SAPIENZA
UNIVERSITÀ DI ROMA

Evolutionary Graph Classification Systems by Granular Computing based Embedding

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica

Dottorato di Ricerca in Information and Communication Technologies – XXXIV Ciclo

Candidate

Luca Baldini
ID number 1348590

Thesis Advisor

Prof. Antonello Rizzi

23rd January 2022

Evolutionary Graph Classification Systems by Granular Computing based Embedding
Ph.D. thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: luca.baldini@uniroma1.it

Contents

List of Figures

List of Tables

List Of Acronyms

***k*-NN** k Nearest Neighbor.

BFS Breadth First Search.

BSAS Basic Sequential Algorithmic Scheme.

DFS Depth First Search.

E-ABC Evolutionary Agent Based Classifier.

GED Graph Edit Distance.

GRALG Granular Computing Approach for Labelled Graph.

GrC Granular Computing.

MAS Multi Agent System.

MinSOD (element that leads to the) Minimum Sum of Distances (syn. *medoid*).

MOO Multi-Objective Optimization.

nBMF Node Best Match First procedure.

NSGA Non Dominated Sorting Algorithm.

PCA Principal Component Analysis.

RBF Radial Basis Function.

SVM Support Vector Machine.

t-SNE t-Distributed Stochastic Neighbor Embedding.

TOPSIS Technique for Order Preference by Similarity to the Ideal Solution.

Abstract

Graphs have gained a lot of attention in the pattern recognition community thanks to their ability to encode both topological and semantic information. Despite their invaluable descriptive power, their arbitrarily complex structured nature poses serious challenges when they are involved in learning systems. Typical approaches aim at building a vectorial representation of the graph in a suitable embedding space by leveraging on the selection of relevant prototypes that enable the use of common pattern recognition methods. An interesting paradigm able to synthesize prototypes in a data-driven fashion can be found in Granular Computing.

This thesis investigates and develops novel techniques for graph embedding methods based on Granular Computing and Multi-Agent based systems in order to solve Pattern Recognition problems. Initially, the proposed methods aim at improving different aspects of an established Granular Computing-based framework designed for graph classification concerning the computational complexity, granulation and embedding ability and optimization problem of the training phase. A lightweight stochastic procedure for the selection of prototypes has been designed in order to mitigate the computational burden of the algorithm. Other proposed techniques focus on improving the granulation phase of the framework by selecting detailed granules of information that characterize the classes of the problem at hand. Concerning the embedding phase, six different graph embedding techniques inspired by the dissimilarity space embedding are proposed to represent graphs into meaningful embedding spaces. Concerning the optimization phase, a novel evolutionary-based approach has been designed for equipping the framework with a class-specific metric learning strategy together with a reformulation in a multi-objective fashion of the problem which aims at jointly optimizing the performance of the classifier, the number of information granules and the structural complexity of the classification model. In the second part of the work, a novel prototypical system for graph classification problems inspired to Multi-Agent Systems principles has been presented. The proposed system investigates a cooperative approach between different groups of agents for synthesizing meaningful granules of information and in turn enabling the graph embedding process via the Granular Computing paradigm. Different publicly available real-world benchmark datasets have been selected in order to show the effectiveness of the proposed methods in comparison with state-of-the-art graph-based classification systems.

Chapter 1

Introduction

1.1 Graph and Structured Domain

In a wide range of scientific areas, interesting systems and phenomena are frequently represented according to a structured prospective. The *structural* facet stems from the fact that a thorough insight of a system behavior or specific phenomenon can emerge only by considering the interactions of single parts composing the whole. A suitable definition of structured information has been expressed by Foggia et al. in [62]:

> *Complex information which can be seen as made of simple parts suitably interconnected.*

Time Series, Sequences and Strings are typical examples of structured information that emerge in many real-world applications: individually, each instance of such structured data is composed by a collection of inputs that are sequentially connected together with a specific relation. In bioinformatics, complex sequences of biological material naturally arise as strings of DNA and RNA or proteins [124]. From a point of view of Natural Language Processing, sentences can be analyzed as sequences of words in order to provide a structural and semantic description [127]. In Economics, the market behavior can emerge by inspecting the financial time series of transactions and stock prices in order to find meaningful trends [123].

In principle, from an algorithmic perspective, any structured information included those discussed above, can be represented as an instance of a specific data structure, namely the *graph.* The outstanding representational power of graphs has attracted scholars from many different disciplines. In the philosophic manuscript *"The mathematical structure of the world"*, Randal Dipert[1] highlights the importance of graphs for the comprehension of real-world phenomena [57]:

> *The concrete world is a single, large structure induced by a single, two place, symmetric relation, and thus best analyzed as a certain sort of graph. Every concrete entity in the world is a part of this structure and*

[1]Randall Roy Dipert (1951–2019) was an American philosopher and professor of philosophy at the State University of New York at Fredonia, the United States Military Academy, and the University at Buffalo where he retired as the C. S. Peirce Chair of American Philosophy. Source: Wikipedia.

is a structure (subgraph) in its own right. (...) Both reality and thoughts are structures of certain sorts and, then, by arguing that the correct or most perspicuous portrayal of this structure is purely relational and, in fact, best portrayed by graphs.

Graphs are ubiquitous mathematical entities which provide a rich representation of many objects and complex systems. In its simplest form, a graph can be defined as a set of *nodes* (or *vertices*) and a set of *edges* (or *arcs*) which define whether there exist a connection between objects in the set of nodes, thus describing a network of relationships between entities. Even more expressive graph types are the so-called *labelled graphs*: along with the sets of nodes and edges, the graph can be enriched with additional information by means of arbitrary data structures defining individually the nodes properties and the relation characteristics, respectively the node and edges *labels*. These aspects allows graphs to capture two levels of information: the *topological* information [183], by encoding the interaction between its constituent parts and the arising structure, and the *semantic* information, according to the arbitrarily complex attributes with which the constituent parts are equipped, i.e. the node attributes. Thanks to this dual abstraction power, graphs can be considered as the most general data structure in the field of Computer Science [33]: a scalar can be modeled as a single node labeled by the scalar value; a time series can be represented as a graph that contains one node per time step, and consecutive steps are linked by an edge; a string is a graph in which each node represents a character, and consecutive characters are connected by an edge [26]. For the generalization characteristics that graphs possess, they have found widespread application in many different fields, typically related (but not limited) to complex networks. For example, in bioinformatics and chemoinformatics [3, 78, 55, 65, 176, 171, 98], the application of graphs has been intensively studied thanks to the straightforward representation of molecular compounds as interacting systems in the graph domain. Graph-based representations have been also employed for applications in the fields of computer vision and image recognition [1, 131, 180, 147, 107, 73, 156], anomaly detection [4], web content mining [33], speech recognition [94], natural language processing [50, 121] and energy distribution networks [144].

1.2 Pattern Recognition and Computational Intelligence in Graph Domain

The ability in recognizing patterns from the surrounding environment is an essential aspect characterizing the highly performative human cognitive function. Indeed, human knowledge and reasoning are fundamentally based on the investigation of such patterns and on their suitable aggregation which defines concepts and rules. Common and relatively simple pattern recognition problems which unconsciously humans solve in every day life include the recognition of written and spoken words, understanding the meaning of traffic signals, the identification of a friend face [155, 151]. Unaccustomed readers may legitimately ask what is meant by the word *pattern* and consequent definition of *pattern recognition.* According to Theodoridis and Koutroumbas [172]:

Pattern recognition is the scientific discipline whose goal is the classification of objects into a number of categories or classes. Depending on the application, these objects can be images or signal waveforms or any type of measurements that need to be classified. We will refer to these objects using the generic term "patterns" .

From this definition, patterns are generic observations of a generating process $P : \mathcal{X} \to \mathcal{Y}$. The process P is intended as an idealization of any concrete or abstract system that provides a specific output according to the input. From this prospective, the input space $\mathcal{X}$ is the formal domain in which the patterns are effectively represented according to a suitable mathematical idealization. Consequently, $\mathcal{Y}$ is the corresponding output space where the recognition happens. Synthesizing a predictive model $\hat{P}$ of the generation process P is the foundation of all the natural sciences and engineering disciplines as well. In *analytical* or *theory-driven* approaches, the synthesis of $\hat{P}$ is performed by human field experts which, according to the observations of the process output in response to suitable inputs, identify the most relevant quantities and their relation according to mathematical equations. On the other hand in *data-driven* methods, the model synthesis phase is carried on automatically by any sort of computational unit according to a series of finite operations, i.e. the *algorithms*, analyzing a finite set of patterns sampled from P, namely the dataset $\mathcal{D}$ [115, 113]. The ultimate goal of pattern recognition is to provide powerful algorithms able to efficiently approximate the generation process P.

Parallel to the development of Pattern Recognition methods, Soft Computing emerged as a discipline encompassing a set of biologically inspired problem solving strategies for dealing with complex environments [199]. Pattern Recognition and Soft Computing are nowadays intimately connected disciplines whose methods interact together and are recognized under an hybrid term "Computational Intelligence".

Nonetheless, a wide range of established Computational Intelligence techniques demand n-uples of real numbers as input data structure. The motivation behind this design choice stems from the fact that vector input spaces (where $\mathcal{X} \subseteq \mathbb{R}^n$) are far more tractable from an algorithmic perspective with respect to structured domains such as the graph domain. Vector domains like Euclidean spaces provide a rich set of algebraic operations that are typically involved in the algorithmic tools, such as the summation or product of two elements. On the contrary, the graph domain is devoid of these essential operations and also lack of a meaningful geometric interpretation, two non-negligible aspects that have strongly limited the proliferation of Computational Intelligence methods addressing the graph domain. Nonetheless, from a representational power point of view, vectors are not able to convey the same amount of information when compared with graphs, as already mentioned in Section 1.1. Indeed, vectors are static entities representing a fixed set of pattern features and they can not describe by no means the interaction between different components of the pattern. Graphs, as instead, are not constrained to a fixed size, but can be adapted to the complexity of each specific considered pattern [151]. On the other hand, this flexibility is counterbalanced in generally expensive procedures from a computational point of view. For example, a simple comparison between two graphs requires to identify all the common parts, hence considering all the possible $\mathcal{O}(2^n)$ subsets of nodes leading to an exponential complexity, whilst the same operation in Euclidean space is linear with respect to the number of features [32]. A nice quotation from Borgwardt [26] summarizes the controversial aspects when dealing with graphs:

Graphs are prisoners of their own flexibility

Graph Embedding techniques emerged as promising solutions for dealing with most of the limitations when the graph domain is chosen as the input space for Computational Intelligence methods. Basically, an embedding of graphs into a vector space allows access to the rich repository of algorithmic tools for pattern recognition. The overall objective of all graph embedding techniques is to condense the high representational power of graphs into a computationally efficient and mathematically convenient feature vector [155]. In these approaches, the input pattern from the structured graphs domain $\mathcal{G}$ is explicitly mapped into an embedding (feature) space $\mathcal{F}$ according to a suitable mapping function $\phi : \mathcal{G} \to \mathcal{F}$ with $\mathcal{F} \subseteq \mathbb{R}^n$. The design of ϕ is obviously crucial and some efforts must be ensured to fill the informative and semantic gap between the two domains. There are several ways to perform explicit embedding in either naïve or automatic approaches. Naïve approaches usually leverage on feature engineering (also known as feature generation), in which the data analyst manually designs the mapping function ϕ by extracting numerical features from the (structured) pattern at hand and concatenates them in a vector form. Amongst the automatic approaches, graph neural networks emerged as a deep learning-based methods for automatically learning low-dimensional graph embeddings [35]. However, as common in deep architectures, the model interpretability is a delicate issue whose addressing is still out of reach [188]. Another automatic approach for explicit graph embedding which, at the same time, returns a fully interpretable model is based on Granular Computing [16].

Granular Computing emerged within the field of computational intelligence as an information processing paradigm able to deal with uncertainties in the data. The rationale behind this approach is inspired by the human problem solving strategies in complex situations and problems with partial knowledge and certainties, where approximations and simplifications are practically needed for making decisions [194, 192]. The fundamental aspect of Granular Computing resides in the definition of *information granules* which can be described as atomic data entities that condense relevant information on the problem (or system), synthesized at a particular level of abstraction. Indeed, granules must reflect common properties related to an aggregation of data in terms of proximity, functionality and similarity according to the indistinguishability rule [58]. Generally speaking, in recent years, Granular Computing has been a dynamic paradigm for synthesizing advanced machine learning and pattern recognition systems both under a practical (see e.g. [104, 105, 115, 157]) and methodological viewpoint (see e.g. [27, 42, 54, 185]). The appeal of Granular Computing-based pattern recognition systems stems from their ability to automatically design (in a data-driven manner) the mapping function. Furthermore, these systems are human-interpretable, as the automatically-extracted information granules can provide field-experts with further knowledge about the modelled system.

1.3 Aim and Objectives

The aim of this work regards the investigation and development of efficient graph embedding methods based on Granular Computing for solving pattern recognition problems. Starting from an established framework designed for graph classification called GRALG, novel techniques are discussed addressing a wide range of aspects in order to overcome limitations and to improve crucial issues:

- Designing a lightweight stochastic procedure for the selection of candidate information granules that is able to mitigate the computational complexity of the training procedure.
- Investigating different topologies as candidates information granules and their impact on the classification performances
- Improving the granulation phase with a class-aware approach in order to select detailed granules of information related to the problem classes.
- Proposing novel embedding techniques based on dissimilarity space embedding
- Designing an evolutionary scheme for the optimization procedure able to provide class-specific dissimilarity measures overcoming the limitation of GRALG in finding only a global dissimilarity measure valid for the whole data space.
- Reformulating the optimization problem in a multi-objective fashion taking into account the learning performance and the structural complexity of the classifier together with the number of granules employed in the embedding phase.
- Introducing a novel prototypical system for graph classification specifically designed for labelled graphs inspired to Multi-Agent Systems. The proposed system leverages on a cooperative mechanism between two groups of agents that are individually in charge of discovering granules of information and then enabling the graph embedding procedure, as opposed to an individualistic approach followed by GRALG.

Different publicly available real-world benchmark datasets have been selected in order to show the effectiveness of the proposed methods in comparison with state-of-the-art graph-based classification systems.

1.4 Thesis Organization

This thesis consists in seven chapters and is organized as follows:

- In Chapter 2, the reader will be introduced to pattern recognition in the graph domain. Specifically, in Section 2.1 some preliminary definitions about common learning paradigms and graph theoretical notions are provided whilst in Section 2.2 the discussion focuses on the State-of-the-Art strategies for dealing with pattern recognition in the graph domain.

- In Chapter 3, the graph classification framework based on Granular Computing, i.e. GRALG, is introduced and described thoroughly.
- The discussion moves on Chapter 4, where novel techniques for improving GRALG are proposed. Specifically, in Section 4.1 three strategies for extracting candidate granules of information are provided. In Sections 4.2 and 4.4, different granulations and embedding strategies are proposed. Next, a novel evolutionary scheme enabling a class-specific metric learning is presented in Section 4.3. Finally, in Section 4.5 a multi-objective optimization is investigated in order to analyze the resulting embedding space under three different perspectives, i.e. the classification performances, the number of information granules required in the embedding process and the structural complexity of the selected classifier.
- Chapter 5 describes the novel graph classification system, namely E-ABC, providing motivations and limitations of the proposed system.
- Chapter 6 presents the computational results on benchmark datasets: in Section 6.2 the discussion focused on the validation of the methods proposed in Chapter 4, whereas in Section 6.3 preliminary results for E-ABC are provided.
- Conclusions are finally drawn in Chapter 7.
- Additionally in Appendix A are shown further graphical results regarding tests carried on for multi-objective optimization described in Section 4.5.

Chapter 2

Pattern Recognition in Graph Domain

2.1 Preliminary Definitions

The automatic discovery of regularities in data and the use of these regularities to make decisions is at the basis of pattern recognition activities. The process of discovery is performed by means of computer algorithms that motivate the automatic characteristic of the discovery [24]. Specifically, from a set of *training data* $\mathcal{D}^{(\mathrm{tr})}$ the system tries to *learn* an approximation $\hat{P}$ of the oriented process $P : \mathcal{X} \rightarrow \mathcal{Y}$ which generates $\mathcal{D}^{(\mathrm{tr})}$. This phase is often referred to as *training phase* (or *learning phase*) since a specific model $\hat{P}$ is trained (or learned) starting from the available data. The model $\hat{P}$ can be employed to predict a target value $y \in \mathcal{Y}$ for a pattern $x \in \mathcal{X}$ which usually belong to an independent *test set* $\mathcal{D}^{(\mathrm{ts})}$. The correct recognition of previously unseen patterns, i.e. test set data not used to train the model, is a fundamental goal in pattern recognition, which usually aims at quantifying the *generalization* ability of the trained model.

Generally speaking, pattern recognition activities are usually divided in different categories according to the information carried along with the training examples:

- Supervised Learning: training data $\mathcal{D}^{(\mathrm{tr})}$ comprises a set of target *labels* which are associated to patterns in the training set. That is, each pattern belonging to the training set is provided in the form (x, y), where $y = P(x)$ is an a-priori information available for pattern x. Two specific problems can be identified in supervised learning: *classification* and *functional approximation* (or *regression*). The former encompasses a class of problems in which the output value y belongs to a finite set of categories (e.g. recognize spam or not spam in email classification, fault or non-fault device in anomaly detection). The latter, instead, identifies those problems where y is a continuous variable (e.g. predict the solubility of a chemical compound, the stock price in financial markets). It is worth noticing that according to these definitions, the output domain $\mathcal{Y}$ of classification problems is a non-normed space. Specifically, it is not possible to establish an order between the pattern labels and a possible concept of distance between categories. Conversely, output values in approximation problems are drawn from domains endowed with a suitable norm.

- Unsupervised Learning: as opposed to supervised learning, the training set lacks of any a-priori information about the class of each pattern, that is no label y for pattern x is available. Hence in this class of problems, the goal is to unravel the underlying similarities between patterns in order to discover groups of data that share similar characteristics. These groups are usually known as *clusters* and the process of finding a division between similar and dissimilar patterns is commonly referred to as *clustering* [1]. The caveat with such approaches is that the possible emerging structures and partitions may not properly be natural, that is, they may not represent the real data division, rather they are artificially imposed by the clustering method. Indeed, clustering algorithms output is highly sensitive to hyper-parameters and initialization conditions whose variations can lead to completely different partitions. More important, the definition of a procedure able to measure the dissimilarity (or similarity) between patterns is often not trivial and the most suitable choice depends on the specific application.

- Reinforcement Learning: in this particular class of problems, the learning algorithm does not rely on both examples or optimal output in contrast to the two approaches discussed above. Rather, an *agent* has to learn an optimal policy by interacting with the surrounding environment in order to take reasonable actions in response to a specific state. Hence, the agents shall determine sequences of actions that lead to a success according to a feedback signal provided by the environment usually referred to as *reward*. Reinforcement learning has successfully applied in a wide range of complex domains such as game playing, robotics, communication networks and biology [103, 87, 100, 169].

It is worth noticing that the discussed above methods should not be necessary intended as "stand-alone" or independent solutions. Efficient and effective pattern recognition systems rather use together the techniques inherited from all the learning methodology especially in hard problems with possible deficiency of labelled data. Indeed, in *semi-supervised* learning settings, the training dataset $\mathcal{D}^{(\text{tr})}$ can be partially annotated with the target labels, that is not all data samples $x \in \mathcal{D}^{(\text{tr})}$ have a corresponding y label [39, 175].

The problem of representing patterns in a suitable formalism such that the pattern recognition system can perform a learning task is of paramount importance. As already introduced in Section 1.2, the selection of a candidate data structure to feed the automatic system is often driven by different factors:

- Information conveyed by the chosen representation

- Computational complexity of the procedures involved for dealing with the selected data structure

Another very important aspect to consider is the notion of *dissimilarity* (or *similarity*) measure available in the input space for mutually compare patterns,

[1]In principle, unsupervised learning is not bounded to clustering [24]. Other approaches usually encompass a set of techniques for dimensionality reduction that comes at hand for data visualization [148, 81].

a task which is very common in pattern recognition algorithms. The following properties allow to formally define a dissimilarity measure:

Definition 2.1.1 (Dissimilarity Measure)**.** Let $\mathcal{X}$ be a generic input space. A function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_0^+$ is a dissimilarity measure if the following conditions hold:

$$\exists d_0 \in \mathbb{R} \quad \text{such that} \quad -\infty < d_0 \leq d\left(x_1, x_2\right) < \infty \tag{2.1}$$
$$d\left(x_1, x_2\right) = d_0 \tag{2.2}$$
$$d\left(x_1, x_2\right) = d\left(x_2, x_1\right) \tag{2.3}$$

for any $x_1, x_2 \in \mathcal{X}$. In case d meets also the following conditions:

$$d\left(x_1, x_2\right) = d_0 \quad \text{if and only if} \quad x_1 = x_2 \tag{2.4}$$
$$d\left(x_1, x_3\right) \leq d\left(x_1, x_2\right) + d\left(x_2, x_3\right) \tag{2.5}$$

for any $x_1, x_2, x_3 \in \mathcal{X}$, then d is metric dissimilarity measure.

Hence, $\mathcal{X}$ is a non-metric space if the function d measuring the pairwise dissimilarities between elements in $\mathcal{X}$ does not satisfy at least one of Eqs. (2.1)-(2.5). In such case, an intuitive concept such as the "position" of the pattern in input space is not available because elements belonging to $\mathcal{X}$ lack of a geometric interpretation. Most of the effort in pattern recognition has been directed to the design of algorithmic tools conceived for vector input spaces $\mathcal{X} \subseteq \mathbb{R}^n$ which are usually endowed with metric dissimilarity measures such as the Euclidean distance.

