IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 4, 2020, accepted January 9, 2021, date of publication January 20, 2021, date of current version January 27, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3053085

An Enhanced Filtering-Based Information Granulation Procedure for Graph Embedding and Classification

ALESSIO MARTINO, (Associate Member, IEEE), AND ANTONELLO RIZZI, (Senior Member, IEEE)
Department of Information Engineering, Electronics and Telecommunications, University of Rome "La Sapienza," 00184 Rome, Italy
Corresponding author: Alessio Martino (alessio.martino@uniroma1.it)

This work was supported in part by the Sapienza University's Research Project "PARADISE: PARAllel and DIStributed Evolutionary Agent-Based Systems for Machine Learning and Big Data Mining," 2018, under Grant RM11816432F5C68A, and in part by the Sapienza University's Research Project "HYD3A: Hypercomplex Deep Learning for 3D Audio Analysis," 2019, under Grant RG11916B88E1942F.

ABSTRACT Granular Computing is a powerful information processing paradigm for synthesizing advanced pattern recognition systems in non-conventional domains. In this article, a novel procedure for the automatic synthesis of suitable information granules is proposed. The procedure leverages a joint sensitivity-vs-specificity score that accounts the meaningfulness of candidate information granules for each class considered in the classification problem at hand. Only statistically relevant granules are retained for a graph embedding procedure towards a geometric space, in which standard classification systems can be used without alterations. Performance tests have been carried out by considering open access datasets of fully labelled graphs with arbitrarily complex nodes and/or edges attributes that, by definition, must rely on inexact graph matching procedures to quantify dissimilarities. Two variants of the procedure are investigated: a standard variant, which aims at automatically finding suitable information granules for solving the classification problem as a whole, and a class-specific metric learning variant, in which the optimization procedure is performed in a class-aware fashion. In the latter case, each class will have its own set of information granules, along with the corresponding parameters defining distinct instances of the dissimilarity measure. Computational results show that the proposed algorithm is able to outperform the vast majority of current approaches for graph classification, while at the same time returning a grey-box model, interpretable by field-experts.

INDEX TERMS Inexact graph matching, graph embedding, granular computing, graph classification, structural pattern recognition, supervised learning.

I. INTRODUCTION

Graph embedding is one of the mainstream approaches when dealing with pattern recognition problems in the graph domain. This 'unconventional' domain has fascinated computer scientists and machine learning engineers alike for more than two decades, given the graphs capabilities of endowing both topological and semantic information on the process to be modelled. Indeed, the widespread use of graphs regarded many research fields including biology, social network analysis, computer vision and text mining. The drawback when dealing with graph-based pattern recognition lies on the computational complexity required to evaluate the (dis)similarity between two graphs, that exponentially grows with respect to the input size [1]. Furthermore, graphs are the quintessential data structure lying in a non-metric space [2]. The same does not hold in 'conventional' pattern recognition problems where patterns are represented as multidimensional feature vectors lying in a multidimensional vector space. In fact, the resulting input space can be equipped by a properly-said *metric* satisfying the known properties of non-negativity, identity, symmetry and triangle inequality [2]–[4].

In the literature, there exist several strategies in order to solve pattern recognition problems in the graph domain [2], [5]–[7], notably feature engineering (see e.g., [8]–[11]), the use of custom dissimilarities working directly

The associate editor coordinating the review of this manuscript and approving it for publication was F. K. Wang.

in the input space (see e.g., [12]–[14]) and embedding techniques. In turn, the embedding towards a vector space can be performed in different ways: kernel methods, for example, exploit positive-definite kernel functions in order to (implicitly) map the input data towards a reproducing kernel Hilbert space (see [15] for a detailed review on graph kernels); neural approaches such as graph neural networks (see e.g. [16]), in which the graph representation is learned via a shallow or deep network, yet their interpretability and the understanding of how they return the embedding is still out of reach nowadays [17], [18]; via dimensionality reduction techniques [19], [20] and information granulation. The latter paradigm is the core of this article and is based upon the Granular Computing (GrC) paradigm [21]–[23]. GrC is an information processing paradigm that mimics the innate human ability to granulate and discretize the world around in order to describe it and to support (more or less complex) decision making activities [24]. Under a data-driven viewpoint, this process of 'granulation' can be intended as the extraction of entities (formally known as *information granules*) arising from both the problem and the data at hand. The importance of granulation resides in the ability to underline properties and relationships between data aggregates and their synthesis shall follow the so-called *indistinguishability rule*, according to which elements that show enough similarity, proximity or functionality shall be grouped together [25]. With this approach, each granule is able to show homogenous semantic information from the problem at hand [26]. Furthermore, a given system can be analyzed at different 'levels' of granularity, depending on which different peculiarities may emerge, with different atomic units that show different representation of the system as a whole [27]–[30]. Hence, an effective learning system able to automatically extract the most informative and meaningful set of information granules for both the data and the problem at hand is of paramount interest and put GrC on the spotlight as a powerful framework for analysing complex systems with the aim of providing human-interpretable results, as confirmed by several research works (see e.g. [5]–[7], [31]–[35] and references therein). Once a suitable set of information granules is returned, a pattern recognition problem can be cast towards the Euclidean space by means of an embedding procedure known as *symbolic histograms*, in which each original pattern (regardless of its domain) is represented as a vector containing the number of occurrences of each information granule within the pattern itself. This allows to move the pattern recognition problem towards a metric space, in which computational intelligence tools can be used without alterations. Usually, data clustering is employed for information granules extraction [5], [6], [32], [34], [36]–[41] due to the tight connection between clusters and information granules as 'groups of similar data'. Despite the effectiveness of clustering procedures, they are usually featured by a non-negligible computational cost, especially when custom dissimilarity measures have to be employed in order to better suit the input space under analysis [36].

In this work, as instead, we leverage a statistical index (originally proposed in ecology) which aims at jointly quantifying the sensitivity and specificity of candidate information granules in order to preserve only those with the highest discriminative power. Hence, our approach can be seen as a pure filtering-based one, where we do not employ any clustering procedure. This approach for extracting information granules has already been explored in [7] and [42], with successful results: in the first case, it has been used for the analysis of metabolic pathways and in the second case for text classification. In both cases, the evaluation of the considered statistical index is straightforward: in fact, finding meaningful chemical reactions in a metabolic pathway[1] can be performed in an *exact* manner by matching the labels of the nodes (metabolites) involved in the chemical reaction (edge); similarly, finding meaningful words in a document can as well be performed in an *exact* manner (i.e., a word does either exist or not). Conversely, in this work we extend the proposed methodology to fully labelled graphs (i.e., graphs with arbitrarily complex attributes on both nodes and edges), where an *inexact* matching procedure is mandatory. More in detail, such statistical index aims at letting statistically relevant information granules emerge from the training data. The set of information granules forms a set of pivotal entities thanks to which the embedding procedure from the graph domain towards the Euclidean space can be performed: in the latter, classification can be performed using any pattern recognition technique. Furthermore, we present two variants of the proposed system: a standard variant, which seeks at automatically tuning suitable *global* parameters using an evolutive metaheuristic and a *class-specific* metric learning variant, where a swarm-based evolutionary optimization aims at finding suitable systems parameters in a class-aware fashion, thereby exploiting also the ground-truth class information. Both variants are equipped with feature selection capabilities in order to return a (possibly) small, yet informative, set of information granules. The latter can be analyzed a-posteriori by field experts in order to get further insights on the modelled problem: this aspect perfectly fits the proposed methodology into the recent ''Explainable Artificial Intelligence'' trend [43], [44].

This article is organized as follows: in Section II we present the proposed methodology in its standard form; in Section III we propose an enhanced version which can also perform class-specific metric learning; in Section IV the datasets used for analysis, along with the computational results are presented and, finally, Section V concludes the paper. This article also features two appendices: in Appendix A we describe in detail the inexact graph matching procedure between labelled graphs, whereas in Appendix B we describe in detail the dissimilarities between nodes and edges for all considered datasets.

[1]Where each metabolic pathway can be represented by a graph with no edge labels and with categorical node labels.

II. PROPOSED METHODOLOGY

Let $\mathcal{D}$ be a dataset of graphs (i.e., a dataset where each pattern is a graph along with its corresponding ground-truth label), properly split into training set ($\mathcal{D}_{\text{TR}}$), validation set ($\mathcal{D}_{\text{VL}}$) and test set ($\mathcal{D}_{\text{TS}}$) and let $\mathbf{l}$ be the corresponding ground-truth class label vector, properly split accordingly ($\mathbf{l}_{\text{TR}}$, $\mathbf{l}_{\text{VL}}$, $\mathbf{l}_{\text{TS}}$) whose values belong to a problem-related set $\mathcal{L}$. In order to spot suitable information granules from the training set, let us recall a unified index called INDVAL (I), originally proposed in [45] for spotting representative species in different environmental condition. Its philosophy is straightforward: a given species s is representative, hence useful for the recognition of a given environmental condition ec, if it satisfies both of the following properties:

1) s must be present in only (or almost only) the ec-positive objects
2) s must be present in all (or the great majority of) the ec-positive objects.

The INDVAL score can be re-stated in order to spot signature subgraphs in a set of training graph as [7], [42]:

$$A_{i,j} = \frac{\#\text{ graphs having subgraph } i \text{ in group } j}{\#\text{ graphs having subgraph } i} \tag{1}$$

$$B_{i,j} = \frac{\#\text{ graphs having subgraph } i \text{ in group } j}{\#\text{ graphs in group } j} \tag{2}$$

$$I_{i,j} = A_{i,j} \cdot B_{i,j} \cdot 100 \tag{3}$$

By definition, since $A_{i,j} \in [0, 1]$ and $B_{i,j} \in [0, 1]$, then $I_{i,j} \in [0, 100]$. The two supporting scores A and B have a straightforward interpretation: the maximum value of A is obtained when the i^{th} subgraph can be found only in patterns (graphs) belonging to class j, whereas the maximum value for B is obtained if all patterns of class j have subgraph i. Finally, the maximum INDVAL I corresponds to the maximum sensitivity and specificity for the i^{th} subgraph within group j: all patterns of class j have subgraph i and no patterns belonging to other classes have subgraph i.

Given these preliminary definitions, the model synthesis procedure and the testing phase follow.

A. TRAINING PHASE

1) ALPHABET SYNTHESIS

A set of subgraphs, say $\mathcal{B}$, has to be extracted from graphs in $\mathcal{D}_{\text{TR}}$ in order to evaluate A and B. Then, one can figure $\mathbf{A}, \mathbf{B}, \mathbf{I} \in \mathbb{R}^{|\mathcal{B}|\times|\mathcal{L}|}$ as compact matrix representations of Eqs. (1)–(3), whose entries (i, j) are therein defined. Given the boundedness of I, a threshold $T \in (0, 100)$ can be used in order to select meaningful subgraphs: in other words, subgraphs corresponding to rows in $\mathbf{I}$ whose I score is greater than (or equal to) T for at least one of the problem-related classes (columns) are included in an alphabet $\mathcal{A}$. The latter contains the pivotal substructures to perform the embedding towards a geometric space.

