized granulation of the input dataset of labeled graphs by means of *automatic tuning of system parameters* and *symbols alphabet compression* stages. Therefore, the overall training process of GRALG consists in the automatic determination and optimization of the embedding space (i.e., the symbolic histograms representation) and in the subsequent synthesis of a suited feature-based classifier. Since the training procedure of GRALG relies on a cross-validation approach, the initial training set is split into a reduced training set $\mathcal{S}_{tr}$ and a validation set $\mathcal{S}_{vs}$. The objective function—to be maximized—used in the system optimization is the classification accuracy achieved on $\mathcal{S}_{vs}$.

The embedding procedure employs a *parametric* core IGM algorithm that operates directly on the input space $\mathcal{G}$. As will be deeply discussed in the following, the behavior of the embedding procedure depends, besides the weights characterizing the core IGM algorithm, on several critical parameters (denoted as Γ). Demanding their setting to the user implies a deep knowledge of the processed data, often requiring in turn a long and inappropriate "trials and errors" session. Therefore, the first stage of the GRALG training process consists in tuning the system's parameters Γ through a genetic algorithm, synthesizing a first optimized version of the alphabet $\mathcal{A}$. The second stage instead consists in compressing the derived optimized alphabet $\mathcal{A}$ selecting relevant features, which in turn contributes to the reduction of the complexity of the procedure as a whole: the lower the size of the alphabet, the simpler the resulting learned model of the data. This second optimization step aims to determine which is the smallest set of symbols that is required for generating the most significant alphabet $\mathcal{A}$ for the data at hand. Feature selection is then performed on the embedding space derived from $\mathcal{A}$, guiding the selection using a fitness function defined as a linear convex combination of the classification accuracy computed on $\mathcal{S}_{vs}$, and a term related to the number of selected symbols. For this very reason, the developed feature selection method falls in the wrapper-based family (Zhao et al. 2011).

As shown in Fig. 2, once the two-stage GRALG training process is terminated, a new test pattern G_x can be classified straightforwardly by using the previously synthesized feature-based classifier on the corresponding embedding space. This computation is fast, compared to the time required for the whole GRALG synthesis, making the classification of new test patterns a quick task once $\mathcal{A}$ has been defined. Indeed, given $\mathcal{A}$, the number of IGM evaluations for computing the dissimilarity between any two graphs G_i and G_j is given by $|\mathcal{A}| \cdot (|\text{expand}(G_i)| + |\text{expand}(G_j)|)$, where the function $\text{expand}(\cdot)$ extracts from a graph the (not complete) set of its subgraphs. Computing the distance between two input graphs is hence linear in the number of derived symbols–information granules.

4.1 High-level explanation of the method

The key assumption underlying the GRALG classification system is that the classes can be discriminated in terms of the frequent subgraphs extracted from the training set $\mathcal{S}_{tr}$. If such a characterization exists, GRALG is able to develop a new vector representation (embedding) of the input graphs in terms of symbolic histograms, which in turns contain the information needed to synthesize a suited classification rule. Moreover, the symbolic histograms

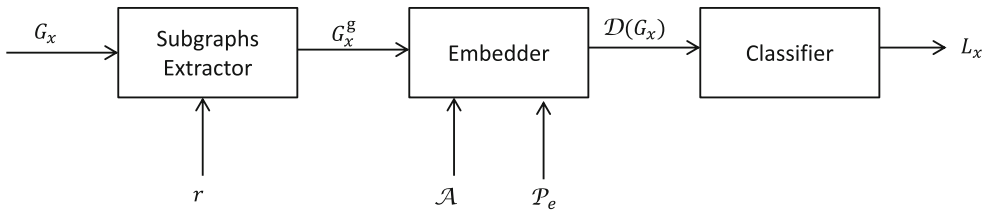


Fig. 2 A new graph G_x, in order to be processed by the feature-based classifier, must be transformed into an embedding vector: at first all the subgraphs, up to order r, will be extracted in a structure G_x^g which will be converted into a vector representation $\mathcal{D}(G_x)$ using the alphabet $\mathcal{A}$ and the learned parameters $\mathcal{P}_e \subset \Gamma$ related to the embedder block. The embedding vector will be then processed by the feature-based classifier that will assign a label L_x to the pattern

representation contains *interpretable* information, which can be exploited by field experts to derive insights for the problem at hand. For example, it is possible to understand which are the features that characterize a class, interpreting the distribution of the alphabet symbols in the histograms that represent its graphs. It is important to underline that the clustering ensemble procedure is in charge of defining a set of clusters of frequent subgraphs, which are candidate to become meaningful information granules (symbols). Optimization stages are performed in order to discover those information granules actually related to the classification task at hand. A symbol is therefore not just a representative of a set of similar subgraphs found in the training set: it is an information granule with a specific semantic value. The semantic value is attributed by the system during the first and second optimization stages, when it is recognized as useful to the final classification task, usually working in some *logic conjunction* with other symbols. Note that the same frequent subgraph discovered as a symbol for a given classification problem, can be completely uninformative for another problem.

Figure 3 shows an example describing the mechanism driving the GRALG classification system. Each input labeled graph G_i is represented in terms of the symbols of $\mathcal{A}$. To this aim, the j-th component of the symbolic histogram associated to G_i contains the number of times the symbol $a_j \in \mathcal{A}$ has been recognized into G_i (see Sect. 3.2.3 for more details). Reasonably, graphs belonging to the same class will be characterized by an analogue distribution in terms of symbol occurrences, while graphs pertaining to different classes will show a discriminative representation. The similarity in terms of symbol occurrences will be reflected by the Euclidean distance of the corresponding symbolic histograms. A suited feature-based classifier (e.g., a neuro-fuzzy network Rizzi et al. 2002) can be trained once the symbolic histograms representation is completely defined.

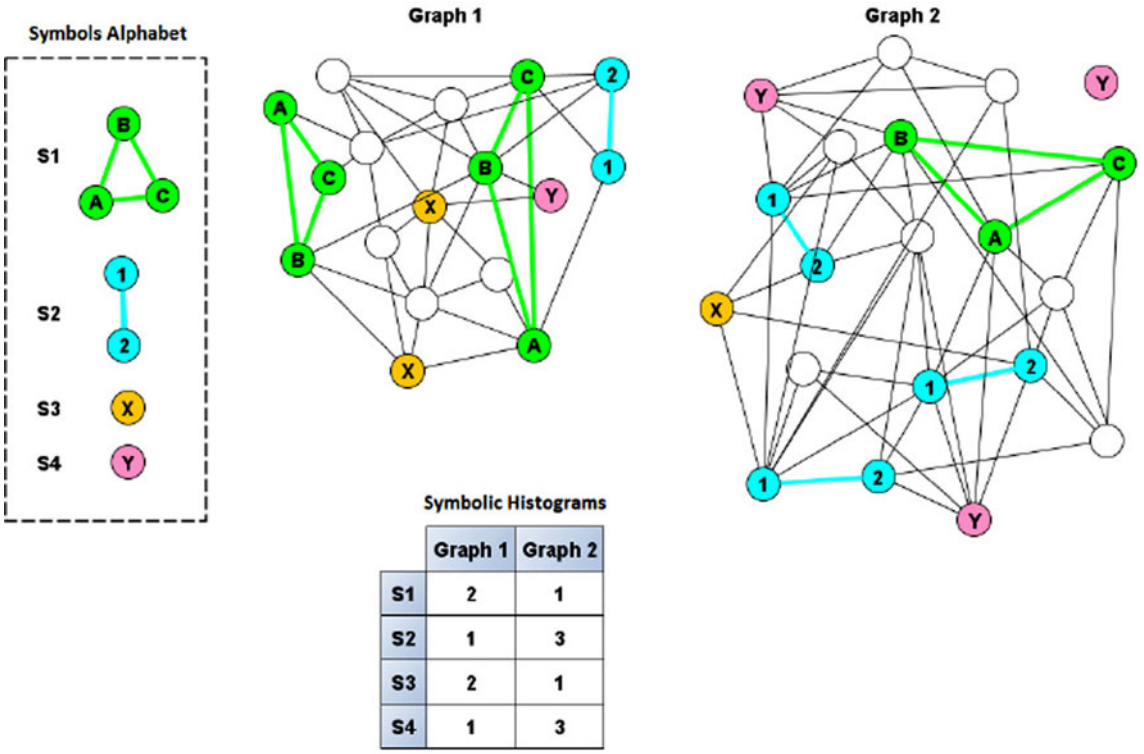
4.2 The adopted core IGM algorithm

In GRALG, we used the GED algorithm known as weighted *Best Match First* (wBMF) as the core dissimilarity measure between labeled subgraphs, mainly for its good trade-off between efficacy and computational complexity (Livi and Rizzi 2012a, b). The matching algorithm consists in computing an assignment of the vertices of G_1 with respect to G_2 on the base of a greedy strategy. Vertices with lowest labels' dissimilarity value are assigned for substitution in each iteration, without the possibility to modify this decision. If G_1 has more (less) vertices than G_2, then additional deletion (insertion) edit operations are considered in the overall edit costs. The operations on the edges are induced considering the ones performed on the vertices. In this paper, we considered the weighting scheme for the edit operations based on six parameters falling within the [0, 1] range, which are used to modulate the importance of the substitution, insertion, and deletion edit operations for both vertices and edges.

The normalization of the outcomes of the core IGM procedure within the [0, 1] interval is a very important aspect in the GRALG system. To this aim, we assume that both vertices and edges dissimilarity functions (i.e., the dissimilarity functions for the vertices and edges labels) have been defined to return values within the [0, 1] interval. If r is the maximum order of the (undirected) graphs to be compared, the maximum number of edit operations required for transforming a complete graph K_i into another complete graph K_j is upper bounded by $\delta = r + r(r-1)/2$. Consequently, fixing to 1 insertion and deletion costs for both vertices and edges, a simple way to obtained an IGM function normalized in [0, 1] consists in dividing the returned value by δ. As we will see in the following, the proposed classification system is conceived to perform IGM computations only between subgraphs of a given maximum order r.

