SN Computer Science (2023) 4:314
https://doi.org/10.1007/s42979-023-01716-1

ORIGINAL RESEARCH

On Information Granulation via Data Filtering for Granular Computing-Based Pattern Recognition: A Graph Embedding Case Study

Alessio Martino[1] · Enrico De Santis[2] · Antonello Rizzi[2]

Received: 30 June 2022 / Accepted: 30 January 2023

Abstract
Granular Computing is a powerful information processing paradigm, particularly useful for the synthesis of pattern recognition systems in structured domains (e.g., graphs or sequences). According to this paradigm, granules of information play the pivotal role of describing the underlying (possibly complex) process, starting from the available data. Under a pattern recognition viewpoint, granules of information can be exploited for the synthesis of semantically sound embedding spaces, where common supervised or unsupervised problems can be solved via standard machine learning algorithms. In this companion paper, we follow our previous paper (Martino et al. in Algorithms 15(5):148, 2022) in the context of comparing different strategies for the automatic synthesis of information granules in the context of graph classification. These strategies mainly differ on the specific topology adopted for subgraphs considered as candidate information granules and the possibility of using or neglecting the ground-truth class labels in the granulation process and, conversely, to our previous work, we employ a filtering-based approach for the synthesis of information granules instead of a clustering-based one. Computational results on 6 open-access data sets corroborate the robustness of our filtering-based approach with respect to data stratification, if compared to a clustering-based granulation stage.

Keywords Structural pattern recognition · Supervised learning · Graph classification · Inexact graph matching · Granular computing · Information granulation · Data mining and knowledge discovery

Introduction

In the early 2000s, Granular Computing emerged as a novel information processing paradigm that exploits pivotal mathematical structures called *granules of information* to describe an underlying set of (likely complex) data, describing a (likely complex) process under analysis [2, 3]. The concept of information granulation dates back to the mid-1990s, thanks to soft computing and fuzzy logic pioneer Lotfi Aliasker Zadeh. In their words:

> *Among the basic concepts which underlie human cognition there are three that stand out in importance. The three are: granulation, organization and causation.*
> L.A. Zadeh [4]

and

> *Informally, granulation of an object A results in a collection of granules of A, with a granule being a clump of objects (or points) which are drawn together by*

This article is part of the topical collection “Computational Intelligence 2021” guest edited by Juan Julian Merelo, Kevin Warwick, Thomas Bäck, Christian Wagner, Jonathan Garibaldi, H. K. Lam and Marie Cottrell.

✉ Alessio Martino
amartino@luiss.it

Enrico De Santis
enrico.desantis@uniroma1.it

Antonello Rizzi
antonello.rizzi@uniroma1.it

[1] Department of Business and Management, LUISS University, Viale Romania 32, 00197 Rome, Italy

[2] Department of Information Engineering, Electronics and Telecommunications, University of Rome “La Sapienza”, Via Eudossiana 18, 00184 Rome, Italy

Published online: 08 April 2023

indistinguishability, similarity, proximity or functionality. In this sense, the granules of a human body are the head, neck, arms, chest, etc. In turn, the granules of a head are the forehead, cheeks, nose, ears, eyes, hair, etc. In general, granulation is hierarchical in nature. A familiar example is granulation of time into years, years in months, months into days and so on.
L.A. Zadeh [4]

This philosophical viewpoint behind the birth of Granular Computing as human-inspired information processing paradigm has been embraced by other well-known scholars, notably Ronald R. Yager. In their words:

Language, which is central to most human cognitive activities, is based on granularization. In addition human facilities for distinctions are limited. Being limited, at the very least by language and perhaps additionally by their ability to perceive, human beings have been developed a granular view of the world. Thus, we see that the objects with which humankind perceives, measures, conceptualizes and reasons are granular.
R.R. Yager and D. Filev [5]

As the 1990s marked the golden age of fuzzy logic and fuzzy-based pattern recognition, L.A. Zadeh further argues that information granules should be inherently fuzzy, since most of human reason and concept formation are fuzzy rather than crisp [4, 6].

Clearly, Granular Computing is not a set of computational pipelines and is not a set of algorithms; rather, it can be considered as a goal-driven umbrella that, according to Y.Y. Yao, in their ”Granular Computing manifesto” [7] should fulfill the following points:

- Be a truthful representation of the real world;
- Be consistent with human thinking and problem solving;
- Allow a simplification of the problem;
- Provide economic and low-cost solutions.

Thus, in the realm of Granular Computing, we can easily find techniques that have not been developed for Granular Computing but nonetheless they satisfy the above goals. Indeed, information granulation can be performed by a plethora of different strategies, notably fuzzy sets [4, 8, 9], rough sets [9–11], shadowed sets [12], interval analysis [13] and data clustering [14–16].

Recently, the granular computing paradigm has been employed for the synthesis of pattern recognition systems, as well as in structured domains, such as graphs, sequences and images [17, 18]. The rationale behind these pattern recognition systems is to automatically extract recurrent and/or meaningful substructures (i.e., subgraphs, subsequences, portions of images) suitable to be considered as granules of information. On the top of these pivotal elements, it is possible to build an embedding space in such a way that the pattern recognition problem is cast from the structured domain towards the Euclidean space. The latter, being a metric space, allows to comfortably use one of the many statistical classifiers currently available in the pattern recognition and machine learning literature [19].

This work is the second part of our analysis in the context of granulation techniques. Our investigation began in [1], where we compared 4 different clustering-based granulation strategies originally proposed in [20–22]. The comparison involved two important aspects:

1. The topology of the candidate information granules (paths extracted via random walks vs. cliques extracted via the maximal clique decomposition);
2. The possibility of exploiting the ground-truth class labels in the granulation procedure (which, being based on data clustering, is unsupervised by definition).

In this work, as instead, we exploit a filtering-based approach originally proposed in [23, 24] and later extended in [25] for the same goal, that is, the automatic synthesis of information granules in the context of graph classification. The information granules reflect pivotal subgraphs extracted from the training data endowed with high discriminative power. On the top of these information granules, we perform an embedding procedure thanks to the symbolic histograms approach [26]. Following our original work, we review and compare two different candidate topologies for the synthesis of granules of information (paths and cliques) and we compare two additional strategies for their synthesis: a stratified approach, where the ground-truth labels of the classification problem play an important role in the information granules synthesis, and a non-stratified approach, where the ground-truth labels are completely discarded in the process.

The remainder of this paper is structured as follows: in the section “High-Level Framework Description”, we describe the four main building blocks of the Granular Computing framework (extraction and synthesis of granules of information, graph embedding and classification), which we exploit to perform our twofold investigation. In the section “Information Granulation Strategies”, we discuss in detail four granulation strategies based on different combination of subgraph topologies (paths vs. cliques) and stratification (class-aware vs. no stratification). In the section “Model Synthesis and Testing”, we detail how these building blocks co-operate to synthesize an optimized model for graph classification. In the section “Tests and Results”, we show the computational results obtained by the four combinations of topology and granulation and, finally, the section “Conclusion” concludes the paper.

High-Level Framework Description

The proposed pattern recognition system is composed of the following four main modules:

- Extractor, which is in charge of extracting, from the training set, a suitable set of candidate information granules;
- Granulator, which is in charge of building an alphabet of symbols starting from the candidate information granules provided by the Extractor block;
- Embedder, which is in charge of mapping a graph data set towards the Euclidean space;
- Classifier, which is in charge of training and testing a suitable classification system in the Euclidean space spanned by the Embedder block.

Extractor and Granulator

Let P be an unknown, oriented and possibly complex process to be modeled, where the inputs are annotated graphs and where the output domain is a finite set of class labels. Furthermore, let $\mathcal{S}$ be an input–output sampling of P and let $\mathcal{S}_{\text{tr}}$, $\mathcal{S}_{\text{val}}$ and $\mathcal{S}_{\text{ts}}$ be a split of $\mathcal{S}$ into training, validation and test sets, respectively. The split should be performed so that each subset should share the same statistics to be considered as valid representation of the same process. Moreover, this split must satisfy the partition properties, notably:

- The union of the three sets yields the original set: $\mathcal{S}_{\text{tr}} \cup \mathcal{S}_{\text{val}} \cup \mathcal{S}_{\text{ts}} = \mathcal{S}$;
- The intersection of any two distinct sets is empty: $\mathcal{S}_{\text{tr}} \cap \mathcal{S}_{\text{val}} = \mathcal{S}_{\text{tr}} \cap \mathcal{S}_{\text{ts}} = \mathcal{S}_{\text{ts}} \cap \mathcal{S}_{\text{val}} = \emptyset$.