Despite the simplicity of the procedure, there are two crucial facets that need to be addressed.

a: HOW TO CHOOSE $\mathcal{B}$

In plain terms, $\mathcal{B}$ can be interpreted as the set of candidate information granules and plays a crucial role in the alphabet synthesis. The most intuitive idea to populate $\mathcal{B}$ may rely on the exhaustive extraction of all possible subgraphs from the training data up to a user-defined order: however, this leads to unfeasible running times and possible memory footprint issues given the combinatorial nature of exhaustive extractors, as thoroughly investigated in [5], [6]. Other ideas in order to shrink the size of $\mathcal{B}$ might include the extraction of narrow families of subgraphs such as cliques [46]–[48] or graphlets [49], [50], whether useful in order to characterize the inner structure of the input graphs. For the sake of generality, in this work we propose simple random walks. Random walks trace back to the beginning of the 20$^{\text{th}}$ century [51] and have been widely studied since, especially in the context of Markov chains. Random walks have been further used in graph theory and network analysis: for example, random walk kernels have been proposed in order to measure similarity between graphs [52] and Twitter (amongst others) uses random walks for its recommender system [53]–[55]. The considered plain random walk generation procedure is summarized in Algorithm 1.

Algorithm 1 Random Walk Sampling Procedure

Input: A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, subgraph order o
Output: A subgraph $\tilde{\mathcal{G}}$

Init $\tilde{\mathcal{G}}$ as empty graph;
Select starting node $u \in \mathcal{V}$ uniformly at random;
Add $u \in \mathcal{V}$ in $\tilde{\mathcal{G}}$;
while order($\tilde{\mathcal{G}}$) $\neq o$ **do**
    Collect $\mathcal{N}(u)$; ▷ $\mathcal{N}(\cdot)$ is the neighbours set
    Select ending node $v \in \mathcal{N}(u)$ uniformly at random;
    Add $v \in \mathcal{V}$ in $\tilde{\mathcal{G}}$;
    Add $(u, v) \in \mathcal{E}$ in $\tilde{\mathcal{G}}$;
    Replace starting node $u \leftarrow v$;
end while
return $\tilde{\mathcal{G}}$;

This procedure is repeated W times, where $W = |\mathcal{B}|$ is the desired user-defined size for the set of candidate information granules, each time selecting from the training set a randomly chosen graph as input of Algorithm 1.

b: HOW TO CHECK FOR SUBGRAPHS INTO GRAPHS

From Eqs. (1)–(3) it is clear that the necessity of checking whether a subgraph exists in a graph emerges. If graphs are equipped with categorical node labels and have unlabelled edges, this matching can be performed in an *exact* manner [7]. In this work, we consider a more general scenario in which nodes and/or edges can be equipped with arbitrarily complex attributes, hence those matches have to be performed in an *inexact* manner. Finding a subgraph $\tilde{\mathcal{G}}$ in a graph $\mathcal{G}$ can be accomplished by the following two-steps procedure:

1) decompose $\mathcal{G}$ into its constituent parts (e.g., walks, cliques, graphlets and the like)
2) match $\tilde{\mathcal{G}}$ against each of the subgraphs from step 1 using a suitable dissimilarity measure between graphs: a match is considered as an hit if the dissimilarity measure is below a predefined threshold τ.

As regards step 1, an exhaustive decomposition would again lead to unfeasible running times. In order to overcome this problem, we consider a sub-optimal solution, summarized in Algorithm 2, in which subgraphs are extracted thanks to a Breadth First Search (BFS) graph traversing procedure while, at the same time, not considering nodes that already appeared in previously-extracted walks as root nodes. This choice assures a complete coverage of the graph while keeping a low number of resulting subgraphs.

Algorithm 2 Graph Constituent Parts Extraction

Input: A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, subgraph order o
Output: List of subgraphs $\mathcal{S}$

Init $\mathcal{S}$ as empty list;
for each node $n \in \mathcal{V}$ **do**
  if n does not exists in paths in $\mathcal{S}$ **then**
    Extract g = BFS(graph=$\mathcal{G}$, seed=n, order=o);
    Add g in $\mathcal{S}$;
  end if
end for
return $\mathcal{S}$;

As regards step 2, let us consider a Graph Edit Distance (GED) as the driving tool in order to measure similarity between graphs [13], [56]–[59]. GEDs inherit the same idea behind well-known dissimilarity measures such as the Levenshtein distance defined on the strings domain [60]: that is, the dissimilarity between two objects is given by the minimum amount of atomic operations (i.e., insertions, deletions and substitutions) to be performed in order to transform the two objects into one another. As graphs are concerned, atomic operations can be performed on both nodes and edges. Within the GEDs family, we consider a node Best Match First (nBMF) strategy in light of its computational complexity, which is linear with the number of nodes in the graphs to be compared [32]. Let $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$ be the two graphs to be compared, which we assume to be labelled on both nodes and edges, with attributes pertaining to suitable sets $\mathcal{L}_v$ and $\mathcal{L}_e$, respectively. Let $d_e : \mathcal{L}_e \times \mathcal{L}_e \to \mathbb{R}$ and $d_v : \mathcal{L}_v \times \mathcal{L}_v \to \mathbb{R}$ be two dissimilarity measures tailored to quantify the dissimilarity between nodes and edges and let d_e and d_v be possibly parametric with respect to a set of parameters Π_e and Π_v, respectively. Given these inputs, nBMF returns the six costs for nodes/edges insertions/deletions/substitutions. The six resulting costs (c^{sub}_{edge}, c^{ins}_{edge}, c^{del}_{edge}, c^{sub}_{node}, c^{ins}_{node}, c^{del}_{node}) can be weighted by six non-negative free parameters (w^{sub}_{edge}, w^{ins}_{edge}, w^{del}_{edge}, w^{sub}_{node}, w^{ins}_{node}, w^{del}_{node}) which encode the importance of each transformation. After weighting the cost of each transformation, the overall dissimilarity between $\mathcal{G}_1$ and $\mathcal{G}_2$ is returned. Appendix A features an in-depth description of nBMF.

In conclusion, in order to check whether a given subgraph $\tilde{\mathcal{G}}$ exist in a graph $\mathcal{G}$, one needs to evaluate the pairwise GEDs between $\tilde{\mathcal{G}}$ and the constituent subgraphs extracted from $\mathcal{G}$ (see Algorithm 2). As anticipated, a match is scored if the dissimilarity is below a threshold τ.

2) EMBEDDING AND CLASSIFICATION

The aim of the alphabet synthesis is to return a set of meaningful information granules properly collected in the alphabet $\mathcal{A}$. As anticipated in Section II-A1, the alphabet contains the pivotal substructures for the embedding stage, by following the *symbolic histogram* paradigm [2]. According to the latter, a given graph $\mathcal{G}$ is mapped into an $|\mathcal{A}|$-length integer-valued vector $\mathbf{h}$ counting in position i the number of times the i^{th} item from $\mathcal{A}$ appears in the graph, that is:

$$\mathbf{h} = [\text{occ}(\mathcal{A}_1, \mathcal{G}), \ldots, \text{occ}(\mathcal{A}_n, \mathcal{G})] \tag{4}$$

where $n = |\mathcal{A}|$ and where $\text{occ}(a, b)$ is a function that counts the number of times a appears in b. This counting (matching) stage procedure follows the same procedures already in Section II-A1.

Hence, a set of graphs to be embedded is transformed into an integer-valued feature matrix with $|\mathcal{A}|$ columns and as many rows as there are patterns, suitable to be fed to a classification algorithm, along with the corresponding ground-truth class labels.

3) OPTIMIZED TRAINING PROCEDURE VIA EVOLUTIONARY METAHEURISTIC

An automatic synthesis of $\mathcal{A}$ is of utmost importance given the number of free parameters that are strictly problem- and data-dependent and hardly known a-priori. To this end, we employ a differential evolution algorithm [61] for automatic parameter tuning and alphabet synthesis. Each individual from the evolving population is a candidate solution to the optimization problem, whose form reads as:

$$\left[T \;\; \tau \;\; \mathbf{w} \;\; \Pi_e \;\; \Pi_v \;\; \mathcal{H}\right] \tag{5}$$

where

- $T \in (0, 100)$ is the INDVAL threshold for promoting candidate symbols to the alphabet
- $\tau \in (0, 1)$ is the GED threshold for scoring hits
- $\mathbf{w} = \left[w^{sub}_{edge}, w^{ins}_{edge}, w^{del}_{edge}, w^{sub}_{node}, w^{ins}_{node}, w^{del}_{node}\right] \in [0, 1]^6$ is the 6-weights vector in charge of weighting atomic operations on nodes and edges
- Π_e and Π_v are the sets of dissimilarity measure parameters for nodes and edges, if applicable
- $\mathcal{H}$ contains the hyper-parameters for the classification system.

The objective function, to be minimized, reads as a linear convex combination between the classifier performances on

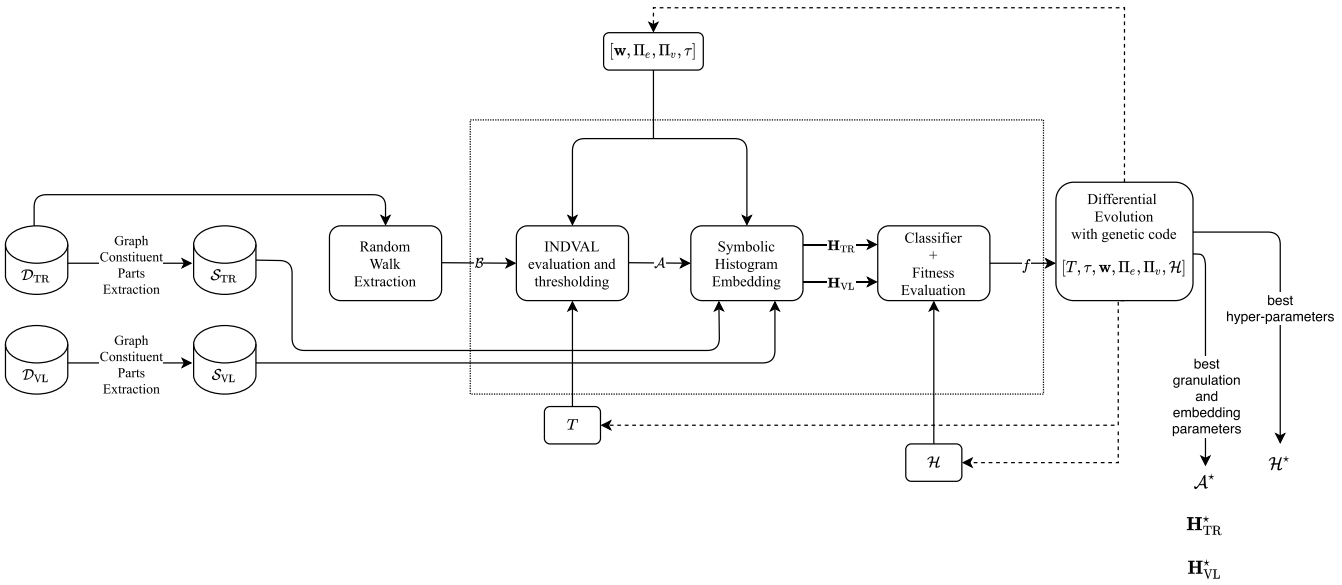

FIGURE 1. Model synthesis block diagram. The behaviour of a single individual is framed in the dotted box.

the validation set and the size of the alphabet, that is:

$$f = \alpha \cdot e + (1 - \alpha) \cdot \frac{|\mathcal{A}|}{|\mathcal{B}|} \tag{6}$$

with e being an error function and $\alpha \in [0, 1]$ being a user defined parameter weighting the two terms. The rightmost term accounts for the dimensionality of the embedding space and can play a huge role in terms of model interpretability.

