Exploiting Cliques for Granular Computing-based Graph Classification

Luca Baldini, Alessio Martino, Antonello Rizzi
Department of Information Engineering, Electronics and Telecommunications
University of Rome "La Sapienza"
Via Eudossiana 18, 00184 Rome, Italy
{luca.baldini, alessio.martino, antonello.rizzi}@uniroma1.it

***Abstract*—The most fascinating aspect of graphs is their ability to encode the information contained in the inner structural organization between its constituting elements. Learning from graphs belong to the so-called Structural Pattern Recognition, from which Graph Embedding emerged as a successful method for processing graphs by evaluating their dissimilarity in a suitable geometric space. In this paper, we investigate the possibility to perform the embedding into a geometric space by leveraging to peculiar constituent graph substructures extracted from training set, namely the maximal cliques, and providing the performances obtained under three main aspects concerning classification capabilities, running times and model complexity. Thanks to a Granular Computing approach, the employed methodology can be seen as a powerful framework able to synthesize models suitable to be interpreted by field-experts, pushing the boundary towards new frontiers in the field of explainable AI and knowledge discovery also in big data contexts.**

***Index Terms*—Structural Pattern Recognition, Supervised Learning, Embedding Spaces, Granular Computing, Graph Edit Distances.**

I. Introduction

The possibility of solving pattern recognition problems in the graphs domain challenged computer scientists and machine learning engineers alike for more than two decades. That is because graphs are able to encode both topological information (namely, relationship between entities) and semantic information (whether nodes and/or edges are equipped with suitable attributes). In turn, this high level of abstraction and customization made graphs suitable mathematical objects for modelling several real-world systems in various application fields such as biology [1]–[3], social networks [4], [5], computer vision and image analysis [6], [7]. The drawback when dealing with graph-based pattern recognition relies on the computational complexity required in order to measure the (dis)similarity between two graphs, which exponentially grows with respect to the input size [8]. This inevitably results in an heavy computational burden when it comes to perform pattern recognition in the graphs domain, especially when also node and/or edge attributes must be taken into account.

In the literature, common strategies include feature engineering, where numerical features are manually extracted from the input patterns and concatenated in a vector form (despite its simplicity, this procedure requires either a deep knowledge on the modelled system in order to list such features or an expensive and time consuming trial-and-error search); kernel methods [9], [10], where semi-positive definite kernel functions are used to measure similarity between input data via reproducing kernel Hilbert spaces (see e.g. [11]–[14]); using ad-hoc dissimilarity measures (e.g., edit distances [15]–[17]) in the input space in order to directly solve the pattern recognition problem without moving towards Euclidean spaces; or by means of (explicit) embedding techniques. Embedding techniques have the same target of feature engineering-based ones: that is, move the pattern recognition problem from the structured input domain towards the Euclidean space in which classification is performed. Nonetheless, an automatic synthesis of the embedding space is a delicate issue that must fill the informative and semantic gap between the two domains [16]. Alongside neural approaches such as [18]–[22], an efficient strategy for solving this task relies on Granular Computing (GrC) [23], [24]. GrC is an information processing paradigm suitable for complex data mining, which are often characterized by different levels of representation. The main objective of a GrC approach is to represent together (group) entities that are indistinguishable at a given level of abstraction [25]: in the technical literature, these groups are known as *information granules*. However, finding a suitable set of meaningful and recurrent information granules is a task that is both problem- and data-dependent, therefore they are hardly known a-priori and designing an intelligent system in order to automatically synthesize the set of information granules is of paramount importance. In the current literature, the GrC paradigm has been successfully employed in order to design effective advanced pattern recognition systems dealing in both graphs and sequences domains (see e.g. [16], [17], [26]–[28] and references therein). A common strategy relies on extraction of substructures from structured data (e.g., paths from graphs, k-mers or n-grams from text corpora) and then use a clustering procedure in order to automatically discover groups of similar/frequent pivotal data aggregates, hence prospective information granules. These approaches have the major drawback of a non-negligible time and space complexity, especially when big datasets have to be analyzed: for example, the number of paths in an n-vertex graph goes like $\mathcal{O}(n!)$ in the worst-case, whereas the prospective number of k-mers in a sequence whose elements are drawn from a finite alphabet of size m goes like m^k. These plain back-of-the-envelope

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

calculations remark the unfeasibility of exhaustive extractions for automatic synthesis of information granules.

In order to overcome these problems and make GrC-based pattern recognition systems appealing towards big datasets, previous works such as [16] and [17] focused on finding lightweight procedures based on random walks, showing that the vast majority of the information is still preserved whilst featuring remarkably lower running times and memory usage with respect to an exhaustive enumeration of the paths [29]. Random walks trace back to the beginning of the 20$^{\text{th}}$ century [30] and have been widely studied since, especially in the context of Markov chains. A random walk on a graph is a particular case of Markov chain, where the transition matrix (namely, the probability of 'jumping' from one node to another) is given by $\mathbf{D}^{-1}\mathbf{A}$, where $\mathbf{D}$ and $\mathbf{A}$ are the degree and the adjacency matrix of the graph, respectively. 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 [13] and Twitter (amongst others) uses random walks for its recommender system [31], [32].

In this paper, we pose our attention to another peculiar substructure that can be drawn from a graph: the clique. Cliques originate in social sciences indicating groups of individuals who interact with one another and share similar interests [33], [34]. The seminal work [35] has brought widespread use of the term 'clique' in graph theory and network analysis, where the authors used complete subgraphs to model (social) cliques in social networks. In fact, this is the current definition of a (graph) clique: a subset of vertices forms a clique if the induced subgraph is complete (i.e., every two distinct vertices in the clique are adjacent). The same does not hold in other type of subgraphs such as graphlets (induced subgraphs) and motifs (partial subgraphs). Theoretically speaking, the number of maximal cliques (i.e., cliques that cannot be made any larger) goes like $\mathcal{O}(3^{n/3})$ in the worst-case scenario [36] for an n-vertex graph: this result suggests that the number of prospective information granules is way lower with respect to the paths case. Furthermore, despite being a well-known NP complete problem, finding the maximal cliques in a graph can be pursued in exponential time, for example thanks to the Bron-Kerbosch algorithm [37]. In order to address whether cliques can be interpreted as meaningful information granules for synthesizing an embedding space for graph classification purposes we consider GRALG, a GrC-based classification system suitable for dealing in the graphs domain. GRALG has been originally proposed in [29] and later applied in the context of image classification [38], [39]: in these works, it has been equipped with an exhaustive subgraphs extractor which, however, turned out to be unfeasible for large datasets. As previously introduced, in [16] and [17] we solved this problem by equipping GRALG with a random walk-based extractor which operates in an unsupervised (the former) and class-aware (the latter) fashion. In this work, as instead, we investigate the possibility of using cliques instead of walks and paths for the very same purpose of automatically synthesizing a set of possibly meaningful information granules for building an embedding space for graph classification.