The Extractor is a block that takes as input $\mathcal{S}_{\text{tr}}$ and returns a bucket $\mathcal{B}$ of subgraphs drawn from graphs in $\mathcal{S}_{\text{tr}}$.

Conversely, the Granulator block takes as input $\mathcal{B}$ and returns an alphabet of symbols $\mathcal{A} \subset \mathcal{B}$, namely, suitable granules of information, by calculating a statistical score called INDVAL and retaining only statistically relevant items from $\mathcal{B}$.

The INDVAL score has been originally proposed in [27] for spotting representative species in different environments. The idea at the basis of the INDVAL score is straightforward: a given species s is representative, hence useful for the recognition of an environmental condition c if both of the following properties are met:

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

The INDVAL score (I) can be re-stated to spot signature subgraphs in a set of training graphs as [23, 24, 28]:

$$A_i^{(j)} = \frac{\#\text{graphs having subgraph } i \text{ in group } j}{\#\text{graphs having subgraph } i} \tag{1}$$

$$B_i^{(j)} = \frac{\#\text{graphs having subgraph } i \text{ in group } j}{\#\text{graphs in group } j} \tag{2}$$

$$I_i^{(j)} = A_i^{(j)} \cdot B_i^{(j)} \cdot 100 \tag{3}$$

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 ith subgraph can be found only in patterns (graphs) belonging to class j;
- 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 ith 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, let us consider the following example useful for understanding how the INDVAL score can be used to spot meaningful subgraphs.

Example 1 Fig. 1 shows a toy data set with 4 graphs equally distributed in two classes.

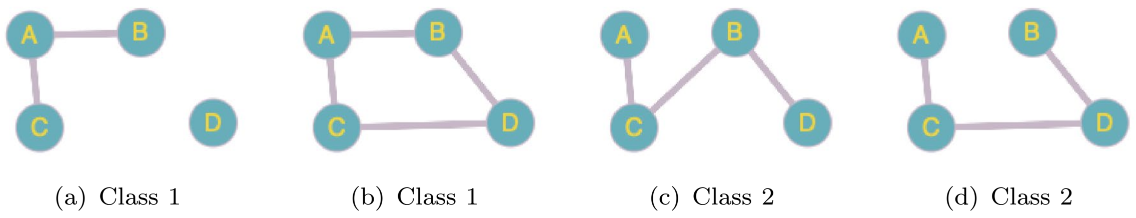


Fig. 1 Toy data set with 2 graphs per class

For the sake of example, let us consider only order-2 subgraphs, notably A–B, B–D, B–C, C–D, A–C. Then, we can calculate the scores A, B and I by means of Eqs. (1)–(3), yielding:

$$
\begin{array}{lll}
A^{(1)}_{\text{A–B}} = 2/2 & B^{(1)}_{\text{A–B}} = 2/2 & I^{(1)}_{\text{A–B}} = 1 \cdot 1 \cdot 100 = 100 \\
A^{(2)}_{\text{A–B}} = 0/2 & B^{(2)}_{\text{A–B}} = 0/2 & I^{(2)}_{\text{A–B}} = 0 \cdot 0 \cdot 100 = 0 \\
A^{(1)}_{\text{B–D}} = 1/3 & B^{(1)}_{\text{B–D}} = 1/2 & I^{(1)}_{\text{B–D}} = 0.3 \cdot 0.5 \cdot 100 = 15 \\
A^{(2)}_{\text{B–D}} = 2/3 & B^{(2)}_{\text{B–D}} = 2/2 & I^{(2)}_{\text{B–D}} = 0.6 \cdot 1 \cdot 100 = 60 \\
A^{(1)}_{\text{B–C}} = 0/1 & B^{(1)}_{\text{B–C}} = 0/2 & I^{(1)}_{\text{B–C}} = 0 \cdot 0 \cdot 100 = 0 \\
A^{(2)}_{\text{B–C}} = 1/1 & B^{(2)}_{\text{B–C}} = 1/2 & I^{(2)}_{\text{B–C}} = 1 \cdot 0.5 \cdot 100 = 50 \\
A^{(1)}_{\text{C–D}} = 1/2 & B^{(1)}_{\text{C–D}} = 1/2 & I^{(1)}_{\text{C–D}} = 0.5 \cdot 0.5 \cdot 100 = 50 \\
A^{(2)}_{\text{C–D}} = 1/2 & B^{(2)}_{\text{C–D}} = 1/2 & I^{(2)}_{\text{C–D}} = 0.5 \cdot 0.5 \cdot 100 = 50 \\
A^{(1)}_{\text{A–C}} = 2/4 & B^{(1)}_{\text{A–C}} = 2/2 & I^{(1)}_{\text{A–C}} = 0.5 \cdot 1 \cdot 100 = 50 \\
A^{(2)}_{\text{A–C}} = 2/4 & B^{(2)}_{\text{A–C}} = 2/2 & I^{(2)}_{\text{A–C}} = 0.5 \cdot 1 \cdot 100 = 50
\end{array}
$$

By looking at the INDVAL scores, we can immediately spot edge A–B as "the perfect subgraph", since it has maximum score for class 1 and null score for class 2: indeed (cf. Figure 1) edge A–B is found only and in all graphs belonging to class 1 while being (at the same time) completely absent in all graphs belonging to class 2. In this example, edge A–B alone will allow us to perfectly discriminate graphs belonging to class 1 and class 2 (i.e., by checking whether is present or not in the graph to be classified). Other notable results include edges C–D and A–C: C–D can be found in half of the graphs, regardless of the class, and A–C can be found in all graphs in the data set. Their INDVAL score ($I = 50$, regardless of the class) reflects the fact that such edges do not significantly characterize graphs belonging to either class 1 or class 2.

The above example shows a simplistic case, where graphs have unlabelled edges and nodes are labelled with plain categorical values. In this case, the evaluation of Eqs. (1)–(3) is straightforward, since the procedure of counting how many times a given subgraph appears in a graph can be done in an *exact* manner [23]. However, graphs are by definition very general data structures whose nodes and edges can be equipped with labels of any nature. This problem has been addressed in [25] and here below we summarize our solution to account for an *inexact* graph matching:

1. Let $\mathcal{G}$ be a graph and let $\tilde{\mathcal{G}}$ be a subgraph to be found in $\mathcal{G}$: we can start by decomposing $\mathcal{G}$ into its constituent parts;
2. Match $\tilde{\mathcal{G}}$ against any of the constituent parts from $\mathcal{G}$ thanks to a suitable dissimilarity measure;
3. A match is considered as a hit if the dissimilarity measure is below a user-defined threshold τ.

As regards step #2, we employ the node Best Match First (nBMF) dissimilarity measure, belonging to the wider family of graph edit distances [29, 30]. Mathematical details on nBMF can be found in [25, Appendix A]. We anticipate that nBMF is a parametric dissimilarity measure, where the importance of insertion, deletion and substitution on nodes and edges can be tuned by the end-user via suitable weights. Similarly, the dissimilarity measures to match (dis)similar nodes and/or edges can be parametric themselves, depending on the data and the problem at hand.

Therefore, the Granulator block receives the bucket of candidate information granules from the Extractor block (the four strategies for populating $\mathcal{B}$ will be separately discussed in Sect. "Information Granulation Strategies") and, for each candidate information granule, its INDVAL score is evaluated against each of the problem-related classes. If, for at least one class, the INDVAL score is greater than a threshold $T \in (0, 100)$, that particular subgraph is included in the alphabet $\mathcal{A}$, otherwise it will be filtered out (i.e., discarded).

Embedder

The Embedder block takes as input the alphabet as returned by the Granulator block and runs an embedding function to cast each graph (belonging to an input graph set, e.g., $\mathcal{S}_{\text{tr}}$) towards the Euclidean space.

The mapping function yields the so-called symbolic histogram [26] by transforming each input graph $\mathcal{G}$ into an n-length feature vector of the form:

$$\mathbf{h}(\mathcal{A}, \mathcal{G}) = [\text{occ}(s_1, \mathcal{G}), \dots, \text{occ}(s_n, \mathcal{G})] \tag{4}$$

where $\mathcal{A} = \{s_1, \dots, s_n\}$ and $\text{occ} : \mathcal{A} \times \mathcal{G} \to \mathbb{N}_0^+$ is the enumeration function that counts the number of times each symbol $s \in \mathcal{A}$ appears in $\mathcal{G}$.