Statistical Pattern Recognition is the specific branch of Pattern Recognition that deals with vector data structures for representing patterns. Formally, in this context, a pattern is intended as a *feature vector* $\mathbf{x} \in \mathbb{R}^n$ of n observation or measurements, i.e. $\mathbf{x} = \left[x^{(1)}, \ldots, x^{(n)}\right]$. As opposed, Structural Pattern Recognition comprises a set of techniques specifically designed for dealing with structured information. In Figure 2.1, it is provided a graphical description of statistical and structural representation for a simple image. As introduced in Section 1.1, graphs generalize the concept of structured data thus Structural Pattern Recognition scholars have widely focused their research activity toward algorithmic solutions in graph domain $\mathcal{G}$. Indeed, common structured data such as *sequences* and *trees* can be considered as particular cases of graphs: sequences are directed graphs whose nodes are consecutively connected by an edge, whilst in trees any two nodes are connected by exactly one path [155]. In order to appreciate the advantages of graph-based representations, a formal definition of a labelled graph is provided:

Definition 2.1.2 (Labelled Graphs)**.** A labelled graph G is a four tuple $G = (\mathcal{V}, \mathcal{E}, \mathcal{L}^v, \mathcal{L}^e)$, where:

1. $\mathcal{V}$ is the set of edges

2. $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of nodes

3. $\mathcal{L}^v$ is the set of node attributes

4. $\mathcal{L}^e$ is the set of edge attributes

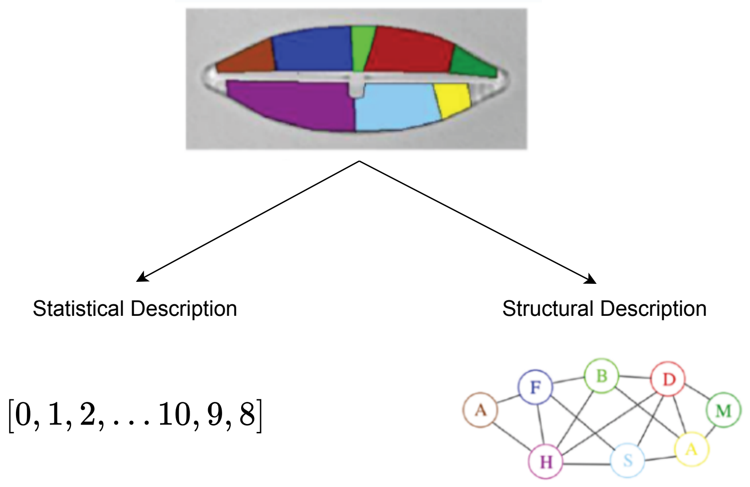


Figure 2.1. Statistical and Structural representation of a diatom image. In the statistical description, the object is represented as a collection of characteristics formally described as a feature vector. The structural description provides instead more information about the object under analysis: functional and relevant components of the diatom are highlighted with different colours and position vectors (semantic level) which define the node labels. Furthermore, the interactions between these parts is revealed by the emerging graph topology (topological level). Images has been taken from [34]

and exists two labelling functions $\mu : \mathcal{V} \rightarrow \mathcal{L}^v$ and $\nu : \mathcal{V} \times \mathcal{V} \rightarrow \mathcal{L}^e$ that respectively assign a node $v \in \mathcal{V}$ to an element of $\mathcal{L}^v$ and an edge $(u, v) \in \mathcal{E}$ to an element of $\mathcal{L}^e$

Analogously, *subgraphs* can be defined as subsets of a parent graph:

Definition 2.1.3 (Subgraph)**.** Let $G_1 = (\mathcal{V}_1, \mathcal{E}_1, \mathcal{L}^v, \mathcal{L}^e)$ and $G_2 = (\mathcal{V}_2, \mathcal{E}_2, \mathcal{L}^v, \mathcal{L}^e)$ bet two labelled graph. G_1 is a *subgraph* of G_2 if the following hold:

1. $\mathcal{V}_1 \subseteq \mathcal{V}_2$
2. $\mathcal{E}_1 \subseteq \mathcal{E}_2 \cap \mathcal{V}_1 \times \mathcal{V}_1$

when $\mathcal{E}_1 = \mathcal{E}_2 \cap \mathcal{V}_1 \times \mathcal{V}_1$ holds in place of #2, G_1 is known as *induced* subgraph of G_2.

In Definition 2.1.3, the subgraph G_1 is assumed to keep the same node and edge labels of the parent subgraph G_2. The sets $\mathcal{L}^v$ and $\mathcal{L}^e$ are the nodes and edges label[2] domains whose elements are defined with arbitrary data structures. For example, nodes can be equipped with vectors such that $\mathcal{L}^v \subseteq \mathbb{R}^2$ and edge set is neglected, hence $\mathcal{L}^e = \emptyset$. In this case, elements in $\mathcal{L}^v$ may represents a specific position of the node in a $\langle x, y \rangle$-plane, offering the chance to visualize the graph in the Cartesian coordinate system. Usually, the number of nodes $q = |\mathcal{V}|$ of the graph G is referred to as *order* (or the *size*) of G. Notable special cases of graphs can be recognized:

[2]The term "feature" is also adopted to indicate the properties annotated on nodes and edges. Often, it is preferred to "label" term especially in node classification problems in order to avoid ambiguity between the target label of the nodes and its properties.

- *unlabelled* graphs arise when both nodes and edges have no attributes, that is $\mathcal{L}^v = \mathcal{L}^e = \emptyset$
- *directed* graphs (also known as *digraph*), where elements of the edge set $\mathcal{E}$ are ordered pairs of nodes. Hence for directed graphs in general $(u, v) \neq (v, u)$ with $u, v \in \mathcal{V}$.

When compared to a feature vector representation, a labelled graph offers two major advantages from a Pattern Recognition point of view:

1. The ability of representing both semantic and topological information of the pattern
2. The flexibility in modeling complexity in data.

Concerning #1, the set of nodes $\mathcal{V}$ identifies the components of a specific structured pattern G according to the definition given in Def. 2.1.2,. Furthermore, the node label space $\mathcal{L}^v$ denotes the semantic level providing a meaningful description of each component. Alongside that, a binary interactions between two entities $u, v \in \mathcal{V}$ are expressed with an edge $(u, v) \in \mathcal{E}$. Consequently, the edge set $\mathcal{E}$ conveys the topological information by collecting all the mutual interactions between the graph nodes. When a set of graphs is organized for training and testing a learning system, different patterns in both $\mathcal{D}^{(\mathrm{tr})}$ and $\mathcal{D}^{(\mathrm{ts})}$ can generally show different sizes (both in terms of number of nodes and edges) keeping unaltered the nature of the input space $\mathcal{X} = \mathcal{G}$. On the contrary, varying the number of features in a vectorial representation of the data sample from n to m implies an alteration of the input domain from $\mathcal{X} = \mathbb{R}^n$ towards $\mathcal{X} = \mathbb{R}^m$. Hence, the flexibility advantage stressed in #2 regards the possibility in adapting each individual pattern with a suitable level of complexity according to a specific number of components (the nodes) and relations (the edges) without affecting the learning process.

A natural question that arises when departing from a vectorial representation in favor of the graph domain regards the evaluation of the similarity between two graphs. In literature, this problem is usually addressed by employing a *graph matching* procedure. Exact graph matching methods are better defined as graph isomorphism procedures that check whether two graphs share the same node sets and corresponding labels while preserving the edge structure [151]:

Definition 2.1.4 (Graph Isomorphism)**.** 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 labelled graphs with node and edge labelling functions respectively μ_1, μ_2 and ν_1, ν_2. A graph isomorphism is a bijective mapping $f : \mathcal{V}_1 \to \mathcal{V}_2$ such that the following hold:

1. $\mu_1(u) = \mu_2(f(u)) \quad \forall u \in \mathcal{V}_1$
2. $e_1 = (u, v) \in \mathcal{E}_1 \implies e_2 = (f(u), f(v)) \in \mathcal{E}_2$ such that $\nu_1(e_1) = \nu_2(e_2) \quad \forall e_1 \in \mathcal{E}_1$
3. $e_2 = (u, v) \in \mathcal{E}_2 \implies e_1 = (f^{-1}(u), f^{-1}(v)) \in \mathcal{E}_2$ such that $\nu_1(e_1) = \nu_2(e_2) \quad \forall e_2 \in \mathcal{E}_2$

G_1 and G_2 are isomorphic if exist an *isomorphism* f between node sets $\mathcal{V}_1$ and $\mathcal{V}_2$.

If on one hand this approach can rely upon well-defined mathematical foundations, it is often ineffective in real-world cases where corrupted or noisy graphs can be found. Inexact graph matching (also referred to as *error-tolerant graph matching*) approaches address this problem by relaxing the constraints and tolerating errors introduced by noise and distortions when measuring the similarity between any two graphs. An intuitive method consists in measuring how much two graphs G_1 and G_2 share according to a notion of maximum common subgraph. Unfortunately, this process involved a subgraph isomorphism that, as opposed to a graph isomorphism, is know to be an NP-complete problem thus not feasible from a computational complexity point of view. Another well-known approach for dealing with inexact graph matching relies on Graph Edit Distances (GED) [63, 151, 44, 166], which can be view as a reformulation of the well-known edit distance for strings, i.e. the Levenshtein distance, in the graph domain.

The GED is a dissimilarity measure d between two graph G_1 and G_2 defined directly into the graph domain $\mathcal{G}$, that is $d : \mathcal{G} \times \mathcal{G} \to \mathbb{R}_0^+$. Intuitively, this method quantifies the dissimilarity between two graphs as the amount of both structural and labels distortions needed to transform G_1 into G_2 according to a set of *edit* operations that comprise insertions, deletions, and substitutions of both nodes and edges.

Definition 2.1.5 (Edit Path)**.** A possible sequence of M edit operations $(\epsilon_1, \dots, \epsilon_M)$ which enables the transformation of G_1 in G_2 is called *edit path.*

According to Def. 2.1.5, let also $\Omega\,(G_1, G_2)$ be the set of all edit paths between G_1 and G_2. In order to find the most suitable edit path among $\Omega\,(G_1, G_2)$, each edit operation ϵ is associated with a specific cost function $c(\epsilon)$ that evaluate the magnitude of the corresponding operation. The idea of such a cost is to define whether or not an edit operation ϵ represents a strong modification of the graph [151]. The GED between two labelled graphs G_1 and G_2 can be defined as follow:

Definition 2.1.6 (Graph Edit Distance)**.** Let $G_1 = (\mathcal{V}_1, \mathcal{E}_1, \mathcal{L}^v, \mathcal{L}^e)$ and $G_2 = (\mathcal{V}_1, \mathcal{E}_1, \mathcal{L}^v, \mathcal{L}^e)$ be two labelled graphs. Let also $(\epsilon_1, \dots, \epsilon_M)$ be a generic sequence of M edit operations that turns G_1 in G_2. The GED defines the dissimilarity $d : \mathcal{G} \times \mathcal{G} \to \mathbb{R}_0^+$ between G_1 and G_2 as:

$$d(G_1, G_2) = \min_{(\epsilon_1,\dots,\epsilon_M)\in\Omega(G_1,G_2)} \sum\nolimits_{i=1}^{k} c\left(\epsilon_i\right) \tag{2.6}$$

The exact computation of Eq. (2.6) is generally approached by employing specific searching methods. For example, A* is a best-first search algorithm able to always provide a solution (if exists) without overestimating the cost necessary to meet the target thanks to a suitable organization of the solution space in an ordered tree-like fashion. Unfortunately, the main drawback of such procedures is the exponential computational complexity with respect to the number of nodes involved, thus limiting the chance to employ GED only for small graphs. Indeed, the exact computation of the GED is known to be an NP-Hard problem [201] hence most of the strategies deployed in real-world applications are based on suboptimal solutions of Eq. (2.6).

2.2 Mainstream Methods

In general, problems on graphs can be broadly divided into the following families [206]:

- node-level tasks: the input data is a graph (which is supposed to be partially known) and the learning task regards node classification, node regression or node clustering. In classification and regression approaches, the model has to predict respectively the class of a node and attributes from continuous values. In node clustering, the goal is to partition the graph into subgraphs (clusters) so that the nodes in the same cluster are 'close' to each other than to those in other clusters.

- edge-level tasks: as in the previous case usually the input data is a single graph and the learning task focuses on edge classification or link prediction, where respectively the model requires to predict the edge attributes or whether an edge exists between two given nodes.

- graph-level tasks: in this case, the problem is defined by a pair of datasets (namely the training and the test set), where each record is a graph, each associated with a given class label. The tasks, i.e. classification and regression, consist in synthesizing a model relying on training data in order to classify or predict numerical attributes of previously unseen graphs belonging to the test set. An example of a classification task is the prediction of a particular molecule activity against a pathogen, whilst for regression task is the prediction of molecular physical properties such as solubility.

In this section, the mainstream pattern recognition approaches for graph-level tasks are discussed being the core of this work. Four main groups have been identified whose relevant characteristics are discussed in the following sections.

2.2.1 Custom Dissimilarities in the Input Domain

A variety of efficient pattern recognition techniques can not be straightforwardly employed in the graph domain due to the lack of fundamental mathematical operations. Notable examples are Bayes Classifiers and Multi Layer Perceptron which leverage on the possibility in multiplying patterns by a constant and/or the summation between elements. On the other hand, the availability of a dissimilarity measure defined directly on the graph domain mitigates this issue enabling the application of particular supervised and unsupervised learning techniques.

The k-Nearest Neighbor (k-NN) [172, 47] is a training-less classifier that only requires the definition of dissimilarity between data in the input space $\mathcal{X}$ for making its decisions about unseen patterns. Given a set of training data $\mathcal{D}^{(\mathrm{tr})}$, k-NN evaluates the pairwise distances between a test pattern $x^{(ts)} \in \mathcal{D}^{(ts)}$ that has to be classified and every sample in $\mathcal{D}^{(\mathrm{tr})}$. Then, the classification is performed by assigning to $x^{(ts)}$ the most frequent label among its k nearest training patterns. Formally speaking, let $\mathcal{N} = \{(x_1, y_1), \ldots, (x_k, y_k)\} \subseteq \mathcal{D}^{(tr)}$ be the set of k nearest patterns to $x^{(\mathrm{ts})}$ whose distances are evaluated according to a dissimilarity measure $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_0^+$ defined in the input space $\mathcal{X}$. Then, the k-NN classifier $\hat{P} : \mathcal{X} \to \mathcal{Y}$ is defined as follows:

$$\hat{P}(x^{(\text{ts})}) = \underset{y \in \mathcal{Y}}{\arg\max} \left|\{(x_i, y_i) \in \mathcal{N} : y_i = y\}\right| \tag{2.7}$$

Hence, k-NN must be informed only about the dissimilarity measure d available in the input domain in order to accomplish the classification. From this point of view, k-NN is a straightforward choice for classification in the graph domain as there exist a great number of dissimilarity measures (such as GED-based methods defined in Def. 2.1.6) while well-defined algebraic operations are missing [155].

Following the same rationale of the discussion above, custom dissimilarity methods are very appealing in the context of unsupervised learning. A large class of clustering methods relies specifically on two distinct concepts: dissimilarity measure for grouping similar patterns together in the same cluster and a cluster representative, that is a prototype pattern that synthesize the group characteristic. Probably the most used technique in the vector domain is the k-means algorithm where patterns $\mathbf{x} \in \mathbb{R}^n$ are assigned to the nearest cluster C evaluated according to Euclidean distance, whose representative is the *centroid* element $\overline{\mathbf{x}}$:

$$\overline{\mathbf{x}} = \frac{1}{|C|} \sum_{\mathbf{x} \in C} \mathbf{x} \tag{2.8}$$

On the other hand, the application of k-means for graph-related problems is not an option: both the multiplication of a graph by a constant and the summation of graphs are not valid operations in the graph domain being a non-metric space. Hence, the definition of a cluster representative is a delicate issue to consider in this scenario. A possible solution is to take advantage of MinSOD (MinSOD) representative [52, 97]. Formally:

Definition 2.2.1 (MinSOD)**.** Let $\mathcal{D} \subset \mathcal{X}$ be a finite set of elements and let $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_0^+$ a dissimilarity measure in the input space $\mathcal{X}$. The MinSOD representative element $x^* \in \mathcal{D}$ is evaluated as:

$$x^* = \underset{x_i \in \mathcal{D}}{\arg\min} \sum_{x_j \in \mathcal{D}} d\left(x_i, x_j\right) \tag{2.9}$$

From Eq. (2.9), it is possible to spot that the MinSOD does not depend on any mathematical operation other than a suitable dissimilarity measure in the input space. Whenever the MinSOD is chosen as cluster representative in place of centroid in k-means, the clustering procedure becomes available also in the graph domain and is usually referred to as k-medoids. Notably, any other clustering method which is based on distance notion and a cluster representative can be extended without much effort in the graph domain. The class of free clustering methods encompasses typical algorithmic solutions that follow the aforementioned rationale. Basic Sequential Algorithmic Scheme (BSAS) [172], as opposed to k-mean/k-medoid, does not need any prior assumption on the number of clusters to find in the dataset leveraging instead on a threshold of inclusion which establishes whether or not a pattern can be part of a specific cluster according to the pattern-to-representative distance.

2.2.2 Graph Neural Network Methods

Graph neural network architectures [189, 205, 187, 29, 164, 204] typically implement convolution layers resembling the well-known convolutional neural networks for image recognition applications. In this case, convolutional neural networks leverage trainable localized filters which scan the images in order to extract high-level features by analyzing pixel relationships in the grid-like structure. On the other hand, the caveat with graphs is their generally irregular structure that makes the convolution operation a challenging aspect [204]. One approach consists in defining the convolution operation in the spectral domain according to eigenvalues and eigenvectors of the graph Laplacian matrix [51, 31, 22]. Let $\mathbf{L}$ be the normalized Laplacian matrix defined as [25]:

$$\mathbf{L} = \mathbf{I}_n - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} \tag{2.10}$$

where $\mathbf{I}_n \in \mathbb{R}^{n\times n}$ is the identity matrix and $\mathbf{D} \in \mathbb{R}^{n\times n}$ and $\mathbf{A} \in \mathbb{R}^{n\times n}$ are respectively the *degree* and *adjacency* matrix of a graph G with $|\mathcal{V}| = n$. Let also be $\mathcal{L}^v = \mathbb{R}^d$ and $\mathcal{L}^e = \emptyset$, then let $\mathbf{x} \in \mathbb{R}^d$ be a *node features* vector defined for a vertex $v \in \mathcal{V}$ also known as *graph signal*[3]. The convolution operation $\mathbf{f} * \mathbf{x}$ between a filter $\mathbf{f}$ and the graph signal $\mathbf{x}$ in the spectral domain is then defined as follow:

$$\mathbf{f} * \mathbf{x} = \mathbf{U}\mathbf{f}\mathbf{U}^\top\mathbf{x} \tag{2.11}$$

where $\mathbf{U}$ is the matrix whose columns are the eigenvectors of $\mathbf{L}$. All Graph Convolutional Neural Networks spectral methods share the definition of convolution operation provided in Eq. (2.11) differentiating each other on how the filter $\mathbf{f}$ (whose coefficient are learned by the network) is implemented [40, 206].

However, it is still nontrivial to transfer the spectral-based graph convolutional network models learned on one graph to another graph whose eigenfunctions are different. To address this issue, filtering and convolution operations are generalized in spatial-based approaches. Typically, these methods perform recursive steps of node features propagation towards the neighbor vertices according to the topological structure. Then, each node aggregates all the information gathered from its neighbors in order to evaluate its own new feature vector [189, 85, 130]. Formally, the node feature representation $\mathbf{x}_i^{(l+1)}$ of i^{th} vertex in the $l+1$ layer is evaluated as follow [7]:

$$\mathbf{x}_i^{(l+1)} = \Gamma^{(l+1)}\left(\mathbf{x}_i^{(l)}, \Psi\left(\left\{\gamma^{(l+1)}\left(\mathbf{x}_j^{(l)}\right)|j \in \Omega(i)\right\}\right)\right) \tag{2.12}$$

where $\Gamma^{(l+1)}$ and $\gamma^{(l+1)}$ are arbitrary non-linear transformations of the input data e.g. multi-layer perceptron, Ψ is a permutation invariant function e.g. sum, mean, or max, and $\Omega(i)$ is the neighbor indices of node i. In Eq. (2.12) it is possible to spot two relevant operations: $\Psi\left(\left\{\gamma^{(l+1)}\left(\mathbf{x}_j^{(l)}\right)|j \in \Omega(i)\right\}\right)$ resembles an aggregation phase where the information about the neighbor node features are combined together according to Ψ in a message $\mathbf{m}_{\Omega(i)}$. Later on, the update phase is carried on by combining the current node feature vector $\mathbf{x}_i^{(l)}$ with the message $\mathbf{m}_{\Omega(i)}$ according to

[3]For the sake of simplicity, the discussion is limited to graph with only node labels, albeit in principle it can be extended to a more general formulation addressing also edge labels [7]

$\Gamma^{(l+1)}$. The whole graph representation in the geometric space can then be obtained through pooling operations by combining in a suitable way the representation vectors of all nodes in the graph [102].