The remainder of this paper is structured as follows: in Section II we introduce GRALG, the GrC-based graph classification system at the basis of this work; in Section III the training and testing phases are described; in Section IV we introduce the datasets used for analysis, along with the five classifiers used for comparison and the proper computational results; finally, Section V concludes the paper.

II. GRALG

In the context of Graph Embedding, GRALG (GRanular computing Approach for Labelled Graphs) [29] is proposed as a classification system for labeled graphs by deploying a GrC approach, which has been shown to be helpful to unravel complex system described by structured data [26]. The main idea behind GRALG is to let emerge, from an initial set of graphs, an *alphabet of symbols* $\mathcal{A} = \{s_i, \ldots, s_n\}$ composed by relevant substructures (i.e., recurrent, meaningful), by exploring different levels of granulation deploying an unsupervised method. These *symbols* will serve as pivotal entities for performing the embedding from the graphs domain into a geometric space via the so-called *symbolic histograms*.

When moving from a structured domain into geometric spaces, an undoubted revenue comes from the possibility to take advantage of the latter's well-defined mathematical properties. Indeed, geometric spaces are often endowed with a straightforward *distance function*, e.g. the Euclidean distance, that has a key role in Pattern Recognition systems. On the other hand, the operations involved for performing the graph embedding rely on a dissimilarity measure (Graph Edit Distance–GED) defined directly in the graphs domain.

Graph Edit Distance: Generally speaking, a GED evaluates the dissimilarity between two graph, say G_1 and G_2, as the minimum cost path needed to transform G_1 into G_2 by applying a sequence of atomic operations (defined as substitution, insertion and deletion) on both nodes and edges. Formally, the distance function $d : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ can be seen as the following minimization problem:

$$d(G_1, G_2) = \min_{(e_1,\ldots,e_k)\in\mathcal{O}(G_1,G_2)} \sum_{i=1}^{k} c\left(e_i\right) \qquad (1)$$

where $\mathcal{O}(G_1, G_2)$ is the set of all possible series of operations that turn G_1 into G_2. An exact solution of Eq. (1) is practically unfeasible due to the combinatorial computational complexity involved [40]. For this reason, a *node Best Match First* (nBMF) [41] strategy is employed. Formally, let $G_1 = (\mathcal{V}_1, \mathcal{E}_1, \mathcal{L}_v, \mathcal{L}_e)$ and $G_2 = (\mathcal{V}_2, \mathcal{E}_2, \mathcal{L}_v, \mathcal{L}_e)$ be two labeled graphs where $\mathcal{L}_v$ and $\mathcal{L}_e$ are the sets of attributes on nodes and edges. Let also $d_v^{\pi_v} : \mathcal{L}_v \times \mathcal{L}_v \to \mathbb{R}$ and $d_e^{\pi_e} : \mathcal{L}_e \times \mathcal{L}_e \to \mathbb{R}$ be the dissimilarity measures needed to compare nodes and edges according to $\mathcal{L}_v$ and $\mathcal{L}_e$ where, for the sake of generalization, $d_v^{\pi_v}$ and $d_e^{\pi_e}$ could be possibly parametric, with parameters sets π_v and π_e. The nBMF evaluates the atomic operation costs c_{edge}^{sub}, c_{edge}^{ins}, c_{edge}^{del}, c_{node}^{sub}, c_{node}^{ins}, c_{node}^{del} using the following greedy procedure:

1) find pair of node matches (i.e., most similar) between G_1 and G_2 according to $d_v^{\pi_v}$: any match accounts for a node substitution
2) node substitutions and deletions are determined by the order discrepancy between the two graphs
3) for each pair of nodes from step 1, assess whether an edge exist in G_1 and G_2: if so, this accounts as an edge substitution; otherwise this accounts as a node deletion or substitution.

The importance of each operation can be quantified by means of suitable weights, respectively w_{node}^{sub}, w_{edge}^{sub}, w_{node}^{ins}, w_{edge}^{ins}, w_{node}^{del}, w_{edge}^{del} bounded in $[0, 1]$. Finally, the dissimilarities between nodes and edges, $d_{\mathcal{V}}(\mathcal{V}_1, \mathcal{V}_2)$ and $d_{\mathcal{E}}(\mathcal{E}_1, \mathcal{E}_2)$, read as:

$$\begin{aligned} d_{\mathcal{V}}(\mathcal{V}_1, \mathcal{V}_2) &= w_{node}^{sub} \cdot c_{node}^{sub} + w_{node}^{ins} \cdot c_{node}^{ins} + w_{node}^{del} \cdot c_{node}^{del} \\ d_{\mathcal{E}}(\mathcal{E}_1, \mathcal{E}_2) &= w_{edge}^{sub} \cdot c_{edge}^{sub} + w_{edge}^{ins} \cdot c_{edge}^{ins} + w_{edge}^{del} \cdot c_{edge}^{del} \end{aligned} \tag{2}$$

In order to overcome skewness due to G_1 and G_2 (possibly) having different sizes, a normalization step is performed by taking into account their respective orders:

$$\begin{aligned} d'_{\mathcal{V}}(\mathcal{V}_1, \mathcal{V}_2) &= \frac{d_{\mathcal{V}}(\mathcal{V}_1, \mathcal{V}_2)}{\max(o_1, o_2)} \\ d'_{\mathcal{E}}(\mathcal{E}_1, \mathcal{E}_2) &= \frac{d_{\mathcal{E}}(\mathcal{E}_1, \mathcal{E}_2)}{\frac{1}{2}\left(\min(o_1, o_2) \cdot (\min(o_1, o_2) - 1)\right)} \end{aligned} \tag{3}$$

with $o_1 = |\mathcal{V}_1|$ and $o_2 = |\mathcal{V}_2|$. The normalization factors for both nodes and edges are defined by considering that the nBMF heuristic computes at most $\min(o_1, o_2)$ matches between nodes in G_1 and G_2, and that the edit operations on edges are induced from those performed on nodes.The overall distance between the two graphs is then given by

$$d(G_1, G_2) = \frac{1}{2}\left(d'_{\mathcal{V}}(\mathcal{V}_1, \mathcal{V}_2) + d'_{\mathcal{E}}(\mathcal{E}_1, \mathcal{E}_2)\right) \tag{4}$$

The remainder of this Section aims at defining individually the blocks needed to perform the graph embedding: in Section II-A–II-B, we give a detailed description of the process needed to synthesize the alphabet starting from the maximal cliques extracted from a given graphs set. Next, in Section II-C, we describe the building block for embedding graphs into a geometric space, with Section II-D addressing the final classification block.