4.3 Synthesis of the GRALG classification model

The key component of the GRALG classification system is the graph embedding procedure, which is founded on the GrC-based technique described in Sect. 3.2.3; Fig. 4 delineates its schematic.



	Graph 1	Graph 2
S1	2	1
S2	1	3
S3	2	1
S4	1	3

Fig. 3 In this example, let us suppose to have found four *symbols* (frequent subgraphs) in the training set $\mathcal{S}_{tr}$: namely S_1, S_2, S_3, and S_4. Consequently, we will represent each input graph G_i with a symbolic histogram of four dimensions/components, each of which will count the number of occurrences of the corresponding *symbol* in the input graph. For example, the graph G_1 contains two occurrences of S_1, one occurrence of S_2, two occurrences of S_3, and finally one occurrence of S_4; thus the associated symbolic *histogram* will then be the following vector $[2, 1, 2, 1]^T$

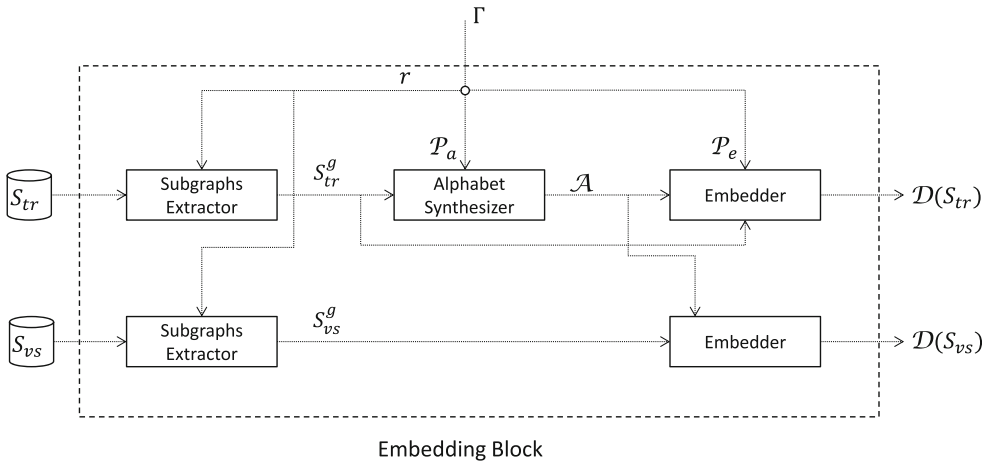


Fig. 4 The overall graph embedding procedure, defined by a set of system's parameters Γ, transforms graphs into $\mathbb{R}^m$ vectors synthesizing the symbols alphabet $\mathcal{A}$

The subgraphs extractor generates all the subgraphs $\mathcal{S}^g_{tr}$ and $\mathcal{S}^g_{vs}$, up to order r, expanding the graphs in $\mathcal{S}_{tr}$ and $\mathcal{S}_{vs}$, respectively. The subgraphs $\mathcal{S}^g_{tr}$ of the training set $\mathcal{S}_{tr}$ are used for computing the alphabet $\mathcal{A}$ (i.e., the set of symbols–information granules), which is related to the recurrent substructures of the training set, using a specific set of parameters $\mathcal{P}_a \subset \Gamma$. A cluster C_k is characterized by a cost function $F(\cdot)$ defined as:

$$F(C_k) = \eta\Phi(C_k) + (1-\eta)\Theta(C_k), \quad \eta \in [0,1]. \tag{3}$$

Equation 3 is defined as a convex linear combination of two cluster's descriptors: the *compactness* $\Phi(C_k)$ and the *size* $\Theta(C_k)$ costs. By defining the subgraph g_k as the representative of the cluster C_k according to the following equation:

$$g_k = \arg\min_{g_j \in C_k} \sum_{g_i \in C_k} d(g_j, g_i), \tag{4}$$

the compactness cost of C_k is defined in turns as the average intra-cluster distances with respect to g_k:

$$\Phi(C_k) = \frac{1}{|C_k| - 1} \sum_{i \neq k} d(g_k, g_i). \quad (5)$$

The size cost instead evaluates if the cluster is sufficiently populated:

$$\Theta(C_k) = 1 - \frac{|C_k|}{|S_{tr}^g|}. \quad (6)$$

Each derived cluster of subgraphs defines a candidate symbol for $\mathcal{A}$. In particular, each cluster C_k generates a candidate symbol which is defined by a triple $(g_k, 1 - F(C_k), \tau_k)$; g_k is the MinSOD subgraph of the cluster C_k (see Eq. 4 for the formulation), $1 - F(C_k)$ is the overall quality of the cluster, and finally τ_k is the symbol-dependent recognition threshold used by the embedder that is defined as $\tau_k = \Phi(C_k) \cdot \epsilon$, where $\epsilon \geq 1$ is a user-defined tolerance parameter. Note that in this way, a cluster is effectively modeled by a single subgraph g_k, which represents operatively and compactly the symbol–information granule. The final step in the alphabet synthesis is based on a filtering threshold τ_F, used to remove (candidate) symbols with low quality. If the cost $F(C_k)$ related to the cluster C_k (equivalently, to the symbol g_k) is lower or equal to the symbols filtering threshold τ_F, the symbol is retained, otherwise it is discarded. Once the filtered $\mathcal{A}$ is obtained, the alphabet is used for building the embeddings $\mathcal{D}(\mathcal{S}_{tr})$ and $\mathcal{D}(\mathcal{S}_{vs})$ of both training and validation set with the embedder component, which produces the symbolic histograms related to the input graphs—See Algorithm 1 for the pseudo-code related to the symbolic histograms representation.

Algorithm 1 Symbolic histograms representation of the input graphs.

Input: The input set of graphs $\mathcal{S}$, the synthesized symbols alphabet $\mathcal{A}$, the maximum subgraphs order r
Output: A set of symbolic histograms representation $\mathcal{D}(\mathcal{S})$

```
for all G_i ∈ S do
    Initialize the relative zero-valued symbolic histogram h^(i)
    Extract all the subgraphs of G_i up to oder r in N_{G_i}
    for all g_j ∈ A do
        for all n_k ∈ N_{G_i} do
            Compute the IGM value d_jk = d(g_j, n_k)
            if d_jk ≤ τ_j then
                h_j^(i) = h_j^(i) + 1
            end if
        end for
    end for
    D(S) = D(S) ∪ {h^(i)}
end for
```

With the aim of tuning the herein described overall graph embedding procedure (alphabet synthesis and representation based on symbolic histograms), a classification model is then learned on $\mathcal{D}(\mathcal{S}_{tr})$, evaluating the performance of the overall embedding as a convex linear combination of the classification accuracy obtained on $\mathcal{D}(\mathcal{S}_{vs})$, and a term related to the number of selected symbols. It is worth to stress that once the input graphs have been embedded, any feature-based classifier can be employed directly. Accordingly, in the following two subsections we detail the two stages characterizing the synthesis of the GRALG classification system. The first one, discussed in Sect. 4.3.1, involves the computation of an optimized symbols alphabet $\mathcal{A}$, while the second one, discussed in Sect. 4.3.2, focuses on the feature selection, i.e., the identification of a relevant and essential subset of $\mathcal{A}$. In both optimization stages we relay on evolutionary global optimization techniques, since the objective function to be maximized is not known in closed form. Consequently it is not possible to employ derivatives-based optimization techniques. Specifically, we adopted a standard version of a genetic algorithm.

4.3.1 *Synthesis of the optimized alphabet*

The genetic algorithm optimizes a code $\underline{\mathbf{c}}^{\mathcal{P}}$ which contains the values of the parameters used to determine the optimized alphabet $\mathcal{A}_{opt}$. The parameters set $\mathcal{P} = \mathcal{P}_a \cup \mathcal{P}_e$ is composed by the six weights of the GED-based core IGM procedure, and two additional parameters used in the symbols extraction stage: the maximum number of allowed clusters that can be generated by the clustering ensemble procedure (which is tuned to avoid overfitting), and the symbols filtering threshold τ_F.

The following optimization procedure consists in determining a sequence of candidate symbols alphabets $\mathcal{A}_i$. Each $\mathcal{A}_i$ is used for building an embedding of both

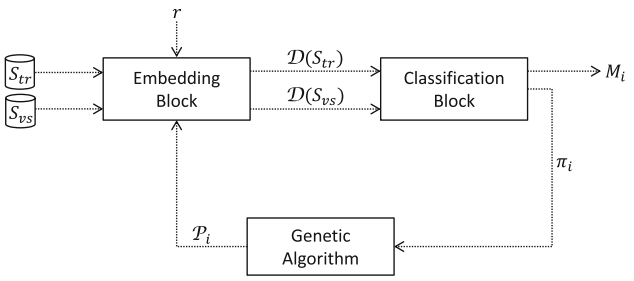


Fig. 5 In this optimization cycle, the genetic algorithm generates an instance $\mathcal{P}_i$ which is a configuration of the parameters $\mathcal{P}_a$ and $\mathcal{P}_e$ used by the embedding block for generating embeddings of $\mathcal{S}_{tr}$ and $\mathcal{S}_{vs}$. The first dataset is then used for learning a classification model M_i, whose performance π_i is evaluated on the latter (classification block). Performance is used for computing the fitness of $\mathcal{P}_i$, guiding the next evolutions of the genetic algorithm

training $\mathcal{D}_i(\mathcal{S}_{tr})$ and validation $\mathcal{D}_i(\mathcal{S}_{vs})$ set. A classifier M_i is then trained using $\mathcal{D}_i(\mathcal{S}_{tr})$ and the achieved recognition rate (*RR*) on $\mathcal{D}_i(\mathcal{S}_{vs})$ is considered as the fitness $f(\underline{\mathbf{c}}_i^{\mathcal{P}})$ of the code $\underline{\mathbf{c}}_i^{\mathcal{P}}$. The optimization ends when a genetic code with a fitness higher than a user-defined threshold τ_{RR} is generated (or when a maximum number of iterations is reached). The code with highest fitness contains the optimal parameters $\mathcal{P}_{opt}$ used for synthesize $\mathcal{A}_{opt}$, from which optimal embeddings $\mathcal{D}_{opt}(\mathcal{S}_{tr})$ and $\mathcal{D}_{opt}(\mathcal{S}_{vs})$ are generated (Fig. 5). The optimization procedure is outlined in Algorithm 2.