Here, the process of counting subgraphs into graphs is subject to the same observations as for the INDVAL case and, likewise, operates as follows:

1. The input graph $\mathcal{G}$ is decomposed into its constituent parts, yielding a decomposition $\mathcal{G}' = \{g_1, \dots, g_k\}$.
2. For the ith symbol in $\mathcal{A}$:

(a) The pairwise dissimilarities between s_i and all subgraphs in $\mathcal{G}'$ are evaluated;
(b) All dissimilarities below a threshold τ are retained and considered as a 'hit';
(c) The ith entry in $\mathbf{h}$ is filled with the number of occurrences (i.e., the number of 'hits');

1. Repeat step 2 for $i = 1, \dots, n$.

One important aspect in this procedure regards the pairwise dissimilarities between symbols and subgraphs, which is evaluated by means of the very same nBMF dissimilarity measure already used by the Granulator block.

Classifier

The Classifier block trains a classification system on the embedded version of $\mathcal{S}_{\mathrm{tr}}$, say $\mathbf{H}_{\mathrm{tr}}$, namely, an $|\mathcal{S}_{\mathrm{tr}}| \times n$ instance matrix with patterns (i.e., graphs from $\mathcal{S}_{\mathrm{tr}}$) organized as rows.

To validate the behavior of the classifier, the Classifier block also needs the embedded version of $\mathcal{S}_{\mathrm{val}}$, say $\mathbf{H}_{\mathrm{val}}$. The ability of the classification system, previously trained on $\mathbf{H}_{\mathrm{tr}}$, in predicting the ground-truth labels of $\mathbf{H}_{\mathrm{val}}$ dictates the performance of the Classifier.

In this work, we use a K-Nearest Neighbors (K-NN) decision rule [31] with $K = 5$ as the classification system.

Information Granulation Strategies

Random Walk

The random walk extractor, proposed in [25], takes as input a bucket $\mathcal{B}$ of candidate information granules extracted via a plain random walk on the graphs belonging to the training set.

In a plain random walk [32, 33], the next-hop $u \in \mathcal{V}$ is chosen uniformly at random among the neighbors of the current node $v \in \mathcal{V}$. Formally, the probability of moving from v to u is

$$p_{v \to u} = \begin{cases} \frac{1}{\deg(v)}, & \text{if} u \in \mathcal{N}(v) \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

where $\mathcal{N}(v)$ is the neighborhood of node v and $\deg(v)$ is its degree, i.e., $\deg(v) = |\mathcal{N}(v)|$.

Populating the bucket $\mathcal{B}$ relies on two important parameters:

- W, the user-defined number of subgraphs in $\mathcal{B}$, i.e., $W = |\mathcal{B}|$;
- o, the user-defined maximum number of nodes for subgraphs in $\mathcal{B}$.

and it works as follows:

1. Start with $\mathcal{B} = \emptyset$;
2. Let $W' = \frac{W}{o}$ be the number of subgraphs to be extracted for each of the candidate subgraph orders;
3. For $l = 1, \dots, o$:

   (a) Let $\mathcal{B}^{(l)} = \emptyset$ be a temporary bucket containing only subgraphs of order l;
   (b) Until $\left|\mathcal{B}^{(l)}\right|$ equals W':

      (i) Extract uniformly at random a graph $\mathcal{G}$ from the training set;
      (ii) Extract uniformly at random a node v from $\mathcal{G}$;
      (iii) Start a simple random walk of length l from node v;
      (iv) The subgraph emerged from the random walk is added to $\mathcal{B}^{(l)}$;

   (c) $\mathcal{B} = \mathcal{B} \cup \mathcal{B}^{(l)}$;

The so-collected bucket $\mathcal{B}$ is the main input to the Granulator module. As anticipated in Sect. "High-Level Framework Description", the Granulator block calculates the INDVAL scores for each subgraph in $\mathcal{B}$, eventually collecting statistically relevant granules in the alphabet $\mathcal{A}$.

Clique

The clique extractor aims at investigating a particular subgraph topology: the clique, namely, an induced subgraph that is complete [34, 35].

In this scenario, the bucket $\mathcal{B}$ will be filled with a subset of the cliques extracted from the graphs belonging to the training set. The end-user is still required to specify W, as for the section "Random Walk", yet the maximal order parameter o is meaningless as cliques are concerned, since the formation (and the size) of a clique is strictly topology-dependent rather than user-defined.

The clique extractor works as follows:

1. Start with $\mathcal{B} = \emptyset$;
2. For each graph $\mathcal{G}$ from the training set:

   (a) Evaluate $\mathcal{C}$ as the maximal clique decomposition of $\mathcal{G}$. The maximal clique decomposition of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ can be evaluated thanks to the Bron–Kerbosh algorithm [36] with a worst-case complexity of $\mathcal{O}(3^{|\mathcal{V}|/3})$ [37];
   (b) $\mathcal{B} = \mathcal{B} \cup \mathcal{C}$;

3. Let $\mathcal{B}\simeq$ be a set of W subgraphs selected uniformly at random from $\mathcal{B}$;
4. $\mathcal{B} \leftarrow \mathcal{B}'$.

The so-collected bucket $\mathcal{B}$ is the main input to the granulator module. The Granulator block works exactly as the one described in the section "Random Walk", yet on a bucket $\mathcal{B}$ composed of cliques only.

Stratified Clique

The two extractors and granulator strategies in the sections. "Random Walk" and "Clique" populate the bucket $\mathcal{B}$ uniformly at random. Such procedures present the following two potential drawbacks:

- The information about the ground-truth labels (freely available in classification problems) is not exploited in the extraction and granulation stages;
- A uniformly at random selection can bias the contents of $\mathcal{B}$, especially in case of unbalanced data sets; indeed, training graphs pertaining to the majority class have a higher change of being selected.

To overcome both problems, we further proposed a stratified clique-based extractor and granulator [20]. The main objective of the stratified extractor is to build the bucket $\mathcal{B}$ as a set-of-sets $\mathcal{B} = \{\mathcal{B}^{(1)}, \dots, \mathcal{B}^{(p)}\}$, with p being the number of classes for the classification problem at hand, with the constraint that $\mathcal{B}^{(i)}$ contains subgraphs drawn from the subset of training graphs belonging to the ith class only.

The stratified clique-based extractor works as follows:

1. For each ground-truth class $i = 1, \dots, p$:

(a) Let $\mathcal{S}_{\text{tr}}^{(i)}$ be the subset of the training set containing only patterns belonging to class i;
(b) Calculate the (relative) frequency of the ith class as $f_i = \lfloor * \rfloor \frac{|\mathcal{S}_{\text{tr}}^{(i)}|}{|\mathcal{S}_{\text{tr}}|} + \frac{1}{2}$, where the operator $\lfloor * \rfloor x + \frac{1}{2}$ rounds x to the nearest integer;
(c) Evaluate $W_i = \lfloor * \rfloor W \cdot f_i + \frac{1}{2}$, namely, the size of $\mathcal{B}^{(i)}$;
(d) Set $\mathcal{B}^{(i)} = \emptyset$;
(e) For each graph $\mathcal{G} \in \mathcal{S}_{\text{tr}}^{(i)}$:

  (i) Evaluate $\mathcal{C}$ as the maximal clique decomposition of $\mathcal{G}$;
  (ii) Update $\mathcal{B}^{(i)} = \mathcal{B}^{(i)} \cup \mathcal{C}$.

(f) If $\left|\mathcal{B}^{(i)}\right| > W_i$, then replace $\mathcal{B}^{(i)}$ with a uniform random selection of W_i of its own subgraphs.

Due to the set-of-sets nature of the stratified bucket, the Granulator described in the section "Random Walk" and later employed in the section Clique" loses its effectiveness. To overcome this problem, we flat the set-of-sets and run the Granulator Section from "Random Walk".