Each individual from the evolving population:

1) receives $\mathcal{B}$ and the expanded versions (according to Algorithm 2) of the graphs belonging to training set and validation set (say $\mathcal{S}_{\mathrm{TR}}$ and $\mathcal{S}_{\mathrm{VL}}$, respectively)
2) evaluates $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{I}$ (see Section II-A1) by scoring matches thanks to a GED equipped with $\mathbf{w}$, Π_e and Π_v (if applicable). The GED is thresholded thanks to τ
3) the matrix $\mathbf{I}$ is thresholded thanks to T and the alphabet $\mathcal{A}$ is built
4) the two sets $\mathcal{S}_{\mathrm{TR}}$ and $\mathcal{S}_{\mathrm{VL}}$ are embedded thanks to $\mathcal{A}$ in order to return the two instance matrices $\mathbf{H}_{\mathrm{TR}} \in \mathbb{R}^{|\mathcal{D}_{\mathrm{TR}}| \times |\mathcal{A}|}$ and $\mathbf{H}_{\mathrm{VL}} \in \mathbb{R}^{|\mathcal{D}_{\mathrm{VL}}| \times |\mathcal{A}|}$. The symbolic histogram counting procedure leverages the same GED as for step 2
5) a classifier with hyper-parameters $\mathcal{H}$ is trained on $\mathbf{H}_{\mathrm{TR}}$ and its error rate is evaluated on $\mathbf{H}_{\mathrm{VL}}$
6) the fitness function is returned.

At the end of the optimization, the best genetic code is retained. This allows to synthesize the optimal alphabet $\mathcal{A}^\star$ against which it is possible to build the instance matrix $\mathbf{H}^\star_{\mathrm{TR}}$.

In Figure 1 we summarize the model synthesis discussed so far. The proposed classification system starts by reading the training set and the validation set. Each graph belonging to these sets is expanded thanks to the BFS traversal strategy (see Algorithm 2), hence returning $\mathcal{S}_{\mathrm{TR}}$ and $\mathcal{S}_{\mathrm{VL}}$. The training set is also fed to the random walk block in charge of sampling candidate information granules (see Algorithm 1) and populating $\mathcal{B}$. With these actors, the optimization phase may take place: each individual from the differential evolution optimizes the INDVAL threshold T, the classifier hyper-parameters $\mathcal{H}$ and the GED parameters and weights (see Eq. (5)) in order to minimize Eq. (6).

4) AN (OPTIONAL) ALPHABET REFINEMENT PHASE

The choice of α in the fitness function (see Eq. (6)) weights the trade-off between performance and number of items in the alphabet. The value for α that better suits this trade-off is hardly known a-priori. In order to refine the alphabet selection and overcome this problem, a second optimization stage can be placed after the alphabet synthesis stage. Recall that the alphabet synthesis stage returns an optimal alphabet $\mathcal{A}^\star$ and the embedded version of the training set $\mathbf{H}^\star_{\mathrm{TR}}$. Similarly, let $\mathbf{H}^\star_{\mathrm{VL}}$ be the embedded version of the validation set against $\mathcal{A}^\star$.

This second (optional) alphabet refinement phase can be performed thanks to an evolutionary metaheuristic procedure as well. Each individual from the evolving population has the form

$$\left[\mathbf{m}\ \mathcal{H}\right] \tag{7}$$

where $\mathcal{H}$ contains the hyper-parameters of the classification system and $\mathbf{m} \in \{0, 1\}^{|\mathcal{A}^\star|}$ is a binary mask in charge of filtering out unpromising features (i.e., corresponding to 0's).

The objective function, to be minimized, reads as a linear convex combination between the classifier performances on the validation set and the ratio of selected features, that is:

$$f_{\mathrm{FS}} = \beta \cdot e + (1 - \beta) \cdot \frac{|\{i : \mathbf{m}_i = 1\}|}{|\mathbf{m}|} \tag{8}$$

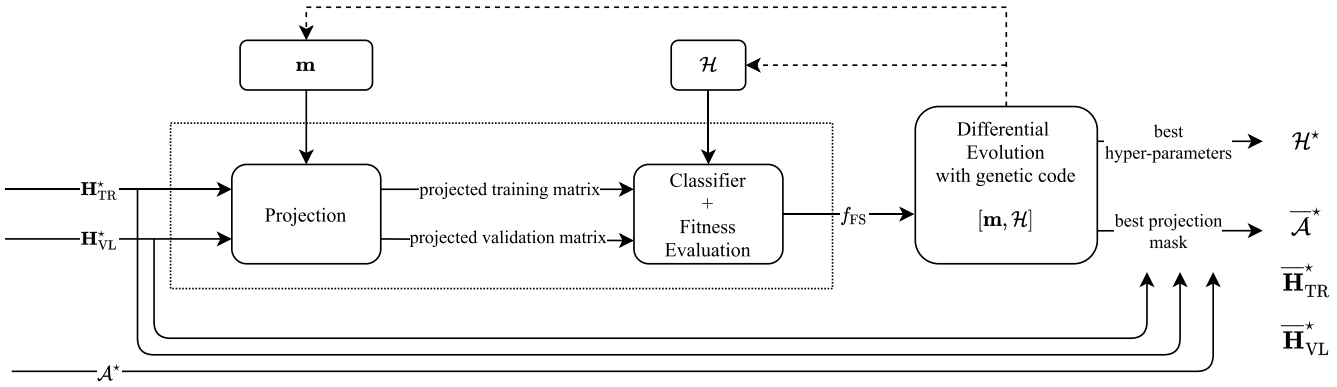

FIGURE 2. Feature selection block diagram. The behaviour of a single individual is framed in the dotted box.

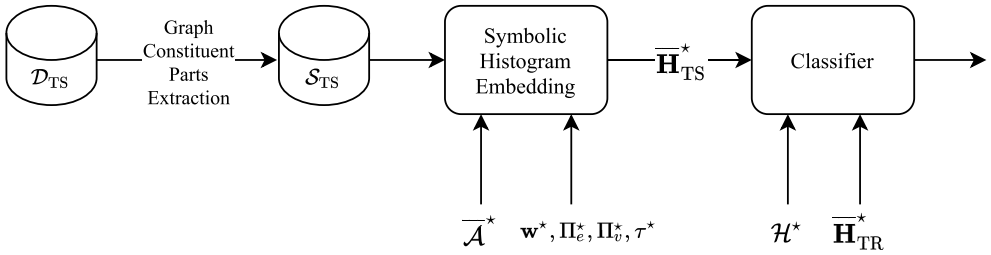

FIGURE 3. Test phase block diagram. Here, we suppose that the feature selection phase has been adopted, otherwise one can easily let $\overline{\mathcal{A}}^{\star} \leftarrow \mathcal{A}^{\star}$, $\overline{\mathbf{H}}^{\star}_{\mathbf{TR}} \leftarrow \mathbf{H}^{\star}_{\mathbf{TR}}$ and $\overline{\mathbf{H}}^{\star}_{\mathbf{TS}} \leftarrow \mathbf{H}^{\star}_{\mathbf{TS}}$.

Hence, each individual from the evolving population:

1) projects $\mathbf{H}^{\star}_{\mathrm{TR}}$ and $\mathbf{H}^{\star}_{\mathrm{VL}}$ on the subspace spanned by 1's in $\mathbf{m}$
2) trains a classifier with hyper-parameters $\mathcal{H}$ on the projected version of $\mathbf{H}^{\star}_{\mathrm{TR}}$ and evaluates its performances on the projected version of $\mathbf{H}^{\star}_{\mathrm{VL}}$
3) evaluates the cost of the binary mask (i.e., rightmost term in Eq. (8))
4) the fitness function is returned.

At the end of the optimization, the best genetic code is retained. This allows to shrink the optimal alphabet $\mathcal{A}^{\star}$ by retaining symbols corresponding to 1's in the best binary mask, say $\overline{\mathcal{A}}^{\star}$. Similarly, the embedded versions of training and validation set can be reduced, i.e. $\mathbf{H}^{\star}_{\mathrm{TR}} \rightarrow \overline{\mathbf{H}}^{\star}_{\mathrm{TR}}$ and $\mathbf{H}^{\star}_{\mathrm{VL}} \rightarrow \overline{\mathbf{H}}^{\star}_{\mathrm{VL}}$. Figure 2 summarizes the feature selection phase described so far.

B. TESTING PHASE

The testing phase consists in expanding the test set (see Algorithm 2) and embedding against the best alphabet $\mathcal{A}^{\star}$ ($\overline{\mathcal{A}}^{\star}$, if the alphabet refinement phase is adopted) in order to get $\mathbf{H}^{\star}_{\mathrm{TS}}$ ($\overline{\mathbf{H}}^{\star}_{\mathrm{TS}}$, if the alphabet refinement phase is adopted). By using the best hyper-parameters $\mathcal{H}^{\star}$, the classifier can be trained on $\mathbf{H}^{\star}_{\mathrm{TR}}$ ($\overline{\mathbf{H}}^{\star}_{\mathrm{TR}}$) and finally tested on the embedded version of $\mathbf{H}^{\star}_{\mathrm{TS}}$ ($\overline{\mathbf{H}}^{\star}_{\mathrm{TS}}$). For the sake of completeness, the testing phase is summarized in Figure 3.

III. ADDING CLASS-SPECIFIC METRIC LEARNING CAPABILITIES

With *metric learning* [62], [63] in pattern recognition one refers to the ability of an intelligent system to learn suitable parameters for a (parametric) dissimilarity measure in order to optimize some predefined criteria (see e.g., [64]–[67]). In Section II, the whole system has been described with a differential evolution scheme in charge of optimizing suitable parameters for the dissimilarity measure (namely $\mathbf{w}$, Π_e and Π_v) in a *global* manner, i.e. by considering the same parameters instance for comparing all possible pairs of patterns. With *local metric learning* one refers to learning suitable weights in a cluster-aware fashion in order to better characterize each cluster [68], [69]. For example, in a multidimensional Euclidean space, different clusters might lie in different subspaces, hence the detection of the cluster involves also the detection of the subspace in which such cluster lies [70], [71]. A third possibility is to design automatic procedures able to perform *class-specific metric learning*, where each decision region (possibly described by the union of more than one cluster) is characterized by its own dissimilarity measure parameters.

In order to introduce the class-specific metric learning variant, recall $|\mathcal{L}|$ be the number of classes for the classification problem at hand. The set of candidate symbols $\mathcal{B}$ is re-defined as a set-of-sets $\mathcal{B} = \{\mathcal{B}_1, \ldots, \mathcal{B}_{|\mathcal{L}|}\}$, where $\mathcal{B}_i$ contains subgraphs (i.e., random walks) drawn from training graphs belonging to class i. The size of each set $\mathcal{B}_i$ is proportional to

the frequency of the i^{th} class within the training set in order to ensure a fair amount of subgraphs for each problem-related class [6]. Specifically, as for Section II-A1, let W be the user-defined size of $\mathcal{B}$, then $\mathcal{B}_i = W \cdot |\mathcal{D}_{\text{TR}}^{(i)}|/|\mathcal{D}_{\text{TR}}|$, where $\mathcal{D}_{\text{TR}}^{(i)}$ denotes the subset of training patterns belonging to the i^{th} class. Hence, it holds that $W = \sum_{i=1}^{|\mathcal{L}|} |\mathcal{B}_i|$.