A. Extractor

This block is in charge to extract atomic substructures belonging to a given graph by following a class-aware strategy as described and tested in [17]. Additionally, a stochastic subsampling method allows to fix the total number of subgraphs to extract with a user-defined parameter in order to address the computational and memory footprint issues [16], typical of exhaustive procedures [29].

In details, let $\mathcal{S} \subset \mathcal{G}$ be a set of graphs with ground-truth class labels $L = 1 \dots N$, this procedure aims at building class-stratified subgraphs sets $\mathcal{S}_g^L$ starting from graphs in $\mathcal{S}$, taking into account the frequency of the L-th class in the dataset. Let also W be the user-defined cardinality of $\bigcup_{L=1}^{N} S_g^L$, then the class-aware extraction can be summarized as follows:

1) Evaluate the absolute frequency f_L for each class $L = 1 \dots N$, such that $\sum_{L=1}^{N} f_L = |\mathcal{S}|$
2) According to W, set the number of subgraphs to be extracted for each class as $N_L = \frac{f_L}{|\mathcal{S}|} \cdot W$
3) For each label L, extract N_L subgraphs from graphs belonging to $\mathcal{S}$ having label L and populate the set $\mathcal{S}_g^L$.

The subgraphs extraction may occur by employing different strategies that take into account distinct peculiarities of the desired substructures. That is, in step 3 various algorithms can be deployed depending on the subgraph's properties intended. For example, in [16], we considered subgraphs emerged by traversing graphs using the well-known Breadth First Search (BFS) and Depth First Search (DFS) algorithms: given a root node, BFS explores all of the neighbor nodes at the present depth prior to moving on to the nodes at the next depth level, whereas DFS acts in an opposite manner by exploring the node branch as far as possible before being backtracking and expand other nodes. In this work, we test the possibility of a clique extractor based on the Bron-Kerbosch algorithm. For sake of clarity, the aforementioned extractor is described separately:

Maximal cliques Extractor based on Bron-Kerbosch: This Extractor aims at enumerating all maximal cliques of a given graph $G \in \mathcal{S}$. For this purpose, the well-known Bron-Kerbosch algorithm has been employed, which uses a recursive backtracking strategy that looks for all maximal cliques. In order to describe the method, let R, P, and X, be three disjoint vertices sets. In each recursion step, R stands for the set containing a possible maximal clique, P makes note of not yet visited vertices, whereas X keeps track for the already visited nodes in earlier steps that serves for avoiding a clique is repeated in the backtracking mechanism. In every call to the main function, the procedure checks whether R is maximal by looking at the set $P \cup X$. Since this set is made up by the vertices which are adjacent to R (a potential maximal cliques), $P \cup X \neq \emptyset$ proves that R is not maximal. In practice, the method works by calling recursively the main procedure for all $v \in P$ for the clique $R \cup \{v\}$ and restricting P and X to the neighborhood $\Gamma(v)$. When a cliques R is signed as maximal, the algorithm backtracks by swapping the vertex v from P to X guaranteeing that a clique is not enumerated multiple times. A detailed description for the procedure can be found in Algorithm 1.

The designed Extractor takes as input a graph G for which is required to enumerate the maximal cliques. Then, when the Bron-Kerbosch algorithm finds a maximal clique in G, a subgraph $g = \{\mathcal{V}_g, \mathcal{E}_g\}$ with the vertices $\mathcal{V}_g \equiv R$ is created in order to save the corresponding clique subgraph. The set C contains all maximal cliques found in G.

B. Granulator

This block defines the operations needed to synthesize an alphabet of symbols $\mathcal{A} = \{s_1, \dots, s_n\}$, i.e. information granules. Thus, starting from a set of subgraphs, the alphabet $\mathcal{A}$ is intended to collect only relevant and meaningful substructures

Algorithm 1 Maximal Cliques Extractor

procedure CLIQUES EXTRACTOR(Graph $G = \{\mathcal{V}, \mathcal{E}\}$)
  P: initialize with vertices $\mathcal{V}$
  $R = \{\}$
  $X = \{\}$
  g: maximal clique subgraph with $\mathcal{V}_g$ vertices
  C: initially empty set of maximal cliques
  procedure BRON-KERBOSCH(P,R,X)
    if $P \cup X = \emptyset$ **then**
      Set $\mathcal{V}_g = R$
      Append g in C
      return R as maximal clique
    for $v \in P$ **do**
      BRON-KERBOSCH$\big(P \cap \Gamma(v), ...$
        $R \cup \{v\}, X \cap \Gamma(v)\big)$
      $P = P \setminus \{v\}$
      $X = X \cup \{v\}$
  end
  return C cliques container
end

for the problem at hand. In our approach, we used to synthesize symbols by means of *clusters* emerged by performing a Basic Sequential Algorithmic Scheme (BSAS) clustering algorithm [16], [17]. A major BSAS benefit relies on the number of clusters being not defined a-priori. Indeed, BSAS only relies on two parameters, θ and Q, defining the inclusion threshold for a pattern to belong to a cluster and the maximum number of allowed clusters. The latter parameter Q aims at bounding the maximum number of clusters since, for small θ, the number of clusters might explode. The parameter θ enables to explore the problem at different level of granularity: by varying the threshold, the granulator orchestrates a *clustering ensemble*, hence generates different partitions where clusters emerge at different levels of resolution, depending on the considered θ value. Then, in all partitions, each cluster $\mathbf{C}$ is evaluated with a cluster quality index $F(\mathbf{C})$ defined as:

$$F(\mathbf{C}) = \eta \cdot \Phi(\mathbf{C}) + (1 - \eta) \cdot \Theta(\mathbf{C}) \tag{5}$$

where the two terms $\Phi(\mathbf{C})$ and $\Theta(\mathbf{C})$ are defined as

$$\Phi(\mathbf{C}) = \frac{1}{|\mathbf{C}| - 1} \sum\nolimits_i d(g^\star, g_i) \tag{6}$$

$$\Theta(\mathbf{C}) = 1 - |\mathbf{C}|/|\mathcal{S}_g| \tag{7}$$

where, in turn, the representative of cluster $\mathbf{C}$ is denoted with $g^\star$ and g_i is the i^{th} pattern in the cluster. Specifically, $g^\star$ is defined as the medoid of the cluster, i.e. the pattern that minimizes the pairwise sum of distances between all patterns in the cluster. As shown in Eq. (5), the quality index of each cluster accounts both its *compactness* (Eq. (6)) and *cardinality* (Eq. (7)) with a trade-off parameter $\eta \in [0, 1]$. Eventually, the method retains only well-compact and populated clusters by discarding clusters whose quality index is below a threshold τ_F. According to the class-aware strategy, the clustering ensemble acts on each $\mathcal{S}_g^L$ set separately, enabling the synthesis of class-related symbols properly organized in different alphabets $\mathcal{A}_L$. As all alphabets $\mathcal{A}_L$ are ready, they will be merged together $\mathcal{A} = \bigcup_{L=1}^{N} \mathcal{A}_L$.