2.2.3 Implicit Graph Embedding

Implicit graph embedding is the embedding technique that leverages on the idea of *kernel* methods. Instead of representing individually each pattern directly into a vector space, implicit methods encode the information about data into the pairwise similarities. In other words, a kernel $\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}_0^+$ is a real-valued similarity function that represent the input space $\mathcal{X}$ in an implicit fashion according to the kernel matrix $\mathbf{K}$:

Definition 2.2.2 (Kernel Matrix)**.** Let $\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}_0^+$ be a kernel function and let $\mathcal{D} \subset \mathcal{X}$ be a finite set of patterns such that $N = |\mathcal{D}|$. The kernel matrix $\mathbf{K}$ is an $N \times N$ matrix that reads as:

$$\mathbf{K} = \begin{bmatrix} \kappa_{11} & \kappa_{12} & \dots & \kappa_{1N} \\ \kappa_{21} & \kappa_{22} & \dots & \kappa_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \kappa_{N1} & \kappa_{N2} & \dots & \kappa_{NN} \end{bmatrix} \tag{2.13}$$

where κ_{ij} with $1 \leq i, j \leq N$ is the kernel function $\kappa(x_i, x_j)$ evaluated between $x_i, x_j \in \mathcal{D}$

According to the data representation provided by $\mathbf{K}$ as defined in Eq. (2.13), pattern recognition methods can be reformulated in terms of kernel matrix (or more in general in terms of dot product as it will be clear soon) in order to operate on the pairwise kernel values in place of the pattern itself. Such methods are extensively applied in a large set of both supervised and unsupervised learning techniques such as Support Vector Machines (SVM) for classification tasks and k-means and Principal Component Analysis (PCA) for clustering and dimensionality reduction [75, 88, 170, 106].

When a specific kernel satisfies the positive definite condition they are usually called *valid* kernels:

Definition 2.2.3 (Positive Definite Kernel)**.** A kernel function $\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}_0^+$ is positive definite if, and only if, $\forall N \in \mathbb{N}$ the following hold:

$$\sum_{i,j=1}^{N} c_i c_j \kappa(x_i, x_j) \geq 0 \tag{2.14}$$

for all $c_i, c_j \in \mathbb{R}$ and for all $x_i, x_j \in \mathcal{X}$

The importance of valid kernels in pattern recognition is highlighted by a relevant result which is usually referred to as *kernel trick* [172, 24]:

Theorem 2.2.1. Let $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_0^+$ be a valid kernel defined on the input space $\mathcal{X}$. There exists a possibly infinite-dimensional Hilbert space [4] $\mathcal{F}$ and a mapping function $\phi : \mathcal{X} \to \mathcal{F}$ such that:

$$\kappa(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle \tag{2.15}$$

for all $x_i, x_j \in \mathcal{X}$ where $\langle \cdot, \cdot \rangle$ denotes the dot product in a Hilbert space $\mathcal{F}$.

Theorem 2.2.1 really encloses the essence of implicit graph embedding and thoroughly explains the power of kernel methods: every valid kernel κ resembles a dot product $\langle \cdot, \cdot \rangle$ in some (implicitly existing) feature space $\mathcal{F}$ also known as *Reproducing Kernel Hilbert Space*. Hence, knowing an explicit representation of the pattern in such space according to the mapping function $\phi(\cdot)$ is unnecessary from the point of view of kernel machines since it is sufficient to evaluate only the kernel function $\kappa(\cdot, \cdot)$ in the input space $\mathcal{X}$ that in turn corresponds to the inner product in $\mathcal{F}$. Eventually, it is possible to say that the Hilbert space $\mathcal{F}$ is not known explicitly but rather through the concept of dot product induced by the kernel function κ [155].

So far, no prior assumption has been made regarding the nature of the input domain $\mathcal{X}$ for what concerns the validity of the kernel trick. In fact, Theorem 2.2.1 clearly relies on a simple yet restrictive hypothesis, that is kernel function κ must be part of the class of positive definite kernel as defined in Def. 2.2.3. Hence, the validity of Theorem 2.2.1 is still intact also in the graph domain (and in all other structured domains) making available a large set of algorithmic tools for performing graph classification and graph clustering. Thus, graph kernels are raised as specific similarity functions $\kappa_g : \mathcal{G} \times \mathcal{G} \to \mathbb{R}_0^+$ evaluated in the graph domain.

2.2.4 Explicit Graph Embedding

In Section 2.2.3, graph kernel methods have been introduced as possible solutions for using kernel machines in the graph domain by implicitly mapping a graph in a Hilbert space $\mathcal{F}$. The implicit term stems from the fact that target space $\mathcal{F}$ is not known but its existence is guaranteed by Th. 2.2.1 assuming κ_g a valid kernel satisfying Eq. (2.2.3). On the other hand, the possibility to explicit map graphs towards a vector space automatically makes available all the arsenal of pattern recognition techniques without limitations. Formally speaking, an (explicit) graph embedding consists in designing a mapping function $\phi : \mathcal{G} \to \mathbb{R}^n$ which maps graphs from the input domain $\mathcal{X} = \mathcal{G}$ towards a suitable vector space $\mathbb{R}^n$. In general, the most challenging aspect concerns the realization of ϕ in order to generate an embedding space semantically sound with respect to the original graph domain. In other words, an ideal embedding process would be able to preserve entirely both the topological and semantic information into $\mathcal{F}$. In practice, mapping entities from a structured toward a vector domain is by no means a lossless operation. In this section, different embedding techniques are presented underlining the approaches pursued for preserving the largest amount of information in the embedding process.

[4]Hilbert spaces can be roughly defined as a generalization of vector spaces with possible infinite dimension equipped with a suitable notion of dot product allowing the definition of mathematical concept such as angles and lenghts.[134]

Feature Engineering

The most natural way to construct explicit embedding spaces is to start from numerical features extracted from the graph dataset (e.g., entropy, centrality, modularity, enumeration of graph properties), which in turn can be combined in order to build a handcrafted mapping function ϕ. Although these feature generation-based methods can be very effective, they often either require a deep knowledge about the domain and the problem at hand or undergo computationally expensive trial-and-error stages in order to select the most informative numerical features. Examples of explicit feature-engineered embedding procedures exploit statistics of nodes' attributes [70], histograms encoding topological and semantic attributes [101], lexicon matching [168], topological data analysis [119], spectral properties [116] and community embedding [37].

Dissimilarity space embedding

This approach stems from the seminal works of Pekalska and Duin [59] that introduced the concept of dissimilarity representation. In pattern recognition problems, the common way for representing patterns is to collect a set of measurements for a specific object and represent it as a point into a vector space whose coordinate determines the numerical values of such features. As opposed, a dissimilarity representation proposes to represent a given object according to the pairwise dissimilarities between other objects. That is, a dissimilarity measure is firstly imposed for the problem at hand and thereafter each object is represented as the vector whose components are individually the distances between all the objects in the dataset. In light of this discussion, the feature vector approach can be seen as an absolute representation while the dissimilarity representation can be interpreted as a relative representation of the underlying patterns [155, 95].

The extension of such approach for graph domain is simple and intuitive: a single graph is represented as a vector with n components evaluated as the pairwise dissimilarity from a reference set of n objects, usually know as *prototypes*. Formally:

Definition 2.2.4 (Dissimilarity Embedding)**.** Let $G \in \mathcal{D}$ be a labelled graph such that $\mathcal{D} \subset \mathcal{G}$ is a finite set of $N = |\mathcal{D}|$ graphs. Let $\mathcal{P} = \{p_1, \dots, p_n\} \subseteq \mathcal{D}$ be the set of prototype graphs with $n \leq N$. The mapping function $\phi^{\mathcal{P}} : \mathcal{G} \to \mathbb{R}^n$ is defined as:

$$\phi^{\mathcal{P}}(G) = [d(G, p_1), \dots, d(G, p_n)] \tag{2.16}$$

where $d(G, p_i)$ is any graph dissimilarity measure between graph G and the i^{th} prototype graph.

From the above definition, it is clear that the graph G is represented as a vector whose component values are defined as the distances between a specific prototype $p \in \mathcal{P}$ and the original graph G according to a notion of dissimilarity $d : \mathcal{G} \times \mathcal{G} \to \mathbb{R}_0^+$. Hence, any kind of dissimilarity measure defined in the graph domain can be exploited for the embedding phase such as the GED defined in Eq. (2.6).

The identification of a suitable prototype set $\mathcal{P}$ is an important aspect that must be faced in dissimilarity space embedding methods. As can be observed from Eq. (2.6), the prototype set cardinality determines the dimensionality n of the resulting embedding space which is known to be a crucial aspect that can affect the performance of pattern recognition algorithms. A naive approach that can be pursued consist in electing all graph from $\mathcal{D}$ to prototype, that is $|\mathcal{P}| = N$. When dealing with large sets $\mathcal{D}$, graphs will be embedded in high-dimensional spaces meeting the well known *curse of dimensionality* in learning system. For such reason, most of the state-of-the-art method focuses on heuristics that empirically suggest how to pick enough elements from the graph set $\mathcal{D}$ in order to reflect the underlying distribution of data and at the same time removing unnecessary, redundant and, in general, uninformative prototypes.

Embedding via Information Granulation

In the fields of Computational Intelligence and Soft Computing, Granular Computing (GrC) emerged as a novel paradigm able to deal with complex systems inspired by the human abilities to unravel complex situations in environments characterized by uncertainties and with limited knowledge [77, 194]. Indeed, the process of granulation can be referred to as the set of techniques that leads to the emergence of meaningful aggregated data at different levels of abstraction known as *information granules.* As a human-thinking problem-solving process, the GrC approach consists in observing and considering the problem at various levels of granularity, retaining only those that are relevant for the task at hand, therefore discarding unnecessary and superfluous information and make the problem tractable for decision making activities [139].

The importance of information granules resides in the ability to underline properties and relationships between data aggregates. These entities can be synthesized according to the so-called *indistinguishability rule*, that is, elements that share enough similarity, structural or functional properties can be condensed into the same group [200], with the goal to pursue a semantic discrimination of the information residing in the data at hand [137].

Furthermore, data can be represented using different levels of 'granularity' and thus different peculiarities of the considered system can emerge [140, 178, 190, 196, 195]. The selection of the most adequate granularity level is one of the most important issue when designing GrC-based systems, being strongly influenced by nature of the problem. Indeed, by varying the resolution at which the problem is observed, the level of abstraction varies accordingly: the higher the resolution, the less the level of abstraction and finer details emerge. Conversely, low resolution or low granularity levels correspond to high level of abstraction where less, but more populated information granules are likely to emerge.

Different approaches can be considered in order to accomplish the process of information granulation. Notable frameworks represent information granules as mathematical entities relying on set theory: fuzzy sets, rough sets and probabilistic set [135, 58, 92, 203]. A straightforward method for the synthesis of meaningful information granules can be found amongst unsupervised learning methods which have been widely explored in the context of GrC [136, 138, 143]. Indeed, clustering algorithms have a direct connection with the concept of 'granules-as-groups', since these methods use to combine similar data into the same cluster according to a notion of proximity. Notwithstanding that, clustering methods must be properly designed in order to unravel groups and regularities in a multi-perspective view according to the GrC principles. Typically, three main factors may affect the resulting data partitioning from a GrC viewpoint [56]:

- (dis)similarity measure, which serves as the main function in order to determine the degree of proximity between data elements
- threshold of inclusion, which determines whether a given pattern can be included in a specific group (cluster) according to the level of (dis)similarity
- cluster representative, which is the pivotal element that compresses the information contained in a cluster.

A typical clustering algorithm that directly relies on the aforementioned parameters is the BSAS algorithm that performs a so-called 'free clustering procedure'. When the input space of the problem corresponds to the graphs domain, the above discussed clustering parameters must be carefully tuned in order to deal with structured nature of data involved. As already discussed in Section 2.2.1, an effective approach involves the MinSOD (see Eq. (2.9)) as the representative element of a cluster, since its evaluation can be performed just in light of the pairwise dissimilarities between the patterns belonging to the cluster itself, overcoming the lack of algebraic structures which characterize non-geometric spaces.

The clusters representatives from the resulting partition, synthesized in GrC fashion, can be considered as symbols belonging to an *alphabet* $\mathcal{A} = \{s_1, ..., s_n\}$. These elements are retained as pivotal elements on the top of which structured data will be mapped in a vector space by means of the *symbolic histograms* paradigm [53], hence performing an explicit graph embedding. It is worth underlining that the alphabet set is composed of information granules elaborated according to an algorithmic implementation of the indistinguishability rule. Hence, they are effectively semantically sound constructs synthesized for the problem at hand which a field expert could examine for gaining more insights about the underlying process [96]. According to symbolic histograms, a graph can be mapped into an embedded space by building an n-length integer valued vector whose i^{th} component counts the number of occurrences of the symbol s_i belonging to the alphabet $\mathcal{A}$ within the graph to be embedded. The resulting embedding space is inherently endowed with well-defined distance measures, such as the Euclidean distance or the dot product, effectively enabling the application of many standard classification systems developed for geometric spaces.

Chapter 3

Granular Approach for Labelled Graphs

3.1 Introduction

In the previous Section 2.2.3, the explicit graph embedding method based on information granulation emerged as an interesting technique from two points of view:

- The ability in automatically extract the pivotal information enabling the graph embedding
- The interpretability of the synthesized embedding model according to the extracted granules

In this chapter, GRanular Computing Approach for Labelled Graph (GRALG) [23] is introduced as a graph classification system based on information granulation graph embedding. Indeed, GRALG relies on the automatic synthesis of information granules extracted from a training set $\mathcal{D}^{(\mathrm{tr})} \subset \mathcal{G}$ collected in a suitable alphabet of symbols [1] $\mathcal{A}$. The symbols are intended as frequent and meaningful subgraphs emerged as representative elements of compacted and populated clusters. In turn, these clusters are the product of a clustering ensemble procedure driven by BSAS algorithm.

At the core of GRALG lies a parametric GED adopted for evaluating the pairwise dissimilarities directly in the graph domain that are required for the granulation process. Alongside, other relevant parameters are involved in the clustering procedure which are rarely know a-priori. For these reasons, the algorithm training phase is carried out in order to automatically determine an optimal embedding space by tuning the crucial parameters involved in the granulation stage for synthesizing an optimal alphabet. The objective function pursued during the optimization is the accuracy of a classifier trained in the emerging embedding space. That is, the graph training set $\mathcal{D}^{(\mathrm{tr})}$ is embedded on a vector space by building the corresponding symbolic histograms and a suitable classifier is trained accordingly. In order to avoid bias and overfitting, a validation set $\mathcal{D}^{(\mathrm{vs})} \subset \mathcal{G}$ such that $\mathcal{D}^{(\mathrm{tr})} \cap \mathcal{D}^{(\mathrm{vs})} = \emptyset$ is embedded as well in the resulting vector space in order to test the classifier accuracy.

[1] Hereinafter, the terms "granule" and "symbol" are used interchangeably.

A second stage of optimization is in charge of compressing the optimized alphabet by removing unnecessary symbols and reducing the complexity of the learned model. The selection criteria, i.e. the optimization objective function, is based on the classification accuracy on the validation set and an additional term which discourages alphabets with high cardinality. After the whole optimization is over, the final optimal alphabet can be exploited for embedding both the training and disjoint test set $\mathcal{D}^{(\mathrm{ts})} \subset \mathcal{G}$ in order to evaluate the final performance of the classification system.

The remainder of this chapter will explain in detail the work flow of the graph classification: in Section 3.2 the definition of the core dissimilarity measure is given. In Section 3.3, the individual description of the building blocks enabling the graph embedding is provided. Finally in Section 3.4.1 and Section 3.4.2, the optimization stages are described thoroughly according to the interaction of the system components.

3.2 Core Dissimilarity in Graph Domain

The core dissimilarity measure adopted in GRALG is a GED. As discussed in Section 2.1, given a set of edit operations defined on nodes and edges (i.e., deletion, insertion and substitution), a GED evaluates the dissimilarity between two graphs as the minimum cost set of operations needed to turn the first graph into the other.A suitable heuristic that considers a suboptimal solution of Eq. (2.6), namely Node Best Match First procedure (nBMF) is adopted in GRALG in order to overcome the impracticability arisen from the intrinsic computational complexity. Let $G_1 = (\mathcal{V}_1, \mathcal{E}_1, \mathcal{L}_v, \mathcal{L}_e)$, $G_2 = (\mathcal{V}_2, \mathcal{E}_2, \mathcal{L}_v, \mathcal{L}_e)$ be two fully labelled graphs with nodes and edges labels set $\mathcal{L}_v$ and $\mathcal{L}_e$ where in general, G_1 and G_2 might have different sizes in terms of both nodes and edges hence $|\mathcal{V}_1| \neq |\mathcal{V}_2|$ and $|\mathcal{E}_1| \neq |\mathcal{E}_2|$. Additionally, let $d_{\Pi^v} : \mathcal{L}_v \times \mathcal{L}_v \to \mathbb{R}$ and $d_{\Pi^e} : \mathcal{L}_e \times \mathcal{L}_e \to \mathbb{R}$ be two custom functions that enable the evaluation of dissimilarities between nodes' and edges' attributes, possibly depending on some real-valued parameter tuple Π^v and Π^e. The procedure is divided in two consecutive routines that evaluate the costs on nodes and edges. The first procedure greedy matches the nodes in graph G_1 with nodes in G_2 according to d_{Π^v}, that is, the first node from $\mathcal{V}_1$ is assigned to the most similar node from $\mathcal{V}_2$. The selected pairs are stored in a set of matched nodes $\mathcal{R}$ and neglected in the next rounds of nodes evaluations. The routine evaluates node operations costs as follows:

- according to d_{Π^v}, each match contributes to the overall node substitution cost
- if $|\mathcal{V}_1| > |\mathcal{V}_2|$, then $|\mathcal{V}_1| - |\mathcal{V}_2|$ counts as node insertions
- if $|\mathcal{V}_1| < |\mathcal{V}_2|$, then $|\mathcal{V}_2| - |\mathcal{V}_1|$ counts as node deletions.

Once the set of matched nodes $\mathcal{R}$ is returned, the second routine takes place by matching induced edges. Specifically, by relying on the set $\mathcal{R}$, the procedure checks whether an edge exists in both $\mathcal{E}_1$ and $\mathcal{E}_2$. For example, let us suppose $(v_1, u_1) \in \mathcal{R}$ and $(v_2, u_2) \in \mathcal{R}$ be two pairs of matched nodes, where $v_1, v_2 \in \mathcal{V}_1$ and $u_1, u_2 \in \mathcal{V}_2$. Let also be $(v_1, v_2) \in \mathcal{E}_1$ an edge of G_1. Then, the procedure checks whether an edge (u_1, u_2) exists in G_2 as well, being u_1 and u_2 the nodes that have been matched to v_1 and v_2, namely the nodes that compose the edge in G_1. In general, the edit operation costs are evaluated as follow:

- if the edge exists on both $\mathcal{E}_1$ and $\mathcal{E}_2$, this counts as an edge substitution and its cost is given by the dissimilarity between edges according to d_{Π^e};
- if the two nodes are connected on G_1 only, this counts as an edge insertion;
- if the two nodes are connected on G_2 only, this counts as an edge deletion.

The overall dissimilarities between nodes and edges, say $d_{\mathcal{V}}(\mathcal{V}_1, \mathcal{V}_2)$ and $d_{\mathcal{E}}(\mathcal{E}_1, \mathcal{E}_2)$, can be defined as:

$$\begin{aligned} d_{\mathcal{V}}(\mathcal{V}_1, \mathcal{V}_2) &= w_{node}^{sub} \cdot c_{node}^{sub} + w_{node}^{ins} \cdot c_{node}^{ins} + w_{node}^{del} \cdot c_{node}^{del} \\ d_{\mathcal{E}}(\mathcal{E}_1, \mathcal{E}_2) &= w_{edge}^{sub} \cdot c_{edge}^{sub} + w_{edge}^{ins} \cdot c_{edge}^{ins} + w_{edge}^{del} \cdot c_{edge}^{del} \end{aligned} \tag{3.1}$$

where c_{edge}^{sub}, c_{edge}^{ins}, c_{edge}^{del}, c_{node}^{sub}, c_{node}^{ins}, c_{node}^{del} are the costs associated to the edit operations on nodes and edges and w_{node}^{sub}, w_{edge}^{sub}, w_{node}^{ins}, w_{edge}^{ins}, w_{node}^{del}, w_{edge}^{del} are the six weights which reflect the importance of each operation individually. For ease of notation, the latter are collected in a tuple $\mathcal{W} \in [0,1]^6$, that is:

$$\mathcal{W} = \left(w_{node}^{sub}, w_{edge}^{sub}, w_{node}^{ins}, w_{edge}^{ins}, w_{node}^{del}, w_{edge}^{del}\right) \tag{3.2}$$

In order to avoid skewness due to the different sizes between G_1 and G_2, dissimilarities in Eq. (3.1) are normalized as follows:

$$\begin{aligned} d'_{\mathcal{V}}(\mathcal{V}_1, \mathcal{V}_2) &= \frac{d_{\mathcal{V}}(\mathcal{V}_1, \mathcal{V}_2)}{\max(o_1, o_2)} \\ d'_{\mathcal{E}}(\mathcal{E}_1, \mathcal{E}_2) &= \frac{d_{\mathcal{E}}(\mathcal{E}_1, \mathcal{E}_2)}{\frac{1}{2}\left(\min(o_1, o_2) \cdot (\min(o_1, o_2) - 1)\right)} \end{aligned} \tag{3.3}$$

and finally:

$$d(G_1, G_2) = \frac{1}{2}\left(d'_{\mathcal{V}}(\mathcal{V}_1, \mathcal{V}_2) + d'_{\mathcal{E}}(\mathcal{E}_1, \mathcal{E}_2)\right) \tag{3.4}$$

For ease of notation, the overall GED parameters, i.e. the nBMF cost weights and the node/edge dissimilarity parameters, are collected in a vector $\mathbf{w}$ which completely define the parametric dissimilarity measure $d_{\mathbf{w}} : \mathcal{G} \times \mathcal{G} \to \mathbb{R}_0^+$ in the graph domain $\mathcal{G}$:

$$\mathbf{w} = [\mathcal{W} \quad \Pi^v \quad \Pi^e] \tag{3.5}$$

3.3 Building Blocks

3.3.1 Substructures Extraction

The purpose of this block is dedicated to the identification of a substructures set $\mathcal{S}$ from a graph set $\mathcal{D}$. Indeed, the *Extractor* block implements the operations needed to expand a graph $G \in \mathcal{D}$ in a set of subgraphs G' in order to compose $\mathcal{S} = \cup_{i=1}^{N} G'_i$, where $N = |\mathcal{D}|$ is the number of graph in the dataset. As it will be clear in next sections, $\mathcal{S}$ is intended as the set of candidate structures to become meaningful granules of information.