The alphabet synthesis phase is methodologically similar to the one described in Section II-A1, but instead of employing a single swarm, $|\mathcal{L}|$ swarms operate in parallel, where the i^{th} swarm processes symbols from $\mathcal{B}_i$. The goal of each swarm is to return class-specific symbols $\mathcal{A}_i$ to be included in the alphabet along with its best genetic code, containing the (sub-)optimal GED parameters, classifier hyper-parameters, weights and thresholds for discriminating the i^{th} class (cf. Eq. (5)). The fitness function for each swarm still reads as Eq. (6). However, since the i^{th} swarm works on the i^{th} class, the fitness function is evaluated in a one-vs-all fashion: training and validation sets are relabelled with the i^{th} class being positive and all other classes being negative. At the end of each swarms' evolution, the overall alphabet $\mathcal{A}^\star$ sees the concatenation of the class-specific alphabets $\mathcal{A}^\star = \{\mathcal{A}_1^\star, \ldots, \mathcal{A}_{|\mathcal{L}|}^\star\}$. The training and validation matrices ($\mathbf{H}_{\text{TR}}^\star$, $\mathbf{H}_{\text{VL}}^\star$) are evaluated according to the symbolic histograms paradigm (cf. Section II-A2): however, the matching of each symbol must be performed thanks to the symbol-specific GED parameters, weights and scoring threshold.

The feature selection phase does not change with respect to Section II-A4: however, in order to keep the ensemble-like nature of the system, $|\mathcal{L}|$ classifiers are trained in a one-vs-all fashion.

The testing phase is the same as for Section II-B: yet, the embedding of the test set must as well follow a symbol-specific matching.

IV. TESTS AND EXPERIMENTS

A. DATASETS DESCRIPTION

In order to validate the proposed approach, five datasets freely available from the IAM Graph Database Repository[2] [72] have been considered:

AIDS: each graph represents a molecular compound showing either activity or inactivity against HIV (2 classes). Graphs are built by considering atoms as nodes and covalent bounds as edges. The dataset contains a total of 2000 graphs, split as: 250 (training set), 250 (validation set) and 1500 (test set). Nodes are compared thanks to a custom non-parametric dissimilarity measure, whereas edge labels are discarded.

GREC: each graph represents an architectural or electronic drawing. Graphs are built by considering corners, circles and intersections as nodes. The datasets contains a total of 1100 graphs, divided in 22 classes and split as: 286 (training set), 286 (validation set) and 528 (test set). Nodes and edges are compared thanks to custom parametric dissimilarity measures.

Letter: each graph represents a distorted (capital) Roman letter drawing. Graphs are built by considering lines as edges and endpoints as nodes. Three different datasets account for different amount of distortion (low, medium, high – hereinafter Letter-L, Letter-M and Letter-H, respectively). These datasets, divided in 15 classes each, are split as: 750 (training set), 750 (validation set), 750 (test set). Edges are unlabelled, whereas the dissimilarity between nodes reads as plain Euclidean distance between node attributes.

Training, validation and test splits have been kept as provided by the IAM Repository. Details on nodes and edges dissimilarities for the five datasets can be found in Appendix B.

B. COMPUTATIONAL RESULTS

In a first test campaign, the performances of the proposed classification system in both of its variants, hereinafter named as 'RECTIFIER'[3] (RECogniTIon oF graphs via Indval gEometRic embedding – Section II) and 'Dual RECTIFIER'[4] (Section III), are thoroughly investigated. The investigation leverages three different indices, which account both performance and structural complexity of the model, namely:

1) the accuracy on the test set
2) the size of $\mathcal{A}^\star$ (the number of symbols, i.e. the dimensionality of the embedding space, after the optimization phase)
3) the size of $\overline{\mathcal{A}^\star}$ (the number of symbols, i.e. the dimensionality of the embedding space, after the feature selection phase).

The differential evolution for the automatic alphabet synthesis routine is configured as follows:

- 20 individuals for maximum 20 generations
- crossover probability: 0.8
- mutation with dithering in range [0.5, 1]
- early stop criterion triggered if $\sigma(\mathcal{F}) \leq 0.01 \cdot \mu(\mathcal{F})$, where $\mu(\cdot)$ and $\sigma(\cdot)$ denote mean and standard deviation and $\mathcal{F}$ contains the absolute value of the fitness values at current generation.

The differential evolution for the feature selection stage is configured as for the previous one, with the following exceptions:

- 100 individuals for maximum 20 generations
- trade-off parameter in the fitness function (see Eq. (8)) $\beta = 0.9$.

As classification system, a plain K-Nearest Neighbours [73] is considered, hence the set of hyper-parameters to be tuned reads as $\mathcal{H} = \{K\}$.

[2]All datasets can be downloaded from http://www.fki.inf.unibe.ch/databases/iam-graph-database.

[3]As the legendary Mesa/Boogie® RECTIFIER® guitar amp head.

[4]Which allows a dual explainability facet in terms of both information granules to be analyzed and class-specific dissimilarity measure parameters, just like the Mesa/Boogie® Dual RECTIFIER® guitar amp head that features dual hi-gain channels.

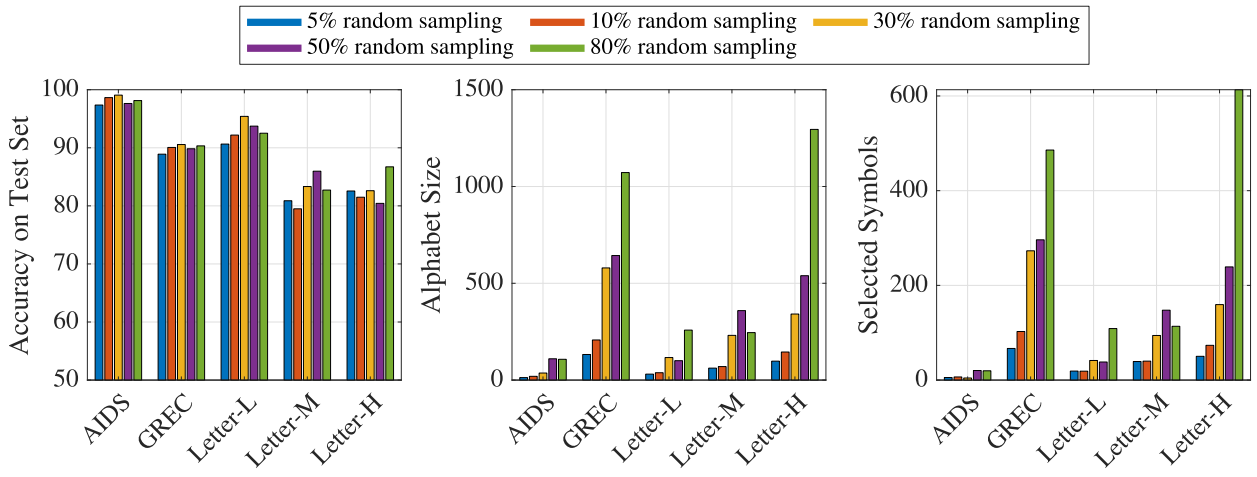

FIGURE 4. RECTIFIER results at $\alpha = 0.5$.

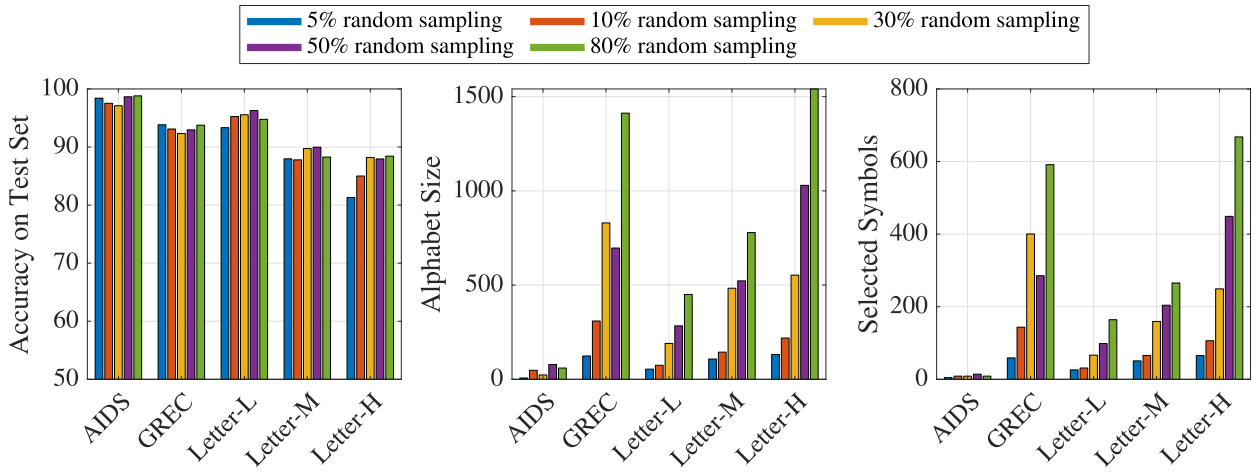

FIGURE 5. RECTIFIER results at $\alpha = 0.7$.

A sensitivity analysis is performed on the trade-off parameter α in the fitness function for automatic alphabet synthesis (see Eq. (6)), where candidate values are $\alpha = \{0.9, 0.7, 0.5\}$, and on the random walk subsampling rate, where candidate values are $W = \{5\%, 10\%, 30\%, 50\%, 80\%\}$ of the maximum number of random walks (of maximum order 5) that can be drawn from the training graphs. Subgraphs in $\mathcal{B}$ are equally distributed across orders that range from 2 to 5.

Due to randomness in the model synthesis, results herein presented have been averaged across 10 runs. Figures 4, 5 and 6 show the results for RECTIFIER for $\alpha = 0.5$, $\alpha = 0.7$ and $\alpha = 0.9$, respectively, whereas Figures 7, 8 and 9 are the counterparts for Dual RECTIFIER.

As both RECTIFIER and Dual RECTIFIER are concerned, the accuracies of the proposed classification systems are quite robust with respect to the subsampling rate, with non-negligible shift performances observed for harder classification problems (Letter-M and Letter-H) at $\alpha = 0.5$. For RECTIFIER, changing α from 0.9 to 0.5 leads to a maximum accuracy shift of 12% for Letter-M and 11% for Letter-H, whereas for the other three datasets shifts in accuracy are always below 7%. Similar results hold for Dual RECTIFIER, where changing α from 0.9 to 0.5 leads to a maximum accuracy shift of 13% for Letter-M and 14% for Letter-H.

Changing the subsampling rate leads to a common (and expected) trend for both RECTIFIER and Dual RECTIFIER: for a given value of α, increasing W (i.e., lowering the subsampling rate) leads to more symbols in the alphabet (both after and before the feature selection phase). Yet, the proper number of selected symbols indeed depends on α, the latter being part of the fitness function: the lower the value, the lower the number of symbols.