C. Embedder

Starting from the alphabet $\mathcal{A} = \{s_1 \dots s_n\}$ synthesized during the granulation phase, the *Embedder* has the role to map graphs towards a geometric Euclidean space. In other words, it aims at building a function $\phi : \mathcal{G} \to \mathcal{D}$ where $\mathcal{D} \subseteq \mathbb{R}^n$ is a suitable n-dimensional space. The function ϕ is designed according to the symbolic histograms approach: symbols in $\mathcal{A}$ serve as pivotal structures for building a vector $\mathbf{h}$, i.e. the symbolic histogram. Formally, let $G \in \mathcal{G}$ be the graph to be embedded and let G_{exp} an appropriate expansion of G in subgraphs, i.e. the set of subgraphs forming G. The symbolic histogram $\mathbf{h}$ is built as follows:

$$\mathbf{h} = \phi^{\mathcal{A}}(G_{exp}) = [occ(s_1, G_{exp}), \dots, occ(s_n, G_{exp})] \tag{8}$$

In plain words, the function $occ : \mathcal{A} \times \mathcal{G} \to \mathbb{N}$ counts the occurrences of a given symbol s_j where $j = 1 \dots n$ within the subgraphs in G_{exp}. According to a suitable dissimilarity measure (i.e., GED), a symbol-to-subgraph match is scored if their dissimilarity is below a symbol-dependent threshold $\tau_j = \Phi(\mathbf{C}_j) \cdot \epsilon$, where $\epsilon \geq 1$ is user-defined parameter defining a tolerance margin in the subgraphs dissimilarity.

In the above discussion, the method used to construct G_{exp} is of utmost importance. Indeed, if this set is build according to an exhaustive extraction, the system is likely to lead to a memory issues due to the combinatorial complexity needed to enumerate all the subgraphs of a given graph. It is worth noting that since the number of matches grows according to the cardinality of the symbols and the number of substructures in G_{exp}, is also fairly possible to run into time issues. In this work, we choose to build the aforementioned set following the same method employed in the extractor described in Section II-A: given a new graph G, it is decomposed in the set of its maximal cliques according to the Bron-Kerbosch algorithm. If, on one hand, the theoretical number of cliques goes like $\mathcal{O}(3^{n/3})$ in the worst-case, in practice a subset of vertices of G has to satisfy strong requirements to be eligible as a clique, notably it has to be a fully connected subgraph that can be no more expanded in a larger clique. This highly restricts the cardinality of the set G_{exp} with respect to an exhaustive enumeration of all subgraphs.

D. Classifier

The final block aims at evaluating the whole system through a suitable classifier. To this end, we employed a simple K-NN decision rule working on the embedding space $\mathcal{D}$ described previously in Section II-C: a new incoming pattern $\mathbf{h}_i$ is assigned to the most frequent label among its K nearest patterns. The performance of the classification system is defined as the ratio of correctly classified patterns.

III. Alphabet Optimization and Features Selection

In Section II, we gave a detailed description of the atomic procedures that made up the algorithm itself. In this Section, we provide the explanation of the system as whole, showing the methods for optimizing the crucial parameters for each block separately and a features selection optimization for selecting only relevant symbols for the problem at hand. For sake of description, let $\mathcal{S}$ be a graph dataset (with graphs possibly labeled on nodes and/or edges) and let $\mathcal{S}_{tr}$, $\mathcal{S}_{vs}$ and $\mathcal{S}_{ts}$ be training, validation and test set drawn from $\mathcal{S}$.

In the first step, the *Extractor* finds all the maximal cliques for each graphs in $\mathcal{S}_{tr}$. Thus, following the class-aware strategy, N class-specific subgraph sets $\mathcal{S}^L_{g,tr}$ are created using the parameter W according to Section II-A. Consequently, the *Granulator* takes as input $\mathcal{S}^L_{g,tr}$ and synthesize N different alphabet sets $\mathcal{A}_L$. In the next phase, these sets will be merged in the set $\mathcal{A} = \bigcup_{L=1}^{N} \mathcal{A}^L$, feeding the *Embedder* block. Thus, for each $G^{tr}_i \in \mathcal{S}_{tr}$ and $G^{vs}_i \in \mathcal{S}_{vs}$, the related vectors $\mathbf{h}^{tr}_i \in \mathcal{D}$ and $\mathbf{h}^{vs}_i \in \mathcal{D}$ are built in the embedding space $\mathcal{D} \subseteq \mathbb{R}^n$. It is worth noting that each graph in $\mathcal{S}_{tr}$ and $\mathcal{S}_{vs}$ needs to be expanded by collecting all its maximal cliques, as discussed in Section II-C. Finally, the K-NN decision rule evaluates the performances on the embedding space $\mathcal{D}$ in classifying the vectors $\mathbf{h}_{vs}$, where $\mathbf{h}_{tr}$ serves as training set.

A. Alphabet Optimization

The procedures described in Section II rely on many parameters which are problem- and data-dependent and hardly known a-priori. Notably, these parameters are responsible for the synthesis of an optimal set of symbols $\mathcal{A}^*$. A genetic algorithm is considered for automatic tuning of the aforementioned parameters, hence the genetic code can be summerized as follows:

$$[Q \quad \tau_F \quad \eta \quad \mathcal{W} \quad \Pi] \tag{9}$$

where

- $Q \in [1, 500]$ is the BSAS parameter that sets the maximum number of allowed clusters
- $\tau_F \in [0, 1]$ defines the threshold for promoting a cluster representative to a symbol
- $\eta \in [0, 1]$ weights compactness and cardinality in Eq. (5)
- $\mathcal{W} = \{w^{sub}_{node}, w^{sub}_{edge}, w^{ins}_{node}, w^{ins}_{edge}, w^{del}_{node}, w^{del}_{edge}\}$ are the insertion, deletion and substitution weights for both nodes and edges involved in the GED dissimilarity measure
- $\Pi = \{\pi_v, \pi_e\}$ are the parameters for the nodes/edges dissimilarity measures $d^{\pi_v}_v$ and $d^{\pi_e}_e$, if applicable.

The genetic algorithm is equipped with elitism, mutation and crossover operators that allow to move candidate solutions (individuals) from one generation to the next, with the accuracy achieved by the K-NN in classifying the validation set $\mathbf{h}_{vs}$ serving as the objective function, to be maximized. At the end of the optimization, together with the optimized alphabet $\mathcal{A}^*$, we also retain the parameters related to the GED, namely $\mathcal{W}^*$ and Π^*.