The main issue of this procedure is the high memory footprint and computational complexity necessary for expanding the parent graph in all possible substructures. Indeed, a graph with q vertices can have almost $2^{\binom{q}{2}}$ subgraphs [61, 84] making the extraction unfeasible for networks that exhibit high number of nodes and for large graph datasets as well. Often, when dealing with an expansion task, one is interested in enumerating all subgraphs avoiding possible repetitions of the same substructure. Consequently, a test of isomorphism should be implemented in order to decide whether the subgraph just discovered has been found in previous iterations. This aspect further exacerbates the demanding computational complexity due to the NP-completeness required for solving the graph isomorphism problem [150].

For the above reasons, in the first GRALG approach [23], the extraction problem had been relaxed by fixing the subgraphs order with an arbitrary parameter o. That is, instead of trying to decompose the graph in all its possible subgraphs, the procedure limits the exploration to those substructures with number of nodes $q = 1, \dots, o$. On the other hand, the uniqueness of subgraphs in G' has been preserved leveraging on a suitable hash table that keeps track of the previously explored substructures. Along with the cost necessary for repetitively check the subgraph presence in the hash table, this approach still requires an expensive memory usage that hinder the procedure feasibility.

3.3.2 Granulation Technique

The granulation process is carried out after the set of atomic entities $\mathcal{S}$ has been provided by the *Extractor* block as defined in Section 3.3.1. This block aims at building a set of relevant symbols, namely the alphabet $\mathcal{A}$, by means of an unsupervised learning approach performed on the subgraphs set $\mathcal{S}$. In the spirit of GrC, each symbol $s_i \in \mathcal{A}$ is an highly informative mathematical entity following the principle of justifiable granularity related to the specific level of abstraction at which the problem has been observed. In the literature, many authors have given several different definitions of information granule: fuzzy sets, rough sets and shadowed sets [141] are just few examples on how granules can be practically formalized. The method adopted in GRALG is a clustering-based approach for formally defining a granule of information [181, 56].

In principle, there is no limitation about which clustering algorithm has to be employed as granulation method. Nonetheless, free clustering based methods can be very effective for the granulation information since they directly relies on resolution parameters that can be exploited in order to change the granularity level. For the discussed reasons, the BSAS algorithm has been chosen as the core granulation method of GRALG. Furthermore, the sequential scheme followed by BSAS allows to limit the number of pattern-to-representative distance evaluations reducing the overall computational effort. On the other hand, this approach makes the partitioning outcome dependent to different presentation ordering of the patterns. Specifically, BSAS relies on:

- U: the maximum number of allowed clusters

- θ: the dissimilarity threshold for pattern inclusion in the nearest cluster

In particular, the θ parameter impacts on the resolution adopted during the clustering procedure affecting consequently the granularity level of the synthesized symbols. Additionally, the dissimilarity measure $d_{\mathbf{w}}$ serves as the main function in order to determine the degree of proximity between data elements for aggregating patterns in meaningful clusters.

The clustering algorithm will return a partition $\mathcal{P}_\theta = \{C_1, \dots, C_k\}$ emerged by employing a particular θ value. Thanks to a binary search method, an ensemble of partitions $\mathcal{P}_{\theta_1}, \dots, \mathcal{P}_{\theta_h}$ can be generated according to $\theta_1, \dots, \theta_h$ resolution parameters. For every cluster C in the resulting partitions, a cluster quality index $F(C)$ is defined as:

$$F(C) = \eta \cdot \Phi(C) + (1 - \eta) \cdot \Theta(C) \tag{3.6}$$

where the two terms $\Phi(C)$ and $\Theta(C)$ are defined respectively as:

$$\Phi(C) = \frac{1}{|C| - 1} \sum_{g \in C} d(g^*, g) \tag{3.7}$$

$$\Theta(C) = 1 - \frac{|C|}{|\mathcal{S}|} \tag{3.8}$$

where, g^* is the MinSOD element of cluster C and g_i the i^{th} pattern in the cluster. Both the clustering algorithm and Eq. (3.7) rely on the GED dissimilarity measure (i.e. the nBMF) described in Section 3.2. The quality index defined in Eq. (3.6) reads as the linear convex combination between *compactness* $\Phi(C)$ and *cardinality* $\Theta(C)$ as defined in Eqs. (3.7) and (3.8), respectively, where $\eta \in [0, 1]$ weights the importance of the two terms. For all partitions in the ensemble, each cluster is filtered thanks to a given threshold τ_F, which aims at selecting only relevant clusters according to the quality index F. In this way, only well-formed clusters (i.e., compact and populated) contribute to shape the alphabet set $\mathcal{A}$. For ease of notation, the granulation stage parameters $\mathbf{p}$ can be expressed in more compact way:

$$\mathbf{p} = [U \quad \tau_F \quad \eta] \tag{3.9}$$

3.3.3 Embedding with Symbolic Histograms

The Embedding step defines the operations needed to perform the mapping function $\phi : \mathcal{G} \to \mathcal{F}$ from the graph domain towards a feature space $\mathcal{F} \subseteq \mathbb{R}^n$. The block aims at building the vectorial representation of a graph, i.e. the *symbolic histogram*, relying on the alphabet $\mathcal{A} = \{s_1, \dots, s_n\}$ provided by the Granulator. Formally, let G be the graph for which the embedding is required and let $G' = \{g_1, \dots, g_m\}$ be a suitable decomposition of G in atomic units. Hence, G' is the set of subgraphs emerged from G according to the extraction block discussed in Section 3.3.1.

In general, the symbolic histogram $\mathbf{h} \in \mathbb{R}^n$ of graph G can be evaluated by an embedding block which takes as input the alphabet $\mathcal{A}$ and G'. Hence, this block implements the mapping function $\phi^{\mathcal{A}}(G')$ defined as follows:

$$\mathbf{h} = \phi^{\mathcal{A}}\left(G'\right) = \left[occ\left(s_1, G'\right), \ldots, occ\left(s_n, G'\right)\right] \tag{3.10}$$

where the function $occ : \mathcal{A} \times \mathcal{G} \rightarrow \mathbb{N}$ counts the occurrences of $s_i \in \mathcal{A}$ with $i = 1, \ldots, n$ among the set of subgraphs G' that compose G:

$$occ\left(s_i, G'\right) = \sum_{g \in G'} \Gamma\left(s_i, g\right) \tag{3.11}$$

where

$$\Gamma\left(s_i, g\right) = \begin{cases} 0 & \text{if } d_{\mathbf{w}}\left(s_i, g\right) > \zeta \\ 1 & \text{if } d_{\mathbf{w}}\left(s_i, g\right) \leq \zeta \end{cases} \tag{3.12}$$

and $\zeta = \Phi(C_{s_i}) \cdot \xi$ is a symbol-dependent threshold, where C_{s_i} is the cluster whose minSOD is the symbol s_i and ξ serves as a tolerance parameter for scoring the match.

3.3.4 Classification in Embedding Space

After having synthesized the candidate alphabet $\mathcal{A}$, each graph from the dataset $\mathcal{D}$ can be embedded toward the resulting embedding space $\mathcal{F}$ spanned by the symbolic histograms emerged from the application of the mapping function $\phi^{\mathcal{A}}$ as described in Section 3.3.3. In this way, it is possible to build a $|\mathcal{D}| \times |\mathcal{A}|$ embedding matrix $\mathbf{F}$ whose rows are the symbolic histograms related to each graph $G \in \mathcal{D}$.

A natural question that arise is how to determine the quality of the emerging embedding space $\mathcal{F}$ in order to evaluate the goodness of $\mathcal{A}$. The solution explored in GRALG consists in training a classifier c directly in the resulting embedding space and evaluating a performance measure $\mathcal{J} : \mathcal{D} \rightarrow \mathbb{R}$ of c in classifying a validation set. In this way, $\mathcal{J}$ can be interpreted as a critic asserted by c about the effectiveness of the embedding space $\mathcal{F}$.

As already discussed, the graph embedding strategies have the undoubted advantage to enable a plethora of well-established pattern recognition tools such as SVM, Neural Networks, Random Forests and many others. Hence in GRALG as well, the choice of the classifier c that will be used is completely arbitrary.

3.4 Automatic Learning Graph Representation with Evolutionary Algorithm

In Section 3.3, the building blocks for embedding a set of graph according to the symbolic histograms paradigm have been thoroughly described. In the discussion, the granulator (see Section 3.3.2) and the embedder blocks (see Section 3.3.3) emerge as key components for correctly embedding the graph into a feature space $\mathcal{F}$. Nonetheless, the granulator block relies on the clustering algorithm ability in discovering effective group of data at different levels of abstraction which is strongly influenced by the power of the dissimilarity measure in capturing the proximity of semantically closed patterns, whilst the embedding phase relies on the same dissimilarity measure which in this case serves for evaluating whether there is a match between a symbol and a subgraph. Consequently, the selection of suitable parameters involved in these phases is crucial for building semantically sound embedding spaces with respect to the input space, i.e. the original graph domain. In GRALG, this task is achieved with two sequential automatic procedures based on a genetic algorithm: a first alphabet optimization select the crucial parameters for each block separately and a features selection optimization selects only relevant symbols for the problem at hand. The procedures relies on three disjoint sets $\mathcal{D}^{(\mathrm{tr})}$, $\mathcal{D}^{(\mathrm{vs})}$ and $\mathcal{D}^{(\mathrm{ts})}$ namely training, validation and test set.

3.4.1 Alphabet Synthesis

Each individual from the evolving population considers the set of subgraphs $\mathcal{S}^{(\mathrm{tr})}$ extracted from $\mathcal{D}^{(\mathrm{tr})}$ and runs several BSAS procedures with different threshold values $\theta_1, \ldots, \theta_h$. Regardless of the θ value under analysis, at most $U \in \mathbf{p}$ clusters can be discovered in each run and the dissimilarity between graphs is evaluated using the nBMF procedure as in Section 3.2 by considering the six weights $\mathcal{W} \in \mathbf{w}$ and the parameters $\Pi^v, \Pi^e \in \mathbf{w}$. It is worth recalling that the latter is only applicable if the vertices and/or nodes dissimilarities are parametric themselves. At the end of the clustering procedures, each cluster is evaluated thanks to the quality index in Eq. (3.6) using the parameter $\eta \in \mathbf{p}$ for weighting the convex linear combination. Hence, clusters whose F value lie above $\tau_F \in \mathbf{p}$ are discarded and their representatives will not form the alphabet.

Once the alphabet $\mathcal{A}$ is synthesized, the *Embedder* (Section 3.3.3) receives the substructure sets $\mathcal{S}^{(\mathrm{tr})}$ and $\mathcal{S}^{(\mathrm{vs})}$ and exploits the alphabet $\mathcal{A}$ in order to map both the training set and the validation set towards the metric space $\mathcal{F}$. Eventually, the procedure leads to the generation of the training and validation sets vectorial representation, i.e. $\mathbf{F}^{(\mathrm{tr})}$ and $\mathbf{F}^{(\mathrm{vs})}$, respectively an $|\mathcal{D}^{(\mathrm{tr})}| \times |\mathcal{A}|$ and an $|\mathcal{D}^{(\mathrm{vs})}| \times |\mathcal{A}|$ matrix, whose rows are the symbolic histograms of each graph in $\mathcal{D}^{(\mathrm{tr})}$ and $\mathcal{D}^{(\mathrm{vs})}$. It goes without saying that in the subgraph-symbol matching operation defined in Eq. (3.12), the same GED previously used for BSAS will be deployed along with the corresponding $\mathbf{w}$ parameters.

The definition of the genetic code stems from the previous discussion: both granulator and the embedder leverage on the dissimilarity measure $d_{\mathbf{w}}$ parametric in $\mathbf{w}$; the granulator instead requires also the fine tuning of additional parameters for the clustering algorithm which are collected in $\mathbf{p}$. Hence the search space for the first genetic optimization reads as:

$$[\mathbf{w} \quad \mathbf{p}] \tag{3.13}$$

Finally, the classifier c can be trained on $\mathbf{F}^{(\mathrm{tr})}$ where its performance measure $\mathcal{J}$ is evaluated as the accuracy in correctly classifying each row of $\mathbf{F}^{(\mathrm{vs})}$. The latter serves as the fitness function f_1 (to be minimized) for the individual itself:

$$f_1 = 1 - \mathcal{J} \tag{3.14}$$

Once the evolutionary strategy is completed, the optimal alphabet $\tilde{\mathcal{A}}$ is retained together with the dissimilarity parameters $\mathbf{w}$ in the optimized genetic code. According to $\tilde{\mathcal{A}}$, it is possible to build the optimal vector representation of $\mathcal{S}^{(\mathrm{tr})}$ and $\mathcal{S}^{(\mathrm{vs})}$, respectively $\tilde{\mathbf{F}}^{(\mathrm{tr})}$ and $\tilde{\mathbf{F}}^{(\mathrm{vs})}$. A schematic representation can be found in Figure 3.1

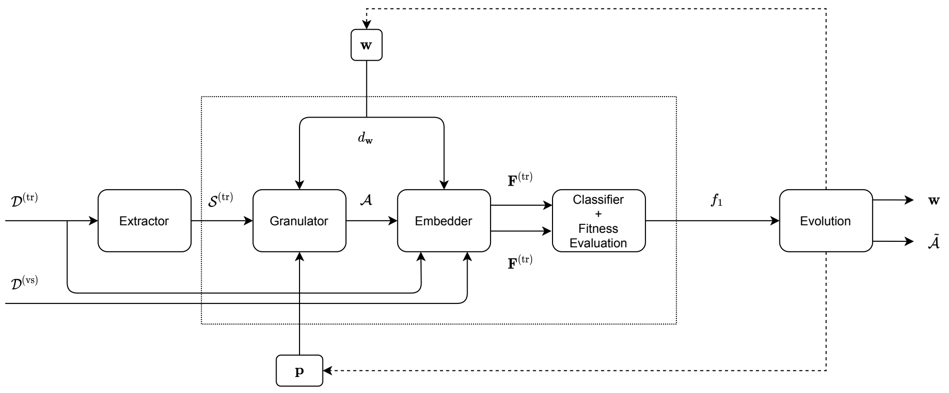


Figure 3.1. Schematic representation of Alphabet Synthesis phase. In the first stage, a subgraph set $\mathcal{S}^{(\text{tr})}$ is extracted from training set $\mathcal{D}^{(\text{tr})}$. The granulator block provides an alphabet $\mathcal{A}$ according to dissimilarity and granulation parameters, respectively $\mathbf{w}$ and $\mathbf{p}$. Both training and validation sets are embedded in corresponding matrices $\mathbf{F}^{(\text{tr})}$ and $\mathbf{F}^{(\text{vs})}$ thanks to the embedder block equipped with $\mathbf{w}$ parameters. The classification performance contributes to the fitness function f_1 evaluation that in turn drives the evolutionary algorithm for the genetic code optimization. At the end, the optimized alphabet $\tilde{\mathcal{A}}$ and the best dissimilarity parameters $\mathbf{w}$ are retained.

3.4.2 Feature Selection

It is not rare that after the alphabet optimization described in the previous section, the cardinality of the optima alphabet set $\tilde{n} = |\tilde{\mathcal{A}}|$ can be very large, that is $\tilde{\mathcal{A}}$ may contain a large number of symbols and thus spanning vectors in high-dimensional spaces since the genetic algorithm is mainly focused on the minimization of the error rate (see (3.14)). A feature selection strategy can be deployed in order to remove uninformative, redundant and in general not necessary features for the classification task. In GRALG, the optimization process is applied on the alphabet set $\tilde{\mathcal{A}}$ in order to select only those symbols that are relevant for the classification problem. The benefits of this additional stage can be summarized in two distinct points:

- enhanced model interpretability: having less pivotal symbols fosters the model interpretability since there will be less symbols to be analyzed by field-experts;
- a faster test phase: as will be stressed in Section 3.5, limiting the alphabet cardinality impacts the number of dissimilarity evaluations needed to embed a test pattern according the symbolic histogram as can be noticed from Eq. (3.10)

A wrapper approach based on an evolutionary algorithm has been designed for this purpose, where the genetic code of each individual is a binary mask $\mathbf{m} \in [0,1]^{\tilde{n}}$ which allows the selection of a subset of features. Hence, each individual:

1. reads $\tilde{\mathbf{F}}^{(\text{tr})}$ and $\tilde{\mathbf{F}}^{(\text{vs})}$
2. according to the 0's in $\mathbf{m}$, the corresponding columns of $\tilde{\mathbf{F}}^{(\text{tr})}$ and $\tilde{\mathbf{F}}^{(\text{vs})}$ are removed, leading to projected matrices $\tilde{\mathbf{F}}'^{(\text{tr})} \in \mathbb{R}^{|\mathcal{D}^{(\text{tr})}| \times n'}$ and $\tilde{\mathbf{F}}'^{(\text{vs})} \in \mathbb{R}^{|\mathcal{D}^{(\text{vs})}| \times n'}$ where $n' = \sum_{i=1}^{\tilde{n}} \mathbf{m}_i$

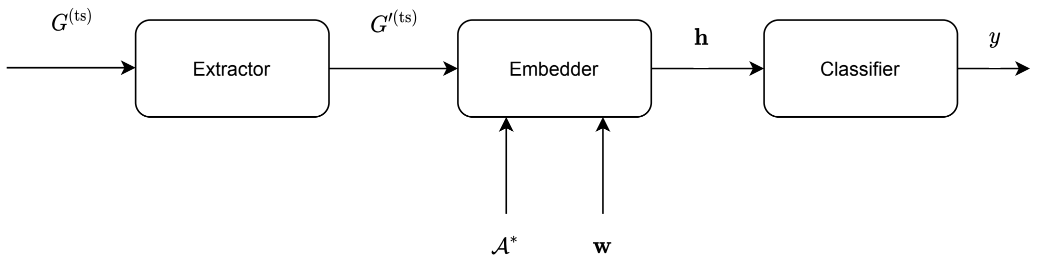


Figure 3.2. Schematic representation of test phase. A previously unseen pattern $G^{(\text{ts})}$ is first decompose in atomic substructures collected in $G'^{(\text{ts})}$. The embedder block provides the symbolic histogram $\mathbf{h}$ for G by leveraging on the optimal alphabet set and dissimilarity parameters, respectively $\mathcal{A}^*$ and $\mathbf{w}$. The classifier trained in the embedding space emits the label y for $G^{(\text{ts})}$.

3. a classifier c is trained on $\tilde{\mathbf{F}}'^{(\text{tr})}$ and its own performance measure $\mathcal{J}$ is computed as the accuracy on correctly classifying $\tilde{\mathbf{F}}'^{(\text{vs})}$

The objective function (to be minimized) is then defined as:

$$f_2 = (1-\sigma)\cdot(1-\mathcal{J}) + \sigma\cdot\frac{n'}{\tilde{n}} \tag{3.15}$$

which reads as a convex linear combination between the error rate $(1-\mathcal{J})$ on the validation set (leftmost term) and the ratio of selected symbols (rightmost term), weighted by a user-defined trade-off parameter $\sigma \in [0,1]$.

Once the optimization is completed, the optimal mask $\mathbf{m}^*$ is retained with $\mathbf{F}^{*(\text{tr})} \in \mathbb{R}^{|\mathcal{D}^{(\text{tr})}|\times n^*}$, $\mathbf{F}^{*(\text{vs})} \in \mathbb{R}^{|\mathcal{D}^{(\text{vs})}|\times n^*}$ as well, where $n^* = \sum_{i=1}^{\tilde{n}} \mathbf{m}_i^*$. Accordingly, the optimal alphabet $\mathcal{A}^* \subset \tilde{\mathcal{A}}$ is created by selecting the features indicated by the 1's in $\mathbf{m}^*$.

3.5 Synthesized Classification Model and Test Phase

From the two genetic optimization procedures, the optimized alphabet $\mathcal{A}^*$, the dissimilarity measure parameters $\mathbf{w}$ and the classifier c trained on $\mathbf{F}^{*(\text{tr})}$ are the main actors which completely characterize the classification model. Let $G^{(\text{ts})} \in \mathcal{D}^{(\text{ts})}$ be a previously unseen graph belonging to a test set $\mathcal{D}^{(\text{ts})}$. It can be classified according to the following procedure (a schematic representation of the test phase is proved in Figure 3.2):

1. Expand $G^{(\text{ts})}$ in the subgraph set $G'^{(\text{ts})}$;

2. Determine the symbolic histogram $\mathbf{h}$ related to $G'^{(\text{ts})}$ by running the *Embedder* block equipped with the optimal alphabet $\mathcal{A}^*$, where $\mathbf{w}$ will be exploited in the symbol-subgraph matching procedure as depicted in Eqs (3.10) and (3.12).