By matching the results in terms of embedding space dimensionality, it can be seen that Dual RECTIFIER always returns a lower number of symbols with respect to RECTIFIER, both before and after feature selection, with the only

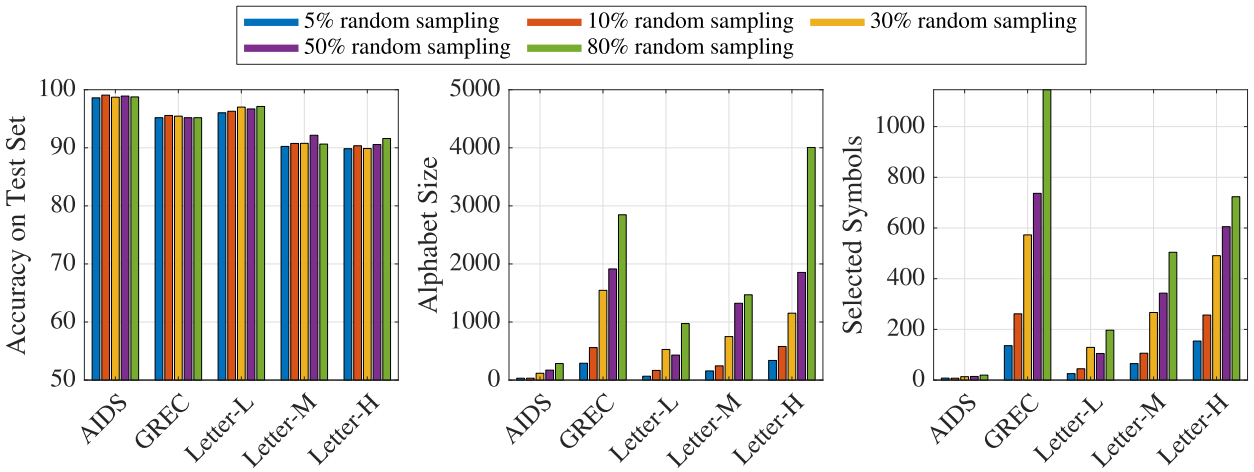

FIGURE 6. RECTIFIER results at $\alpha = 0.9$.

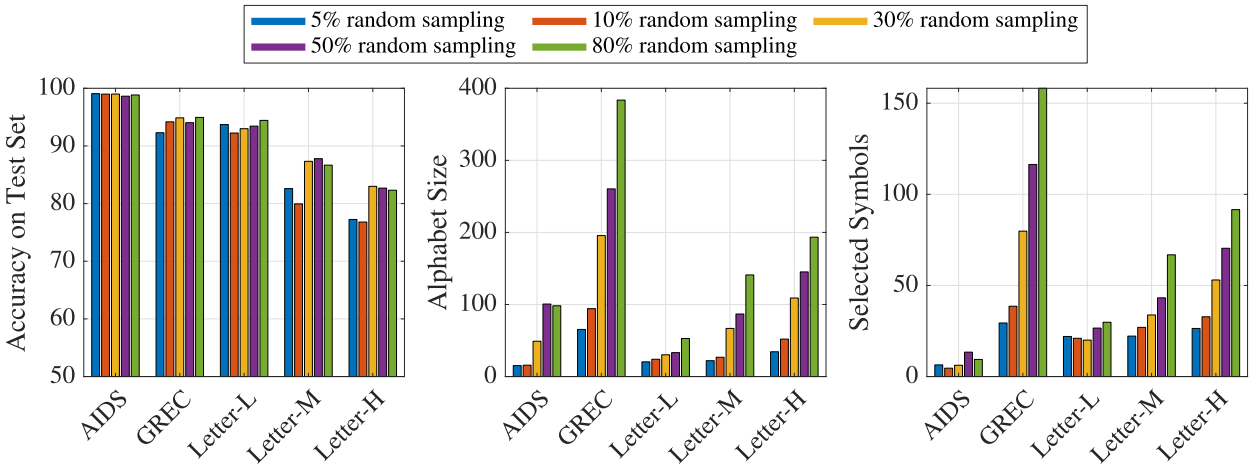

FIGURE 7. Dual RECTIFIER results at $\alpha = 0.5$.

exception being AIDS: in fact, since AIDS contains only two classes, only one swarm would suffice. Another interesting aspect of Dual RECTIFIER is that for some cases (Letter-M and Letter-L, $\alpha = 0.5, 5-10\%$ subsampling rate) the number of selected symbols after feature selection is approximately equal to the number of original symbols, suggesting that the class-aware alphabet synthesis approach already returns a suitable set of symbols and the feature selection phase would be superfluous.

In conclusion, despite the two approaches have comparable results in terms of performances, Dual RECTIFIER seems to be preferable in terms of knowledge discovery because of the following two aspects:

1) smaller alphabet: this fosters the a-posteriori knowledge discovery phase by field-experts, since less symbols have to be analyzed;
2) each problem-related class has its own dissimilarity measure parameters and weights, making each class easier to characterize.

On the other hand, the training phase of Dual RECTIFIER is slower with respect to RECTIFIER. In fact, we experimentally observed that the former is (on average) 35%, 32%, 36%, 41% and 57% slower with respect to the latter for Letter-L, Letter-M, Letter-H, AIDS and GREC, respectively.

C. COMPARISON AGAINST CURRENT APPROACHES

In this second test campaign, the comparison involves the INDVAL embedding (both RECTIFIER and Dual RECTIFIER variants) and current approaches in graph classification, with Table 1 summarizing the results. The comparison is restricted to the five datasets considered in this work, with a dash (-) indicating that a given dataset has not been tested in the literature on the corresponding model. Competitors span a variety of approaches for graph classification, including classifiers working on the top of pure graph matching similarities [72], [74], kernel methods [77], [80] and several embedding techniques [75], [76], including GrC-based [5], [6], [33] and

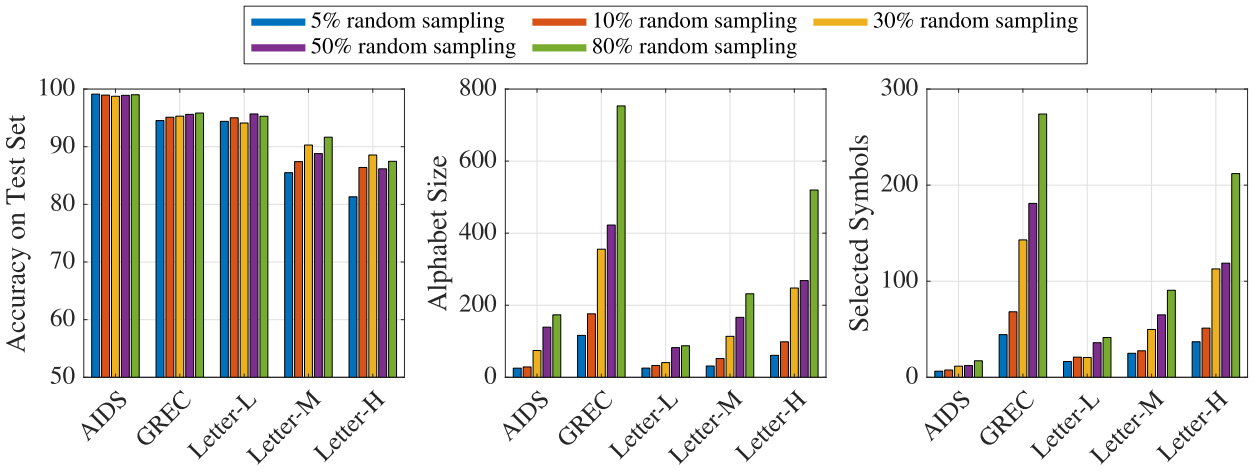

FIGURE 8. Dual RECTIFIER results at $\alpha = 0.7$.

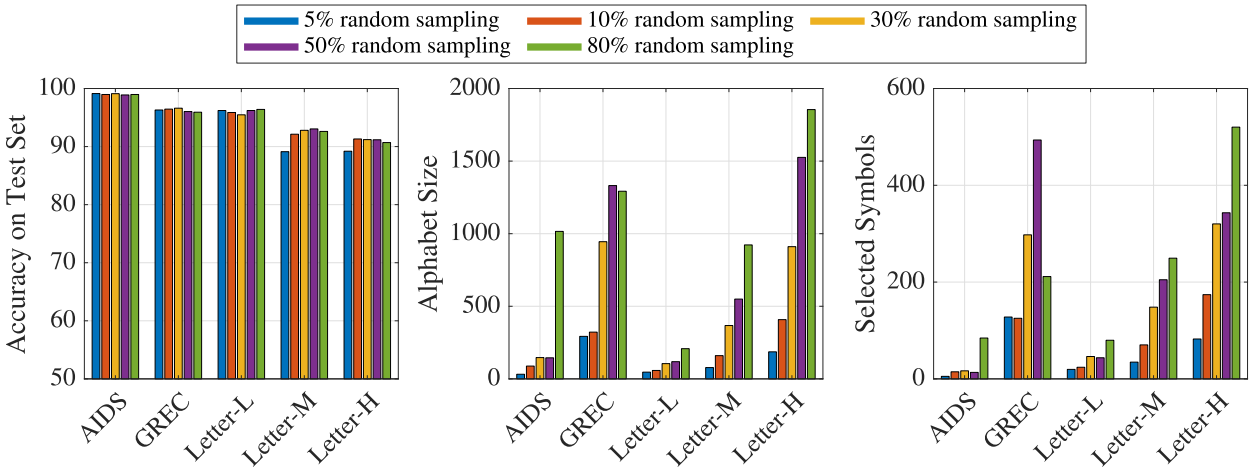

FIGURE 9. Dual RECTIFIER results at $\alpha = 0.9$.

neural ones [78], [79]. As regards subsampling-based implementations, in Table 1 are reported the performances obtained at different subsampling rates in the form of min–max range. The accuracies of the last ten methods are obtained from our own experiments. For the remaining competing algorithms, we directly quote the values from their respective papers.

Clearly, the proposed approaches are able to reach state-of-the-art performances on all five datasets: the shift in accuracy with respect to the most performing technique is below 1% for AIDS, GREC and Letter-H, and below 3% for the remaining two datasets (Letter-M and Letter-L). In particular, the proposed approaches greatly outperform ODD kernel and CGMM on AIDS and G-*-Perceptron on GREC, Letter-M and Letter-H. Especially on harder problems (i.e., Letter-H and Letter-M), the proposed approaches also outperform GRALG (another GrC-based classification system which can also be equipped with random walk procedures in order to reduce the computational complexity[5]) and C-*-NN. The proposed approaches are also comparable with the Lipschitz Embedding (the overall most performing technique amongst the competing ones).

Nonetheless, the proposed approaches (alongside [5], [6], [33]) are able to return an interpretable model: in fact, the resulting symbols from the alphabet (possibly after feature selection) can be analyzed by field-experts and allow a further a-posteriori knowledge discovery phase, as anticipated in Section IV-B. Conversely, neural approaches such as G-*-Perceptron and C-*NN are well known of lacking any interpretability. The same holds for kernel methods (ODD, HCK, WJK, EK, SEK) and plain dissimilarity-based classifiers.

[5]For GRALG 10%, 30% and 50% subsampling rates have been considered to foster a fair comparison.

TABLE 1. Test set accuracy (expressed in percentage) comparison amongst RECTIFIER, Dual RECTIFIER and current approaches for graph classification. In bold, for each dataset, we highlight the most performing approach.