B. Feature Selection

The optimized alphabet $\mathcal{A}^\star$ obtained in the previous Section, may show a large cardinality and is likely to contain unnecessary symbols. For this reason, we designed a feature selection phase based on a genetic algorithm whose goal is to retain only significant and essential symbols. Recalling that $\mathbf{h}_{tr} \in \mathcal{D}$ with $\mathcal{D} \subseteq \mathbb{R}^n$, we filter relevant symbols by multiplying component-wise each vector in $\mathbf{h}_{tr}$ with a projection mask $\mathbf{m} \in \{0, 1\}^{|\mathcal{A}^\star|}$. Consequently, a reduced embedding space $\overline{\mathcal{D}} \subseteq \mathbb{R}^m$ with $m = |\{i : \mathbf{m}_i = 1\}| \leq n$ is spanned by the projected vectors $\overline{\mathbf{h}}_{tr}$. Accordingly, the validation set $\mathbf{h}_{vs}$ is projected as well using the same mask $\mathbf{m}$ and, finally, the classification system evaluates the accuracy ω on classifying $\overline{\mathbf{h}}_{vs}$. In order to drive the evolution in finding the best projection mask, the individuals' fitness function J, to be maximized, is defined as a convex linear combination between the accuracy ω and the cost of the mask $\mu = |\{i : \mathbf{m}_i = 1\}|/|\mathbf{m}|$:

$$J = \alpha \cdot \omega + (1 - \alpha) \cdot (1 - \mu) \tag{10}$$

where $\alpha \in [0, 1]$ is a trade-off parameter that weights the relevance between ω and μ. At the end of the optimization, the best individual's mask $\mathbf{m}^\star$ is retained and used to generate the reduced embedding space $\overline{\mathcal{D}}^\star \subseteq \mathbb{R}^m$ and the reduced alphabet $\overline{\mathcal{A}}^\star$ with cardinality $|\overline{\mathcal{A}}^\star| = m$.

C. Model Evaluation

The final stage of the algorithm is the evaluation of the synthesized model on the test set. Thus, for all graphs $G \in \mathcal{S}_{ts}$, the expansion in the set of maximal cliques occurs and consequently, these graphs are embedded in the geometric space $\overline{\mathcal{D}^\star}$ by the embedder block using the optimize parameters $\mathcal{W}^\star$, $\Pi^\star$ and $\overline{\mathcal{A}}^\star$. At this point, vectors in $\mathbf{h}_{ts}$ are available for being classified by the K-NN decision rule: the accuracy on the test set serves as the performance measure for the whole GRALG classification system.

IV. Experiments

A. Datasets Description

In our experiments, we tested the classification system on five different datasets taken from the IAM repository [42]:

Letter: three datasets of hand-written letters with growing level of distortion: low (L), medium (M), high (H). Node dissimilarity measure is set as a Euclidean distance, whereas edges are matched with a delta distance since they are unlabelled.

GREC: this dataset contains graphs that represent architectural and electronic drawings symbols. A custom dissimilarity measure is employed for matching vertices, since they are labeled with a complex data structure, as well as the edge dissimilarity measure. Furthermore, both dissimilarity measures rely on five parameters bounded in $[0, 1]$ that define the set Π described in Section III-A

AIDS: each pattern represents molecule that shows or not activities against HIV. Nodes are atoms whereas edges are the related covalent bonds. Dissimilarity measure on nodes is a non parametric custom distance function.

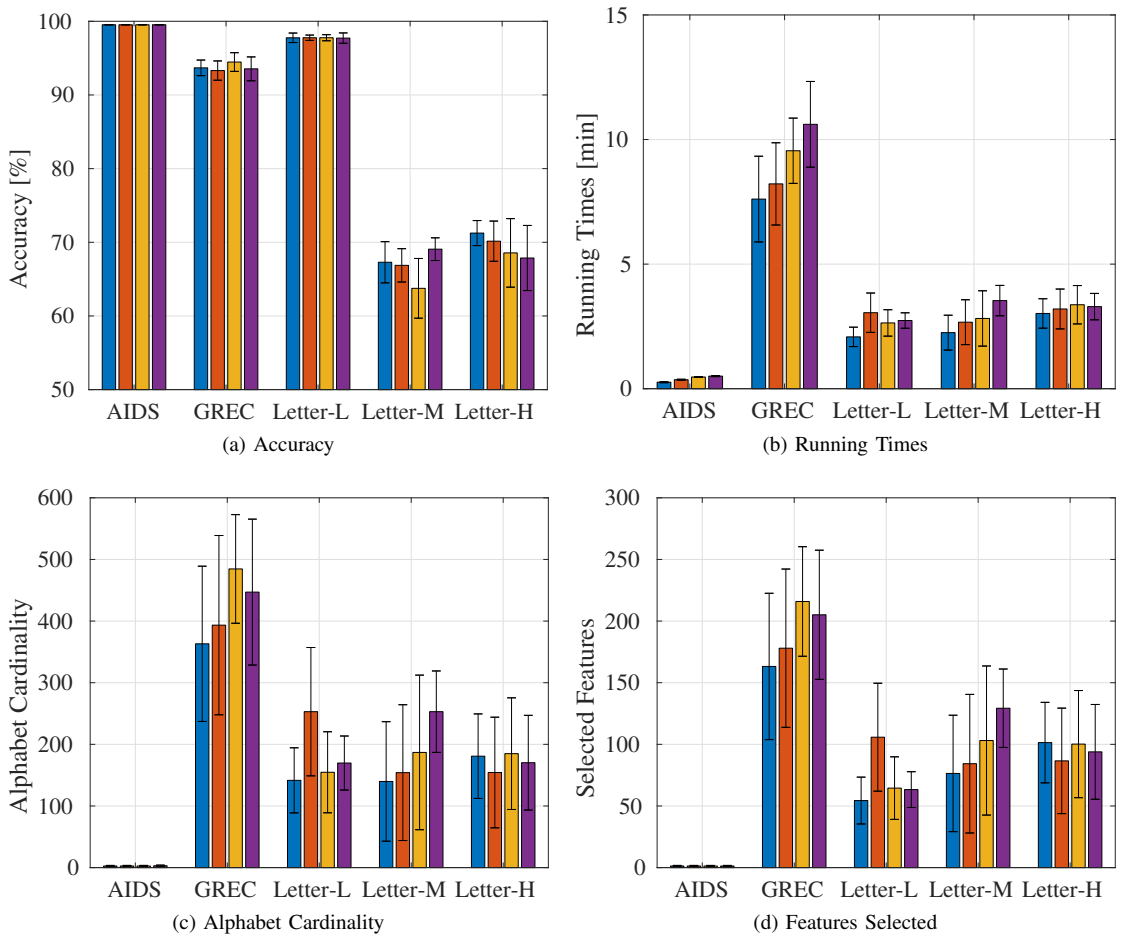


Fig. 1. Results for cliques extractor. Blue, red and yellow bars correspond to sampling rates $W = 40\%, 60\%, 80\%$, respectively. Purple bars refer to maximal cliques enumeration. Whiskers indicate the standard deviation.