3. The classifier c trained with $\mathbf{F}^{*(\text{tr})}$ is now able to predict a label y for the embedded graph $\mathbf{h}$

It is worth to notice that the classification problem takes place in the embedding space spanned by $\mathbf{F}^{*(\mathrm{tr})}$, that is $\mathbb{R}^{n^*}$. Hence, the test graph $G^{(\mathrm{ts})}$ is coherently embedded according to $\mathcal{A}^*$ giving rise to a symbolic histogram $\mathbf{h} \in \mathbb{R}^{n^*}$. Accordingly, the synthesized feature based classification model leverages on a fast computation scheme since it is driven by a lower number of dissimilarity computations involved in the embedding process. Indeed, the procedure for classifying a single graph $G^{(ts)}$ requires to match $|\mathcal{A}^*|$ symbols with the subgraphs set $|G'^{(\mathrm{ts})}|$ that make up the test graph, leading to $|\mathcal{A}^*| \cdot |G'^{(\mathrm{ts})}|$ dissimilarity computations.

Chapter 4

Studies and Novelties for Granular Graph Embedding

In this chapter, the novelties introduced for GRALG classification system are discussed in depth. In particular, the chapter is organized as follows:

- In Section 4.1, a novel stochastic lightweight method for the subgraph extraction is introduced to mitigate the high computational cost and memory footprint witnessed in the original GRALG implementation due to the exhaustive extraction procedure. Hence, the method concerns the design of a new Extractor module (see Section 3.3.1) which implements a random subgraph sampling according to a specific traversal strategy, i.e. Breadth First Search (BFS) and Depth First Search (DFS). On the other hand, the aforementioned traversal strategies limit the subgraphs extracted to path- and star-like topologies. For this reason, in Section 4.1.1, the extractor block has been further revised in order to sample also clique-based subgraphs.

- In Section 4.2, a class-aware granulation procedure is proposed in order to improve the quality of the synthesized granules of information. Indeed, in the original GRALG version, the granulation method neglects the ground-truth information carried on with the training set $\mathcal{D}^{(\mathrm{tr})}$. Hence, the proposed new Granulator block provides a specific set of symbols which characterized individually the classes of the problem. Additionally, in Section 4.2.1, a description of limitations for the proposed method is provided together with possible solutions.

- In Section 4.3, an evolutionary swarm like optimization algorithm is designed in order to enable a class-specific metric learning strategy that allows learning different dissimilarity measures in the graph domain, each of which is specifically tailored for a particular class of the problem at hand. This method overcomes the main limitation of the GRALG optimization strategy described in Section 3.4.1 which leads to the definition of a single global dissimilarity measure valid for all patterns regardless of the class they belong to.

- In Section 4.4, six different variants of the symbolic histogram embedding method are explored. In particular, the evaluation of the occurrences for the symbolic histogram is relaxed by using three different types of soft functions (*sum*, *mean* and *median*). That is, instead of relying on counting the number of occurrences with an hard-limiting function that triggers the counter whether the dissimilarity measure between the specific prototype and a subgraph is below a given threshold, the proper dissimilarity values is instead considered when searching for granules occurrences within the graph to be embedded, following the rationale behind dissimilarity space embedding [59].

- In Section 4.5, it is investigated a Multi-Objective Optimization approach for the synthesis of a GRALG driven classification model that simultaneously minimizes the misclassification error, the number of features (i.e., information granules) and the structural complexity validated in the embedding space. Indeed, in the original GRALG optimization strategy the 'best' embedding space has always been addressed via single-objective evolutionary optimization driven only by maximizing classifier's accuracy on a validation set projected onto the embedding space. If on one side the learning performance is an essential aspect of a reliable classification system, on the other hand minimizing the number of symbols and the structural complexity lead to models that are likely to be more interpretable and more prone to generalize.

Table 4.1. Summary of novel methods and techniques introduced in GRALG.

Name	**Section**	**Block involved**	**Limitation Addressed**
Stochatic Subgraph Sampling	4.1	Extractor	Running Times and Memory Footprint
Class-Aware Strategy	4.2	Extractor and Granulator	Loss of ground-truth information in granulation
Class Specific Metric Learning	4.3	All	Single dissimilarity measure optimization
Relaxed Symbolic Histograms	4.4	Embedder	Expressiveness of the hard-limit symbolic histogram
MOO optimization	4.5	Optimizer	Considering error rate as the only performance measure for a classification system

4.1 Stochastic Substructures Extraction

The extraction phase discussed in Section 3.3.1 is of utmost importance in granular graph embedding approaches based on GRALG. Indeed, the subgraph set $\mathcal{S}$ is the key component defining the candidate granules of information for the alphabet generation carried out by the granulator block. As already detailed in Section 3.3.1, the extractor block has a primary influence in the computational and memory footprint problems of the whole algorithm which derived respectively from graph isomorphism complexity and the large number of possible subgraphs that can be extracted from a single graph.

A lightweight stochastic extraction procedure has been proposed in [8] addressing the discussed limitation of the exhaustive subgraph extraction. In this novel approach, the problem is further relaxed by fixing the cardinality of the subgraph set $\mathcal{S}$ and admitting simultaneously multiple repetitions of the same subgraphs in $\mathcal{S}$. Hence, the memory issues has been tackled by imposing a limitation on the number of candidate subgraphs to be considered as prospective granules of information, whilst the computational complexity has been coped by giving up the possibility of relying on unique substructures. Two well-known traversal strategies, i.e. BFS and DFS [46] are in charge of exploring the parent graph for building the substructures set.

In this procedure, a graph $G = \{\mathcal{V}, \mathcal{E}, \mathcal{L}^v, \mathcal{L}^e\}$ is randomly drawn from the graph set $\mathcal{D}$ where $\mathcal{V}$ and $\mathcal{E}$ indicate as usual the set of nodes and edges, respectively. Then, a node $v \in \mathcal{V}$ is selected as seed node for a traversal strategy based on either BFS and DFS in order to extract a subgraph $g = \{\mathcal{V}_\text{g}, \mathcal{E}_\text{g}, \mathcal{L}^v, \mathcal{L}^e\}$. Both extractions (graph G from $\mathcal{D}$ and node v from $\mathcal{V}$) are performed with uniform random distribution. Alongside o (maximum subgraph order), a new user-defined parameter W determines the desired cardinality of the set of candidate substructure $\mathcal{S}$ for the granulation phase.

The procedure can be summarized as follow:

1. Initialize $\mathcal{S}$ as an empty set
2. For each candidate subgraph order $q = 1 \dots o$
   (a) Draw a random graph G from $\mathcal{D}$
   (b) Draw a random vertex v from $\mathcal{V}$
   (c) Traverse graph G starting from node root v until q vertices are found
   (d) Collect edges and nodes traversed in a subgraph g
   (e) Append g in $\mathcal{S}$
3. Repeat step #2 until $|\mathcal{S}| = W$

In step #2c, the graph traverse is performed by using one of two following well-known algorithms:

Breadth First Search: starting from a node v, BFS performs a traverse throughout the graph exploring first the adjacent nodes of v and then moving farther only after the neighbourhood is totally discovered. A First-In-First-Out policy is in charge to organize the list of neighbours for the considered vertex, in order to give priority to adjacent nodes. The algorithm can be summarized as follow:

1. Select the starting vertex v.
2. Push v in a queue list $S^{(\text{queue})}$
3. Pop u, the first element of the queue from $S^{(\text{queue})}$
4. For each neighbour t of u, push t in $S^{(\text{queue})}$ if t is not mark as "visited"
5. Mark u as a "visited" vertex
6. Repeat 3-5 until $S^{(\text{queue})}$ is empty.

Depth First Search: in this strategy, a given graph is traversed starting from a seed vertex v, but unlike the BFS search, the visit follows a path with increasing length from v and backtracks only after all the vertices from the selected path are discovered. A Last-In-First-Out policy is in charge to organize the list of neighbours for the considered vertex, in order to visit in-depth vertices first. The steps of the algorithm are:

1. Select the starting vertex v
2. Push v in a stack list $S^{(\text{stack})}$
3. Pop u the last element from stack $S^{(\text{stack})}$
4. For each neighbour t of u, push t in $S^{(\text{stack})}$ if t is not marked as "visited"
5. Mark u as "visited"
6. Repeat 3-5 until $S^{(\text{stack})}$ is empty.

These methods are employed to populate the set of vertices $\mathcal{V}_\text{g}$ and edges $\mathcal{E}_\text{g}$ for the subgraph g: a vertex is added to $\mathcal{V}_g$ as soon as it is marked as "visited", whereas an edge is added to $\mathcal{E}_\text{g}$ by considering the current and the last visited vertices. A complete pseudocode can be found in Algorithm 1

Notes on Subgraph Set for Embedding Phase

The original embedding procedure in GRALG (see Section 3.3.3) used to expand the graph G in atomic substructures up to a defined order, following an exhaustive extraction strategy (alike the *Extractor* block, see Section 3.3.1) and then compare all the obtained subgraphs against all alphabet symbols in $\mathcal{A}$ via the GED dissimilarity. However, this approach still unfeasible both in terms of running time and memory footprint for medium/large graphs datasets hence a lightweight strategy has been employed in order to avoid the issues related to an exhaustive expansion. The algorithm starts either a BFS or DFS traversal according to the traverse strategy selected for the Extractor block. A seed node v belonging to the node set $\mathcal{V}$ is chosen in order to start the exploration for extracting a set of subgraphs up until a given order o. In order to mitigate the number of subgraphs, when a new prospective seed node $u \in \mathcal{V}$ is considered, the procedure firstly checks whether u already appeared in one of the previously extracted subgraphs and eventually neglect it as starting node for the traversal strategy. This procedure finally returns the expanded set of subgraphs G' whose cardinality is reduced with respect to an exhaustive expansion.

Algorithm 1 Lightweight stochastic subgraph extraction

```
procedure EXTRACT(Graph $G = \{\mathcal{V}, \mathcal{E}, \mathcal{L}^v, \mathcal{L}^e\}$, $o$ maximum number subgraph order)
    Initialize subgraph $g = \{\mathcal{V}_\text{g}, \mathcal{E}_\text{g}, \mathcal{L}^v, \mathcal{L}^e\}$ with $\mathcal{V}_g = \emptyset$ and $\mathcal{E}_g = \emptyset$
    Uniform at random extract a vertex $v \in \mathcal{V}$
    Set $v$ as seed node for BFS/DFS strategy
    repeat
        $\{\mathcal{V}_\text{g}, \mathcal{E}_\text{g}\} \leftarrow$ Traverse $G$ using BFS/DFS
    until $|\mathcal{V}_g| = o$
    return $g$
end procedure
```

4.1.1 Clique Extraction

In the stochastic extraction method discussed in Section 4.1, the extractor block has become responsible of the topology that characterized the granules of information. In other words, the subgraph collected in $\mathcal{S}$ as well as the symbols appearing in the alphabet $\mathcal{A}$ will show *star*- or *path*-like topologies according to the traversal strategy adopted respectively BFS and DFS. Even though this point can give flexibility to our approach, the selection of an appropriate traversal algorithm is not trivial and likely influences the performance of a graph classification system driven by finding meaningful subgraphs for graph embedding purposes since the topological properties of the original graphs must be reflected in the subgraphs to be extracted.

In [10], the extraction strategy has been adapted in order to explore only the maximal clique subgraph from all the graph in $\mathcal{D}$ in order to address whether this particular topology can be interpreted as meaningful information granules for synthesizing an embedding space for graph classification purposes. Cliques originate in social sciences indicating groups of individuals who interact with one another and share similar interests [162, 173]. The seminal work [99] 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 $\mathcal{O}(3^{q/3})$ in the worst-case scenario [126] for an q-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 [28].

This method has been employed as core algorithm in the extractor proposed, 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 $\Omega(v)$. 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.

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.

4.2 Class-Aware Granulation

The Granulator block introduced in Section 3.3.2 plays a fundamental role in the whole graph embedding procedure. Indeed, it is responsible in the synthesis of meaningful symbol composing the alphabet set $\mathcal{A}$ upon which the model synthesis occurs. The relevant parameters needed to the granulator are optimized according to a genetic algorithm in order to generate a suitable alphabet of symbols effective for the graph embedding, as already discussed in Section 3.4.1. In this phase, each individual in the population receives the set of candidate substructure $\mathcal{S}^{(\mathrm{tr})}$ from which the information granules are extracted according to their genetic codes. Hence, $\mathcal{S}^{(\mathrm{tr})}$ shall convey the maximum amount of information in order to put the granulator in condition of synthesizing valuable alphabets.

Nonetheless, the granulation block described in Section 3.3.2 has been designed without properly considering all the information contained in the subgraph set $\mathcal{S}^{(\mathrm{tr})}$. Indeed, the training set $\mathcal{D}^{(\mathrm{tr})}$ from which $\mathcal{S}^{(\mathrm{tr})}$ is extracted carries also the information related to the ground-truth classes about patterns, which is completely neglected during the process of granulation, missing the opportunity to exploit information potentially useful for this task. In other words, this a-priori information about the data can be used by the *Granulator* for synthesizing specific symbols for each of the problem-related classes and hopefully improve the overall quality of the alphabet [145]. To this end, the *Extractor* block defined in Section 4.1 has been revisited as well, being the core module in charge of forwarding an appropriate set of subgraphs $\mathcal{S}^{(\mathrm{tr})}$ which is essential for the granulation phase.

The subgraphs extraction has been redesigned in a stratified (class-aware) [14] way that takes into account the frequency of the classes in the training set and, consequently, populate different target sets $\mathcal{S}_l^{(\mathrm{tr})}$ for $l = 1, \dots, L$ with L being the number of classes in the dataset. The frequency measure serves for keeping unaltered the class distribution of subgraphs with respect to that of the starting set $\mathcal{D}^{(\mathrm{tr})}$. This class-aware extraction, described in detail in Algorithm 2, can be summarized as follows:

1. For each class l with $l = 1, \dots, L$ in the dataset $\mathcal{D}^{(\mathrm{tr})}$, evaluate the absolute

frequency f_l, namely the number of patterns of class l, such that $\sum_{l=1}^{L} f_l = |\mathcal{D}^{(\mathrm{tr})}|$;

2. Let W be the user-defined desired cardinality of $\mathcal{S}^{(\mathrm{tr})}$, then evaluate the number of subgraphs to be extracted for each class as $N_l = \frac{f_l}{|\mathcal{D}^{(\mathrm{tr})}|} \cdot W$;

3. Extract N_l subgraphs by performing the stochastic procedure from Section 4.1 on graphs belonging to l-th class only and collect them in $\mathcal{S}_l^{(\mathrm{tr})}$.

4. Repeat from Step #2 for $l = 1, \dots, L$

Algorithm 2 Enhanced Extractor

```
procedure ClassAwareExtr(Graph Set $\mathcal{D} = \{G_1, \dots, G_n\}$, $W$ max size of
subgraphs set $\mathcal{S}_l$, $o$ max order of extracted subgraph)
    $\mathcal{S}_l$: initially empty set of class specific subgraphs
    $L$: number of target classes in the problem
    for class $l = 1 \dots L$ do
        Evaluate class frequency $f_l$
        Compute target number of subgraphs per class $N_l = \frac{f_l}{|\mathcal{D}|} \cdot W$
        while $|\mathcal{S}_l| \leq N_l$ do
            for order $q = 1 \dots o$ do
                Random extract a graph $G$ from $\mathcal{D}$
                if $G$ belongs to class $l$ then
                    $g =$ Extract$(G, q)$                ▷ Extraction as defined in Alg. 1
                    $\mathcal{S}_l = \mathcal{S}_l \cup g$
                end if
            end for
        end while
    end for
    return $L$ class specific subgraph sets $\{\mathcal{S}_1, \dots, \mathcal{S}_L\}$
end procedure
```

After the extraction phase has completed, the class-aware granulation can take place. Recall that $\mathcal{S}_l^{(\mathrm{tr})}$ is the set of subgraphs extracted from graphs from $\mathcal{D}^{(\mathrm{tr})}$ belonging to the l-th class: the *Class-Aware Granulator* performs L different BSAS-driven clustering ensemble procedures by considering the class-specialized set of subgraphs $\mathcal{S}_l^{(\mathrm{tr})}$. In this way, each instance of a clustering procedure outputs his own alphabet, say $\mathcal{A}_l$, whose cardinality may differ for each class. Since in this novel method the clustering algorithms are fed with class-specific subgraphs set, the granulation process should be able in principle to discover more accurate granules of information characterizing in details the class itself. When all of the L granulation stages have been completed, the alphabet $\mathcal{A}$ collects all symbols in the L alphabets $\mathcal{A}_l$ returned so far and the synthesis moves towards the *Embedder* block and the symbolic histograms evaluation thanks to $\phi^{\mathcal{A}}(\cdot)$. A detailed description of the proposed procedure is listed in Algorithm 3.

Algorithm 3 Class Aware Granulator

```
procedure GRANULATE($L$ subgraph sets $\mathcal{S}_l^{(\mathrm{tr})}$, $\boldsymbol{\theta}$ vector of thresholds, $U$ max number of cluster)
    $\mathcal{A}$: initially empty set of overall alphabet
    $\mathcal{A}_l$: initially empty set of class related alphabet
    $g^*$: minSOD subgraph cluster representative
    $L$: number of target classes in the problem
    for all class $l = 1 \dots L$ do
        $partitions = clusteringEnsemble(\mathcal{S}_l^{(\mathrm{tr})}, \boldsymbol{\theta}, U)$
        for all cluster $C$ in $partitions$ do
            if $F(C) \geq \tau_F$ then
                Append $g^*$ in $\mathcal{A}_l$
            end if
        end for
    end for
    Merge all $\mathcal{A}_l$ in $\mathcal{A}$
    return $\mathcal{A}$
end procedure
```

4.2.1 Limitations and Solutions

The main drawback of the *Class-Aware Granulator* is an augmented complexity of the underlying model in terms of cardinality of the symbols alphabet $\mathcal{A}$.It is worth recalling that the BSAS ensemble clustering procedure employed in the *Granulator* block relies on two parameters, U (maximum number of allowed clusters) and $\theta_1, \dots, \theta_h$ (different threshold dissimilarity parameters under which a pattern is included into a cluster determining $\mathcal{P}_1, \dots, \mathcal{P}_h$ partitions). Furthermore, in the optimization phase, the genetic algorithm described in Section 3.4.1 takes care of selecting a suitable value of U in a defined range $U \in [1, U_{max}]$, where U_{max} is user-defined. Since the *Granulator* block generates an ensemble of partitions for each $\theta_1, \dots, \theta_h$, the number of clusters for a single granulation can be at most $\mathcal{O}(h \cdot U_{max})$, where h trivially depends on depth level of the binary-search for the clustering ensemble procedure. Consequently, when the *Class-Aware Granulator* is employed, the cardinality $|\mathcal{A}|$ for the alphabet can be at most $\mathcal{O}(L \cdot h \cdot U_{max})$. This may lead to very high-dimensional embedding spaces, especially when the number of classes of the problem at hand is large, affecting also the computational time required to build the symbolic histograms. In order to face this problem, two different approaches have been investigated in order to reduce the model complexity by bounding the maximum values of U_{max} according to the number of classes involved.

Class-Aware Granulator with Uniform U Scaling

The first method is a simple uniform scaling of U_{max} with respect to the L classes for the classification problem at hand. That is, when the l-th granulation occurs, U is bounded in range $U \in [1, U_{max}/L]$, for all of the L classes in the dataset. Of course, this reduces the prospective number of symbols by shrinking the range of admissible values for U and therefore limits the cardinality of $\mathcal{A}$. In plain words, $[1, U_{max}/L]$ are the genetic algorithm lower and upper bounds for the U parameter involved in Algorithm 3.

Class-Aware Granulator with Frequency-based U scaling

If on one hand the previous approach may work well for balanced dataset, the uniform scaling may not work properly when the classes in dataset are not equally distributed. Indeed, the number of subgraphs to be extracted for each class is proportional to the frequency of the class itself. For this reason, a different strategy is deployed to scale U in a class-aware fashion as well by considering the frequency f_l of the l-th class in the training set:

1. For each class l with $l = 1 \dots L$ in $\mathcal{D}^{(\text{tr})}$, evaluate the absolute frequency f_l such that $\sum_{l=1}^{L} f_l = |\mathcal{D}^{(\text{tr})}|$;

2. For each class l with $l = 1 \dots L$ evaluate the scaled value $U^l_{max} = \frac{f_l}{|\mathcal{D}^{(\text{tr})}|} \cdot U_{max}$;

3. Perform the Class-Aware Granulator for class l with $U \in \left[1, U^l_{max}\right]$

Therefore, each time a partition has to be built, a class-specific value U^l_{max} is selected for the related U range by the *Class-Aware Granulator*. It is worth noting that for this specific method, the length of the genetic code differ from what described in Section 3.4.1. Indeed, when the frequency-based U scaling is employed, we need to optimize L different U values for each *Class-Aware Granulator* instance: this inevitably results in an increased genetic code length by a factor of $L-1$ with respect to the former case (cf. Eq. (4.24)). In this case, the granulator parameter vector reads as follow:

$$\mathbf{p} = [U_1 \dots U_L \quad \tau_F \quad \eta] \tag{4.1}$$

4.3 Class-specific Metric Learning for Graph Embedding

The recognition ability of the graph classification system based on a granular graph embedding approach (i.e. GRALG) introduced in Chapter 3 extensively relies on a suitable notion of parametric dissimilarity $d_{\mathbf{w}}$ working on the graph domain. Indeed, the GED heuristic adopted (see Section 3.2) appears in different crucial aspects of GRALG:

- In the granulation step $d_{\mathbf{w}}$ serves as core method for aggregating similar subgraphs together in order to synthesize meaningful clusters which are essential for composing the alphabet set $\mathcal{A}$ (see Section 3.3.2).