Stratified Random Walk

The stratified path-based extractor, originally proposed in [21], works as follows:

1. For each ground-truth class $i = 1, \dots, p$:

(a) Let $\mathcal{S}_{\text{tr}}^{(i)}$ be the subset of the training set containing only patterns belonging to class i;
(b) Calculate the (relative) frequency of the ith class as $f_i = \lfloor * \rfloor \frac{|\mathcal{S}_{\text{tr}}^{(i)}|}{|\mathcal{S}_{\text{tr}}|} + \frac{1}{2}$;
(c) Evaluate $W_i = \lfloor * \rfloor W \cdot f_i + \frac{1}{2}$, namely, the size of $\mathcal{B}^{(i)}$;
(d) Evaluate $W'_i = \lfloor * \rfloor W_i \cdot o + \frac{1}{2}$, namely, the number of subgraphs to be extracted for each of the candidate subgraphs order, yet considering only graphs belonging to class i;
(e) Set $\mathcal{B}^{(i)} = \emptyset$;
(f) For $l = 1, \dots, o$:

  (i) Set $\mathcal{B}^{(i,l)} = \emptyset$, namely, a temporary bucket that will hold subgraphs of order l extracted from graphs of class i;
  (ii) Until $\left|\mathcal{B}^{(i,l)}\right|$ is equal to W'_i:

(A) Extract uniformly at random a graph $\mathcal{G}$ from $\mathcal{S}_{\text{tr}}^{(i)}$;
(B) Extract uniformly at random a node v from $\mathcal{G}$;
(C) Start a simple random walk of length l from node v;
(D) The subgraph emerged from the random walk is added to $\mathcal{B}^{(i,l)}$;

  (iii) $\mathcal{B}^{(i)} = \mathcal{B}^{(i)} \cup \mathcal{B}^{(i,l)}$

The Granulator block works exactly as the one described in the section "Stratified Clique", yet on a bucket-of-buckets $\mathcal{B}$ composed of random walks (see the section "Random Walk").

Model Synthesis and Testing

In this section, we explain in detail how the above-described four blocks (Extractor, Granulator, Embedder and Classifier) co-operate to synthesize the classification model and subsequent testing of its final performance on the test set. Since there are several hyper-parameters involved in the model synthesis, notably:

- The thresholds T (used in the Granulator block) and τ (used in both Granulator and Embedder);
- The weights for the nBMF dissimilarity measure (used in both Granulator and Embedder).

We employ a differential evolution algorithm [38] for an automatic tuning of these parameters, hence driving the overall model synthesis.

The model synthesis starts by triggering the Extractor block, which yields the set of candidate information granules $\mathcal{B}$ by properly processing the training graphs. As $\mathcal{B}$ is returned, the differential evolution optimization scheme can take place.

The search space for the optimization procedure is defined as

$$\begin{bmatrix} T & \tau & w_{\text{sub}}^{\text{node}} & w_{\text{ins}}^{\text{node}} & w_{\text{del}}^{\text{node}} & w_{\text{sub}}^{\text{edge}} & w_{\text{ins}}^{\text{edge}} & w_{\text{del}}^{\text{edge}} & \Pi_{\text{edge}} & \Pi_{\text{node}} \end{bmatrix} \tag{6}$$

Each individual from the evolving population forwards $\mathcal{B}$ to the Granulator block. As anticipated in the section "High-Level Framework Description", the Granulator calculates the INDVAL score by exploiting a suitable dissimilarity measure $d(\cdot,\cdot)$ and a threshold $\tau \in [0,1]$ and, finally, filters out any subgraph whose INDVAL score is below T. Recall that the dissimilarity measure between any two patterns is evaluated by a parametric nBMF dissimilarity measure that, in turn, depends on:

- Six real-valued weights that account for the importance of each atomic transformation (insertion, deletion and substitution) on nodes and edges: $w_{\text{sub}}^{\text{node}}$, $w_{\text{ins}}^{\text{node}}$, $w_{\text{del}}^{\text{node}}$, $w_{\text{sub}}^{\text{edge}}$, $w_{\text{ins}}^{\text{edge}}$, $w_{\text{del}}^{\text{edge}}$;
- A set Π_{edge} of parameters, if needed, to drive the dissimilarity measure between edges;
- A set Π_{node} of parameters, if needed, to drive the dissimilarity measure between nodes.

At the end of the Granulation stage, the alphabet $\mathcal{A}$ is available for feeding the Embedder block, which yields the embedded version of $\mathcal{S}_{\text{tr}}$ and $\mathcal{S}_{\text{val}}$, namely, $\mathbf{H}_{\text{tr}}$ and $\mathbf{H}_{\text{val}}$.

These two instance matrices are finally fed to the Classifier block, which trains a K-NN decision rule on $\mathbf{H}_{\text{tr}}$ and predicts the ground-truth labels on $\mathbf{H}_{\text{val}}$.

Each individual is evaluated by means of a fitness function (to be minimized) formalized as follows:

$$f = \alpha \times \pi + (1-\alpha) \times \sigma \tag{7}$$

where π is an error term and σ is a penalty term. Specifically:

$$\pi = 1 - \frac{1}{p}\sum_{i=1}^{p} \text{rec}(i) \tag{8}$$

where rec(i) is the recall of class i. Hence, Eq. (8) reads as the complement of the balanced accuracy [39]. On the other hand, σ is a penalty term defined as

$$\sigma = \frac{|\mathcal{A}|}{|\mathcal{B}|} \tag{9}$$

to penalize large alphabets and foster the optimization towards smaller alphabets.

At the end of the optimization stage, the best individual is retained, along with the best alphabet $\mathcal{A}^\star$ synthesized with its genetic code and the two instance matrices $\mathbf{H}_{\text{tr}}^\star$ and $\mathbf{H}_{\text{val}}^\star$ embedded against $\mathcal{A}^\star$.

The choice of the trade-off parameter $\alpha \in [0,1]$ in Eq. (7) that weights performance against dimensionality can hardly be set a priori. To overcome this problem, a second lightweight optimization stage can be employed to further reduce the size of the alphabet.

In this second optimization stage, the genetic code is simply a binary mask:

$$\mathbf{m} = \{0,1\}^{|\mathcal{A}^\star|} \tag{10}$$

and each individual from the evolving population:

1. Projects $\mathbf{H}_{\text{tr}}^\star$ and $\mathbf{H}_{\text{val}}^\star$ on the subset of columns spanned by the indices $\{i : \mathbf{m}_i = 1\}$, say $\mathbf{H}_{\text{tr}}^{\star\prime}$ and $\mathbf{H}_{\text{val}}^{\star\prime}$;
2. Trains the classifier on $\mathbf{H}_{\text{tr}}^{\star\prime}$ and validates its performance on $\mathbf{H}_{\text{val}}^{\star\prime}$.

The fitness function (to be minimized) reads as

$$f' = \beta \times \pi + (1-\beta) \times \frac{\left|\{i : \mathbf{m}_i = 1\}\right|}{\left|\mathcal{A}^\star\right|} \tag{11}$$

where the leftmost term reads as in Eq. (8) and the rightmost term aims at fostering the evolution towards smaller alphabets by preferring sparse binary masks.

At the end of this second optimization stage, the (possibly) reduced alphabet $\mathcal{A}^{\star\prime} \subseteq \mathcal{A}^\star$ is retained, along with the projected training instance matrix $\mathbf{H}_{\text{tr}}^{\star\prime}$. The test set is itself embedded against $\mathcal{A}^{\star\prime}$, yielding $\mathbf{H}_{\text{ts}}^{\star\prime}$. The classifier is finally trained on $\mathbf{H}_{\text{tr}}^{\star\prime}$ and its final performance is evaluated on $\mathbf{H}_{\text{ts}}^{\star\prime}$.

Tests and Results

Data sets Description

The proposed comparison among different data granulation strategies involves 6 different data sets, with the first 5 data sets being taken from the IAM Repository [40] and the remaining one being taken from the TUDataset Repository [41]. A brief description of the data sets, along with the formal definition of the dissimilarity measures between nodes and edges, follows:

AIDS: The AIDS data set consists of 2000 graphs representing molecules showing activity or not against HIV (two classes). Molecules are converted into graphs in a straightforward manner by representing atoms as nodes and the covalent bonds as edges. Nodes are labeled by a three-element tuple that collects the 2D $\langle x, y\rangle$ coordinates of the atom, the chemical symbol (categorical) and its charge (integer). Although edges are originally labeled with the valence of the linkage, such a value has been discarded, since it is not useful for the classification task.