Further properties of the datasets and formal formulation of nodes and vertices dissimilarities can be found in [17].

B. Computational Results Against Walk-Based Extractors

In a first test campaign, we show the GRALG performances by using two different extractors: the proposed procedure described in Section II-A based on maximal cliques is compared against the results obtained when a Breadth First Search algorithm is employed for the subgraphs extraction [16]. In both cases, we follow a Class-Aware granulator strategy [17] together with a stochastic sampling strategy in order to reduce the cardinality of the training set for the granulation stage. As consequence, the W parameter in charge to fix the cardinality of $\bigcup_{L=1}^{N} \mathcal{S}_{g,tr}^{L}$ is chosen as follows:

- for cliques, we let $W = 40\%, 60\%, 80\%$ of the number of maximal cliques enumerated using Bron-Kerbosch
- for paths, we let $W = 10\%, 30\%, 50\%$ of the number of subgraphs extracted exhaustively [29]

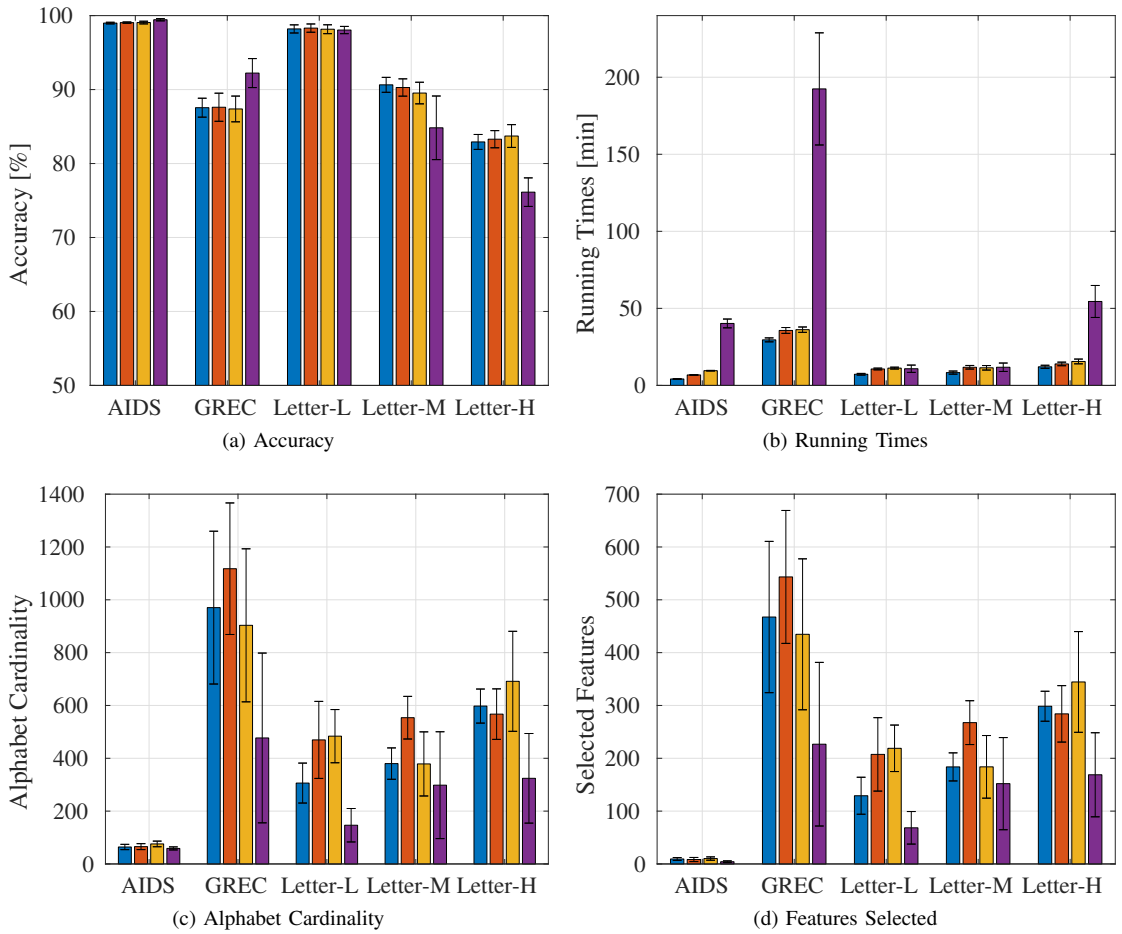
where the total numbers of paths and cliques are shown in Table I. It is worth remarking that when a BFS strategy is employed, an additional parameter o is needed, which defines the maximum order for the subgraphs to be extracted. Conversely, when the extractor is based on maximal cliques, this parameter is unnecessary since the Bron-Kerbosch procedure returns a complete clique decomposition where the order of the cliques is strictly topology-related rather than user-defined. For the sake of completeness, we considered the performances achieved without subsampling strategy with both Bron-Kerbosch and the exhaustive extraction procedures [29].

TABLE I
EXHAUSTIVE NUMBER OF SUBGRAPHS EXTRACTED FROM $\mathcal{S}_{tr}$.

Subgraph Type	**Letter-L**	**Letter-M**	**Letter-H**	**GREC**	**AIDS**
Path ($o = 5$)	8193	8582	21165	27119	35208
Clique	2377	2398	2493	3321	3961

Remaining parameters are chosen as follows:

- $K = 5$ (number of neighbours for K-NN)
- 20 individuals per population (both genetic algorithms)
- 20 generations (first genetic algorithm – alphabet optimization)
- 100 generations (second genetic algorithm – feature selection)



(a) Accuracy (b) Running Times (c) Alphabet Cardinality (d) Features Selected

Fig. 2. Results for BFS extractor. Blue, red and yellow bars correspond to sampling rates $W = 10\%, 30\%, 50\%$, respectively. Purple bars refers to the exhaustive extraction. Whiskers indicate the standard deviation.

- $\alpha = 0.99$ in the fitness function for the second genetic algorithm (very minor weight to sparsity)
- $\epsilon = 1.1$ as tolerance value for the symbolic histograms evaluation.