Technique	AIDS	GREC	Letter-L	Letter-M	Letter-H	Reference
Bipartite Graph Matching + K-NN	-	86.3	91.1	77.6	61.6	[74]
Lipschitz Embedding + SVM	98.3	96.8	99.3	**95.9**	**92.5**	[75]
Graph Edit Distance + K-NN	97.3	95.5	**99.6**	94	90	[72]
Graph of Words + K-NN	-	**97.5**	98.8	-	-	[76]
Graph of Words + kPCA + K-NN	-	97.1	97.6	-	-	[76]
Graph of Words + ICA + K-NN	-	58.9	82.8	-	-	[76]
ODD ST_+ kernel	82.06*	-	-	-	-	[77]
ODD ST_+^{TANH} kernel	82.54*	-	-	-	-	[77]
CGMM + linear SVM	84.16*	-	-	-	-	[78]
G-L-Perceptron	-	70	95	64	70	[79]
G-M-Perceptron	-	75	98	87	81	[79]
C-1NN	-	-	96	93	84	[79]
C-M-1NN	-	-	98	81	71	[79]
Hypergraph Embedding + SVM	99.3†	-	-	-	-	[33]
Hypergraph Kernel (HCK)	89.5†*	-	-	-	-	[80]
Hypergraph Kernel (WJK)	99.5†*	-	-	-	-	[80]
Hypergraph Kernel (EK)	99.5†*	-	-	-	-	[80]
Hypergraph Kernel (SEK)	**99.6**†*	-	-	-	-	[80]
GRALG (exhaustive extraction)	99.44	92.23	98.05	84.83	76.13	[5]
GRALG (random walk extraction)	99.09–99.16	83.08–84.04	96.36–96.58	85.28–87.89	72.93–73.78	[5]
GRALG (class-aware random walk extraction)	98.99–99.06	87.38–87.61	98.16–98.31	89.53–90.64	82.82–83.72	[6]
RECTIFIER + K-NN	97.11–99.07	88.90–95.57	90.64–97.12	79.49–92.16	80.43–91.60	This work
Dual RECTIFIER + K-NN	98.63–99.13	92.29–96.61	92.24–96.40	79.95–93.04	76.80–91.31	This work

*Results refer to 10-fold cross-validation.
†Only chemical symbol type (categorical) as node label. Full details in Appendix B.

V. CONCLUSION

In this article, we proposed a novel filtering-based information granulation procedure. The filtering operation relies on a unified index called INDVAL which accounts both specificity and sensitivity of substructures (i.e., candidate information granules) stochastically drawn from the training data with respect to the problem-related classes, with the final goal of electing as information granules only substructures endowing the highest discriminative power. The set of resulting information granules (i.e., the alphabet) composes the pivotal substructures in order to perform the embedding procedure from the original input space (labelled graphs, in this work) towards a Euclidean space. The overall synthesis is driven by an evolutionary optimization procedure in order to tune thresholds, dissimilarity measure parameters and, possibly, the classifier hyper-parameters. A second (optional) evolutionary optimization procedure aims at reducing the size of the alphabet in order to enhance the knowledge discovery capabilities of the proposed system. In fact, not only the considered index has a straightforward interpretation, but also the set of resulting information granules allow the definition of an explainable model.

Two variants of the same methodology have been proposed and investigated: a standard variant, where the optimization stage takes care of synthesizing an alphabet and dissimilarity measure parameters for solving the overall classification problem globally, and a class-specific metric learning variant, where class-aware swarms aim at extracting class-specific alphabets along with class-specific dissimilarity measure parameters.

In order to validate the proposed approaches (with and without class-specific metric learning), five datasets for graph classification have been considered, with graphs having attributes on nodes and/or edges. Both approaches are able to reach remarkable results even with low subsampling rates and even when using simple (uniform) random walks to extract candidate information granules.

The proposed scheme allows some degrees of generalization: in fact, plain random walks can be replaced by extraction procedures based on cliques, graphlets or other types of random walks [81]–[86]; the classifier can as well be personalised; the graph dissimilarity does not necessarily have to be the nBMF and other GEDs can be employed instead [14], [74]. On a higher level, the proposed scheme can be personalized towards virtually any structured input domain, provided that one can extract candidate information granules and fill $\mathcal{B}$ (e.g., k-mers in case of sequences) and define a suitable dissimilarity measure for matching (e.g., the Levenshtein distance between k-mers).

APPENDIX A
THE nBMF DISSIMILARITY MEASURE

In order to properly introduce the nBMF heuristic, let us recall some basic notation from Section II-A1:

- let $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$ be the two graphs to be compared. For the sake of generalization, let us assume that graphs have different sizes, i.e. $|\mathcal{V}_1| \neq |\mathcal{V}_2|$
- let $\mathcal{L}_v$ and $\mathcal{L}_e$ be the sets of possible nodes and edges attributes

- let d_e and d_v be two dissimilarity measures tailored to quantify the dissimilarity between nodes and edges. For the sake of generalization, let us assume these dissimilarity measures to be parametric
- let Π_e and Π_v be the sets of parameters for d_e and d_v, respectively
- let w_{edge}^{sub}, w_{edge}^{ins}, w_{edge}^{del}, w_{node}^{sub}, w_{node}^{ins} and w_{node}^{del} be six non-negative free parameters weighting each of six edit costs (c_{edge}^{sub}, c_{edge}^{ins}, c_{edge}^{del}, c_{node}^{sub}, c_{node}^{ins}, c_{node}^{del}).

The nBMF strategy starts by matching most similar nodes as follows:

1) initialize node substitutions, insertion and deletion costs as $c_{node}^{sub} = c_{node}^{ins} = c_{node}^{del} = 0$
2) initialize a set which will collect matched nodes, say $\mathcal{P} = \emptyset$
3) for a given node $u \in \mathcal{V}_1$, find the most similar node $v \in \mathcal{V}_2$ according to d_v
4) add the pair (u, v) to $\mathcal{P}$ and count this operation as a node substitution with cost $c_{node}^{sub} += d_v(u, v, \Pi_v)$
5) remove u from $\mathcal{V}_1$ and remove v from $\mathcal{V}_2$
6) repeat from step 3 until either $\mathcal{V}_1$ or $\mathcal{V}_2$ is empty
7) if $|\mathcal{V}_1| > |\mathcal{V}_2|$, then this counts as node deletions with $c_{node}^{del} = |\mathcal{V}_1| - |\mathcal{V}_2|$
8) if $|\mathcal{V}_1| < |\mathcal{V}_2|$, then this counts as node insertions with $c_{node}^{ins} = |\mathcal{V}_2| - |\mathcal{V}_1|$.

nBMF now proceeds in matching induced edges:

1) initialize edge substitutions, insertion and deletion costs as $c_{edge}^{sub} = c_{edge}^{ins} = c_{edge}^{del} = 0$
2) for a given (u, v) pair in $\mathcal{P}$, check:
   - whether edge $e_1 = (u, v)$ exists in $\mathcal{E}_1$ and whether $e_2 = (u, v)$ exists in $\mathcal{E}_2$: if so, this counts as an edge substitution with cost $c_{edge}^{sub} += d_e(e_1, e_2, \Pi_e)$
   - whether (u, v) only exists in $\mathcal{E}_1$: if so, this counts as an edge insertion with $c_{edge}^{ins} += 1$
   - whether (u, v) only exists in $\mathcal{E}_2$: if so, this counts as an edge deletion with $c_{edge}^{del} += 1$
3) repeat from step 2 until all pairs in $\mathcal{P}$ are processed.

The overall dissimilarities between nodes and edges ($d_{\mathcal{V}}$ and $d_{\mathcal{E}}$, respectively) are defined as:

$$d_{\mathcal{V}}(\mathcal{V}_1, \mathcal{V}_2) = w_{node}^{sub} \cdot c_{node}^{sub} + w_{node}^{ins} \cdot c_{node}^{ins} + w_{node}^{del} \cdot c_{node}^{del} \tag{9}$$

$$d_{\mathcal{E}}(\mathcal{E}_1, \mathcal{E}_2) = w_{edge}^{sub} \cdot c_{edge}^{sub} + w_{edge}^{ins} \cdot c_{edge}^{ins} + w_{edge}^{del} \cdot c_{edge}^{del} \tag{10}$$

and in order to avoid skewness due to graphs having different sizes, are normalised as:

$$d'_{\mathcal{V}}(\mathcal{V}_1, \mathcal{V}_2) = \frac{d_{\mathcal{V}}(\mathcal{V}_1, \mathcal{V}_2)}{\max(|\mathcal{V}_1|, |\mathcal{V}_2|)} \tag{11}$$

$$d'_{\mathcal{E}}(\mathcal{E}_1, \mathcal{E}_2) = \frac{d_{\mathcal{E}}(\mathcal{E}_1, \mathcal{E}_2)}{\frac{1}{2}\left(\min(|\mathcal{V}_1|, |\mathcal{V}_2|) \cdot (\min(|\mathcal{V}_1|, |\mathcal{V}_2|) - 1)\right)} \tag{12}$$

Finally, the dissimilarity between $\mathcal{G}_1$ and $\mathcal{G}_2$ reads as:

$$d(\mathcal{G}_1, \mathcal{G}_2) = \frac{1}{2}\left(d'_{\mathcal{V}}(\mathcal{V}_1, \mathcal{V}_2) + d'_{\mathcal{E}}(\mathcal{E}_1, \mathcal{E}_2)\right) \tag{13}$$

APPENDIX B
DISSIMILARITY MEASURES BETWEEN NODES AND EDGES

A. LETTER-L, LETTER-M AND LETTER-H

Node labels are real-valued 2-dimensional vectors $\mathbf{v}$ of (x, y) coordinates, therefore the dissimilarity measure d_v between two given nodes, say $v^{(a)}$ and $v^{(b)}$, is defined as the plain Euclidean distance:

$$d_v(v^{(a)}, v^{(b)}) = \|\mathbf{v}^{(a)} - \mathbf{v}^{(b)}\|_2$$

Conversely, edges are not labelled.

B. AIDS

Node labels are composed by a string value S_{chem} (chemical symbol), an integer N_{ch} (charge) and a real-valued 2-dimensional vector $\mathbf{v}$ of (x, y) coordinates. For any two given nodes, their dissimilarity is evaluated as:

$$d_v(v^{(a)}, v^{(b)}) = \|\mathbf{v}^{(a)} - \mathbf{v}^{(b)}\|_2 + |N_{ch}^{(a)} - N_{ch}^{(b)}| + d_s(S_{chem}^{(a)}, S_{chem}^{(b)})$$

with $d_s(\cdot, \cdot)$ reading as a plain delta distance:

$$d_s(S_{chem}^{(a)}, S_{chem}^{(b)}) = \begin{cases} 1 & \text{if } S_{chem}^{(a)} \neq S_{chem}^{(b)} \\ 0 & \text{otherwise} \end{cases}$$

Conversely, edge attributes are discarded since not useful for the classification task.