- In the embedding phase depicted in Section 3.3.3, the counting function $occ(\cdot)$ relies on the dissimilarity measure $d_{\mathbf{w}}$ in order to determine whether a subgraph-symbol match occurs (see Eq. (3.12)).

On the other hand, the ability of $d_{\mathbf{w}}$ in correctly capturing the proximity of semantically closed pattern is strongly influenced by the set of GED parameters $\mathbf{w}$ which are typically problem dependent and hardly ever are know a-priori. Deploying an effective strategy that automatically tailors these parameters on the specific data and problem under analysis can positevely impact on the learning abilities of the classification system.

In literature, this task is usually referred to as *metric learning* [19, 177]. Many methods approach the problem from the point of view of mathematical optimization, attempting to find a single global metric that maximize an objective function (see e.g., [38, 167, 191, 197]). In this case, a single instance of the dissimilarity measure parameters is learned and the metric is valid for the whole input space of the model and for all the decision regions (one for each class). More often, the underlying data distribution can be extremely complex and heterogeneous, where a single global metric could not provide enough semantic description. In such situations, learning different metrics from data can help in improving the recognition abilities of the whole system [128, 20, 142]. Generally, this approaches can be classified in two categories:

- local metric learning, where different regions of the input space can be characterized by specific metrics. For example, if decision regions are described and approximated by a set of clusters, each cluster is characterized by a possibly specific instance of the dissimilarity measure parameters;

- class-specific metric learning, where each decision region can be characterized by a specific instance of the dissimilarity measure parameters. The set of clusters describing a decision region associated to a given class label will share the same metric.

Although GRALG already envisages an automatic mechanism for learning suitable dissimilarity measure parameters, it neglects the potential information carried by classes individually. The alphabet optimization procedure described in Section 3.4.1 eventually provides the final genetic code of a single best individual evaluated in the whole evolution in which is encoded the optimal dissimilarity parameter set $\mathbf{w}$ (see Eq. (3.13)), hence it considers only a global solution. The evolutionary strategy can instead be designed ad-hoc in order to equip GRALG with class-specific metric learning capabilities [11]. In a more practical setting, this means that rather than learning a global GED parameters set (see Eq. (3.5)), suitable cost weights $\mathcal{W}$ and dissimilarity measure parameters (Π^v and Π^e) have to be learned in a class-aware fashion:

$$\{\langle \mathcal{W}_1, \Pi^v_1, \Pi^e_1\rangle, \ldots, \langle \mathcal{W}_L, \Pi^v_L, \Pi^e_L\rangle\} \tag{4.2}$$

with L being the number of problem-related classes. Therefore, it is possible to define a class-related dissimilarity measure $d_l : \mathcal{G} \times \mathcal{G} \to \mathbb{R}^+_0$ expressed as follow:

$$d_l\left(G_1, G_2\right) = d\left(G_1, G_2\right)\big|_{\mathbf{w}=\mathbf{w}_l} \tag{4.3}$$

where $\mathbf{w}_l = [\mathcal{W}_l, \Pi^v_l, \Pi^e_l]$ is the set of parameters tailored for the target class l.

4.3.1 Evolutionary strategy for class-specific metric learning

The problem of both local metric learning and optimal alphabet synthesis is faced according to a swarm-like optimization: L swarms (one for each class label) learn concurrently optimal parameters with the goal to individually synthesize a class-aware alphabet.

Initially, each individual in the l-th swarm receives:

- $\mathcal{S}_l^{(\mathrm{tr})}$ namely the set of subgraphs extracted from training patterns belonging to class l.
- a re-labelled version of the training and validation sets, respectively $\mathcal{D}^{(\mathrm{tr})}$ and $\mathcal{D}^{(\mathrm{vs})}$ in a One-vs-All fashion, where the l-th label is considered as positive and all other labels are considered as negative.

This operations enable each group of individuals, i.e. the swarms, in working exclusively on the data related to their own l-th class. In other words, the l-th swarm of individual focuses on the optimization of an alphabet $\mathcal{A}_l$ composed by symbols which are extracted from the class-aware set of candidate subgraphs $\mathcal{S}_l^{(\mathrm{tr})}$. Therefore, each individual in l-th swarm moves on performing the granular graph embedding procedure:

1. a single individual runs the Granulator on $\mathcal{S}_l^{(\mathrm{tr})}$ by exploiting the dissimilarity measure d_l with $\mathbf{w}_l$ and $\mathbf{p}_l$, where $\mathbf{p}_l$ is the class aware critical parameters characterizing the clustering procedure:

$$\mathbf{p}_l = [U_l \quad \tau_l \quad \eta_l] \tag{4.4}$$

Each cluster returned by the l-th Granulator, regardless of its partition, is evaluated by the following internal quality measure $F_l\,(C)$:

$$F_l\,(C) = \eta_l \cdot \Phi_l(C) + (1-\eta_l)\cdot \Theta_l(C) \tag{4.5}$$

By spotting Eq. (4.5), it can be noticed that this formulation is the class-aware analogous of the cluster quality index introduced in Eq. (3.6), where $\Phi_l(C)$ and $\Theta_l(C)$ are respectively the average dispersion and the coverage of the cluster:

$$\Phi_l(C) = \frac{1}{|C|-1}\sum_{g\in C} d_l(g^*, g) \tag{4.6}$$

$$\Theta_l(C) = 1 - \frac{|C|}{|\mathcal{S}_l^{(\mathrm{tr})}|} \tag{4.7}$$

The subgraphs set $\mathcal{S}_l^{(\mathrm{tr})}$ is processed according to d_l and $\mathbf{p}_l$ tailored for the class l. The set of partitions $\mathcal{P}_l = \{\mathcal{P}_{\theta_1}, \ldots, \mathcal{P}_{\theta_h}\}_{(l)}$ contains the clusters emerged by varying the clustering resolution $\theta_1, \ldots, \theta_h$. Thanks to the class-dependent threshold τ_l, low quality clusters according to their corresponding F_l are discarded whilst the remaining are promoted to be symbols. That is, the MinSoDs of the clusters preserved after applying the threshold τ_l are collected together in class-aware alphabet $\mathcal{A}_l = \{s_1, \ldots, s_n\}_{(l)}$, regardless of the partition $\mathcal{P}_\theta \in \mathcal{P}_l$ they belong to.

2. The resulting alphabet $\mathcal{A}_l$ synthesized by a single individual is then processed by the Embedder for building the vectorial representation $\mathbf{h}_l$ of each graph G in the training and validation set. Accordingly, this block receives d_l, that is the core dissimilarity measure with which $\mathcal{A}_l$ has been synthesized. In this way, the emerging symbolic histogram $\mathbf{h}_l$ of graph G is conveniently built by exploiting the representational power conveyed in d_l according to the semantic content in $\mathcal{A}_l$. For this reasons when G actually belongs to the class l, the symbols-subgraphs matches is expected to occur frequently since d_l should emphasizes the similarity between entities in class l, encouraging a match as defined in Eq. (3.12). By inspecting Eq. (3.10), the corresponding symbolic histogram should consequently count high occurrences of symbols in $G_\prime$ when $\mathcal{A}_l$ is considered and when G (hence $G_\prime$) belongs to class l. Otherwise, if G does not belong to class l, the counting procedure should return low values.

3. Eventually, the embedding phase carried on by the individual enable to build two matrices $\mathbf{F}_l^{(\mathrm{tr})}$ and $\mathbf{F}_l^{(\mathrm{vs})}$, respectively an $|\mathcal{D}^{(\mathrm{tr})}| \times |\mathcal{A}_l|$ and an $|\mathcal{D}^{(\mathrm{vs})}| \times |\mathcal{A}_l|$ matrix, whose rows are the symbolic histograms $\mathbf{h}_l$ of each graph in $\mathcal{D}^{(\mathrm{tr})}$ and $\mathcal{D}^{(\mathrm{vs})}$

4. The individually trains a binary classifier on $\mathbf{F}_l^{(\mathrm{tr})}$ and validates its performance on $\mathbf{F}_l^{(\mathrm{vs})}$;

5. Eventually, each individual evaluates its fitness f_1 for the optimization problem.

The fitness function, to be minimized, for a given individual reads as the following convex linear combination

$$f_1 = 1 - \mathcal{J}_1 \tag{4.8}$$

where $\mathcal{J}_1 \in [0,1]$ is defined as the (normalized) informedness [146]:

$$\mathcal{J}_1 = \frac{\text{Sensitivity} + \text{Specificity}}{2} \tag{4.9}$$

This formulation for the objective function stems from the following observation: the informedness is known to be among the most reliable performance indices for binary classification problems, especially when classes are unbalanced. The latter scenario is indeed common in One-vs-All learning systems. At the end of the evolution, the l-th swarm returns the optimal solution

$$\begin{bmatrix} \tilde{\mathbf{w}}_l & \tilde{\mathbf{p}}_l \end{bmatrix} \tag{4.10}$$

along with the alphabet $\tilde{\mathcal{A}}_l$ synthesized thanks to the best genetic code and the embedded versions of training and validation sets against the best alphabet, say $\tilde{\mathbf{F}}_l^{(\mathrm{tr})}$ and $\tilde{\mathbf{F}}_l^{(\mathrm{vs})}$, respectively an $|\mathcal{D}^{(\mathrm{tr})}| \times |\tilde{\mathcal{A}}_l|$ and $|\mathcal{D}^{(\mathrm{vs})}| \times |\tilde{\mathcal{A}}_l|$ matrix.

However, let recall that the above procedure holds for a single swarm. After all swarms converge to their respective solutions, the output of the training procedure results as:

- a set of class-aware optimal alphabets $\tilde{\mathcal{A}}_l|_{l=1}^{L}$

- a set of class-aware dissimilarity measure parameters $\tilde{\mathbf{w}}_l|_{l=1}^{L}$
- a set of embedded training data $\tilde{\mathbf{F}}_l^{(\mathrm{tr})}|_{l=1}^{L}$,
- a set of embedded validation data $\tilde{\mathbf{F}}_l^{(\mathrm{vs})}|_{l=1}^{L}$

Without loss of generality, the alphabets $\tilde{\mathcal{A}}_l|_{l=1}^{L}$ returned from described optimization can be concatenated together in order to return the final, optimal alphabet:

$$\tilde{\mathcal{A}} = \left[\tilde{\mathcal{A}}_1, \dots, \tilde{\mathcal{A}}_L\right] \tag{4.11}$$

where $|\tilde{\mathcal{A}}| = \sum_{l=1}^{L} |\tilde{\mathcal{A}}_l|$. However, the size of the overall alphabet $\tilde{\mathcal{A}}$ can be further reduced by applying a feature selection phase, also driven by an evolutionary strategy. Let

$$\tilde{\mathbf{F}}^{(\mathrm{tr})} = \left[\tilde{\mathbf{F}}_1^{(\mathrm{tr})}, \dots, \tilde{\mathbf{F}}_L^{(\mathrm{tr})}\right] \tag{4.12}$$

and

$$\tilde{\mathbf{F}}^{(\mathrm{vs})} = \left[\tilde{\mathbf{F}}_1^{(\mathrm{vs})}, \dots, \tilde{\mathbf{F}}_L^{(\mathrm{vs})}\right] \tag{4.13}$$

be the concatenation of the class-aware embedded versions of training and validation data, coherently with Eq. (4.11). Hence, $\tilde{\mathbf{F}}^{(\mathrm{tr})}$ and $\tilde{\mathbf{F}}^{(\mathrm{vs})}$ are respectively $|\mathcal{D}^{(\mathrm{tr})}| \times |\tilde{\mathcal{A}}|$ and $|\mathcal{D}^{(\mathrm{vs})}| \times |\tilde{\mathcal{A}}|$ matrices.

This second evolution is in charge of discarding unpromising symbols among the ones contained in $\tilde{\mathcal{A}}$, hence the genetic code reads as a binary mask, say $\mathbf{m} \in \{0,1\}^{|\tilde{\mathcal{A}}|}$. The procedure is equivalent to the feature selection discussion hold in Section 3.4.2. For sake of clarity, the objective function has been reported in this Section as well:

$$f_2 = \sigma \cdot \mathcal{J}_2 + (1-\sigma) \cdot \frac{n'}{\tilde{n}} \tag{4.14}$$

where $n' = \sum_{i=1}^{\tilde{n}} \mathbf{m}_i$ and $\tilde{n} = |\tilde{\mathcal{A}}|$, respectively the number of feature selected and the cardinality of $\tilde{\mathcal{A}}$. It is worth noting that this second optimization phase exploits the training and validation data with their respective original labels, with no binary relabelling involved, and the error rate $\mathcal{J}_2 \in [0,1]$ refers to the plain multi-class error rate (i.e., the ratio of incorrectly classified patterns)

At the end of this second optimization stage, the resulting optimal mask $\mathbf{m}^*$ helps in reducing the size of the starting alphabet $\tilde{\mathcal{A}}$, hence returning:

$$\mathcal{A}^* = [\mathcal{A}_1^*, \dots, \mathcal{A}_N^*] \tag{4.15}$$

where $\mathcal{A}_l^* \subseteq \tilde{\mathcal{A}}_l$ for all $l = 1, \dots, L$, along with a projected training set, say $\mathbf{F}^{*(\mathrm{tr})}$, i.e. an $|\mathcal{D}^{\mathrm{tr}}| \times |\mathcal{A}^*|$ matrix with which a classifier c is trained.

The finalization of the training stage leads to the synthesis of a classification model characterized by the optimized alphabet $\mathcal{A}^*$, along with the set of locally learnt metrics $d_l|_{\mathbf{w}=\tilde{\mathbf{w}}^{(l)}}$ with $l = 1, \dots, L$.

4.4 Soft Symbolic Histogram based Embedding Strategies

From Section 3.3.3, let recall the original symbolic histogram. In short, it aims at representing the graph to be embedded as a vector collecting (in the i^{th} position) the number of occurrences of the i^{th} symbol from the alphabet $\mathcal{A}$ within the graph to be embedded G, with the latter being properly decomposed in $G' = \{g_1, \ldots, g_m\}$ to facilitate the search-and-matching procedure. It is clear that the original symbolic histogram reads as an integer-valued vector, where the proper symbol-to-subgraphs dissimilarity values, i.e. $d(s_i, g)$[1], in Eq. (3.12) are not taken into account. Inspired by dissimilarity spaces (see Dissimilarity Space Embedding in Section 2.2.4), where the dissimilarity value amongst data is the core of the embedding procedure, here below six different strategies are presented in order to populate the symbolic histogram, while at the same time taking into account the dissimilarities between symbols in the alphabet and the constituent parts of the graph to be embedded [13].

Sum

The sum strategy aims at collecting, in the i^{th} position of the symbolic histogram, the sum of distances between the i^{th} symbol in the alphabet and the constituent parts of the graph to be embedded. Formally,

$$\mathbf{h} = \phi^{\mathcal{A}}\left(G'\right) = \left[\text{sum}\left(s_1, G'\right), \ldots, \text{sum}\left(s_n, G'\right)\right] \tag{4.16}$$

where the sum : $\mathcal{A} \times \mathcal{G} \to \mathbb{R}$ operator reads as

$$\text{sum}(s_i, G') = \sum_{g \in G'} d(s_i, g) \tag{4.17}$$

Mean

The sum strategy is characterized by a couple of caveats: a) dissimilarity values are summed up regardless of their magnitude, b) the number of dissimilarities that are summed up is not taken into account. Especially in light of the second caveat, the mean strategy accounts for the mean of distances between the i^{th} symbol in the alphabet and the constituent parts of the graph to be embedded. Formally,

$$\mathbf{h} = \phi^{\mathcal{A}}\left(G'\right) = \left[\text{mean}\left(s_1, G'\right), \ldots, \text{mean}\left(s_n, G'\right)\right] \tag{4.18}$$

where the mean : $\mathcal{A} \times \mathcal{G} \to \mathbb{R}$ operator reads as

$$\text{mean}(s_i, G') = \frac{1}{|G'|} \sum_{g \in G'} d(s_i, g) \tag{4.19}$$

It is worth noting that $|G'|$ varies on a graph-based fashion (i.e., each symbolic histogram has its own scaling factor as it depends on the graph G to be embedded) and it is not equivalent to a constant scaling of all symbolic histograms.

[1]For sake of clarity, the notation $d_{\mathbf{w}}$ used to indicate the GED has been avoided since in this section the edit operation parameters $\mathbf{w}$ are not relevant for the discussion

Median

It is well known that outliers might have a non-negligible impact on the mean of a set of scalar values. In our case, this reflects on very high or very low dissimilarities that might skew the mean value. In order to mitigate this effect, a more robust statistic based on the median value is considered. Formally,

$$\mathbf{h} = \phi^{\mathcal{A}}\left(G'\right) = \left[\text{median}\left(s_1, G'\right), \ldots, \text{median}\left(s_n, G'\right)\right] \tag{4.20}$$

where the median : $\mathcal{A} \times \mathcal{G} \to \mathbb{R}$ operator reads as

$$\text{median}(s_i, g) = \begin{cases} \mathbf{d}_{\frac{|G'|}{2}} & \text{if } |G'| \text{ is even} \\ \frac{\mathbf{d}_{\frac{|G'|-1}{2}} + \mathbf{d}_{\frac{|G'|+1}{2}}}{2} & \text{if } |G'| \text{ is odd} \end{cases} \tag{4.21}$$

and $\mathbf{d} \in \mathbb{R}^{1\times|G'|}$ is a vector that collects the pairwise dissimilarities between s_i and all items in $g \in G'$, sorted in ascending order.

Thresholded-Sum

The three strategies discussed so far in Sections 4.4–4.4 aim at aggregating, according to different operators, the pairwise symbol-to-subgraphs dissimilarities for populating a given entry of the symbolic histogram. Yet, as introduced in Section 4.4, all dissimilarities (regardless of their magnitude) are entitled to contribute to a given entry of the symbolic histogram vector. Taking inspiration from the original symbolic histogram (Section 3.3.3), the dissimilarities contribute to a given operator if and only if their magnitude is below a given threshold.

As the sum operator is concerned, in Eq. (4.16), the sum$(\cdot,\cdot)$ operator, formerly Eq. (4.17), is replaced by the following thresholded sum (or, for short, t-sum : $\mathcal{A} \times \mathcal{G} \to \mathbb{R}$) operator

$$\text{t-sum}(s_i, G') = \sum_{\substack{g\in G' \\ d(s_i,g)\leq\zeta_i}} d(s_i, g) \tag{4.22}$$

where, recall, ζ_i is a symbol-aware inclusion threshold.

Thresholded-Mean

The thresholded mean follows the same rationale behind t-sum$(\cdot,\cdot)$: only dissimilarities below a given threshold are entitled to contribute to the mean value for populating the symbolic histogram entries. That is, the mean$(\cdot,\cdot)$ operator defined in Eq. (4.19) to be plugged into the symbolic histogram (see Eq. (4.18)) is replaced by the following thresholded mean (or, for short, t-mean : $\mathcal{A} \times \mathcal{G} \to \mathbb{R}$) operator

$$\text{t-mean}(s_i, G') = \frac{\sum_{\substack{g\in G' \\ d(s_i,g)\leq\zeta_i}} d(s_i, g)}{|\{g : d(s_i, g) \leq \zeta_i, \forall g \in G'\}|} \tag{4.23}$$

Thresholded-Median

Finally, the thresholded median (or, for short, t-median : $\mathcal{A} \times \mathcal{G} \to \mathbb{R}$) reads as the median$(\cdot, \cdot)$ operator in Eq. (4.21). The major difference is that the (sorted) vector $\mathbf{d}$ will only contain dissimilarities below the symbol-related thresholds ζ_i.

4.5 Multi Objective Optimization for Granular Graph Embedding

As discussed in Section 3.3.4, the classification stage is performed in a suitable vectorial space $\mathcal{F} \subseteq \mathbb{R}^n$ spanned by the symbolic histogram according to the number of symbols n synthesized in the alphabet set $\mathcal{A}$ enabling the chance to employ common machine learning and pattern recognition classifiers which were not available for structured domain such the graph domain $\mathcal{G}$.