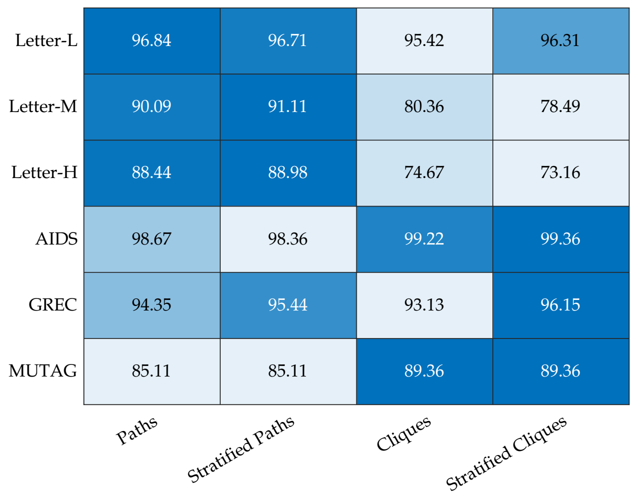
Letter-L: The Letter-L data set involves graphs that represent distorted letter drawings with a low level of distortion. The recognition task involves the 15 capital letters of the Roman alphabet that can be represented by straight lines only. Each handwritten letter is transformed into a graph by representing lines as edges and endpoints of lines as nodes. Each node is labeled by a two-dimensional real-valued vector giving its position within a reference coordinate system. Conversely, edges are unlabeled.

Letter-M: Same as Letter-L, but with medium level of distortion in handwritten digits.

Letter-H: Same as Letter-L, but with high level of distortion in handwritten digits.

GREC: The GREC data set consists of symbols from electronic and architectural drawings and, after suitable pre-processing, graphs are extracted from such images. Ending points, corners, intersections and circles are represented by nodes and labeled with a two-dimensional attribute giving their position. The nodes are connected by undirected edges, which are labeled as ”line” or ”arc”. An additional attribute specifies the ”angle” with respect to the horizontal direction or the diameter in case of arcs.

MUTAG: The MUTAG data set consists of 188 graphs corresponding to chemical compounds divided into two classes according to their respective mutagenic effect on a bacterium. Both nodes and edges are equipped with categorical labels: node labels identify the atom type and edge labels identify the bond type (single, double, triple or aromatic).



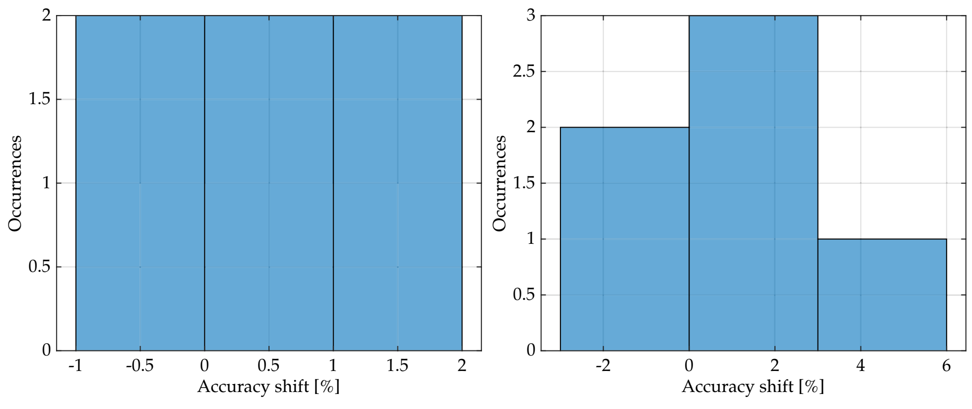
	Paths	Stratified Paths	Cliques	Stratified Cliques
Letter-L	96.84	96.71	95.42	96.31
Letter-M	90.09	91.11	80.36	78.49
Letter-H	88.44	88.98	74.67	73.16
AIDS	98.67	98.36	99.22	99.36
GREC	94.35	95.44	93.13	96.15
MUTAG	85.11	85.11	89.36	89.36

Fig. 2 Accuracy on the test set (in percentage)

Brief statistics about the data sets can be found in Table 1, along with the reference paper in which each data set has been originally presented, to which we refer the interested reader for more information. Specifically, for each data set, we show the number of graphs in the whole data set, the average number of nodes and edges amongst graphs in the data set, the number of classes for the classification problem, whether the classes are balanced or not and the data set application domain.

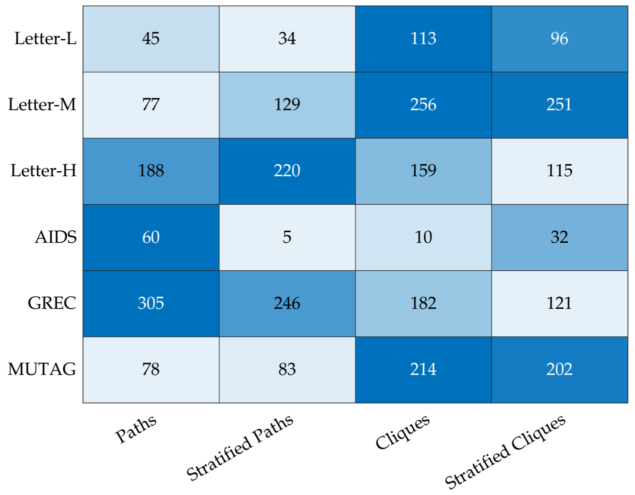
Table 1 Data set statistics

Data set Name	# Graphs	Avg. # Nodes	Avg. # Edges	# Classes	Balanced	Domain	References
AIDS	2000	15.69	16.20	2	No	Chemoinformatics	[40]
Letter-L	2250	4.7	3.1	15	Yes	Computer Vision	[40]
Letter-M	2250	4.7	3.2	15	Yes	Computer Vision	[40]
Letter-H	2250	4.7	4.5	15	Yes	Computer Vision	[40]
GREC	1100	11.5	12.2	22	Yes	Electronics	[40, 42]
MUTAG	188	17.93	19.79	2	No	Chemoinformatics	[43, 44]



(a) Paths (b) Cliques

Fig. 3 Distribution of the accuracy boosts for paths and cliques across the 10 data sets. A positive shift means that the stratified approach outperforms the non-stratified approach



	Paths	Stratified Paths	Cliques	Stratified Cliques
Letter-L	45	34	113	96
Letter-M	77	129	256	251
Letter-H	188	220	159	115
AIDS	60	5	10	32
GREC	305	246	182	121
MUTAG	78	83	214	202

Fig. 4 Size of the alphabet after feature selection

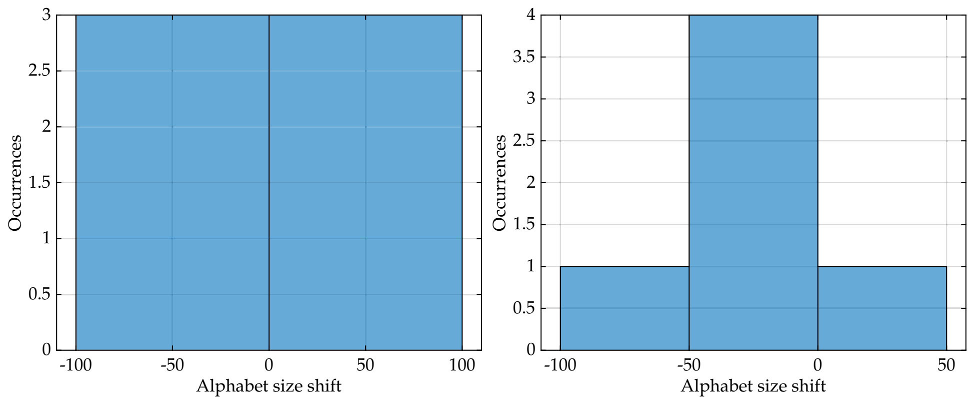
Each data set has been split into three disjoint sets: training, validation and test. For the 5 IAM data sets, we used the very same training/validation/test splits provided in the repository, whereas for MUTAG we had to perform our own splits with the following ratio: training set (50%), validation set (25%) and test set (25%). The splitting procedure has been performed in a stratified manner in order to preserve labels' distribution across the three sets.

For Letter-L, Letter-M, Letter-H, GREC and AIDS, details about the dissimilarity measures on nodes and edges can be found in [25, Appendix B]. We anticipate that GREC is the only data set with parametric dissimilarity measures on nodes and edges, i.e., for which $\Pi_{\text{edge}} \neq \emptyset$ and $\Pi_{\text{node}} \neq \emptyset$.