C. GREC

Node labels are composed by a string (*type*) and a real-valued 2-dimensional vector $\mathbf{v}$. The dissimilarity measure d_v between two different nodes reads as:

$$d_v(v^{(a)}, v^{(b)}) = \begin{cases} (1-\chi) \cdot \frac{1}{\sqrt{2}}\|\mathbf{v}^{(a)} - \mathbf{v}^{(b)}\|_2 & \text{if } type^{(a)} = type^{(b)} \\ \chi + (1-\chi) \cdot \frac{1}{\sqrt{2}}\|\mathbf{v}^{(a)} - \mathbf{v}^{(b)}\|_2 & \text{otherwise} \end{cases}$$

Edge labels are defined by an integer value *freq* (frequency) that defines the number of *(type,angle)*-pairs where, in turn, *type* is a string which may assume two values (namely, `arc` or `line`) and *angle* is a real number. Given two edges, say $e^{(a)}$ and $e^{(b)}$ their dissimilarity is defined as follows:

1) If $freq^{(a)} = freq^{(b)} = 1$

$$d_e(e^{(a)}, e^{(b)}) = \begin{cases} \alpha \cdot d^{line}(angle^{(a)}, angle^{(b)}) \\ \quad \text{if } type^{(a)} = type^{(b)} = \texttt{line} \\ \beta \cdot d^{arc}(angle^{(a)}, angle^{(b)}) \\ \quad \text{if } type^{(a)} = type^{(b)} = \texttt{arc} \\ \gamma \quad \text{otherwise} \end{cases}$$

2) If $freq^{(a)} = freq^{(b)} = 2$, $d_e(e^{(a)}, e^{(b)})$, as shown at the top of the next page.

$$d_e(e^{(a)}, e^{(b)}) = \begin{cases} \frac{\alpha}{2} \cdot d^{line}(angle_1^{(a)}, angle_1^{(b)}) + \frac{\beta}{2} \cdot d^{arc}(angle_2^{(a)}, angle_2^{(b)}) & \text{if } type^{(a)} = type^{(b)} = \texttt{line} \\ \frac{\alpha}{2} \cdot d^{line}(angle_2^{(a)}, angle_2^{(b)}) + \frac{\beta}{2} \cdot d^{arc}(angle_1^{(a)}, angle_1^{(b)}) & \text{if } type^{(a)} = type^{(b)} = \texttt{arc} \\ \gamma & \text{otherwise} \end{cases}$$

3) If $freq^{(a)} \neq freq^{(b)}$

$$d_e(e^{(a)}, e^{(b)}) = \delta$$

where $d^{line}(\cdot, \cdot)$ and $d^{arc}(\cdot, \cdot)$ are the module distance normalized respectively in $[-\pi, \pi]$ and $[0, arc_{max}]$. Parameters $\chi, \alpha, \beta, \gamma, \delta \in [0, 1]$ compose the sets $\Pi_v = \{\chi\}$ and $\Pi_e = \{\alpha, \beta, \gamma, \delta\}$ defined in Section II-A1 (also Appendix A) which shall be optimized by the genetic algorithm.

REFERENCES

[1] H. Bunke, "Graph-based tools for data mining and machine learning," in *Machine Learning and Data Mining in Pattern Recognition*, P. Perner and A. Rosenfeld, Eds. Berlin, Germany: Springer, 2003, pp. 7–19.

[2] A. Martino, A. Giuliani, and A. Rizzi, "Granular computing techniques for bioinformatics pattern recognition problems in non-metric spaces," in *Computational Intelligence for Pattern Recognition* (Studies in Computational Intelligence), vol. 777, W. Pedrycz and S.-M. Chen, Eds. Cham, Switzerland: Springer, 2018, pp. 53–81.

[3] E. Pękalska and R. P. Duin, *The Dissimilarity Representation for Pattern Recognition: Foundations and Applications*. Singapore: World Scientific, 2005.

[4] D. Weinshall, D. W. Jacobs, and Y. Gdalyahu, "Classification in non-metric spaces," in *Advances in Neural Information Processing Systems*, M. J. Kearns, S. A. Solla, and D. A. Cohn, Eds. Cambridge, MA, USA: MIT Press, 1999, pp. 838–846.

[5] L. Baldini, A. Martino, and A. Rizzi, "Stochastic information granules extraction for graph embedding and classification," in *Proc. 11th Int. Joint Conf. Comput. Intell.*, vol. 1. Setúbal, Portugal: SciTePress, 2019, pp. 391–402.

[6] L. Baldini, A. Martino, and A. Rizzi, "Towards a class-aware information granulation for graph embedding and classification," in *Proc. 11th Int. Joint Conf., Comput. Intell. (IJCCI)*, Vienna, Austria, Sep. 2019.

[7] A. Martino, A. Giuliani, V. Todde, M. Bizzarri, and A. Rizzi, "Metabolic networks classification and knowledge discovery by information granulation," *Comput. Biol. Chem.*, vol. 84, Feb. 2020, Art. no. 107187.

[8] G. Shi, H. Huang, and L. Wang, "Unsupervised dimensionality reduction for hyperspectral imagery via local geometric structure feature learning," *IEEE Geosci. Remote Sens. Lett.*, vol. 17, no. 8, pp. 1425–1429, Aug. 2020.

[9] F. Luo, H. Huang, Y. Duan, J. Liu, and Y. Liao, "Local geometric structure feature for dimensionality reduction of hyperspectral imagery," *Remote Sens.*, vol. 9, no. 8, p. 790, Aug. 2017.

[10] A. Martino, A. Rizzi, and F. M. Frattale Mascioli, "Supervised approaches for protein function prediction by topological data analysis," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2018, pp. 1–8.

[11] A. Martino, E. Maiorino, A. Giuliani, M. Giampieri, and A. Rizzi, "Supervised approaches for function prediction of proteins contact networks from topological structure information," in *Proc. 20th Scandin. Conf., Image Anal. (SCIA)*, P. Sharma and F. M. Bianchi, Eds., Tromsø, Norway. Cham, Switzerland: Springer, Jun. 2017, pp. 285–296.

[12] H. Bunke and K. Shearer, "A graph distance metric based on the maximal common subgraph," *Pattern Recognit. Lett.*, vol. 19, nos. 3–4, pp. 255–259, Mar. 1998.

[13] H. Bunke, "On a relation between graph edit distance and maximum common subgraph," *Pattern Recognit. Lett.*, vol. 18, no. 8, pp. 689–694, Aug. 1997.

[14] Z. Abu-Aisheh, R. Raveaux, J.-Y. Ramel, and P. Martineau, "An exact graph edit distance algorithm for solving pattern recognition problems," in *Proc. Int. Conf. Pattern Recognit. Appl. Methods*, vol. 1, Setúbal, Portugal: SciTePress, 2015, pp. 271–278.

[15] S. Ghosh, N. Das, T. Gonçalves, P. Quaresma, and M. Kundu, "The journey of graph kernels through two decades," *Comput. Sci. Rev.*, vol. 27, pp. 88–111, Feb. 2018.

[16] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Trans. Knowl. Data Eng.*, early access, Mar. 17, 2020, doi: 10.1109/TKDE.2020.2981333.

[17] F. Xu, H. Uszkoreit, Y. Du, W. Fan, D. Zhao, and J. Zhu, "Explainable AI: A brief survey on history, research areas, approaches and challenges," in *Natural Language Processing and Chinese Computing*, J. Tang, M.-Y. Kan, D. Zhao, S. Li, and H. Zan, Eds. Cham, Switzerland: Springer, 2019, pp. 563–574.

[18] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–17.

[19] J. Gou, Y. Yang, Z. Yi, J. Lv, Q. Mao, and Y. Zhan, "Discriminative globality and locality preserving graph embedding for dimensionality reduction," *Expert Syst. Appl.*, vol. 144, Apr. 2020, Art. no. 113079.

[20] J. Gou, Y. Xue, H. Ma, Y. Liu, Y. Zhan, and J. Ke, "Double graphs-based discriminant projections for dimensionality reduction," *Neural Comput. Appl.*, vol. 32, no. 23, pp. 17533–17550, May 2020.

[21] A. Bargiela and W. Pedrycz, "The roots of granular computing," in *Proc. IEEE Int. Conf. Granular Comput.*, May 2006, pp. 806–809.

[22] W. Pedrycz, "Granular computing: An introduction," in *Proc. Joint 9th IFSA World Congr., 20th NAFIPS Int. Conf.*, vol. 3, 2001, pp. 1349–1354.

[23] A. Bargiela and W. Pedrycz, *Granular Computing: An Introduction*. Boston, MA, USA: Kluwer, 2003.

[24] Y. Yao, "A triarchic theory of granular computing," *Granular Comput.*, vol. 1, no. 2, pp. 145–157, Jun. 2016.

[25] L. A. Zadeh, "Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic," *Fuzzy Sets Syst.*, vol. 90, no. 2, pp. 111–127, Sep. 1997.

[26] W. Pedrycz, "Human centricity in computing with fuzzy sets: An interpretability quest for higher order granular constructs," *J. Ambient Intell. Humanized Comput.*, vol. 1, no. 1, pp. 65–74, Mar. 2010.

[27] Y.-Y. Yao, "The rise of granular computing," *J. Chongqing Univ. Posts Telecommun., Natural Sci. Ed.*, vol. 20, no. 3, pp. 299–308, 2008.

[28] W. Pedrycz and W. Homenda, "Building the fundamentals of granular computing: A principle of justifiable granularity," *Appl. Soft Comput.*, vol. 13, no. 10, pp. 4209–4218, Oct. 2013.

[29] Y. Yao and L. Zhao, "A measurement theory view on the granularity of partitions," *Inf. Sci.*, vol. 213, pp. 1–13, Dec. 2012.

[30] G. Wang, J. Yang, and J. Xu, "Granular computing: From granularity optimization to multi-granularity joint problem solving," *Granular Comput.*, vol. 2, no. 3, pp. 105–120, Sep. 2017.

[31] A. Martino, A. Giuliani, and A. Rizzi, "The universal phenotype," *Organisms. J. Biol. Sci.*, vol. 3, no. 2, pp. 8–10, 2019.

[32] F. M. Bianchi, S. Scardapane, A. Rizzi, A. Uncini, and A. Sadeghian, "Granular computing techniques for classification and semantic characterization of structured data," *Cognit. Comput.*, vol. 8, no. 3, pp. 442–461, Jun. 2016.

[33] A. Martino, A. Giuliani, and A. Rizzi, "(Hyper) graph embedding and classification via simplicial complexes," *Algorithms*, vol. 12, no. 11, p. 223, Oct. 2019.

[34] L. Baldini, A. Martino, and A. Rizzi, "Exploiting cliques for granular computing-based graph classification," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–9.

[35] A. Martino, F. M. Frattale Mascioli, and A. Rizzi, "On the optimization of embedding spaces via information granulation for pattern recognition," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–8.

[36] A. Martino, A. Rizzi, and F. M. Frattale Mascioli, "Efficient approaches for solving the large-scale k-medoids problem: Towards structured data," in *Proc. 9th Int. Joint Conf. Comput. Intell. (IJCCI)*, Funchal-Madeira, Portugal, C. Sabourin, J. J. Merelo, K. Madani, and K. Warwick, Eds. Cham, Switzerland: Springer, Nov. 2019, pp. 199–219.

[37] A. Martino, A. Rizzi, and F. M. Frattale Mascioli, "Efficient approaches for solving the large-scale k-medoids problem," in *Proc. 9th Int. Joint Conf. Comput. Intell.*, vol. 1, Setúbal, Portugal: SciTePress, 2017, pp. 338–347.

[38] F. M. Frattale Mascioli, A. Rizzi, M. Panella, and G. Martinelli, "Scale-based approach to hierarchical fuzzy clustering," *Signal Process.*, vol. 80, no. 6, pp. 1001–1016, Jun. 2000.

[39] W. Pedrycz, *Knowledge-Based Clustering: From Data to Information Granules*. Hoboken, NJ, USA: Wiley, 2005.

[40] W. Pedrycz, "Proximity-based clustering: A search for structural consistency in data with semantic blocks of features," *IEEE Trans. Fuzzy Syst.*, vol. 21, no. 5, pp. 978–982, Oct. 2013.

[41] S. Ding, M. Du, and H. Zhu, "Survey on granularity clustering," *Cognit. Neurodynamics*, vol. 9, no. 6, pp. 561–572, Dec. 2015.