While the classifying method is no doubt arbitrary, the specific choice can influence different aspects of the whole algorithm:

- Alphabet sparsity
- Structural complexity

Regarding the alphabet sparsity, the embedding space and thus the alphabet set $\mathcal{A}$ that had generated it are indirectly evaluated by means of the performance measure $\mathcal{J}$ obtained by the classifier trained on $\mathbf{F}^{(\mathrm{tr})}$ and tested on the embedded validation set $\mathbf{F}^{(\mathrm{vs})}$. During the evolution for the alphabet optimization (see Section 3.4.1), different kind of classifiers may require larger set of symbols for attaining good performances in terms of $\mathcal{J}$, eventually impacting on the cardinality of the final alphabet $\tilde{\mathcal{A}}$. On the other hand, one of the most intriguing aspect of the GrC embedding method relies on the chance of analyzing the information granules by field expert enabling a knowledge discovery phase [43, 17, 93]. For this reason, typically the optimization aims at selecting the smallest subset of symbols that simultaneously maximize the pattern recognition system performances [111, 113, 9]. For what concerning the structural complexity, it plays a vital role in terms of classification performance: most challenging problems usually require more complex model able to approximate possibly non-linear decision boundaries in order to attain higher classification performance, e.g. more hidden layers and neurons in Neural Network, large number of support vector in SVM, high number of neighbors to consider in k-NN classifier. Nonetheless, complex models likely have generalization drawbacks that lead to *overfit* the data hand [24, 172]. According to the selected classifier hyperparameters, it is clear that a correct balance between the classification performance and the model complexity must be carefully addressed.

In [9], GRALG has been equipped with several supervised classifiers in the embedding space namely, k-NN [47], SVMs [165] and Neuro-Fuzzy Min-Max Classifiers [158]. Comparison amongst classifiers regards their performance on the test set, their structural complexity and the alphabet sparsity as well. However, the search for a suitable set of information granules in GRALG that leads to the ‘best’ embedding space (i.e., in which classification is more promising) has been addressed via single-objective evolutionary optimization (i.e., maximization of the accuracy of the classifier in the embedding space). Therefore, the structural complexity and the alphabet sparsity emerge as consequences of the maximization of the performance on the validation data and does not play any role in the optimization procedure. After these considerations, some natural questions arise:

- Can the three relevant quantities, i.e. the classification performance, the sparsity of the alphabet and the structural complexity be optimized simultaneously?
- How do these three quantities compete against each other?
- Furthermore, is there any correlation between the low-dimensional space and the smoothness of the decision boundary?

A possible approach to rigorously answer these questions consists in designing suitable fitness functions that consider individually the discussed quantities on a Multi-Objective Optimization (MOO) problem [12]. Multi-Objective Optimization is an area of mathematical optimization that regards problems with more than one objective function, to be optimized simultaneously [120, 5, 108, 182]. Under a decision making viewpoint, MOO is particularly suited where optimal decisions must be taken by considering trade-offs amongst conflicting objective functions. The caveat with MOO is that (for non-trivial problems, at least) there is no single solution that simultaneously optimizes each objective function, due to their conflicting nature. As instead, the set of (usually finite) non-dominated solutions, namely solutions for which it is impossible to improve one of the objective functions without degrading some of the other objective function values, populate the so-called *Pareto Front* (or *Frontier*) [129].

For the sake of description, the critical parameters to be optimized in the proposed MOO optimization are given in a schematic fashion according to the procedures and/or functional blocks in which they are employed:

nBMF: the 6 weights for the edit operations $\mathcal{W}$ (deletions, insertions and substitutions of nodes and edges) and the additional parameters Π^v, Π^e for the node\edge distances: $\mathbf{w} = [\mathcal{W}, \Pi^v, \Pi^e]$ (see Section 3.2).

Granulation: U the maximum number of admissible cluster for the BSAS procedure; τ_F the quality threshold for promoting a MinSoD to a symbol; η trade-off parameter that weights compactness and cardinality collected in $\mathbf{p}$ (see Section 3.3.2);

Classifier: the set of hyperparameters for a ν-SVM classifier collected in $\mathbf{c}$ according to kernel employed:

- Linear: only the regularization term $\nu \in (0, 1]$;
- Radial Basis Function (RBF): along with $\nu \in (0, 1]$, the kernel shape $\gamma \in (0, 100]$ is tuned as well.

Hence, the variable space for the MOO problem reads as follows:

$$\mathbf{s} = [\mathbf{w} \quad \mathbf{p} \quad \mathbf{c}] \tag{4.24}$$

The set of parameters collected in $\mathbf{s}$ drives the pipeline of granulation, embedding and classification described in Section 3.3. Specifically, the MOO optimization technique leverage on the Non Dominated Sorting Genetic Algorithm II (NSGA) [198] algorithm which individuals in the corresponding population:

1. Run the granulation procedure on the set of subgraphs $\mathcal{S}^{(\mathrm{tr})}$ extracted by following the stochastic variant described in Section 4.1 using the BFS extractor. The dissimilarity measure parameters for the clustering algorithm are determined by $\mathbf{w}$ and $\mathbf{p}$. The granulator parameters (i.e. U, τ and ρ) are employed as well for building the alphabet $\mathcal{A}$;

2. Both $\mathcal{D}^{(\mathrm{tr})}$ and $\mathcal{D}^{(\mathrm{vs})}$ can now be embedded in $\mathbf{F}^{(\mathrm{tr})}$ and $\mathbf{F}^{(\mathrm{vs})}$ according to symbolic histogram paradigm exploiting the current alphabet $\mathcal{A}$. The dissimilarity measure parameters $\mathbf{w}$ and $\mathbf{p}$ are thus employed for the evaluation of Eq. (3.12);

3. The set of classifier hyperparameters $\mathbf{c}$ are exploited for training c on $\mathbf{F}^{(\mathrm{tr})}$. Finally, the performance $\mathcal{J}\left(\mathbf{F}^{(\mathrm{vs})}\right)$ is returned.

The three objective functions are formalized as follows:

$$f_1 = \mathcal{J}\left(\mathbf{F}^{(\mathrm{vs})}\right) \tag{4.25}$$

$$f_2 = \frac{SV}{|\mathcal{D}^{(\mathrm{tr})}|} \tag{4.26}$$

$$f_3 = \frac{|\mathcal{A}|}{|\mathcal{S}^{(\mathrm{tr})}|} \tag{4.27}$$

The first objective function $f_1 \in [0,1]$ in Eq. (4.25) is defined as the error rate of the classifier c in predicting $\mathbf{F}^{(\mathrm{vs})}$. The second objective function in Eq. (4.26) is defined as the ratio between the number of support vectors (SV) used in the classification problem and the cardinality of the training set and accounts for the structural complexity of the classification model. In this way f_2 is bounded in $[0,1]$ as well, being $|\mathcal{D}^{(\mathrm{tr})}|$ the maximum number of admissible support vectors (i.e., all training patterns are elected as support vectors). Finally, the third objective function f_3 in Eq. (4.27) reads as the ratio between the alphabet cardinality and the number of subgraphs employed during the granulation phase. In this way, f_3 is bounded in $[0,1]$ since $|\mathcal{S}^{(\mathrm{tr})}|$ is the cardinality of the largest alphabet derivable from $\mathcal{S}^{(\mathrm{tr})}$ (i.e., all candidate information granules are promoted as information granules). The optimization problem will therefore aim at performing a joint minimization of f_1, f_2 and f_3.

4.5.1 Selection of Solutions from the Pareto Front

At the end of the optimization phase, NSGA-II returns a set of N non-dominated solutions $X = \{\mathbf{s}_1, \ldots, \mathbf{s}_N\}$. More in details, in a multi-objective minimization problem a solution $\mathbf{s}_i$ is said to be dominated by $\mathbf{s}_j$ if the following hold [60]:

$$\begin{aligned} f_m(\mathbf{s}_j) \leq f_m(\mathbf{s}_i) \quad \forall m \in \{1, \ldots, M\} \\ \exists m \in \{1, \ldots, M\} : f_m(\mathbf{s}_j) < f_m(\mathbf{s}_i) \end{aligned} \tag{4.28}$$

where, recall, M is the number of objective functions. Practically speaking, since X contains the observed non-dominated solutions, it is actually approximating the Pareto front of the multi-objective problem, which can not be known in closed-form in the context of machine learning. The Pareto front can then be viewed as the set of compromise solutions that are trying to optimize simultaneously all the objective functions. It goes without saying that, in light of Eq. (4.28), a solution $\mathbf{s}_i \in X$ can not be preferred to any other $\mathbf{s}_j \in X$ without further indications [83]. On the other hand, in real applications it is often mandatory to select a single suitable solution or a group of efficient alternatives. Multi-criteria decision making helps in proposing a set of methods in order to possibly overcome this problem. A very popular procedure in multi-criteria decision making is Technique for Order Preference by Similarity to the Ideal Solution (TOPSIS) [79] which has shown good reliability, low computational cost and an intuitive logical reasoning [160, 182]. Starting from the decision matrix $\mathbf{D} \in \mathbb{R}^{N \times M}$, i.e. the matrix whose i^{th} row is the vector $[f_1(\mathbf{s}_i), \dots, f_M(\mathbf{s}_i)]$, TOPSIS ranks elements in X according to their Euclidean distance to the positive-ideal and negative-ideal solutions. That is, the optimum will be chosen as the solution that is simultaneously showing the smallest distance from the positive-ideal and the largest distance from the negative-ideal. Additionally, TOPSIS allows to weight individually the importance of each objective function in the ranking process by scaling each rows of $\mathbf{D}$ according to a weight vector $\mathbf{u} \in \mathbb{R}^M$.

4.5.2 Ensemble of Classifiers for Test Phase

In light of the discussion in Section 4.5.1, the top K solutions $X^* = \{\mathbf{s}_1^*, \dots, \mathbf{s}_K^*\}$ ranked by TOPSIS according to the weight vector $\mathbf{u}$ are retained from the optimization stage. Alongside, the corresponding alphabets $\mathcal{A}^* = (\mathcal{A}_1^*, \dots, \mathcal{A}_K^*)$ and the training embedded matrices $\mathcal{T}^* = \left(\mathbf{F}_1^{*(\text{tr})}, \dots, \mathbf{F}_K^{*(\text{tr})}\right)$ obtained by embedding $\mathcal{D}^{(\text{tr})}$ thanks to the alphabets in $\mathcal{A}^*$ are retained as well. In order to exploit simultaneously the information carried by the K solutions, an ensemble of classifiers $\mathcal{C}$ addresses the final performance on the test set $\mathcal{D}^{(\text{ts})}$. The design of $\mathcal{C}$ goes as follows:

1. For each alphabet $\mathcal{A}_i^* \in \mathcal{A}^*$, the vectorial representation $\mathbf{F}_i^{(\text{ts})}$ of $\mathcal{D}^{(\text{ts})}$ is built according to the symbolic histogram paradigm;
2. From each solution $\mathbf{s}_i^* \in X^*$, the hyperparameters $\mathbf{c}_i^*$ of the optimized classifier is retrieved;
3. The classifier $c_i \in \mathcal{C}$ is trained with $\mathbf{F}_i^{*(\text{tr})} \in \mathcal{T}^*$ with hyperparameters $\mathbf{c}_i^*$;
4. All the trained classifiers c_i with $i = 1, \dots, K$ are collected in the stack $\mathcal{C}$.

It is worth remarking that each classifier c_i operates in a specific embedding space (to which $\mathbf{F}_i^{*(\text{tr})}$ and $\mathbf{F}_i^{*(\text{ts})}$ belong) generated by the alphabet $\mathcal{A}_i^*$ emerged from the solution $\mathbf{s}_i^*$.

This approach allows to establish a common Winner-Takes-All policy when predicting a single pattern from the test set. Let $G_k \in \mathcal{D}^{(\text{ts})}$ be the k^{th} graph in the test set; the steps that lead to the prediction of its class label by the ensemble $\mathcal{C}$ go as follow:

1. let $\left(\mathbf{h}_1^{(k)}, \ldots, \mathbf{h}_K^{(k)}\right)$ be the vectorial representations of G_k in K different embedding spaces $\mathcal{F}_1, \ldots, \mathcal{F}_K$, where in general $\mathcal{F}_i \neq \mathcal{F}_j$ for all $i \neq j$. This step corresponds in gathering the k^{th} rows of $\left(\mathbf{F}_1^{*(\text{ts})}, \ldots, \mathbf{F}_K^{*(\text{ts})}\right)$;

2. each classifier $c_i \in \mathcal{C}$ individually predicts the label $y_i^{(k)}$ for $\mathbf{h}_i^{(k)}$ (for $i = 1, \ldots, K$);

3. all predicted labels are collected in the output stack $\left(y_1^k, \ldots, y_K^k\right)$;

4. the most frequent label in $\left(y_1^k, \ldots, y_K^k\right)$ is finally chosen as the predicted class for G_k.

However, especially in multi-class problems, ties between different labels can occur in the voting process. For this reason, it is necessary to define a suitable decision rule acting as a tie-breaker which eventually leads to a single label output. This problem have been approached by defining a confidence measure $\gamma_i\left(y_l\right)$ of classifier c_i in predicting a specific label y_l [2], where $l = 1, \ldots, L$, being L the number of classes. The confidence $\gamma_i\left(y_l\right)$ for class y_l is chosen as the (multi-class) precision score obtained in classifying the training set $\mathbf{F}_i^{(\text{tr})}$.

Let R be the subset of L classes that tie. The resolution of a dispute works as follows:

1. Group the classifiers involved in the dispute according to the labels they predict;

2. Ask each classifier its confidence about the prediction;

3. For each label in R (hence, for each group of classifiers), evaluate the average within-group confidence;

4. Emit the final predicted label as the one with highest within-group confidence.

[2]With an abuse of notation, $y_1, \ldots, y_L$ are all the class labels for the problem at hand

Chapter 5

Evolutive Agent Based Framework for Granular Graph Embedding

5.1 Introduction

Multi-Agent Systems (MAS) [133] emerged in the last years as powerful approaches able to solve complex pattern recognition problems. This is due to the innate nature of multi-agent systems, where independent atomic computational units (i.e., the agents) cooperate in order to solve a complex problem by a divide-and-conquer approach.

The first immediate attractive characteristic of multi-agent systems is the distributed organization of the entities which locally interact together without a centralized control. This aspect offers the opportunity to employ such systems in highly parallel hardware architectures looking for relevant information also in large datasets. Indeed, MAS can be an efficient approach for facing the problems raised by the Big Data advent [18, 179]. In such problems, the whole dataset can not be entirely stored in a single memory device due to its large dimension, rather it has to be decomposed towards several processing units which resemble the aforementioned MAS characteristic.

When multi-agent systems are combined with an adaptive mechanism, the self-organization aspects that arise can be very effective for solving real problems that show high level of complexity. In MAS, the self-organization consists in several simple elements that have limited information for adapting themselves in a specific possibly dynamical environment where the optimization of agent abilities and qualities can be accomplished only by local interactions with other elements [72]. In this context, the interaction activity is an essential part of the learning process and is not only considered as a mere exchange of data. Instead, it manifests itself in the form of spirit of cooperation and collaboration during a negotiation phase between the agents solutions which are found according to the knowledge acquired by each agent individually. The interesting point is that MAS agents are organized in order to accomplish a division of works and tasks under two usual different point of views, i.e. functional and/or by data characteristics division [184]: the former is intended

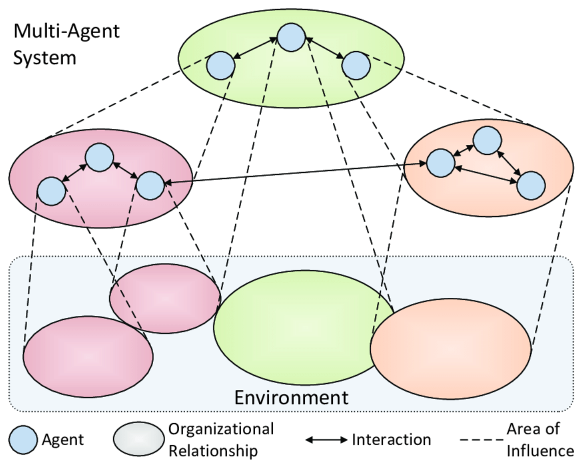


Figure 5.1. Schematic view of a multi-agent system organization [125].

to enable learning roles differentiation between groups of agents, the latter for specializing agents in extracting knowledge from different data regions. In the collaboration stage, the agent's products collected according to the two aspects just described in the moment of investigation are then mutually exploited improving the ability to unravel the complexity of the problem. The described features and capabilities of MAS drive the development of a new granular computing system that address the challenges of complex systems modeling tasks.

The basic rationale behind the development of a new approach for modeling complex problems starts from the main hypothesis that complexity demands complexity. The most straightforward example can be found in the biological brain that can be seen as a complex system that has evolved in harmony with the surrounded complexity of the world around. Due to its nature, there is no a clear definition of complex system that is jointly accepted in all the disciplines [76, 122]. Nonetheless, most of the different views agree that a complex system is characterized by some basic properties [90]:

- It is the ensemble of many atomic interacting elements.
- Atomic elements perform a functional role (a physical causal relationship, or even a computation task) whose mutual interactions are nonlinear in nature.
- The network of these relationships contains loops and feedback.
- Show self-organization properties of the atomic entities in possible hierarchical groups.

The atomic entities are subsystem components that perform simple tasks with respect to the global system behaviour. For example, entities in the biological brain could be represented by the neurons, each one performing a non-linear simple task. On the other hand, the mutual interactions between such entities give rise to complex cognitive abilities, that is the emergent behaviour of the complex system. Loops and feedbacks define how each part of the system will change in time according to other components. That is, the behaviour of a group or single entity at a specific time

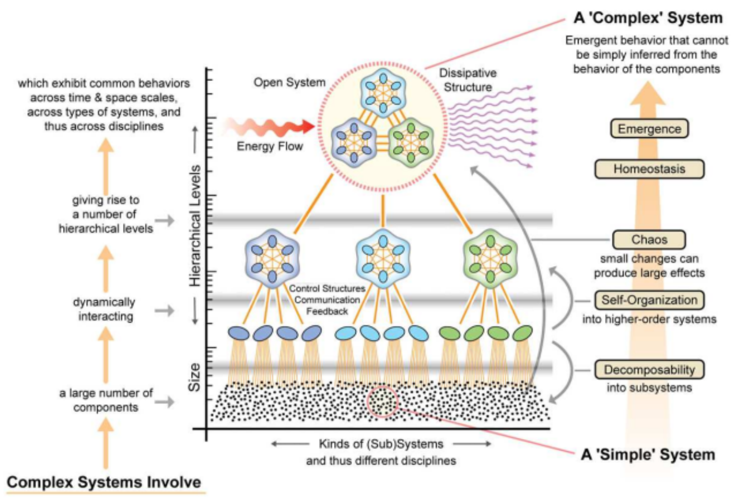

A 'Complex' System
Emergent behavior that cannot be simply inferred from the behavior of the components
which exhibit common behaviors across time & space scales, across types of systems, and thus across disciplines
Open System
Dissipative Structure
Energy Flow
Emergence
Homeostasis
giving rise to a number of hierarchical levels
Hierarchical Levels
Chaos
small changes can produce large effects
Control Structures Communication Feedback
dynamically interacting
Self-Organization
into higher-order systems
Decomposability
into subsystems
a large number of components
Size
Kinds of (Sub)Systems and thus different disciplines
A 'Simple' System
Complex Systems Involve

Figure 5.2. Multi-view schematic description of a complex system architecture. The hierarchical organization is highlighted in the vertical axis. Image taken from `https://www.idiagram.com/examples/complexity.html`

is influenced by its neighbor behaviour and by the mutual interactions happened previously. Usually, behaviors defining each atomic element generate collectively some form of self-organization, giving rise to a hierarchical structure, built as a sequence of functional (and semantic) layers. Each layer is composed by entities consisting of an aggregation of previous level entities. The overall system is thus organized as a nested system of systems, each one being a time-variant, nonlinear, dynamical one. Two schematic representation of the discussed characteristics of complex system are depicted in Figures 5.1 and 5.2. In light of this observations, GrC propose a way of thinking that fit very well with the hierarchical and nested nature of complex systems. Indeed, from a philosophical perspective, GrC underlines the urgency to apply a structural approach in the representation process of a problem in order to reflect the multiple level organization of the system [193].

Emerging behaviors arising from complex systems are predictable, to some extent, if the modeled system is able to automatically synthesize a hierarchy of information granules, organized in successive semantic levels, following an information granulation approach, and possess the ability to identify useful patterns in higher semantic levels that have a predictive value. Building predictive models of such complex systems is a challenging task, yet of utmost importance for advancements in natural sciences, as well for achieving technological goals, e.g., cybersecurity, biological systems, natural language processing, clinical diagnostic systems, precision medicine, smart grids, intelligent transportation system [113, 161, 86, 82].

5.2 High Level Framework Description

In this section, a novel framework conceived for modeling complex systems that follows the principle of Granular Computing and Multi Agent Systems named Evolutionary Agent Based Classifier (E-ABC) is introduced. As discussed in Section

5.1, the granular information processing approach aims at unraveling the complexity of the system by extracting suitable granules of information which in our framework will be used for inferring properties about the problem at hand. That is, the whole framework has the final purpose to build an efficient classification system which leverage on the granules of information synthesized during the training phase according to a multi-level analysis.

5.2.1 Data Granulation

Initially, the system comprises raw data acquired by the observation of the process under investigation. The nature of the data clearly depends on the specific application field and thus can be represented with different data structures. This is a critical facet for the development of our system since it influences how the patterns can be compared together during the granulation phase. Indeed, from our point of view, the granulation procedure is performed by means of a clustering algorithm which commonly relies on a *dissimilarity measure* that must be able to correctly grab the proximity of semantically close pattern for grouping data in meaningful clusters. Typically, data distribution is complex and not homogeneous in the input space, hence using a fixed dissimilarity might be inefficient to enable the emergence of relevant information. On the other hand, by letting the distance be parameterized, it is possible to select suitable values tuned accordingly to the data itself, an approach commonly known as *metric learning*. In a *global metric learning* context, a single dissimilarity measure valid for all the decision regions would be learned, that is only a single set of $\mathbf{w}$ will be found, possibly neglecting the high heterogeneity of the data distribution. In this case, a *local metric learning* approach can be preferred: different regions of the input space will be characterized by specific metrics. In case decision regions are described and approximated by a set of clusters, in local metric learning framework it is possible to learn different dissimilarity measures, one for each cluster, each of which minimize the pairwise dissimilarity between elements belonging to the same cluster. The most straightforward example of local metric learning when the input space $\mathcal{X} = \mathbb{R}^n$ lies on subspace clustering [2, 74], where different clusters might lie in different subspaces so the target of the learning system is to jointly find clusters and the subspace $\mathbb{R}^m$ with $m < n$, in which they lie.