projection mask μ which reduces each symbolic histogram $\underline{\mathbf{h}}$ into a lower dimensional vector $\hat{\underline{\mathbf{h}}}$, according to the selected symbols in $\mathcal{A}_{opt}$. The optimal mask μ_{opt} is determined trough a second optimization procedure, aiming in discovering significant and essential subsets of symbols semantically related to the classification problem at hand, improving at the same time the overall generalization capability, according to the well-known Ockham's Razor criterion (Rizzi et al. 2002; Theodoridis and Koutroumbas 2006). The genetic code in this case coincides with the mask μ_j (a tuple of binary digits), which, according to the selected symbols of the induced subset $\mathcal{A}_j$, projects the embeddings $\mathcal{D}_{opt}(\mathcal{S}_{tr})$ and $\mathcal{D}_{opt}(\mathcal{S}_{vs})$, derived from the previous optimization, in the corresponding subspaces $\widehat{\mathcal{D}}^j_{opt}(\mathcal{S}_{tr})$ and $\widehat{\mathcal{D}}^j_{opt}(\mathcal{S}_{vs})$. A classifier M_j is once again trained with $\widehat{\mathcal{D}}^j_{opt}(\mathcal{S}_{tr})$ and its performances are evaluated classifying the embedded validation set $\widehat{\mathcal{D}}^j_{opt}(\mathcal{S}_{vs})$. This time the fitness $g(\mu_j)$ is a convex linear combination of the achieved recognition rate of the classifier on $\widehat{\mathcal{D}}^j_{opt}(\mathcal{S}_{vs})$ and the cost of the mask $mc(\mu_j)$, which is defined as:

$$mc(\mu_j) = \frac{|\mathcal{A}_j|}{|\mathcal{A}|}. \tag{7}$$

Optimization ends when a mask with a fitness higher than a given threshold τ_{FS} is generated (or when a maximum number of iterations is reached). Eventually, the mask with highest fitness μ_{opt} is used for generating the final embeddings for training $\widehat{\mathcal{D}}^*_{opt}(\mathcal{S}_{tr})$ and test $\widehat{\mathcal{D}}^*_{opt}(\mathcal{S}_{ts})$ sets, respectively—see Fig. 6. Algorithm 3 summarizes the feature selection procedure.

Algorithm 2 Parameters optimization procedure for the alphabet synthesis.

Input: Training Set $\mathcal{S}_{tr}$, Validation Set $\mathcal{S}_{vs}$
Output: Optimal Embeddings $\mathcal{D}_{opt}(\mathcal{S}_{tr})$ and $\mathcal{D}_{opt}(\mathcal{S}_{vs})$, $\mathcal{A}_{opt}$
1: Optimize parameters $\mathcal{P}$ with a genetic algorithm where:
- Stop criterion: fitness $f(\mathcal{P}_i) \geq \tau_{RR}$ or maximum number of evolutions is reached
- Individual: parameters $\mathcal{P}_i$
- Fitness computation:
  - Synthesize alphabet $\mathcal{A}_i$ using current parameters $\mathcal{P}_i$
  - Embed the training $\mathcal{D}_i(\mathcal{S}_{tr})$ and validation set $\mathcal{D}_i(\mathcal{S}_{vs})$ graphs using $\mathcal{A}_i$
  - Train a classifier M_i with $\mathcal{D}_i(\mathcal{S}_{tr})$ and compute the *RR* on $\mathcal{D}_i(\mathcal{S}_{vs})$

2: Let $\mathcal{P}_{opt}$ be the parameters extracted from the individual with highest fitness
3: Synthesize $\mathcal{A}_{opt}$ using $\mathcal{P}_{opt}$
4: Build graph embedding $\mathcal{D}_{opt}(\mathcal{S}_{tr})$ and $\mathcal{D}_{opt}(\mathcal{S}_{vs})$ using $\mathcal{A}_{opt}$
5: **return** $\mathcal{D}_{opt}(\mathcal{S}_{tr})$, $\mathcal{D}_{opt}(\mathcal{S}_{vs})$, and $\mathcal{A}_{opt}$

4.3.2 Feature selection algorithm

A second optimization stage is performed using another dedicated genetic algorithm in charge of defining the most significant subset of symbols of $\mathcal{A}_{opt}$ for the problem at hand. Relevant features are selected with a

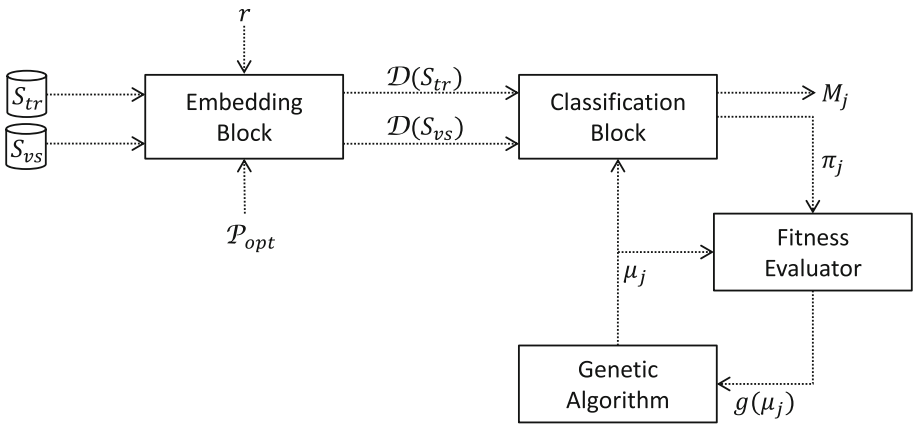


Fig. 6 The embedding vectors generated by the optimal parameters $\mathcal{P}_{opt}$, which are determined by the first optimization procedure, are used for generating a classification model M_j, whose training and validation phase is performed using a projection of the input embedding vectors, according to a mask μ_j. The performances of M_j and the numbers of features selected by μ_j determine the fitness $g(\mu_j)$ of the mask, used by a genetic algorithm for guiding the evolution of the next population

Algorithm 3 Feature selection algorithm.

Input: Embeddings $\mathcal{D}_{opt}(\mathcal{S}_{tr})$ and $\mathcal{D}_{opt}(\mathcal{S}_{vs})$

Output: Projected optimal embeddings $\widehat{\mathcal{D}}^*_{opt}(\mathcal{S}_{tr})$ and $\widehat{\mathcal{D}}^*_{opt}(\mathcal{S}_{ts})$, and the reduced alphabet $\widehat{\mathcal{A}}_{opt}$

1: Optimize the binary mask μ with a genetic algorithm where:
- Stop criterion: fitness $g(\mu_j) \geq \tau_{FS}$ or maximum number of evolutions is reached
- Individual: projection mask μ_j
- Fitness computation:
  - Project $\mathcal{D}_{opt}(\mathcal{S}_{tr})$ into $\widehat{\mathcal{D}}^j_{opt}(\mathcal{S}_{tr})$ (and $\mathcal{D}_{opt}(\mathcal{S}_{vs})$ into $\widehat{\mathcal{D}}^j_{opt}(\mathcal{S}_{vs})$) using the symbols subset $\widehat{\mathcal{A}}_j$ derived by means of μ_j
  - Train a classifier M_j with $\widehat{\mathcal{D}}^i_{opt}(\mathcal{S}_{tr})$ and compute the RR on $\widehat{\mathcal{D}}^j_{opt}(\mathcal{S}_{vs})$
  - Return $g(\mu_j) = \alpha \cdot RR + (1-\alpha) \cdot mc(\mu_j)$

2: Let μ_{opt} be the projection mask associated to the individual with highest fitness

3: Project $\mathcal{D}_{opt}(\mathcal{S}_{tr})$ into $\widehat{\mathcal{D}}^*_{opt}(\mathcal{S}_{tr})$, and $\mathcal{D}_{opt}(\mathcal{S}_{ts})$ into $\widehat{\mathcal{D}}^*_{opt}(\mathcal{S}_{ts})$, using μ_{opt}

4: **return** $\widehat{\mathcal{A}}_{opt}$, $\widehat{\mathcal{D}}^*_{opt}(\mathcal{S}_{tr})$, and $\widehat{\mathcal{D}}^*_{opt}(\mathcal{S}_{ts})$

4.4 Incremental Granules search

In the version of the GRALG system described so far, all subgraphs, up to a given order r, are extracted at the same time from the training set, defining thus the initial set of subgraphs to be partitioned with a clustering ensemble algorithm based on the BSAS (Rizzi and Del Vescovo 2006). Consequently, the set of subgraphs on which is performed the clustering ensemble procedure can be easily characterized by a very high cardinality (depending on the training set size and the graphs topology), which can be difficult to manage, especially from the in-memory footprint viewpoint. In order to speed-up the whole GRALG training procedure, we developed an incremental strategy to populate the set to be clustered. It consists in extracting subgraphs progressively in $r-1$ different iterations. In the following, we refer to the herein described granulation strategy as GRALGv2, and consequently to the version described in previous sections as GRALGv1.