[42] A. Martino, E. De Santis, and A. Rizzi, "An ecology-based index for text embedding and classification," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–8.

[43] D. Doran, S. Schulz, and T. Besold, "What does explainable ai really mean? A new conceptualization of perspectives," in *Proc. CEUR Workshop*, vol. 2071, 2018, pp. 1–8.

[44] S. T. Mueller, R. R. Hoffman, W. Clancey, A. Emrey, and G. Klein, "Explanation in human-AI systems: A literature meta-review, synopsis of key ideas and publications, and bibliography for explainable AI," 2019, *arXiv:1902.01876*. [Online]. Available: http://arxiv.org/abs/1902.01876

[45] M. Dufrêne and P. Legendre, "Species assemblages and indicator species: The need for a flexible asymmetrical approach," *Ecol. Monographs*, vol. 67, no. 3, pp. 345–366, 1997.

[46] N. Tichy, "An analysis of clique formation and structure in organizations," *Administ. Sci. Quart.*, vol. 18, no. 2, pp. 194–208, 1973.

[47] R. D. Luce and A. D. Perry, "A method of matrix analysis of group structure," *Psychometrika*, vol. 14, no. 2, pp. 95–116, Jun. 1949.

[48] N. J. Salkind, *Encyclopedia of Educational Psychology*. Newbury Park, CA, USA: Sage, 2008.

[49] N. Przulj, D. G. Corneil, and I. Jurisica, "Modeling interactome: Scale-free or geometric?" *Bioinformatics*, vol. 20, no. 18, pp. 3508–3515, Dec. 2004.

[50] N. Przulj, "Biological network comparison using graphlet degree distribution," *Bioinformatics*, vol. 23, no. 2, pp. e177–e183, Jan. 2007.

[51] K. Pearson, "The problem of the random walk," *Nature*, vol. 72, no. 1867, p. 342, 1905.

[52] S. V. N. Vishwanathan, N. N. Schraudolph, I. R. Kondor, and K. M. Borgwardt, "Graph kernels," *J. Mach. Learn. Res.*, vol. 11, pp. 1201–1242, Mar. 2010.

[53] L. Backstrom and J. Leskovec, "Supervised random walks: Predicting and recommending links in social networks," in *Proc. 4th ACM Int. Conf. Web Search Data Mining (WSDM)*, 2011, pp. 635–644.

[54] D. Fogaras, B. Rácz, K. Csalogány, and T. Sarlós, "Towards scaling fully personalized PageRank: Algorithms, lower bounds, and experiments," *Internet Math.*, vol. 2, no. 3, pp. 333–358, Jan. 2005.

[55] P. Gupta, A. Goel, J. Lin, A. Sharma, D. Wang, and R. Zadeh, "WTF: The who to follow service at Twitter," in *Proc. 22nd Int. Conf. World Wide Web (WWW)*. New York, NY, USA: ACM, 2013, pp. 505–514.

[56] L. Livi and A. Rizzi, "Graph ambiguity," *Fuzzy Sets Syst.*, vol. 221, pp. 24–47, Jun. 2013.

[57] K. Riesen, X. Jiang, and H. Bunke, "Exact and inexact graph matching: Methodology and applications," in *Managing and Mining Graph Data*. Boston, MA, USA: Springer, 2010, pp. 217–247.

[58] M. Neuhaus and H. Bunke, *Bridging the Gap Between Graph Edit Distance and Kernel Machines*, vol. 68. Singapore: World Scientific, 2007.

[59] H. Bunke and G. Allermann, "Inexact graph matching for structural pattern recognition," *Pattern Recognit. Lett.*, vol. 1, no. 4, pp. 245–253, May 1983.

[60] A. Cinti, F. M. Bianchi, A. Martino, and A. Rizzi, "A novel algorithm for online inexact string matching and its FPGA implementation," *Cognit. Comput.*, vol. 12, no. 2, pp. 369–387, Mar. 2020.

[61] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, Dec. 1997.

[62] F. Wang and J. Sun, "Survey on distance metric learning and dimensionality reduction in data mining," *Data Mining Knowl. Discovery*, vol. 29, no. 2, pp. 534–564, Mar. 2015.

[63] A. Bellet, A. Habrard, and M. Sebban, "A survey on metric learning for feature vectors and structured data," *CoRR*, vol. abs/1306.6709, pp. 1–59, Jun. 2013.

[64] C.-C. Chang, "A boosting approach for supervised mahalanobis distance metric learning," *Pattern Recognit.*, vol. 45, no. 2, pp. 844–862, Feb. 2012.

[65] C. Shen, J. Kim, F. Liu, L. Wang, and A. van den Hengel, "Efficient dual approach to distance metric learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 2, pp. 394–406, Feb. 2014.

[66] L. Yang, R. Jin, L. Mummert, R. Sukthankar, A. Goode, B. Zheng, S. C. H. Hoi, and M. Satyanarayanan, "A boosting framework for visuality-preserving distance metric learning and its application to medical image retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 1, pp. 30–44, Jan. 2010.

[67] X. Yin, T. Shu, and Q. Huang, "Semi-supervised fuzzy clustering with metric learning and entropy regularization," *Knowl.-Based Syst.*, vol. 35, pp. 304–311, Nov. 2012.

[68] Y. Mu, W. Ding, and D. Tao, "Local discriminative distance metrics ensemble learning," *Pattern Recognit.*, vol. 46, no. 8, pp. 2337–2349, Aug. 2013.

[69] M. Bereta, W. Pedrycz, and M. Reformat, "Local descriptors and similarity measures for frontal face recognition: A comparative analysis," *J. Vis. Commun. Image Represent.*, vol. 24, no. 8, pp. 1213–1231, Nov. 2013.

[70] C. C. Aggarwal, J. L. Wolf, P. S. Yu, C. Procopiuc, and J. S. Park, "Fast algorithms for projected clustering," *SIGMOD Rec.*, vol. 28, no. 2, pp. 61–72, Jun. 1999.

[71] A. Martino, M. Giampieri, M. Luzi, and A. Rizzi, "Data mining by evolving agents for clusters discovery and metric learning," in *Neural Advances in Processing Nonlinear Dynamic Signals*, A. Esposito, M. Faundez-Zanuy, F. C. Morabito, and E. Pasero, Eds. Cham, Switzerland: Springer, 2019, pp. 23–35.

[72] K. Riesen and H. Bunke, "IAM graph database repository for graph based pattern recognition and machine learning," in *Structural, Syntactic, and Statistical Pattern Recognition*, N. da Vitoria Lobo, T. Kasparis, F. Roli, J. T. Kwok, M. Georgiopoulos, G. C. Anagnostopoulos, and M. Loog, Eds. Berlin, Germany: Springer, 2008, pp. 287–297.

[73] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. IT-13, no. 1, pp. 21–27, Jan. 1967.

[74] K. Riesen and H. Bunke, "Approximate graph edit distance computation by means of bipartite graph matching," *Image Vis. Comput.*, vol. 27, no. 7, pp. 950–959, 2009.

[75] K. Riesen and H. Bunke, "Graph classification by means of Lipschitz embedding," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 6, pp. 1472–1483, Dec. 2009.

[76] J. Gibert, E. Valveny, and H. Bunke, "Dimensionality reduction for graph of words embedding," in *Graph-Based Representations in Pattern Recognition*, X. Jiang, M. Ferrer, and A. Torsello, Eds. Berlin, Germany: Springer, 2011, pp. 22–31.

[77] G. Da San Martino, N. Navarin, and A. Sperduti, "Ordered decompositional DAG kernels enhancements," *Neurocomputing*, vol. 192, pp. 92–103, Jun. 2016.

[78] D. Bacciu, F. Errica, and A. Micheli, "Contextual graph Markov model: A deep and generative approach to graph processing," in *Proc. 35th Int. Conf. Mach. Learn. (ICML)*, vol. 1, 2018, pp. 495–504.

[79] M. Martineau, R. Raveaux, D. Conte, and G. Venturini, "Learning error-correcting graph matching with a multiclass neural network," *Pattern Recognit. Lett.*, vol. 134, pp. 68–76, Jun. 2020.

[80] A. Martino and A. Rizzi, "(Hyper) graph kernels over simplicial complexes," *Entropy*, vol. 22, no. 10, p. 1155, Oct. 2020.

[81] M. I. Yousuf and S. Kim, "Guided sampling for large graphs," *Data Mining Knowl. Discovery*, vol. 34, no. 4, pp. 905–948, Jul. 2020.

[82] A. H. Rasti, M. Torkjazi, R. Rejaie, N. Duffield, W. Willinger, and D. Stutzbach, "Respondent-driven sampling for characterizing unstructured overlays," in *Proc. IEEE 28th Conf. Comput. Commun. (INFOCOM)*, Apr. 2009, pp. 2701–2705.

[83] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou, "Practical recommendations on crawling online social networks," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 9, pp. 1872–1892, Oct. 2011.

[84] M. Al Hasan and M. J. Zaki, "Output space sampling for graph patterns," *Proc. VLDB Endowment*, vol. 2, no. 1, pp. 730–741, Aug. 2009.

[85] W. K. Hastings, "Monte Carlo sampling methods using Markov chains and their applications," *Biometrika*, vol. 57, no. 1, pp. 97–109, Apr. 1970.

[86] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *J. Chem. Phys.*, vol. 21, no. 6, pp. 1087–1092, Jun. 1953.

ALESSIO MARTINO (Associate Member, IEEE) graduated in communications engineering *(summa cum laude)* from the University of Rome "La Sapienza," Italy, in October 2016. His bachelor's and master's degrees theses regarded EU-FP7 and EU-FP8 projects, respectively. From November 2016 to November 2019, he has served as a Ph.D. Research Fellow in information and communications technologies with the Department of Information Engineering, Electronics and Telecommunications, University of Rome "La Sapienza," with a final dissertation on pattern recognition techniques in non-metric domains. He currently holds a Postdoctoral Research Fellow position at the University of Rome "La Sapienza." He also served as a Scientific Collaborator with Consortium for Research in Automation and Telecommunication, Rome, Italy. His research interests include machine learning, computational intelligence, and knowledge discovery. His current research interests include large-scale machine learning, advanced pattern recognition systems, big data analysis, parallel and distributed computing, granular computing, and complex systems modeling, in applications, including bioinformatics and computational biology, natural language processing, and energy distribution networks.

ANTONELLO RIZZI (Senior Member, IEEE) received the Ph.D. degree in information and communication engineering from the University of Rome "La Sapienza." In September 2000, he joined the Information and Communication Department, as an Assistant Professor. Since July 2010, he has been with the Department of Information Engineering, Electronics and Telecommunications (DIET), University of Rome "La Sapienza," where he currently serves as an Associate Professor. Since 2008, he has been the Scientific Coordinator and the Research and Development Technical Director of the Intelligent Systems Laboratory, Research Center for Sustainable Mobility of Lazio region, Italy. He is also working on smart grids and microgrids modeling and control, intelligent systems for sustainable mobility, battery management systems, granular computing, data mining and knowledge discovery, computational biology, machine learning in non-metric spaces, graph and sequence matching, agent-based clustering, and parallel and distributed computing. He has (co)authored more than 190 international journal/conference papers and book chapters. His major research interests include computational intelligence and pattern recognition, including supervised and unsupervised machine learning techniques, neural networks, fuzzy systems, and evolutionary algorithms. His research interest includes the design of automatic modeling systems, focusing on classification, clustering, function approximation, and prediction problems.