For MUTAG, since nodes and edges are equipped with categorical labels, the dissimilarity between nodes and the dissimilarity between edges are set as the plain discrete distance between labels (i.e., their distance is 1 if they are not equal and 0 otherwise) [45, Chapter 1].

Algorithmic Setup

The software has been fully implemented in Python with the support of the following third-party libraries: NetworkX [46] and Little Ball Of Fur [47] for handling and processing graph data structures, Scikit-Learn [48] for machine learning routines and SciPy [49] for optimization routines.



(a) Paths (b) Cliques

Fig. 5 Distribution of the differences of the alphabet size for paths and cliques across the 10 data sets (after feature selection). A positive shift means that the stratified approach underperforms the non-stratified approach

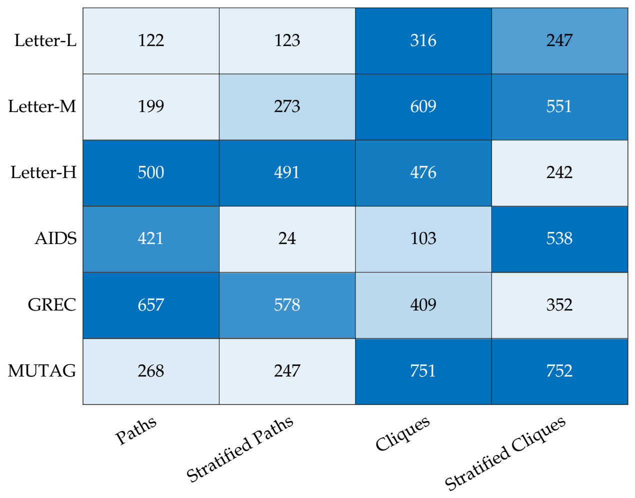


Fig. 6 Size of the alphabet before feature selection

Other algorithm parameters have been configured as follows:

- $\alpha = 0.9$ in the fitness function of the first genetic optimization (cf. Equation (7));
- $\beta = 0.9$ in the fitness function of the second genetic optimization (cf. Equation (11));
- Maximum number of 20 generations for both genetic optimizations;
- A total of 20 and 100 individuals in the population for the first and second genetic optimization, respectively;
- Maximum walk length $o = 5$;
- The user-defined bucket size W has been chosen according to a given percentage of the maximum number of paths or cliques that can be drawn from the training graphs. The percentages are chosen according to the following rule: the larger the data set, the higher the subsampling rate. For the sake of shorthand, we omit any sensitivity analysis on the behavior of the Granulators as a function of W. In this regard, we refer the interested reader to our previous work [25]. The values for W for both clique-based and path-based Granulators can be found in [1, Table 2].

Computational Results

In this section, we proceed in comparing the four different granulation strategies proposed in the section "Information Granulation Strategies", namely:

- The random walk-based granulator, in the following also called *Paths* for the sake of shorthand, presented in the section "Random Walk"
- The clique-based granulator, in the following also called *Cliques*, presented in the section "Clique"
- The stratified clique-based granulator, in the following also called *Stratified Cliques*, presented in the section "Stratified Clique", and
- The stratified random walk-based granulator, in the following also called *Stratified Paths*, presented in the section "Stratified Random Walk".

In Figs. 2 and 4 we show the accuracy on the test set and the number of resulting alphabet symbols, respectively, with Figs. 3 and 5 showing their respective detailed breakdown. For the sake of completeness, in Fig. 6, we show also the size of the alphabet before the second genetic optimization stage. Due to the stochastic nature of the model synthesis procedure, the results shown below have been obtained by averaging across 10 different runs of the algorithm. Each figure shows a heatmap whose color scale has been normalized by rows (i.e., independently for each data set) and ranges from white (lower values) towards blue (higher values).

By considering Fig. 2, it is possible to observe that (regardless of the topology) a stratified approach leads to generally better performance (i.e., higher accuracy on the test set). The breakdown of the shifts in accuracy is shown in Fig. 3: for both path-based and clique-based granulators, the majority of the differences between stratified and non-stratified approaches are positive. For path-based extractors, the only two data sets for which the opposite is true are AIDS and Letter-L. Conversely, for clique-based extractors, the two data sets showing the same phenomenon are Letter-M and Letter-H. By looking at the magnitude of the differences in terms of accuracy between stratified and non-stratified approaches, it can be observed that such differences are within $\pm 3\%$ at most. On the other hand, clustering-based extractors [1] have been shown to benefit more from a stratified approach (i.e., up to a $+10\%$ accuracy boost): this should not surprise, since clustering-based extractors are unsupervised by definition and the advantages of exploiting the ground-truth class labels are straightforward. Conversely, the INDVAL-based granulator (see the section "Extractor and Granulator") natively exploits the ground-truth class labels (cf. Eqs (1)–(3)).

By looking at Fig. 4, we can observe the number of symbols (i.e., the size of the alphabet) after the second genetic-driven feature selection stage. In principle, a clique-based approach is likely to yield a smaller alphabet: this is due to the fact that an n-vertex graph can have at most $\mathcal{O}(3^{n/3})$ cliques, whereas the number of paths can grow as $\mathcal{O}(n!)$ in the worst case, so the set of prospective information granules (even without any subsampling) is smaller. As for the accuracy case, a stratified approach yields some benefits. As can be seen in the breakdown (Fig. 5), as paths are concerned, the stratification yields a smaller alphabet for 3 data sets out

Table 2 Comparison against state-of-the-art graph classification system in terms of classification accuracy

Technique	AIDS	GREC	Letter-L	Letter-M	Letter-H	MUTAG	References
Bipartite Graph Matching + K-NN	–	86.3	91.1	77.6	61.6	–	[50]
Lipschitz Embedding + SVM	98.3	96.8	99.3	95.9	92.5	–	[55]
Graph Edit Distance + K-NN	97.3	95.5	99.6	94	90	–	[40]
Graph of Words + K-NN	–	97.5	98.8	–	–	–	[56]
Graph of Words + kPCA + K-NN	–	97.1	97.6	–	–	–	[56]
Graph of Words + ICA + K-NN	–	58.9	82.8	–	–	–	[56]
Topological embedding	99.4	–	–	–	–	–	[51, 63]
FMGE	99.0	–	–	–	–	–	[51, 64]
Attribute Statistics	99.6	–	–	–	–	–	[51, 65]
Hypergraph Embedding + SVM	99.3	–	–	–	–	84.6	[57]
ODD ST_+ kernel	82.06 *	–	–	–	–	–	[52]
ODD ST_+^{TANH} kernel	82.54 *	–	–	–	–	–	[52]
Laplacian kernel	92.6	–	–	–	–	–	[51, 66]
Treelet kernel	99.1	–	–	–	–	–	[51, 67]
Treelet kernel with MKL	99.7	–	–	–	–	–	[51, 68]
WJK Hypergraph kernel + SVM	99.5 *	–	–	–	–	90.9 *	[53]
CGMM + linear SVM	84.16 *	–	–	–	–	91.18 *	[59]
G-L-Perceptron	–	70	95	64	70	–	[60]
G-M-Perceptron	–	75	98	87	81	–	[60]
C-1NN	–	–	96	93	84	–	[60]
C-M-1NN	–	–	98	81	71	–	[60]
EigenGCN-1	–	–	–	–	–	–	[58]
EigenGCN-2	–	–	–	–	–	–	[58]
EigenGCN-3	–	–	–	–	–	–	[58]
GCN with logical descriptors	–	96.93	96.64	85.27	79.91	–	[61]
MPNN	–	89.5	91.3	81.2	64.24	–	[62]
MPNN (no set2set)	–	92.98	94.8	86.1	75.7	–	[62]
Deep Graphlet Kernel	–	–	–	–	–	82.66 *	[54]
GRALG Paths	99.09	79.13	96.80	91.11	88.31	83.69	[1]
GRALG Stratified Paths	99.02	82.15	97.20	92.09	90.00	85.82	[1]
GRALG Cliques	97.93	90.3	96.31	58.36	74.4	88.65	[1]
GRALG Stratified Cliques	99.22	92.74	97.24	80.22	77.82	90.07	[1]
INDVAL Paths	98.67	94.35	96.84	90.09	88.44	85.11	This work
INDVAL Stratified Paths	98.36	95.44	96.71	91.11	88.98	85.11	This work
INDVAL Cliques	99.22	93.13	95.42	80.3	74.67	89.36	This work
INDVAL Stratified Cliques	99.36	96.15	96.31	78.49	73.16	89.36	This work

$*$ Results refers to cross-validation rather than a separate test set

of 6. As cliques are concerned, only 1 data set (i.e., AIDS) sees a larger alphabet when no stratification is done.