During the first iteration, subgraphs of order $o = 1$ (i.e., single vertices) are extracted and stored in a set $\mathcal{S}^{(1)}$. In general, at iteration i, with $i \leq r$, subgraphs of order $o = i$ are extracted in a set $\mathcal{S}^{(i)}$ containing also all the previously defined lower order subgraphs, i.e. $o < i$. For each $i = 1 \rightarrow r$, the subgraphs of $\mathcal{S}^{(i)}$ are partitioned with the same clustering ensemble algorithm employed in GRALGv1. The clusters characterized by high compactness and cardinality descriptors form the symbols of the alphabet $\mathcal{A}^{(i)}$ of the i-th iteration. Conversely, the discarded clusters are now marked as *bad*. Elements of bad clusters are tracked with a boolean matrix $\mathbf{B}^{(i)} \in \{0,1\}^{m \times n}$, where m is the number of subgraphs and n the

number of partitions derived from $\mathcal{S}^{(i)}$. An entry $b_{l,j}^{(i)}$ is set to 1 if the subgraph g_l is in a bad cluster in the partition P_j. If a subgraph belongs to a percentage of bad clusters that is higher than a threshold τ_{bad}, the subgraph is marked as *bad subgraph*. Bad subgraphs will not be expanded in the next iterations, and, additionally, if their order is 1 (therefore they are vertices), they are removed from the original graphs. In fact, such vertices cannot belong to significant substructures of higher order because, if they are not recurrent as single vertices, more complex substructures which include these vertices cannot be recurrent as well. For the same reason, when a subgraph of order 2 is marked as bad, the edge connecting the two vertices of the subgraph is deleted from the original graph. Using $\mathcal{A}^{(i)}$, the embeddings $\mathcal{D}^{(i)}(\mathcal{S}_{tr})$ and $\mathcal{D}^{(i)}(\mathcal{S}_{vs})$ are built and then a classifier $M^{(i)}$ is trained and tested in the same way as in GRALGv1. If the recognition rate of $M^{(i)}$ is higher than a user-defined threshold τ_{RR} or order i is equal to r (i.e., to the maximum subgraphs order), the learned classification model is returned, otherwise the algorithm proceeds extracting and processing subgraphs of order $i + 1$.

With this granulation strategy, only a subset of subgraphs are expanded in a given iteration and even if the maximum order r is reached, the total number of processed subgraphs will be lower with respect to the original method, since the subgraphs marked as bad are removed dynamically at each iteration. This is an important fact because the bottleneck of the entire procedure is the calculation of the ensemble of partitions, and dealing with less elements will reduce the in-memory footprint as well as the general resources allocation of the procedure.

The procedure for building a classification model starting from training and validation sets is implemented accordingly to the pseudo-code of Algorithm 4. $F(C_k)$ is the function that evaluates the total cost of a cluster (see Eq. 3), using the two cluster descriptors $\Phi(C_k)$ and $\Theta(C_k)$ for the compactness and size costs (see Eqs. 5, 6, respectively). Even this variant depends on some parameters that are tuned by a suited genetic algorithm. Specifically, in addition to the parameters optimized by GRALGv1, the threshold τ_{bad} is considered; as a direct consequence of the granulation procedure performed by GRALGv2, a wrapper-based feature selection is not performed.

Algorithm 4 Granulation and classifier synthesis procedure of GRALGv2.

Input: Training Set $\mathcal{S}_{tr}$, Validation Set $\mathcal{S}_{vs}$
Output: Synthesized classification model M

Subgraphs Container $\mathcal{S}^{(i)} = \emptyset$, alphabet container $\mathcal{A}^{(i)} = \emptyset$, order $o_i = 1$
while Order $o_i \leq$ maximum order r AND $RR < \tau_{RR}$ **do**
  Extract subgraphs of order o_i in $\mathcal{S}^{(i)}$
  Create ensemble of partitions $P_1^{(i)}, .., P_n^{(i)}$ of $\mathcal{S}^{(i)}$
  Create boolean matrix $\mathbf{B}^{(i)} \in \{0,1\}^m \times \{0,1\}^n$ with $m = |\mathcal{S}^{(i)}|$
  for all Partitions $P_j^{(i)}$ **do**
    for all Cluster C_k in $P_j^{(i)}$ **do**
      Compute $F(C_k) = (1 - \eta) \cdot \Phi(C_k) + \eta \cdot \Theta(C_k)$
      if $F(C_k) < \tau_F$ **then**
        Create new symbol a_k and put in $\mathcal{A}^{(i)}$
      else
        for all Subgraphs $S_l \in C_k$ **do**
          Set to 1 the element $b_{l,j}$ of $\mathbf{B}^{(i)}$
        end for
      end if
    end for
  end for
  for all Rows b_l of $\mathbf{B}^{(i)}$ **do**
    if $\sum_j b_{l,j} > \tau_{bad}$ **then**
      Remove S_l from $\mathcal{S}^{(i)}$
      if Order of $S_l = 1$ **then**
        Remove the vertex of S_l from the original graph in $\mathcal{S}_{tr}$
      else if Order of $S_l = 2$ **then**
        Remove the edge of S_l from the original graph in $\mathcal{S}_{tr}$
      end if
    end if
  end for
  Create embeddings $\mathcal{D}^{(i)}(\mathcal{S}_{tr})$ and $\mathcal{D}^{(i)}(\mathcal{S}_{vs})$ with $\mathcal{A}^{(i)}$
  Train a classifier $M^{(i)}$ with $\mathcal{D}^{(i)}(\mathcal{S}_{tr})$ and compute the RR on $\mathcal{D}^{(i)}(\mathcal{S}_{vs})$
  Set order $o_i = o_i + 1$
end while
return $M^{(i)}$ with the highest RR

5 Performance evaluation

In this section, we show and discuss the experimental evaluation of the two versions of the GRALG system, comparing the obtained results with other state-of-the-art procedures. In Sect. 5.1, we face classification problem instances defined over synthetically generated data. In Sect. 5.2 we test the proposed systems on well-known benchmarking datasets for graph-based pattern recognition systems evaluation. In all experiments concerning GRALG, subgraphs are expanded up to order 3.

5.1 Tests on synthetic data

Tests on synthetic data have been conceived to asses the performances of the considered graph-based pattern recognition systems on different classification problem instances, characterized by a decreasing difficulty. Each synthetic dataset has been generated using the same *Markov chains* based method described by Livi et al. (2012a). We have conceived 15 different two-classes classification problem instances. Each problem is defined by a training, a validation, and a test set, containing 300 graphs each of order 30 and a variable size between 40 and 75. The hardness of the problem has been controlled generating the labels of both vertices and edges as numeric vectors falling in $[0, 1]^5$. Each random vector has been constructed sampling numbers from two different Gaussian distributions (i.e., one for each class), distributed with means μ_1, μ_2 and standard deviation σ_1, σ_2, respectively. Both standard deviations have been fixed to 0.1. The hardness is directly controlled varying the mean of the second class, μ_2, and letting the first mean fixed to $\mu_1 = 0.5$. The mean μ_2 varies in a thin interval between 0.51 and 0.65, with an incremental step of 0.01. In addition, the stochastic generation process contributes in generating graphs with a randomized topology.

In this experiment, we have considered five different systems, always adopting a classifier based on the k-NN rule. The first two systems are based on the two versions of the proposed GRALG system (GRALGv1 and GRALGv2), described in Sects. 4.3 and 4.4, respectively. Then we tested two k-NN based systems that operate directly on the input space $\mathcal{G}$ of graphs, which adopt the wBMF GED algorithm using the 6 weights (PD6W) (Livi and Rizzi 2012a, b) and the triple (TWEC) (Rizzi and Del Vescovo 2006) weighting schemes for the edit distance computation. Finally, a graph seriation-based system equipped with the DTW distance (Seriation), tailored for sequences of five-dimensional real vectors, is considered. Tests have been performed using three values for the number k of the considered nearest neighbors in the k-NN classification rule, namely 1, 3, and 5. Finally, due to the stochastic nature of the optimization procedure, we repeated the tests on each instance ten times, reporting the average classification accuracy.

Figure 7a shows the classification accuracy on test sets obtained by the five systems using $k = 1$ over the batch of tests, while Fig. 7b shows the corresponding standard deviations (except for the seriation-based system, since it is not affected by stochastic optimization heuristics). The accuracy of the GRALG systems is considerably higher with respect to the one of the other systems, in particular considering the first version (GRALGv1). Notwithstanding the standard deviation of GRALG results to be higher in the first three tests, while it drops rapidly to lower values. The same type of behavior is observed in the results obtained setting $k = 3$ and $k = 5$ (see Figs. 8, 9).

5.1.1 PCA analysis

Principal Component Analysis (PCA) analysis can be used to assess visually the quality of the derived embedding, observing the (linear) separability of the embedding vectors. For this purpose, Figs. 10, 11 show the scatter plot of the first two principal components concerning a PCA performed on an instance of an hard and a medium difficulty

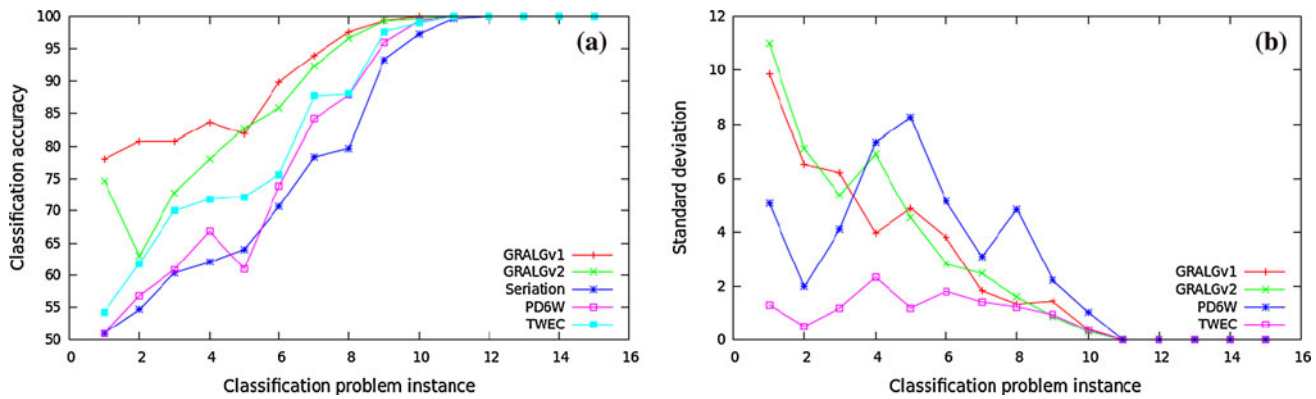


Fig. 7 Results on Markov chains data with $k = 1$. **a** Classification accuracies on test sets. **b** Standard deviations