The software has been developed in C++ using the SPARE[1] and Boost Graph[2] libraries. Tests have been performed on a Linux Ubuntu 19.10 machine, equipped with a 4-core Intel i7-3770K @3.50GHz and 32GB of RAM.

In Fig. 1 and Fig. 2, we compare the extractors based on maximal cliques and BFS paths and their exhaustive counterparts, by taking into account four different aspects:

- accuracy on the test set (Fig. 1a and Fig. 2a)
- wall-clock time (Fig. 1b and 2b)
- number of symbols in the optimized alphabet $\mathcal{A}^\star$ (Section III-A) (Fig. 1c and Fig. 2c)
- selected features after the second optimization phase (Section III-B) (Fig. 1d and Fig. 2d).

Results are averaged on 10 runs in order to account the randomness of the synthesis procedure. In Fig. 1a, *AIDS* and *Letter-L* show comparable levels of accuracy with respect to the BFS extractor in Fig. 2a., whereas in case of *GREC*, better results can be observed, proving that cliques are significant substructures for these problems. Conversely, when the clique extractor acts on *Letter-M* and *Letter-H*, the accuracy is strongly worsen if compared to the BFS strategy, suggesting that paths are better than cliques when it comes to identify useful symbols for these problems. In fact, medium/high level of distortion (i.e., adding or removing vertices) in handwritten letters might destroy useful cliques to characterize letters (e.g., the triangle in 'A'). A major improvement achieved by the clique extractor can be spotted by observing results in Fig. 1c: in all configurations (i.e., regardless of the subsampling percentage from the set of all maximal cliques), the number of symbols is significantly reduced when compared to Fig. 2c. Indeed, when the subsampling occurs, all datasets show a reduced number of symbols necessary to build the alphabet for the embedding phase. Considering that every symbol in the alphabet must be matched with all subgraphs that compose a graph to be embedded (see Section II-C), a straightforward revenue can be observed as running times are considered: by matching Fig. 1b with Fig. 2b, the clique-based extractor outperforms the BFS strategy for each subsample size W,

[1]https://sourceforge.net/projects/libspare/
[2]http://www.boost.org/

TABLE II
COMPARISON AGAINST CURRENT APPROACHES IN TERMS OF ACCURACY. ASTERISKS INDICATE THAT RESULTS REFERS TO CROSS-VALIDATION RATHER THAN A SEPARATE TEST SET.

Technique	AIDS	GREC	Letter L	Letter M	Letter H	Reference
Bipartite Graph Matching + K-NN	-	86.3	91.1	77.6	61.6	[43]
Lipschitz Embedding + SVM	98.3	96.8	99.3	95.9	92.5	[44]
Graph Edit Distance + K-NN	97.3	95.5	99.6	94	90	[42]
Hypergraph Embedding + SVM	99.3	-	-	-	-	[28]
INDVAL Embedding + SVM	98.5	-	-	-	-	[27]
Graph of Words + K-NN	-	97.5	98.8	-	-	[45]
Graph of Words + kPCA + K-NN	-	97.1	97.6	-	-	[45]
Graph of Words + ICA + K-NN	-	58.9	82.8	-	-	[45]
ODD ST_+ kernel*	82.06	-	-	-	-	[14]
ODD ST_+^{TANH} kernel*	82.54	-	-	-	-	[14]
CGMM + linear SVM*	84.16	-	-	-	-	[18]
GRALG (exhaustive path extraction)	99.44	92.23	98.05	84.83	76.13	[16], also in Fig. 2a
GRALG (class-aware clique extraction)	99.53	93.31–94.47	97.72–97.77	63.75–69.07	67.87–71.25	This work, also in Fig. 1a
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	[17], also in Fig. 2a
GRALG (random walk extraction)	99.09–99.16	83.08–84.04	96.36–96.58	85.28–87.89	72.93–73.78	[16], [17]

showing an heavy reduction on the wall clock time even when all the maximal cliques are employed for the granulation phase. Besides the time improvements, another remarkable result achieved thanks to the low-cardinality alphabet is the interpretability of the trained model. Indeed, starting from a reduced set of symbols in the alphabet for the training stage, the following feature selection phase (Fig. 1d) further shrinks the alphabet cardinality, leading to a more explainable learning system. Notable is the case of *AIDS* where, after the feature selection phase, by solely using a single symbol (clique), about 99.5% of the test set is correctly recognized. This peculiar aspect certainly deserves further investigation. Nonetheless, the variance of the alphabet sizes is quite high both as cliques and paths are concerned: we expect to be able to reduce the variance by a proper tweaking of the fitness function of the first genetic algorithm (e.g., by adding a penalty term towards large alphabets).

C. Computational Results Against State of the Art Techniques

In a second test campaign, the comparison involves GRALG and current approaches in graph classification, with Table II 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 (see Section I), including classifiers working on the top of GEDs [42], [43], kernel methods [14] and several embedding techniques [44], [45], including GrC-based [27], [28] and neural [18] ones. As regards subsampling-based GRALG variants, in Table II are reported the performances obtained at different subsampling rates in the form of min-max range. Clearly, GRALG is able to reach state-of-the-art performances on three over five datasets (*AIDS*, *GREC*, *Letter-L*), outperforming ODD and CGMM. On the other hand, performance decays can be observed on *Letter-M* and *Letter-H*, with GRALG scoring -5% in the former case and -9% in the latter case against Lipschitz Embedding (the overall most performing technique). Nonetheless, GRALG is one of the very few amongst the considered techniques (alongside [27], [28]) able to return an interpretable model.

V. CONCLUSIONS

In this paper, we proposed a classification system based on the GrC paradigm for labelled graphs by considering a specific subgraphs type, namely cliques, for building the alphabet upon which the entire model relies on. Symbols emerged after the granulation phase serve as pivotal structures for building the symbolic histograms representation, whereby the classification of the pattern can be achieved in a suitable (geometric) embedding space. The latter, is finally optimized thanks to two different meta-heuristic algorithms (genetic optimization) which are in charge to tune relevant parameters and selecting significant symbols for the final model. With this approach, we aim to understand whether a peculiar subgraph (i.e., clique), can play a relevant role in the synthesis of meaningful information granules and whether these granules can still be efficient in terms of recognition capabilities. Our investigations took place on five datasets, relying on different real problems, by considering accuracy, running times and number of symbols required for building the model as performance indices. In a first test campaign, we compared these performances against the same system equipped with an extractor based on paths emerged with a BFS strategy. Our tests show that, for three out of five considered datasets, cliques emerged as fundamental structure for the problem addressed. Nonetheless, since cliques are very exclusionary kind of structures, the complexity of the underlying model is strongly limited, affecting both running times and the interpretability thereof. A second test campaign compares GRALG against current approaches in the literature, with GRALG showing remarkable performances on three out of five datasets. Overall, these results propose the methodology behind GRALG as an effective framework for knowledge discovery in big data contexts, where the explanability of the classification model is of utmost relevance. Future research can investigate the possibility of mining social networks with the proposed clique-based embedding.