In Table 2, we finally show a brief comparison against current approaches for graph classification. Competitors span a variety of techniques, including classifiers working on the top of GEDs [40, 50], kernel methods [51–54] and several embedding techniques [51, 55, 56], including Granular Computing-based [25, 57] and those based on neural networks and deep learning [58–62]. For the sake of completeness, we also include the results obtained with clustering-based granulators in our companion paper [1]. We can see that our method has comparable performances against current approaches in the graph classification literature. It is worth remarking that, aside from the mere numerical results, our approach has considerable higher perspectives to be deployed for analytical investigation by field experts thanks to its interpretability advantages, a common facet of Granular Computing-based systems [1, 25, 57], since the resulting set of automatically extracted granules of information can be analyzed by field experts to gather further insights on the problem at hand, paving the way for knowledge discovery.

Conclusion

In this work, we performed a two-fold investigation of filtering-based strategies for the automatic synthesis of information granules in the graph domain for solving (graph) classification problems on labeled graphs.

Our investigation jointly considers the subgraph topology to extract granules of information (i.e., cliques vs. paths extracted via a random walk) and the possibility of performing a class-aware, stratified procedure over the set of candidate information granules to build class-specific alphabets of symbols.

These strategies are tested in a graph embedding environment, where the set of automatically extracted granules of information serve as pivotal subgraphs for building and embedding space. The soundness of the latter is addressed by a statistical classifier working on the embedding space. A two-stage optimization process takes care of tuning the hyper-parameters of the classification model and performing an additional feature selection to ensure the selection of optimal granules of information.

To address the behavior of the granulation strategies, we performed a two-fold comparison in terms of classification performance and number of granules of information. To this end, 6 different open-access data sets of labeled graphs pertaining to different application domains have been considered. Computational results show that, conversely to a clustering-based granulator, a stratified approach does not yield any particular benefit in terms of classification accuracy, since the proposed filtering method natively performs a class-aware evaluation of the statistical significance of each candidate information granule.

Funding Open access funding provided by Luiss University within the CRUI-CARE Agreement. This research received no external funding.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Ethical Approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Martino A, Baldini L, Rizzi A. On information granulation via data clustering for granular computing-based pattern recognition: a graph embedding case study. Algorithms. 2022;15(5):148. https://doi.org/10.3390/a15050148.
2. Bargiela A, Pedrycz W. Granular computing: an introduction. Boston, USA: Kluwer Academic Publishers; 2003.
3. Pedrycz W, Skowron A, Kreinovich V. Handbook of granular computing. England: Wiley; 2008.
4. Zadeh LA. Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic. Fuzzy Sets Syst. 1997;90(2):111–27.
5. Yager RR, Filev D. Operations for granular computing: mixing words and numbers. In: 1998 IEEE International Conference on Fuzzy Systems Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98CH36228), vol. 1, p. 123–1281 (1998). https://doi.org/10.1109/FUZZY.1998.687470.
6. Zadeh LA. Fuzzy logic = computing with words. IEEE Trans Fuzzy Syst. 1996;4(2):103–11. https://doi.org/10.1109/91.493904.
7. Yao Y. Perspectives of granular computing. In: 2005 IEEE International Conference on Granular Computing. IEEE. vol. 1, p. 85–90 (2005).
8. Pedrycz A, Hirota K, Pedrycz W, Dong F. Granular representation and granular computing with fuzzy sets. Fuzzy Sets Syst. 2012;203:17–32.
9. Dubois D, Prade H. Bridging gaps between several forms of granular computing. Granul Comput. 2016;1(2):115–26.
10. Pawlak Z. Rough sets. Int J Comput Inf Sci. 1982;11(5):341–56. https://doi.org/10.1007/BF01001956.
11. Zhang Q, Zhang Q, Wang G. The uncertainty of probabilistic rough sets in multi-granulation spaces. Int J Approx Reason. 2016;77(C):38–54. https://doi.org/10.1016/j.ijar.2016.06.001.
12. Pedrycz W. Shadowed sets: representing and processing fuzzy sets. IEEE Trans Syst, Man, Cybern, Part B (Cybern). 1998;28(1):103–9. https://doi.org/10.1109/3477.658584.
13. Kreinovich V. Interval computation as an important part of granular computing: an introduction. England: Wiley; 2008. p. 1–31. https://doi.org/10.1002/9780470724163.ch1.
14. Pedrycz W. Proximity-based clustering: a search for structural consistency in data with semantic blocks of features. IEEE Trans Fuzzy Syst. 2013;21(5):978–82.
15. Ding S, Du M, Zhu H. Survey on granularity clustering. Cogn Neurodynamics. 2015;9(6):561–72.
16. Peters G, Weber R. DCC: a framework for dynamic granular clustering. Granul Comput. 2016;1(1):1–11.
17. Livi L, Del Vescovo G, Rizzi A. Graph recognition by seriation and frequent substructures mining. In: ICPRAM 2012 - Proceedings of the 1st International Conference on Pattern Recognition Applications and Methods, vol. 1, p. 186–191 (2012).
18. Rizzi A, Del Vescovo G. Automatic image classification by a granular computing approach. In: 2006 16th IEEE Signal Processing Society Workshop on Machine Learning for Signal Processing, p. 33–38 (2006). https://doi.org/10.1109/MLSP.2006.275517.
19. Hastie T, Tibshirani R, Friedman J. The elements of statistical learning: data mining, inference, and prediction. 2nd ed. New York: Springer; 2009.
20. Baldini L, Martino A, Rizzi A. Exploiting cliques for granular computing-based graph classification. In: 2020 International Joint Conference on Neural Networks (IJCNN), IEEE, p. 1–9 (2020). https://doi.org/10.1109/IJCNN48605.2020.9206690.
21. Baldini L, Martino A, Rizzi A. Towards a class-aware information granulation for graph embedding and classification. In: Merelo, J.J., Garibaldi, J., Linares-Barranco, A., Warwick, K., Madani, K. (eds.) Computational Intelligence: 11th