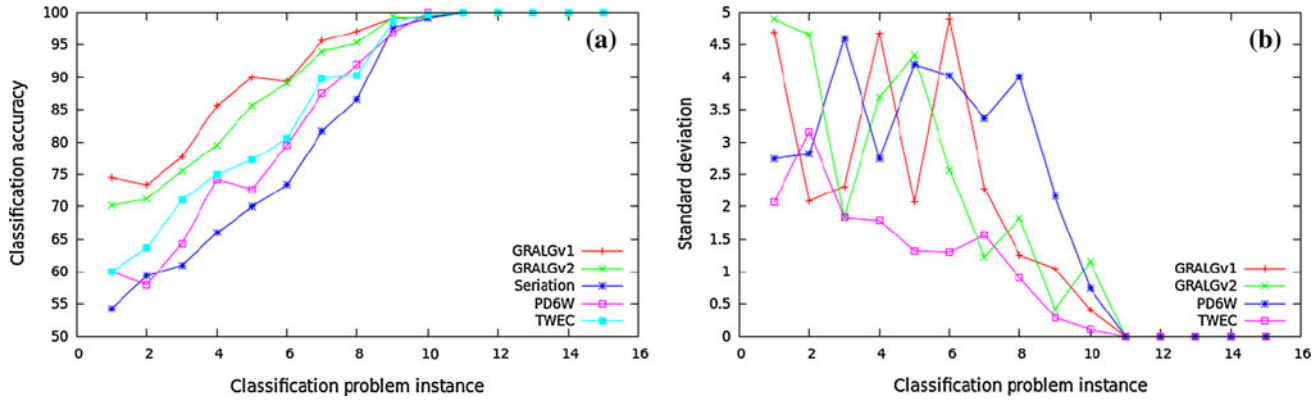


Fig. 8 Results on Markov chains data with $k = 3$. **a** Classification accuracies on test sets. **b** Standard deviations

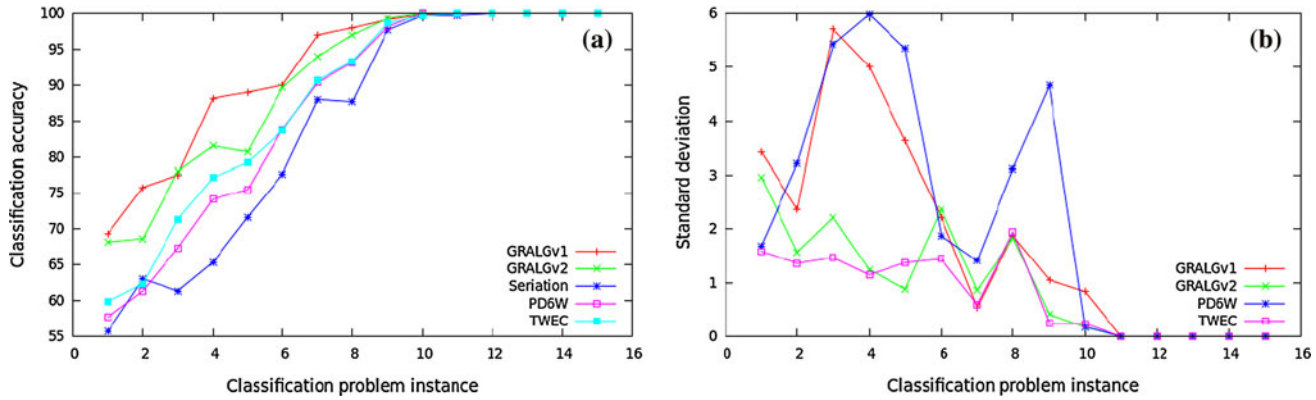


Fig. 9 Results on Markov chains data with $k = 5$. **a** Classification accuracies on test sets. **b** Standard deviations

classification problem; the first and the tenth instance. As it is possible to observe, in the first case the classes overlap, while in the latter we achieve a very discriminative embedding, for both training and test sets. Indeed, the achieved recognition rate on the first instance, regardless the considered k, is considerably worse with respect to the one obtained on the 10-th instance, which is nearly 1 (see Figs. 7a, 8a, 9a). However, PCA performs just a linear transformation of the data, which in some cases is not sufficient enough to fully characterize the difficulty of the problem in terms of class separability.

5.2 Experiments on IAM datasets

In this section, we provide different comparative experimental evaluations over the *IAM* database (Riesen and Bunke 2008). This shared set of datasets has been already used by different authors Riesen and Bunke (2009a, b), Fankhauser et al. (2011), Livi et al. (2012b), Gibert et al. (2011), Riesen and Bunke (2009a, b), Jain et al. (2010), Jain and Obermayer (2011), providing a good evaluation benchmark for graphs-based pattern recognition systems. In Sect. 5.2.3 we show the results achieved with the two variants of the GRALG system, together with other state-of-the-art graph embedding based systems. In Sect. 5.2.4 we discuss the results of graphs classification systems that operate directly in the input space $\mathcal{G}$. Finally, in Sect. 5.2.5 we present the results obtained for classification systems based on graph seriation techniques. In the following tables, the symbol "-" means that the result is not available, and the grayed rows denote results introduced in this paper.

5.2.1 *Description of the datasets*

The datasets *Letter LOW*, *Letter MED*, and *Letter HIGH* are composed of a triple of training, validation and test sets, each of 750 patterns. The first dataset is composed of letters with a low level of distortion. The patterns of the second and the third dataset are affected by medium and high level of distortions. Each dataset contains equally-distributed patterns from 15 different classes. The *AIDS* dataset is a not-equally distributed two-class set of graphs with 250, 250 and 1,500 samples for the training, validation and test set, respectively. The represented data are molecular compounds, denoting or not activity against *HIV*. The atoms are represented directly through the vertices, and covalent bonds by the edges of the graph. Vertices are labeled with the chemical symbol and edges by

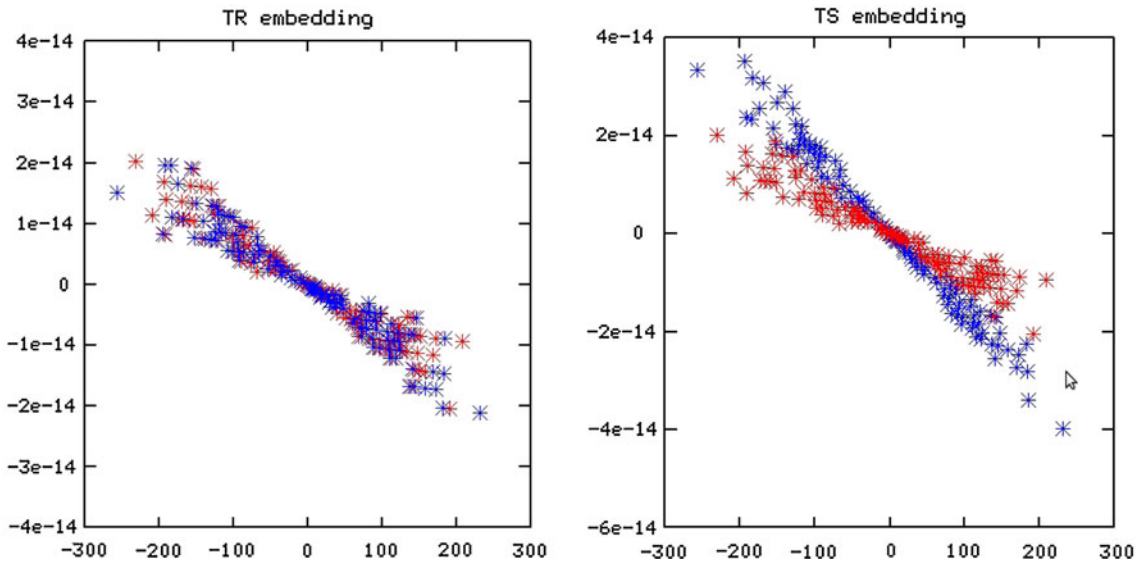


Fig. 10 The two principal components derived from the PCA performed on the embeddings of the graphs coming from the first synthetic dataset

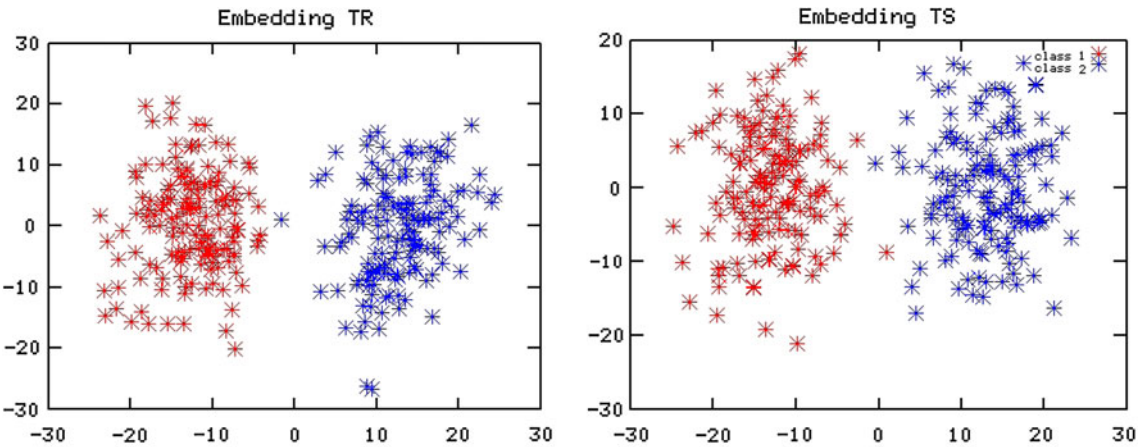


Fig. 11 The two principal components derived from the PCA performed on the embeddings of the graphs coming from the 10-th synthetic dataset

the valence of the linkage. The *COIL-DEL* dataset is composed of 2400, 500 and 1,000 graphs for the training, validation and test set, respectively, which are equally-distributed among 100 classes. The *Proteins* dataset, already studied in Borgwardt et al. (2005), is composed of 200 graphs for each set, equally distributed among six classes. A labeled graph is constructed considering the secondary structure elements of the protein. For this purpose, vertices are labeled with their type (helix, sheet, or loop) and their amino acid sequence. The connecting undirected edges between vertices are defined by considering the three nearest elements, and assigning a label denoting the type and the distance expressed in angstroms. The *GREC* dataset consists of graphs representing symbols from architectural and electronic drawings. The images occur at five different distortion levels, and each graph has been distorted multiple times. The dataset contains 1,100 graphs uniformly distributed over 22 classes, appropriately separated into a training and a validation set of size 286 each, and a test set of size 528. Finally, the *Mutagenicity* dataset contains chemical compounds that are converted into labeled graphs in a straightforward manner by representing atoms as vertices and the covalent bonds as edges. Vertices are labeled with the number of the corresponding chemical symbol and edges by the valence. The Mutagenicity dataset is divided into two classes, depending on their mutagenicity properties. The training, validation and test set contain 1500, 500 and 2,337 graphs, respectively.