REFERENCES

[1] A. Giuliani, S. Filippi, and M. Bertolaso, "Why network approach can promote a new way of thinking in biology," *Frontiers in Genetics*, vol. 5, p. 83, 2014.

[2] L. Di Paola and A. Giuliani, "Protein–protein interactions: The structural foundation of life complexity," in *eLS*. American Cancer Society, 2017, pp. 1–12.

[3] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A.-L. Barabási, "The large-scale organization of metabolic networks," *Nature*, vol. 407, no. 6804, p. 651, 2000.

[4] S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications*. New York, USA: Cambridge University Press, 1994.

[5] J. Scott, "Social network analysis," *Sociology*, vol. 22, no. 1, pp. 109–127, 1988.

[6] J. Richiardi, S. Achard, H. Bunke, and D. Van De Ville, "Machine learning with brain graphs: predictive modeling approaches for functional imaging in systems neuroscience," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 58–70, 2013.

[7] X. Bai, *Graph-Based Methods in Computer Vision: Developments and Applications: Developments and Applications*. IGI Global, 2012.

[8] 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, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 7–19.

[9] J. Shawe-Taylor and N. Cristianini, *Kernel methods for pattern analysis*. Cambridge University Press, 2004.

[10] B. Schölkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[11] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels," *Journal of Machine Learning Research*, vol. 12, no. Sep, pp. 2539–2561, 2011.

[12] T. Horváth, T. Gärtner, and S. Wrobel, "Cyclic pattern kernels for predictive graph mining," in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 158–167.

[13] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt, "Graph kernels," *Journal of Machine Learning Research*, vol. 11, no. Apr, pp. 1201–1242, 2010.

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

[15] A. Cinti, F. M. Bianchi, A. Martino, and A. Rizzi, "A novel algorithm for online inexact string matching and its fpga implementation," *Cognitive Computation*, 2019.

[16] L. Baldini, A. Martino, and A. Rizzi, "Stochastic information granules extraction for graph embedding and classification," in *Proceedings of the 11th International Joint Conference on Computational Intelligence - Volume 1: NCTA, (IJCCI 2019)*, INSTICC. SciTePress, 2019, pp. 391–402.

[17] ——, "Towards a class-aware information granulation for graph embedding and classification," in *Computational Intelligence: 11th International Joint Conference, IJCCI 2019 Vienna, Austria, September 17-19, 2019 Revised Selected Papers*, To appear in.

[18] D. Bacciu, F. Errica, and A. Micheli, "Contextual graph markov model: A deep and generative approach to graph processing," in *35th International Conference on Machine Learning, ICML 2018*, vol. 1, 2018, pp. 495–504.

[19] C. Gallicchio and A. Micheli, "Graph echo state networks," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*, 2010, pp. 1–8.

[20] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *International Conference on Learning Representations*, 2019.

[21] N. Navarin, D. V. Tran, and A. Sperduti, "Universal readout for graph convolutional neural networks," in *2019 International Joint Conference on Neural Networks (IJCNN)*, 2019, pp. 1–7.

[22] A. Micheli, "Neural network for graphs: A contextual constructive approach," *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 498–511, 2009.

[23] A. Bargiela and W. Pedrycz, "The roots of granular computing," in *2006 IEEE International Conference on Granular Computing*, 2006, pp. 806–809.

[24] W. Pedrycz, "Granular computing: an introduction," in *Proceedings Joint 9th IFSA World Congress and 20th NAFIPS International Conference*, vol. 3. IEEE, 2001, pp. 1349–1354.

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

[26] 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*, W. Pedrycz and S.-M. Chen, Eds. Cham: Springer International Publishing, 2018, pp. 53–81.

[27] A. Martino, A. Giuliani, V. Todde, M. Bizzarri, and A. Rizzi, "Metabolic networks classification and knowledge discovery by information granulation," *Computational Biology and Chemistry*, vol. 84, p. 107187, 2020.

[28] A. Martino, A. Giuliani, and A. Rizzi, "(hyper)graph embedding and classification via simplicial complexes," *Algorithms*, vol. 12, no. 11, 2019.

[29] F. M. Bianchi, L. Livi, A. Rizzi, and A. Sadeghian, "A granular computing approach to the design of optimized graph classification systems," *Soft Computing*, vol. 18, no. 2, pp. 393–412, 2014.

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

[31] L. Backstrom and J. Leskovec, "Supervised random walks: predicting and recommending links in social networks," in *Proceedings of the fourth ACM international conference on Web search and data mining*. ACM, 2011, pp. 635–644.

[32] P. Gupta, A. Goel, J. Lin, A. Sharma, D. Wang, and R. Zadeh, "Wtf: The who to follow service at twitter," in *Proceedings of the 22nd International Conference on World Wide Web*, ser. WWW '13. New York, NY, USA: ACM, 2013, p. 505–514.

[33] N. J. Salkind, *Encyclopedia of educational psychology*. SAGE publications, 2008.

[34] N. Tichy, "An analysis of clique formation and structure in organizations," *Administrative Science Quarterly*, vol. 18, no. 2, pp. 194–208, 1973.

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

[36] J. W. Moon and L. Moser, "On cliques in graphs," *Israel Journal of Mathematics*, vol. 3, no. 1, pp. 23–28, 1965.

[37] C. Bron and J. Kerbosch, "Algorithm 457: Finding all cliques of an undirected graph," *Commun. ACM*, vol. 16, no. 9, pp. 575–577, 1973.

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

[39] F. M. Bianchi, S. Scardapane, L. Livi, A. Uncini, and A. Rizzi, "An interpretable graph-based image classifier," in *2014 International Joint Conference on Neural Networks (IJCNN)*, 2014, pp. 2339–2346.

[40] H. Bunke, S. Günter, and X. Jiang, "Towards bridging the gap between statistical and structural pattern recognition: Two new concepts in graph matching," in *International Conference on Advances in Pattern Recognition*. Springer, 2001, pp. 1–11.

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

[42] K. Riesen and H. Bunke, "Iam graph database repository for graph based pattern recognition and machine learning," in *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*. Springer, 2008, pp. 287–297.

[43] ——, "Approximate graph edit distance computation by means of bipartite graph matching," *Image and Vision Computing*, vol. 27, no. 7, pp. 950 – 959, 2009, 7th IAPR-TC15 Workshop on Graph-based Representations (GbR 2007).

[44] K. Riesen and H. Bunke, "Graph classification by means of lipschitz embedding," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 6, pp. 1472–1483, 2009.

[45] 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, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 22–31.