International Joint Conference, IJCCI 2019, Vienna, Austria, September 17–19, 2019, Revised Selected Papers. Springer, Cham; 2021. p. 263–290. https://doi.org/10.1007/978-3-030-70594-7_11.
22. Baldini L, Martino A, Rizzi A. 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), SciTePress, INSTICC, p. 391–402 (2019). https://doi.org/10.5220/0008149403910402.
23. Martino A, Giuliani A, Todde V, Bizzarri M, Rizzi A. Metabolic networks classification and knowledge discovery by information granulation. Comput Biol Chem. 2020;84: 107187. https://doi.org/10.1016/j.compbiolchem.2019.107187.
24. Martino A, Giuliani A, Rizzi A. The universal phenotype. Organisms. J Biol Sci. 2019;3(2):8–10.
25. Martino A, Rizzi A. An enhanced filtering-based information granulation procedure for graph embedding and classification. IEEE Access. 2021;9:15426–40. https://doi.org/10.1109/ACCESS.2021.3053085.
26. Baldini L, Martino A, Rizzi A. Relaxed Dissimilarity-based Symbolic Histogram Variants for Granular Graph Embedding. In: Proceedings of the 13th International Joint Conference on Computational Intelligence - NCTA, p. 221–235. SciTePress, INSTICC (2021). https://doi.org/10.5220/0010652500003063.
27. Dufrêne M, Legendre P. Species assemblages and indicator species: the need for a flexible asymmetrical approach. Ecol Monogr. 1997;67(3):345–66. https://doi.org/10.2307/2963459.
28. Martino A, De Santis E, Rizzi A. An ecology-based index for text embedding and classification. In: 2020 International Joint Conference on Neural Networks (IJCNN), p. 1–8 (2020). https://doi.org/10.1109/IJCNN48605.2020.9207299
29. Sanfeliu A, Fu K-S. A distance measure between attributed relational graphs for pattern recognition. IEEE Trans Syst, Man, Cybern. 1983;SMC–13(3):353–62. https://doi.org/10.1109/TSMC.1983.6313167.
30. Gao X, Xiao B, Tao D, Li X. A survey of graph edit distance. Pattern Anal Appl. 2010;13(1):113–29. https://doi.org/10.1007/s10044-008-0141-y.
31. Cover T, Hart P. Nearest neighbor pattern classification. IEEE Trans Inf Theory. 1967;13(1):21–7.
32. Lovász L. Random walks on graphs: a survey. Combinatorics. 1993;2:1–46.
33. Göbel F, Jagers AA. Random walks on graphs. Stoch Process Appl. 1974;2(4):311–36. https://doi.org/10.1016/0304-4149(74)90001-5.
34. Tichy N. An analysis of clique formation and structure in organizations. Adm Sci Q. 1973;18(2):194–208.
35. Luce RD, Perry AD. A method of matrix analysis of group structure. Psychometrika. 1949;14(2):95–116. https://doi.org/10.1007/BF02289146.
36. Bron C, Kerbosch J. Algorithm 457: finding all cliques of an undirected graph. Commun ACM. 1973;16(9):575–7. https://doi.org/10.1145/362342.362367.
37. Moon JW, Moser L. On cliques in graphs. Israel J Math. 1965;3(1):23–8. https://doi.org/10.1007/BF02760024.
38. Storn R, Price K. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim. 1997;11(4):341–59. https://doi.org/10.1023/A:1008202821328.
39. Grandini M, Bagli E, Visani G. Metrics for multi-class classification: an overview. arXiv (2020). https://doi.org/10.48550/ARXIV.2008.05756.
40. Riesen K, Bunke H. IAM graph database repository for graph based pattern recognition and machine learning. In: da Vitoria Lobo N, Kasparis T, Roli F, Kwok JT, Georgiopoulos M, Anagnostopoulos GC, Loog M, editors. Structural, syntactic, and statistical pattern recognition. Berlin, Heidelberg: Springer; 2008. p. 287–97. https://doi.org/10.1007/978-3-540-89689-0_33.
41. Morris C, Kriege NM, Bause F, Kersting K, Mutzel P, Neumann M. Tudataset: A collection of benchmark datasets for learning with graphs. In: ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020) (2020). www.graphlearning.io.
42. Dosch P, Valveny E. Report on the second symbol recognition contest. In: Liu W, Lladós J, editors. Graphics recognition. Ten years review and future perspectives. Berlin, Heidelberg: Springer; 2006. p. 381–97.
43. Debnath AK, de Compadre RLL, Debnath G, Shusterman AJ, Hansch C. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds correlation with molecular orbital energies and hydrophobicity. J Med Chem. 1991;34(2):786–97. https://doi.org/10.1021/jm00106a046.
44. Kriege N, Mutzel P. Subgraph matching kernels for attributed graphs. In: Proceedings of the 29th International Coference on International Conference on Machine Learning. ICML'12. Omnipress, Madison, WI, USA; 2012. p. 291–298.
45. Deza MM, Deza E. Encyclopedia of distances. 1st ed. Berlin, Heidelberg: Springer; 2009. p. 1–583.
46. Hagberg AA, Schult DA, Swart PJ. Exploring network structure, dynamics, and function using networkx. In: Varoquaux G, Vaught T, Millman J (eds) Proceedings of the 7th Python in Science Conference, Pasadena, CA USA; 2008. p. 11–15.
47. Rozemberczki B, Kiss O, Sarkar R. Little ball of fur: a python library for graph sampling. In: Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), ACM; 2020. p. 3133–3140.
48. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E. Scikit-learn: machine learning in python. J Mach Learn Res. 2011;12:2825–30.
49. ...Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, Burovski E, Peterson P, Weckesser W, Bright J, van der Walt SJ, Brett M, Wilson J, Millman KJ, Mayorov N, Nelson ARJ, Jones E, Kern R, Larson E, Carey CJ, Polat İ, Feng Y, Moore EW, VanderPlas J, Laxalde D, Perktold J, Cimrman R, Henriksen I, Quintero EA, Harris CR, Archibald AM, Ribeiro AH, Pedregosa F, van Mulbregt P. SciPy 1.0: fundamental algorithms for scientific computing in python. Nat Methods. 2020;17:261–72. https://doi.org/10.1038/s41592-019-0686-2.
50. Riesen K, Bunke H. Approximate graph edit distance computation by means of bipartite graph matching. Image Vis Comput. 2009;27(7):950–9.
51. Conte D, Ramel J-Y, Sidère N, Luqman MM, Gaüzère B, Gibert J, Brun L, Vento M. A comparison of explicit and implicit graph embedding methods for pattern recognition. In: Kropatsch WG, Artner NM, Haxhimusa Y, Jiang X, editors. Graph-based representations in pattern recognition. Berlin, Heidelberg: Springer; 2013. p. 81–90. https://doi.org/10.1007/978-3-642-38221-5_9.
52. Da San Martino G, Navarin N, Sperduti A. Ordered decompositional DAG kernels enhancements. Neurocomputing. 2016;192:92–103.
53. Martino A, Rizzi A. (hyper)graph kernels over simplicial complexes. Entropy. 2020;22(10):1155. https://doi.org/10.3390/e22101155.
54. Yanardag P, Vishwanathan SVN. Deep graph kernels. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '15. Association for Computing Machinery, New York, NY, USA; 2015. p. 1365–1374. https://doi.org/10.1145/2783258.2783417.

55. Riesen K, Bunke H. Graph classification by means of lipschitz embedding. IEEE Trans Syst, Man, Cybern Part B (Cybern). 2009;39(6):1472–83.
56. Gibert J, Valveny E, Bunke H. Dimensionality reduction for graph of words embedding. In: Jiang X, Ferrer M, Torsello A, editors. Graph-based representations in pattern recognition. Berlin, Heidelberg: Springer; 2011. p. 22–31.
57. Martino A, Giuliani A, Rizzi A. (hyper)graph embedding and classification via simplicial complexes. Algorithms. 2019;12(11):223. https://doi.org/10.3390/a12110223.
58. Ma Y, Wang S, Aggarwal CC, Tang J. Graph convolutional networks with eigenpooling. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. KDD '19. Association for Computing Machinery, New York, NY, USA; 2019. p. 723–731. https://doi.org/10.1145/3292500.3330982.
59. Bacciu D, Errica F, Micheli A. Contextual graph markov model: a deep and generative approach to graph processing. In: 35th International Conference on Machine Learning, ICML 2018, vol. 1, pp. 495–504 (2018).
60. Martineau M, Raveaux R, Conte D, Venturini G. Learning error-correcting graph matching with a multiclass neural network. Pattern Recognit Lett. 2020;134:68–76. https://doi.org/10.1016/j.patrec.2018.03.031.
61. Kajla NI, Missen MMS, Luqman MM, Coustaty M. Graph neural networks using local descriptions in attributed graphs: an application to symbol recognition and hand written character recognition. IEEE Access. 2021;9:99103–11. https://doi.org/10.1109/ACCESS.2021.3096845.
62. Riba P, Dutta A, Lladós J, Fornés A. Graph-based deep learning for graphics classification. In: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), vol. 02, p. 29–30 (2017). https://doi.org/10.1109/ICDAR.2017.262.
63. Sidère N, Héroux P, Ramel J-Y. Vector representation of graphs: application to the classification of symbols and letters. In: 2009 10th International Conference on Document Analysis and Recognition, p. 681–685 (2009). https://doi.org/10.1109/ICDAR.2009.218.
64. Luqman MM, Ramel J-Y, Lladós J, Brouard T. Fuzzy multilevel graph embedding. Pattern Recognit. 2013;46(2):551–65. https://doi.org/10.1016/j.patcog.2012.07.029.
65. Gibert J, Valveny E, Bunke H. Graph embedding in vector spaces by node attribute statistics. Pattern Recognit. 2012;45(9):3072–83.
66. Brun L, Conte D, Foggia P, Vento M. A graph-kernel method for re-identification. In: Kamel M, Campilho A, editors. Image analysis and recognition. Berlin, Heidelberg: Springer; 2011. p. 173–82.
67. Gaüzère B, Brun L, Villemin D. Two new graphs kernels in chemoinformatics. Pattern Recogn Lett. 2012;33(15):2038–47. https://doi.org/10.1016/j.patrec.2012.03.020.
68. Gaüzère B, Brun L, Villemin D, Brun M. Graph kernels based on relevant patterns and cycle information for chemoinformatics. In: Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012), p. 1775–1778 (2012).