In the Tables 1, 3 and 5, we analyze the Letter LOW (L-L), Letter MED (L-M), Letter HIGH (L-H), AIDS, COIL-DEL (C-D), Proteins (P), GREC (G) and Mutagenicity (M) datasets.

Table 1 Test set classification results achieved using different features-based classifiers and graph embedding algorithms

Classification system	Datasets							
	L-L	L-M	L-H	AIDS	C-D	P	G	M
OV + *k*-NN Gibert et al. (2011)	98.8	–	–	–		–	97.5	–
kPCA + *k*-NN Gibert et al. (2011)	97.6	–	–	–	–	–	80.6	–
ICA + *k*-NN Gibert et al. (2011)	82.8	–	–	–	–	–	63.3	–
sk + *SVM* Riesen and Bunke (2009a, b)	99.7	85.9	79.1	97.4	–	–	94.4	55.4
le + *SVM* Riesen and Bunke (2009a, b)	99.3	95.9	92.5	98.3	–	–	96.8	74.3
lgq Jain et al. (2010)	81.5	–	–	–	–	–	86.2	–
$bayes_1$ Jain and Obermayer (2011)	80.4	–	–	–	–	–	80.3	–
$bayes_2$ Jain and Obermayer (2011)	81.3	–	–	–	–	–	89.9	–
GRALGv1	98.2	79.8	74.5	99.7	94.0	68.0	97.7	73.0
GRALGv2	97.6	89.6	82.6	99.7	97.8	64.7	97.6	73.0

5.2.2 *Vertex and edge label dissimilarities*

For every considered IAM dataset, a normalized (and parametric) dissimilarity measure has been defined for both vertex and edge labels. When the label type is a complex structure with different heterogeneous fields, we have added suited parameters with the aim of combining those fields in a flexible and consistent way. Therefore, we include those parameters in the global optimization process of every considered graph-based pattern recognition systems (e.g., the GRALG system described in Sect. 4). In this way, we give automatically more importance to the fields of the structured label that result to be more correlated with the problem at hand, that is, to the fields carrying useful information for the classification problem at hand, with the aim to improve the overall classification accuracy discarding useless information.

In the following paragraphs, we detail the adopted vertices and edges dissimilarity measures for each considered IAM dataset.

5.2.2.1 *Letter LOW, letter MED, and letter HIGH*

- *Vertices*: The labels are two dimensional real-valued vectors and they are compared with a normalized Euclidean dissimilarity d_E.
- *Edges*: There are no labels on edges, a constant dissimilarity measure $d_c = 0$ was used.

5.2.2.2 *AIDS*

After some experiments, we have noticed that considering only topological information of the input graphs was sufficient for classifying the data with good results, making the information on both vertices and edges labels unnecessary. The discovery is motivated by the fact that the considered molecular compounds are intrinsically identified by the number of atoms, together with their neighborhoods.

5.2.2.3 *GREC*

- *Vertices*: The labels are defined by a structure containing a discrete value named *type*, which is compared with a delta dissimilarity d_d (i.e., $d_d(x, z) = 1$ if $x \neq z$, otherwise is 0), and a two dimensional real-valued vector $\underline{\mathbf{v}}$, which is compared with an normalized Euclidean dissimilarity d_e. Given two vertex labels $\mu(v_i)$ and $\mu(v_j)$, the resulting dissimilarity value D will be obtained as,

$$D = \begin{cases} 1 & \text{if } d_d(\mu(v_1).type, \mu(v_2).type) = 1, \\ d_e(\mu(v_1).\underline{\mathbf{v}}, \mu(v_2).\underline{\mathbf{v}}) & \text{otherwise.} \end{cases} \tag{8}$$

- *Edges*: The labels are characterized by a variable length structure containing an integer value *frequency* and one or two pairs of the type (*type*, *angle*), where the first element is a string which can assume two values, namely *arc* and *line*, and the second element of the pair is a real value in $[0, +\infty)$ if *type* = *arc*, or in $[-\pi, \pi]$ if *type* = *line*. The value of *frequency* represents the number of the pairs. A delta dissimilarity d_d is used for *frequency* and *type*, while the field *angle* is compared with a module distance d_m^l normalized in $[-\pi, \pi]$ if *type* = *line*, or a module distance d_m^a normalized in $[0, arc_{max}]$ if *type* = *arc*. Given two edge labels $\nu(e_i)$ and $\nu(e_j)$, three different dissimilarities can be computed, depending on the value of the *frequency* field:

1. If $\nu(e_i).frequency = \nu(e_j).frequency == 1$

$$D = \begin{cases} \alpha \cdot d_m^l(\nu(e_i).angle0, \nu(e_j).angle0) \\ \quad \text{if } \nu(E_i).type0 = \nu(E_j).type0 = line, \\ \beta \cdot d_m^a(\nu(e_i).angle0, \nu(e_j).angle0) \\ \quad \text{if } \nu(e_i).type0 = \nu(E_j).type0 = arc, \\ \gamma \text{ otherwise.} \end{cases} \tag{9}$$

2. If $\nu(e_i).frequency = \nu(e_j).frequency == 2$

$$D = \begin{cases} \frac{\alpha}{2} \cdot d_m^l(v(e_i).angle0, v(e_j).angle0) \\ \quad + \frac{\beta}{2} \cdot d_m^a(v(e_1).angle1, v(e_2).angle1) \\ \quad \text{if } v(e_1).type0 = v(e_2).type0 = line, \\ \frac{\alpha}{2} \cdot d_m^l(v(e_1).angle1, v(e_2).angle1) \\ \quad + \frac{\beta}{2} \cdot d_m^a(v(e_1).angle0, v(e_2).angle0) \\ \quad \text{if } v(e_1).type0 = v(e_2).type0 = arc, \\ \gamma \text{ otherwise.} \end{cases} \quad (10)$$

3. If $\nu(e_1).frequency \neq \nu(e_2).frequency$

$$D = \delta. \quad (11)$$

α, β, γ and δ parameters, all defined within the [0, 1] interval, are parameters optimized with the genetic algorithm.

5.2.2.4 Mutagenicity

- *Vertices*: The labels store a discrete value *chem* which is compared with a delta dissimilarity.
- *Edges*: The labels store a discrete value *valence* which is compared with a delta dissimilarity.

5.2.2.5 Protein

- *Vertices*: The labels are defined by a structure containing a discrete value *type*, which is compared with a delta dissimilarity measure d_d, and a string *sequence*, representing the aminoacids sequence, which is compared with a normalized Levenshtein distance d_L. Given two vertex labels $\mu(v_i)$ and $\mu(v_j)$, the resulting dissimilarity will be evaluated as,

$$D = \alpha \cdot d_d(\mu(v_i).type, \mu(v_j).type) + (1 - \alpha) \cdot d_L(\mu(v_i).sequence, \mu(v_j).sequence), \quad (12)$$

where $\alpha \in [0, 1]$ is a parameter optimized with the genetic algorithm.

- *Edges*: The labels are defined by a variable length structure containing an integer value *frequency* and one or two pairs (*type*, *distance*), where the first one is a discrete value and the second one a real number. Both *frequency* and *type* are compared with a delta dissimilarity d_d, while the field *distance* is compared with a normalized module distance d_m. Given two edge labels $\nu(e_i)$ and $\nu(e_j)$, three cases can occur, depending on the value of *frequency* field:

1. If $\nu(e_i).frsequency = \nu(e_j).frequency == 1$

$$D = \alpha \cdot d_d(v(e_i).type0, v(e_j).type0) + (1 - \alpha) \cdot d_m(v(e_i).distance0, v(e_j).distance0); \quad (13)$$

2. If $\nu(e_i).frequency = \nu(e_j).frequency == 2$

$$D = \frac{1}{2}(\alpha \cdot d_d(v(e_i).type0, v(e_j).type0) + (1 - \alpha) \cdot d_m(v(e_i).distance0, v(e_j).distance0)) + \frac{1}{2}(\alpha \cdot d_d(v(e_i).type1, v(e_j).type1) + (1 - \alpha) \cdot d_m(v(e_i).distance1, v(e_j).distance1)); \quad (14)$$

3. If $\nu(e_i).frequency \neq \nu(e_j).frequency$

$$D = 1. \quad (15)$$

$\alpha \in [0, 1]$ is a parameter to be tuned by the global parameters optimization process.

5.2.2.6 COIL-DEL

- *Vertices*: The labels are two dimensional real-valued vectors and they are compared with a normalized Euclidean dissimilarity d_E.
- *Edges*: The labels are a discrete value *valence* which is compared with a delta dissimilarity measure.

5.2.3 Results of graph embedding algorithms using different classification systems

Table 1 shows the classification accuracy percentages obtained using different explicit graph embedding methods, employed in different classification systems. In Table 2 are reported the standard deviations for the two different GRALG versions. The GRALG systems have been optimized executing 20 evolutions and measuring its achieved classification accuracy on the validation set. The best configuration of those parameters has been used to test the accuracy on the test set.

In the dataset L-L the recognition rate is pretty high in both the results obtained using GRALGv1 and GRALGv2, having at the same time a pretty low variance in the results. This is because L-L is the easiest Letter dataset and the value of the parameters is not so critical in the performance of the classification, which can be performed correctly even with different configurations of the parameters. In the other Letter datasets, i.e., L-M and L-H, the performances decrease, but this was expected because the higher distortion of the letters makes the classification task more difficult. However, we can observe an increment on the variance caused by the fact that the harder dataset depends more critically from the parameters values, which are tuned during the optimization procedure. In the AIDS dataset a good solution is always found by the two versions of GRALG. Notably, on the base of the opinion gathered from a field expert, we neglected the dissimilarities of both edge

Table 2 Standard deviations of the results of Table 1

System	Datasets							
	L-L	L-M	L-H	AIDS	C-D	P	G	M
GRALGv1	1.1019	3.5646	3.2423	0.0000	7.0000	0.4175	0.5964	0.6586
GRALGv2	0.5598	0.8550	1.8064	0.0000	0.5319	0.3340	0.5233	0.9859

and vertex labels, considering only the topological structure for classifying a graph. In the GREC and COIL-DEL dataset, processed with GRALGv2, the results are good in every run, even with different parameters configurations, showing that the classification results do not depend too much on the parameters setup. On the other hand, the results obtained on COIL-DEL processed with GRALGv1 are worse, in particular in terms of the variance. The results obtained in the Protein dataset, with both versions of GRALG, are aligned with the ones shown in the next section in Table 3. Finally, the recognition rate achieved on the Mutagenicity dataset is aligned with the state-of-the-art methods. Performances on test and validation set were coherent, denoting a good generalization capability, and the low variance in the result showed that data have been processed reliably.

5.2.4 Classification results using the k-NN rule

Tables 3 and 4 show, respectively, the classification accuracy percentages and standard deviations obtained with the *k*-NN rule using the 6 weights-BMF (PD6W) and the *Graph Coverage* (GC) (Livi et al. 2012b) algorithms. For what concerns the results introduced in this paper (the grayed ones), we performed an IGM parameters learning stage that has been executed on the validation set of each considered dataset, using a genetic algorithm-based optimization procedure. The best performing configuration of those parameters is retained for the test set evaluation.

Classification systems based on direct graph matching are much faster than GRALG systems and in general of any other system based on graph embedding techniques. However their generalization capability, and stability in terms of variance of the results, is usually inferior. Indeed, they totally rely on the capability of the matching algorithm, without the possibility to discover and filter noisy or irrelevant information. Furthermore, beside the mere number of the recognition rate, the GRALG system discover also an alphabet of symbols $\mathcal{A}$, which contains information on the structures which are (qualitatively) recurrent in the input dataset, and relevant for the problem at hand, which in turn can be used for understanding what characterizes a class of patterns. Moreover, it is a general tool for further semantic-oriented analysis of data. Notwithstanding, in some situations trading accuracy and robustness for computing time can be a choice and an advantage, especially considering the case of very large datasets.

Table 3 Test set classification results using a classifier based on the *k*-NN rule

Algorithm	Datasets							
	L-L	L-M	L-H	AIDS	C-D	P	G	M
Heuristic-A^* Riesen and Bunke (2009a, b)	91.0	77.9	63.0	–	93.3	–	–	–
Beam(10) Riesen and Bunke (2009a, b)	91.1	78.5	63.9	96.2	93.3	–	76.7	–
BP-M Riesen and Bunke (2009a, b)	91.1	77.6	61.6	97.0	93.3	–	86.3	–
BP-H Fankhauser et al. (2011)	99.6	94.2	89.8	99.2	–	68.0	97.7	68.3
BP-V Fankhauser et al. (2011)	99.6	94.2	89.8	98.9	–	67.0	97.7	67.6
GC	98.8	82.2	76.3	99.2	–	–	87.1	67.5
PD6W	99.6	96.6	90.6	99.2	62.7	69.2	97.3	69.7

Table 4 Standard deviations of the results of Table 3

Algorithm	Datasets							
	L-L	L-M	L-H	AIDS	C-D	P	G	M
GC	0.1755	1.2021	1.4773	2.3614	–	–	0.5315	0.9077
PD6W	0.1827	0.6204	1.5121	0.4034	12.7	6.3116	1.0854	0.6541

Table 5 Test set classification results achieved using different seriation-based approaches

Classification System	Datasets							
	L-L	L-M	L-H	AIDS	C-D	P	G	M
GRADIS + *SVM* Livi et al. (2012c)	–	–	–	98.5	–	–	–	59.0
RL-GRADIS + *SVM* Livi et al. (2012c)	–	–	–	98.0	–	–	–	67.1
Seriation+*k*-NN	95.0	85.4	68.0	99.0	50.3	–	97.5	71.1

5.2.5 *Results of graph seriation techniques*

Using the seriation-based approaches described in Sect. 3.2.2, we repeated the experiments on the same pool of datasets. Table 5 shows the achieved results. As it is possible to observe, the *k*-NN based system, equipped with a DTW matching scheme configured with a suited dissimilarity measure able to cope with each specific vertex label type, achieves results that are comparable with the other state-of-the-art methods (see Tables 1, 3). It is worth to underline that this approach results to be very fast in general, regardless the specific dataset under analysis. Rather, the seriation stage resulted to be a little bit more sensitive to the order of the considered graphs, because of the cubic computational complexity due to the matrix decomposition algorithm. Moreover, this technique is deterministic and consequently there is no need to compute the standard deviation of the results.

5.3 Discussion on classification settings and results

The main goal that we have tried to accomplish in this work is the design of an automatic classification system able to cope with classification problems defined on virtually any labeled graphs space $\mathcal{G}$, which is capable of describing effectively and discriminatingly the classes in terms of its recurrent, and significant, substructures. The approach adopted in the performance evaluation tests was mainly finalized to the assessment of the embedding procedure properties, rather than to obtain the *highest* performances in terms of classification. For this reason, there have been done no efforts at all for tuning the algorithm specifically for each dataset in order to maximize the achieved recognition rate. Another critical decision was the choice of the $\mathbb{R}^n$ classifier adopted in the GRALG system: the *k*-NN. Even if it is common the adoption of classifiers such as support vector machines and neuro-fuzzy networks (Theodoridis and Koutroumbas 2006; Rizzi et al. 2002), the *k*-NN classifier is still a valuable tool in pattern recognition. In particular, in this case the generalization capability depends completely on the configuration of the embedded training and test sets. As a consequence, it can be used as a direct quality measure in terms of class separability. Having obtained an high recognition rate can be translated that the embeddings are well-shaped and they are capable of clearly describe the classes.

6 Conclusions and future directions

In this paper we have described a general-purpose graph classification system based on GrC modeling techniques. The system is able to face problems defined in a labeled graph space $\mathcal{G}$. For a given classification problem at hand, once suited parametric dissimilarity measures have been defined in the vertex and edge label spaces, the system is completely automatic, since it does not require any manual parameter tuning, relaying instead on genetic algorithm optimization procedures. The adopted graph embedding procedure permits to semantically analyze the input data. In fact, the alphabet of symbols $\mathcal{A}$ contains the significant substructures extracted from the input training set; specific subsets of those symbols are assumed to be able to characterize a class of patterns by means of the symbolic histograms representation. We have extensively tested and compared the performance of the proposed classifier in terms of classification accuracy on both controlled and real-world benchmarking datasets. The system showed comparable (and often better) results with respect to other state-of-the-art methods.

6.1 Future directions

With this work, we have accomplished the objective of designing an effective classification system in terms of generalization capability. The experiments conducted on synthetic and well-known benchmarking datasets were focused on the recognition rate performance on the test set. However, the nature of the method allows further analysis of the input data, defining symbols-depended *local metrics*. In the described system, the alphabet is created using a clustering algorithm relying on a given dissimilarity measure, which in our case is a GED-based dissimilarity configured with a set of parameters $\mathcal{P}$. These parameters strongly influence the matching algorithm, determining the weights of the considered edit operations. Additionally, the parameters characterizing the inner dissimilarity functions defined for vertex and edge labels play an important role.

By using different settings of those parameters $\mathcal{P}_i, i = 1, 2, \ldots, k$, we effectively compare the data using k different metrics, i.e., it is possible to evaluate in k different ways the dissimilarity degree between two graphs. Changing the IGM parameters $\mathcal{P}_i$ influences the clustering procedure adopted for the frequent substructures search procedure, allowing the definition of symbols characterized by ad hoc instances of $\mathcal{P}_i$.

The idea of the local metrics consists in generating an alphabet of symbols $\mathcal{A}$ determined using an heterogeneous set of matching parameters $\mathcal{P}_i$. To this aim, each symbol would be described also by the particular instance of $\mathcal{P}_i$. The symbolic histogram associated to an input graph would be constructed considering the particular setting of $\mathcal{P}_i$, associated to the symbols of $\mathcal{A}$. This is what we call "local metric", and it differs from the herein adopted *global metric* where the symbols are extracted from clusters that are all obtained using the same setup for the IGM algorithm parameters. Local metrics may result very useful when a pattern is composed by heterogeneous elements, which should not be compared using the same dissimilarity, or when, for instance, the characteristics of a class c_1 of the dataset are better caught using $\mathcal{P}_1$, while a class c_2 would be better defined by a metric adopting $\mathcal{P}_2$. Consequently, a future work will consists in implementing a local metric mechanism in the GRALG system.

Additional effort will be devoted to increasing the performance, in terms of computation time, of the GRALG classifier synthesis; solutions towards this aim vary from adopting parallel algorithm implementations of IGM algorithms (Livi and Rizzi 2012a, b) to the design of dedicated electronic circuits for specific computationally intensive system components (Cinti and Rizzi 2011).

References

Bargiela A, Pedrycz W (2003) Granular Computing: an introduction. Number v. 2002 in Kluwer international series in engineering and computer science. Kluwer, London. ISBN 9781402072734

Batista L, Granger E, Sabourin R (2010) Applying dissimilarity representation to off-line signature verification. In: Proceedings of the 2010 20th international conference on pattern recognition, ICPR '10. IEEE Computer Society, Washington, DC, pp 1293–1297. doi:10.1109/ICPR.2010.322. ISBN 978-0-7695-4109-9

Bello R, Falcón R, Pedrycz W, Kacprzyk J (2008) Granular Computing: at the junction of rough sets and fuzzy sets. Studies in Fuzziness and Soft Computing. Springer, Berlin. ISBN 9783540769729

Borgwardt KM, Ong CS, Schönauer S, Vishwanathan SVN, Smola AJ, Kriegel H-P (2005) Protein function prediction via graph kernels. Bioinformatics 21:47–56. doi:10.1093/bioinformatics. ISSN 1367-4803

Carli A, Castellani U, Bicego M, Murino V (2010) Dissimilarity-based representation for local parts. In: Workshop on cognitive information processing, pp 299–303. June. ISBN 978-1-4244-6457-9

Carli A, Figueiredo MAT, Bicego M, Murino V (2012) Generative embeddings based on Rician mixtures: application to kernel-based discriminative classification of magnetic resonance images. In: Proceedings of the first international conference on pattern recognition applications and methods 2012, vol 1, pp 113–122

Cinti A, Rizzi A (2011) Neurofuzzy min-max networks implementation on FPGA. In: International joint conference on computational intelligence (IJCCI). Neural Comput Theories Anal. ISBN 978-989-8425-84-3

Del Vescovo G, Rizzi A (2007a) Automatic classification of graphs by symbolic histograms. In: Proceedings of the 2007 IEEE international conference on granular computing, GRC '07. IEEE Computer Society, pp 410–416. doi:10.1109/GRC.2007.46. ISBN 0-7695-3032-X

Del Vescovo G, Rizzi A (2007b) Online handwriting recognition by the symbolic histograms approach. In: Proceedings of the 2007 IEEE international conference on granular computing, GRC '07. IEEE Computer Society, Washington, DC, pp 686–700. doi:10.1109/GRC.2007.116. ISBN 0-7695-3032-X

Del Vescovo G, Livi L, Rizzi A, Frattale Mascioli FM (2011) Clustering structured data with the SPARE library. In: Proceedings of 2011 4th IEEE international conference on computer science and information technology, vol 9, pp 413–417. ISBN 978-1-61284-834-1

Escolano F, Bonev B, Lozano M (2011) Information-geometric graph indexing from bags of partial node coverages. In: Jiang X, Ferrer M, Torsello A (eds) Graph-based representations in pattern recognition, volume 6658 of LNCS. Springer Berlin, pp 52–61. doi:10.1007/978-3-642-20844-7_6. ISBN 978-3-642-20843-0.

Fankhauser S, Riesen K, Bunke H (2011) Speeding up graph edit distance computation through fast bipartite matching. In: Jiang X, Ferrer M, Torsello A (eds) Graph-based representations in pattern recognition, volume 6658 of LNCS. Springer Berlin, pp 102–111. doi:10.1007/978-3-642-20844-7_11. ISBN 978-3-642-20843-0

Gao X, Xiao B, Tao D, Li X (2008) Image categorization: graph edit direction histogram. Pattern Recognit 41(10):3179–3191. doi:10.1016/j.patcog.2008.03.025. ISSN 0031-3203

Gao X, Xiao B, Tao D, Li X (2010) A survey of graph edit distance. Pattern Anal Appl 13(1):113–129. doi:10.1007/s10044-008-0141-y. ISSN 1433-7541

Gärtner T (2008) Kernels for structured data. Number v. 72 in Kernels For Structured Data. World Scientific, Singapore. ISBN 9789812814555

Gibert J, Valveny E, Bunke H (2011) Dimensionality reduction for graph of words embedding. In: Jiang X, Ferrer M, Torsello A, (eds) Graph-based representations in pattern recognition, volume 6658 of LNCS. Springer, Berlin, pp 22–31. doi:10.1007/978-3-642-20844-7_3. ISBN 978-3-642-20843-0

Jain B, Obermayer K (2011) Maximum likelihood for gaussians on graphs. In: Jiang X, Ferrer M, Torsello A (eds) Graph-based representations in pattern recognition, volume 6658 of LNCS. Springer, Berlin, pp 62–71. doi:10.1007/978-3-642-20844-7_7. ISBN 978-3-642-20843-0

Jain BJ, Srinivasan SD, Tissen A, Obermayer K (2010) Learning graph quantization. In: Proceedings of the 2010 joint IAPR international conference on structural, syntactic, and statistical pattern recognition, SSPR&SPR'10. Springer, Berlin, pp 109–118. ISBN 3-642-14979-0, 978-3-642-14979-5

Kashima H, Tsuda K, Inokuchi A (2003) Marginalized kernels between labeled graphs. In: Proceedings of the twentieth international conference on machine learning. AAAI Press, pp 321–328

Livi L, Rizzi A (2012) The graph matching problem. Pattern Anal Appl. doi:10.1007/s10044-012-0284-8. ISSN 1433-7541

Livi L, Rizzi A (2012) Parallel algorithms for tensor product-based Inexact Graph Matching. In: Proceedings of the 2012 international joint conference on neural networks (IJCNN). IEEE, Berlin, pp 2276–2283. June. doi:10.1109/IJCNN.2012.6252681. ISBN 978-1-4673-1489-3

Livi L, Del Vescovo G, Rizzi A (2012a) Graph recognition by seriation and frequent substructures mining. In: Proceedings of the first international conference on pattern recognition applications and methods, vol 1, pp 186–191, Feb. doi:10.5220/0003733201860191. ISBN 978-989-8425-98-0

Livi L, Del Vescovo G, Rizzi A (2012b) Inexact Graph Matching through graph coverage. In: Proceedings of the first international conference on pattern recognition applications and methods, vol 1, pp 269–272, Feb. doi:10.5220/0003732802690272. ISBN 978-989-8425-98-0

Livi L, Del Vescovo G, Rizzi A (2012c) Combining graph seriation and substructures mining for graph recognition. Advances in Intelligent and Soft Computing. Springer, Berlin. http://dx.doi.org/10.1007/978-3-642-36530-0_7

Martins AFT, Smith NA, Xing EP, Aguiar PMQ, Figueiredo MAT (2009) Nonextensive information theoretic kernels on measures. J Mach Learn Res 10:935–975. ISSN 1532-4435

Neuhaus M, Bunke H (2007) Bridging the gap between graph edit distance and kernel machines. Series in machine perception and artificial intelligence. World Scientific, Singapore. ISBN 9789812708175

Neuhaus M, Riesen K, Bunke H (2006) Fast suboptimal algorithms for the computation of graph edit distance. In: Structural, syntactic, and statistical pattern recognition. LNCS. Springer, Berlin, pp 163–172

Pekalska E, Duin R (2005) The dissimilarity representation for pattern recognition: foundations and applications. Series in machine perception and artificial intelligence. World Scientific, Singapore. ISBN 9789812565303

Pedrycz W (2010) Human centricity in computing with fuzzy sets: an interpretability quest for higher order granular constructs. J Ambient Intell Human Comput 1:65–74. doi:10.1007/s12652-009-0008-0. ISSN 1868-5137

Príncipe JC (2010) Information theoretic learning: Renyi's entropy and Kernel perspectives. Information Science and Statistics. Springer, Berlin. ISBN 9781441915696

Riesen K, Bunke H (2008) IAM graph database repository for graph based pattern recognition and machine learning. In: Proceedings of the 2008 joint IAPR international workshop on structural, syntactic, and statistical pattern recognition, SSPR & SPR '08. Springer, Berlin, pp 287–297. doi:10.1007/978-3-540-89689-0_33. ISBN 978-3-540-89688-3

Riesen K, Bunke H (2009a) Graph classification by means of Lipschitz embedding. IEEE Trans Syst Man Cybern Part B 39:1472–1483. doi:10.1109/TSMCB.2009.2019264. ISSN 1083-4419

Riesen K, Bunke H (2009b) Approximate graph edit distance computation by means of bipartite graph matching. Image Vis Comput 27:950–959. doi:10.1016/j.imavis.2008.04.004. ISSN 0262-8856

Riesen K, Bunke H (2010) Graph classification and clustering based on vector space embedding. Series in Machine Perception and Artificial Intelligence. World Scientific Pub Co Inc, Singapore. ISBN 9789814304719

Rizzi A, Del Vescovo G (2006) Automatic image classification by a granular computing approach. In: Proceedings of the 2006 16th IEEE signal processing society workshop on machine learning for signal processing, pp 33–38. doi:10.1109/MLSP.2006.275517

Rizzi A, Panella M, Frattale Mascioli FM (2002) Adaptive resolution min-max classifiers. IEEE Trans Neural Netw 13:402–414. ISSN 1045-9227

Robles-Kelly A, Hancock ER (2005) Graph edit distance from spectral seriation. IEEE Trans Pattern Anal Mach Intell 27:365–378. doi:10.1109/TPAMI.2005.56. ISSN 0162-8828

Robles-Kelly A, Hancock ER (2007) A Riemannian approach to graph embedding. Pattern Recognit 40(3):1042–1056

Sakoe H (1978) Dynamic programming algorithm optimization for spoken word recognition. IEEE Trans Acoust Speech Signal Process 26:43–49

Schölkopf B, Smola A (2002) Learning with kernels: support vector machines, regularization, optimization, and beyond. Adaptive computation and machine learning. MIT Press. ISBN 978026 2194754

Theodoridis S, Koutroumbas K (2006) Pattern recognition. Elsevier/Academic Press. ISBN 9780123695314

Tun K, Dhar P, Palumbo M, Giuliani A (2006) Metabolic pathways variability and sequence/networks comparisons. BMC Bioinform 7(1):24. doi:10.1186/1471-2105-7-24. ISSN 1471-2105

Xiao B, Gao X, Tao D, Li X (2008) HMM-based graph edit distance for image indexing. Int J Imaging Syst Technol 18(2–3): 209–218. doi:10.1002/ima.20146

Yu H, Hancock ER (2006) String Kernels for matching seriated graphs. In: Proceedings of the 18th international conference on pattern recognition, volume 4 of ICPR '06, IEEE Computer Society, Washington, DC, pp 224–228. doi:10.1109/ICPR.2006.1081. ISBN 0-7695-2521-0

Zhao Z, Wang L, Liu H, Ye J (2011) On similarity preserving feature selection. IEEE Trans Knowl Data Eng 99. ISSN 1041-4347. doi:10.1109/TKDE.2011.222 (pre print)