From this perspective, the granulation procedures aims at finding relevant information each of which is matched with an instance of the dissimilarity measure $d_{\mathbf{w}}$ employed for the clustering algorithm at hand. According to the latter, the data is aggregated in a multi-level fashion according to the GrC perspective. The modalities with which different abstraction levels and granularity are explored depends on the specific design choice about the clustering methods to exploit. For example, hierarchical clustering algorithms are the most straightforward approaches in granular computing by clustering due to their ability in highlighting the dependencies between granules. On the other hand, these approaches often show possible computational drawbacks, whilst another class of clustering methods, i.e. free clustering algorithms, provides low computational complexity solutions and could explore different granularity levels by varying the resolution for including patterns in the same clusters. As opposed to hierarchical methods, free clustering algorithms does not provide a clear relation between clusters in the arisen partitions.s Let s be a *symbol*, that is the representative element of a specific cluster emerged for an instance of the granulation phase. This entity is thought as the result of the first computational layer that has the scope to raise the level of abstraction from the lowest (the raw data) to the very next one, that is, data aggregated by means of distance and compressed in the cluster representative element.

In the next layer, the granularity level is increased by generating an aggregation of symbols $s_1, \ldots, s_n$ into the alphabet set $\mathcal{A}$. Hence, the classification model built upon $\mathcal{A}$ provides a performance measure $\mathcal{J}$ evaluated in a supervised way in order to have a direct external critic about the synthesized group of granules, i.e. a supervised feedback of the alphabet effectiveness.

According to this discussion, a symbol s carries three types of information:

- The MinSod representative of the emerged cluster
- The dissimilarity measure $d_{\mathbf{w}}$ parameterized in $\mathbf{w}$ with which it has been found
- The *quality* attribute Q_s build upon a *reward-based* criteria assigned according to the performance measures $\mathcal{J}$ observed for all alphabets $\mathcal{A}$ to which the symbol s belongs.

5.2.2 Agents and Swarms Organization

Having introduced the two level of abstraction from the granulation point of view, i.e. symbols and alphabets, the discussion can be moved towards the organization of main actors responsible for generating them.

The atomic entity is the *Granulator Agent* which is enlisted to work on the raw data for making symbols. These agents are organized in different swarms $\mathcal{U}$, i.e. groups of agents, each of which is in charge to analyze a specific portion of the dataset. Hence, in the spirit of MAS approach, the work division in this layer is oriented to refine the agents ability to extract information and knowledge from a specific data subset, which can be determined for example by the pattern labels of the training dataset. Notably, each agent is directly matched with a specific dissimilarity measure it employed in the granulation phase. That is, each agent carries the information about a specific dissimilarity measure by exploring the space parameters $\mathbf{w}$ in order to find the most reliable ones for synthesizing suitable symbols. Needless to say, in turns the symbols inherit $d_{\mathbf{w}}$ information by its own agent parent. By this means, each swarm synthesizes a set of promising symbols $\mathcal{H}$ at which we will referred to as *swarm candidate symbols*. This terminology is driven by the nature of these symbols, that is, they are relevant information emerged in the first computational layer which potentially can be part of a single or more alphabets. Symbols in $\mathcal{H}$ undergo a *consensus* phase, which can be seen as a form of collaboration and negotiation between agents in $\mathcal{U}$. Indeed, the goal of this phase is devoted to compare the agents results in order to emphasize those symbols who looks similar in the same parameter space $\mathbf{w}$. The rationale behind this procedure is that if two or more agents agree about the parametric dissimilarity measure $d_{\mathbf{w}}$, than it is possible to advance the hypothesis that $\mathbf{w}$ is an interesting set of values for our problem. Accordingly, similar symbols are interpreted as real informative granules hence they are rewarded by increasing their qualities Q_s. In such way, it is possible to emphasize the importance of both symbol and dissimilarity measure according to the parameter space $\mathbf{w}$ in which they have been found.

At the next hierarchical level, a new swarm of agents $\mathcal{Z}$ called *Alphabet Selector Agents* is in charge to select symbols for the alphabet formation phase. By observing the performance measure $\mathcal{J}$ obtained by an alphabet $\mathcal{A}$, a single agent in $\mathcal{Z}$ tries to aggregate symbols that positively impact on $\mathcal{J}$ in novel sets, so that more efficient alphabets could emerge.

The granulator agents in $\mathcal{U}$ are then judged according to the information perceived by the swarm $\mathcal{Z}$ during the classification phase in the form of supervised performance $\mathcal{J}$ and an internal cluster validity measure $F(\cdot)$. Hence, the granulator agent quality Q_a reflects both its own perception about the symbol validity via $F(\cdot)$ and the symbols relevance it synthesized intended as component of valuable alphabets according to $\mathcal{J}$. From the MAS philosophy perspective, $\mathcal{U}$ and $\mathcal{Z}$ respect the principle of work division by role separation: the former, working on a first abstraction level, synthesizes the granules which in turns are managed by the latter for alphabet formation. Hence, such alphabets serves for achieving the final scope of the system, i.e. the design of performing classification models whose refinement is realised through an exchange of information according to the feedback signal $\mathcal{J}$ towards the symbols and in turns towards the agents. The whole system optimization is conceived as an evolutionary strategy which drives the first swarm of agents $\mathcal{U}$ towards promising parameters $\mathbf{w}$ and refine their granulation abilities, whilst the evolution of the second swarm $\mathcal{Z}$ is designed in order to maximize the alphabets performances $\mathcal{J}$ when they are used as starting point for a classification model.

5.3 Evolutionary Agent Based Classifier in Vector Space

Initially, E-ABC has been conceived as an unsupervised algorithm able to perform subspace clustering [112], i.e. a form of local metric learning for problem where input space coincides with $\mathbb{R}^n$. Each agent is entailed to run a clustering algorithm on a subset of data sampled from the training set $\mathcal{S}^{(\mathrm{tr})}$. With the goal of exploring suitable subspaces, the dissimilarity measure employed in the clustering algorithm is evaluated as a weighted Euclidean distance:

$$d_{\mathbf{w}}(\mathbf{a}, \mathbf{b}) = \sqrt{\sum_{i=1}^{n} \mathbf{w}_i \left(\mathbf{a}_i - \mathbf{b}_i\right)^2} \tag{5.1}$$

In this way, the parameter vector $\mathbf{w} \in \{0,1\}^n$ act as a binary mask used for projecting $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ into a lower dimensional vector space where the comparison between the two patterns $\mathbf{a}, \mathbf{b}$ by means of $d_{\mathbf{w}}$ takes place. The agents fitness is designed according to an internal quality measure which is a linear combination between the average clusters compactness and cardinality. Hence, the agents parameters, i.e. clustering parameters and $\mathbf{w}$ are optimized by means of an evolutionary algorithm in order to maximize the agent fitness.

A first attempt to extend this framework towards supervised learning tasks can be found in [68]. In this work, the algorithm relies on two separated phases whose contributions are differently weighted: initially, the procedure is devoted to synthesize promising clusters according to an internal quality measure, i.e. compactness and cardinality, equivalently to as the previous unsupervised version of E-ABC; later on, well-formed discovered clusters are used as starting point for building a classification model, i.e. decision cluster classifier [89, 202]. According to this discussion, each cluster is evaluated simultaneously according to its performance as part of a classification model and to its internal quality measure. The evolution orchestration is driven by a genetic algorithm in charge to select suitable clustering parameters and subspaces for improved agents. Although the procedure shown promising results from recognition point of view and metric learning capabilities, the authors noticed a bias in the selection of informative subspaces. Indeed plain genetic algorithms are usually unable to discover multiple different solutions in the landscape, rather they return a single (sub-) optimum. In [67], a multi-modal genetic optimization [186, 91, 71] has been tested in order to encourage the population of individuals in exploring heterogeneous subspaces according to *fitness sharing* technique [163, 132, 49]. Following this approach, agents act as if they were sharing limited resources, hence close agents (in terms of search subspace) are penalized. The more agents share the same subspace, the more their fitness values are reduced. In this way, agents are compelled to explore new subspaces that no other are investigating. Additionally, in the same work, the authors assessed the algorithm scalability using parallel strategies in multi-core architectures.

5.3.1 Proof of Concept

Starting from the architectures and properties of the above described E-ABC framework, a new scheme has been designed by introducing a feedback mechanism devoted to back propagate the cluster effectiveness information as component of a classification model to its quality and a consensus strategy enabling the collaboration between agents. Furthermore, the system relies on different swarms which individually works on a specific region of dataset establishing a task division from the data point of view. In light of these observations, this latest implementation reflects most of the properties described in Section 5.2 and can be considered as the first proof of concept to assess the proposed framework in problems whose input domain $\mathcal{X} = \mathbb{R}^n$, that is Euclidean spaces. As in the other implementations, the core dissimilarity is chosen as the parametric Euclidean distance as defined in Eq. (5.1).

Bootstrap Phase

In the *bootstrap* phase, the swarms are entailed to synthesize relevant information from the data at hand that later serves for building a classification model. According to the number of classes L in the problem, $\mathcal{U}_1, \ldots, \mathcal{U}_L$ swarms receive each a class stratified training set $\mathcal{D}_1^{(\mathrm{tr})}, \ldots, \mathcal{D}_L^{(\mathrm{tr})}$. As discussed in Section 5.2, these swarms are composed by granulation agents which are in charge to raise the level of abstraction from raw data to candidate symbols collected in class specific sets $\mathcal{H}$. Hence, with the scope of building a specific bucket of class l, i.e. $\mathcal{H}_l$, each agent $a \in \mathcal{U}_l$ performs the following operations:

1. Extracts a sample $\mathcal{S}^{(\mathrm{tr})}$ from $\mathcal{D}_l^{(\mathrm{tr})}$

2. Setup the dissimilarity measure according to the binary mask $\mathbf{w}$
3. Runs a clustering algorithm and searches the best partitions
4. Evaluates the reliability (to be maximized) of each cluster $F(C)$ extracted by the agent:
$$F(C) = \eta \cdot \Phi(C) + (1 - \eta) \cdot \Theta(C) \tag{5.2}$$
In Eq. (5.2), Φ and Θ are respectively the cluster compactness and cardinality whose importance is weighted by a trade-off parameter η:
$$\Phi(C) = 1 - \frac{1}{|C|} \sum_{x \in C} d_{\mathbf{w}}(x, \mu) \tag{5.3}$$
$$\Theta(C) = \frac{|C|}{|\mathcal{S}^{(\text{tr})}|} \tag{5.4}$$
where μ is the centroid of cluster C.
5. Promotes the clusters with sufficient reliability according to a threshold τ and inserts them in $\mathcal{H}_l$.

In point 3, the clustering algorithm employed is the *Reinforcement Learning* variant of BSAS (RL-BSAS) [157], which aims at removing uninformative spurious clusters from the final partition according to a rewarding mechanism. RL-BSAS relies on two fundamental parameters: δ, the reward-forgetting ratio value for strengthening or removing clusters and θ, the maximal cluster representative-to-pattern distance for including the pattern into the cluster. Even though δ is selected by means of the deployed evolutionary strategy, θ is optimized autonomously by the agent with a specific heuristic. When all swarms have finished their tasks, each bucket $\mathcal{H}_l$ undergoes a *consensus* strategy that highlights clusters identified simultaneously by different agents in the same swarm by enhancing their quality Q_c. The procedure goes as follow:

1. Group the clusters according to their binary mask $\mathbf{w}$
2. Evaluate the pairwise distances $d_{\mathbf{w}}(\mu_i, \mu_j)$, where μ_i, μ_j are respectively the centroids of cluster C_i, C_j in the same group
3. According to a threshold τ_{fus}, merge the two clusters in a new cluster C_k and evaluate the new centroid μ_k. The new cluster quality $Q_c(C_k)$ read as follows:
$$Q_c(C_k) = Q_c(C_i) + Q_c(C_j) \tag{5.5}$$

The relevance of this procedure is two fold: by merging the clusters into a novel data agglomeration, we are limiting the memory footprint carried by $\mathcal{H}_l$ that in large datasets and complex problems can be a critical facet; more importantly, the system exploits a collaborative approach for evaluate the information extracted by the agents. Indeed, when two or more agents agree they have found essentially similar information, the system gives value to these results by combining the agents product into a novel and enhanced granule represented by C_k. Later on, the improved quality with respect to the native clusters will positively impact the parents qualities giving them more chance to survive at the next evolution.

Classification Model Synthesis

After the class-aware buckets of symbols $\mathcal{H}_1, \ldots, \mathcal{H}_L$ are ready, the synthesized information is exploited for building several classification systems by means of decision clusters. This type of classifier relies on a set of clusters which are labelled with a specific class of the problem at hand according to the set $\mathcal{H}_l$ from which they are extracted. Hence, the construction of a classification model $\mathcal{M}$ is evaluated as follows:

1. Determine the maximum number of clusters $z_{max} \in \mathbb{N}$ that can be extracted from each set of candidate symbols $\mathcal{H}_l$ as:
$$z_{max} = \min\{|\mathcal{H}_1|, \ldots, |\mathcal{H}_L|\} \tag{5.6}$$
2. Determine the number of clusters $z_l \in \mathbb{N}$ to extract from $\mathcal{H}_l$ by drawing uniform at random $z_l \in [1, z_{max}]$
3. For $l = 1, \ldots, L$, draw uniform at random without replacement z_l clusters from the set of candidate symbols $\mathcal{H}_l$ and insert them in $\mathcal{M}$.

The classification model $\mathcal{M}$ endows an inference mechanism which allows to assign one label amongst the L that characterized the problem to an unseen pattern x given in input to $\mathcal{M}$.

Quality Assignment

The classification model is evaluated according to a performance measure $\mathcal{J} : \mathcal{M} \times \mathcal{D} \to \mathbb{R}$ which assesses the effectiveness of $\mathcal{M}$ in a supervised way. The model $\mathcal{M}$ is evaluated on a validation set $\mathcal{D}_{vs}$ composed by previously unseen pattern where $\mathcal{J}$ is defined as the accuracy of $\mathcal{M}$ in classifying $\mathcal{D}_{vs}$. The feedback received by $\mathcal{M}$ in the form of $\mathcal{J}$ can be used to define the quality Q_c of each cluster $C \in \mathcal{M}$. The cluster quality Q_c is updated as follows:

$$Q_c\,(C)^{(new)} = Q_c\,(C)^{(old)} + \alpha \tag{5.7}$$

The right-most term denoted with α in Eq. (5.7) is the reward value that is determined according to $\mathcal{J} \in [0, 1]$:

$$\alpha = \begin{cases} -1 & \text{if} \quad \mathcal{J} \leq 0.5 \\ 1 & \text{if} \quad 0.5 < \mathcal{J} \leq 0.9 \\ 10 & \text{if} \quad \mathcal{J} > 0.9 \end{cases} \tag{5.8}$$

Given M classification models $\mathcal{M}_1, \ldots, \mathcal{M}_M$, the cluster qualities composing each model are thus determined by applying Eqs. (5.7) and (5.8). It is worth to notice that a single cluster C can appear in different models, hence its quality Q_c will change in line with different performances $\mathcal{J}$ observed. Notably, those clusters whose quality is negative will be removed from their respective $\mathcal{H}$ set.

From our point of view, the agent quality Q_a is intended as a measure determining the agent ability in extracting relevant information with prediction capabilities. Since the classification model core elements are identified with the clusters discovered by the agents, the evaluation of Q_a reflects the overall cluster qualities it synthesized. The quality of agent a is updated with the following rule:

$$Q_a(a)^{(new)} = Q_a(a)^{(old)} + \beta \tag{5.9}$$

where β represents the rewarding factor considering the N_a clusters discovered by a:

$$\beta = \frac{1}{N_a} \sum_{i=1}^{N_a} Q_c(C_i) \tag{5.10}$$

By investigating Eqs. (5.9) and (5.10), it is clear that the agent quality is updated according to the mean quality value obtained by the N_a clusters whose a is the owner.

Evolution

The initial evolution generations g_{bst} are solely spent in the *bootstrap* phase which is addressed in discovering the most informative clusters in unsupervised way. That is, the evolution is totally driven by the cluster reliability $F(C)$ defined in Eq. (5.2) and the agent fitness in the bootstrap step $F_{bst}(a)$ is defined as the best cluster reliability:

$$F_{bst}(a) = \max\{F(C_1), \dots, F(C_{N_a})\} \tag{5.11}$$

The motivation behind this approach is that in the early stages the information synthesized might not be informative enough to be safely exploited in a classification model. This is because at early stages the rewards mechanism is still not effective in producing meaningful clusters, and only unsupervised quality measures, such as compactness and separability of clusters, can be taken into account. Consequently, these clusters could have poor performance when used in a classification system based on decision clusters. Hence, the agents are firstly trained in order to improve their abilities in discovering well-formed clusters and only after that the system moves on the model synthesis phase. In this step, the agent fitness F_{pred} is then evaluate as:

$$F_{pred}(a) = \hat{Q}_a(a) \tag{5.12}$$

where $\hat{Q}_a(a)$ is the agent quality $Q_a(a)$ properly scaled in $[0, 1]$ according to the agent qualities in the swarm. Each agent genetic code is composed by the main clustering parameter δ and the vector $\mathbf{w}$ identifying the binary mask for the dissimilarity measure $d_{\mathbf{w}}$. The genetic algorithm aims at maximizing the two mentioned fitness functions depending on the stage in which the system is in by leveraging on standard genetic operators between genetic codes, i.e. mutation, crossover, elitism and selection.

In each generation, the system selects the best-so-far $\mathcal{M}_1^*, \dots, \mathcal{M}_K^*$ classification models with $K < M$ by exploiting a greedy approach. At the generation g, $\mathcal{M}_1, \dots, \mathcal{M}_M$ are compared with $\mathcal{M}_1^*, \dots, \mathcal{M}_K^*$ by ranking the $K + M$ models according to the performances attained on the validation set. Hence, the top-K models are retained for the next $g + 1$ generation where possibly substitution caused by the observation of better performances in the newest M models can take place.

Test Phase

The swarm training phase (bootstrap and model synthesis) ends after a number of predefined generations g_{stop} where $g_{stop} > g_{bst}$. In this generation, $\mathcal{M}_1^*, \ldots, \mathcal{M}_K^*$ represents the best classification models synthesized during the training phase. In order to exploit most of the information retained by these classifiers, we build an ensemble of classifier $\mathcal{C}$ composed by $\mathcal{M}_1^*, \ldots, \mathcal{M}_K^*$ for testing the final performance on a test set $\mathcal{D}^{(\mathrm{ts})}$. The ensemble receives in input a pattern $x \in \mathcal{D}^{(\mathrm{ts})}$ for which is asked to emit a class prediction ω amongst $l = 1, \ldots, L$ label in the problem. In turns, x is individually classified by the K models whose class estimations are collected in a vector of prediction $[y_1, \ldots, y_k]$. The final prediction $\hat{\omega}$ is obtained by counting the most frequent label according to a *Winner Takes All* policy where ties are broken randomly between the group of label assigned by the classifiers involved in the dispute. The system performance is evaluated as the ensemble classifier accuracy $\mathcal{J}^{\mathcal{C}}\left(\mathcal{D}^{(\mathrm{ts})}\right)$ in classifying patterns in the test set $\mathcal{D}^{(\mathrm{ts})}$.

5.4 Designing an Agent Based Classifier in Non-Geometric Space

In this section, Graph E-ABC [15] is proposed as a graph classification system which aims at extending the vector classifier (i.e. E-ABC) discussed in Section 5.3 toward the graph domain. If on one hand Graph E-ABC and E-ABC share the same principles and most of the designing scheme, the main purpose of Graph E-ABC is to build meaningful embedding spaces where the graphs are mapped in a vector representation. Hence, it is possible to spot two major differences between Graph E-ABC and its vector counterpart:

1. Different stage for selection of candidate substructures

2. Introduction of an alphabet synthesis phase for graph embedding

The rationale behind the approach followed by Graph E-ABC is to identify relevant substructures in order to exploit the Symbolic Histogram paradigm where granules, i.e. relevant substructures, collected in the alphabet set $\mathcal{A}$ are the key components of the procedure. Hence, the agents in charge for granulating data require an extractor block in order to sample candidate subgraphs from the data at hand. From this perspective, the optimization carried on by Graph E-ABC aims at building optimal alphabet sets in order to improve the quality of the embedding space according to the performance of a suitable classification model synthesized in the corresponding embedding space. Conversely, in the plain E-ABC approach, granules are intended as clusters of patterns that are directly sampled from the data without employing any substructure extraction since no embedding procedure is necessary. Nonetheless, at the heart of both systems relies a reward base mechanism for determining the quality of each agent activities. In both case, the agent quality Q_a is indeed conceived as measure that try to track the quality of the agents product, i.e. a cluster quality Q_c or a symbol quality Q_s respectively for E-ABC and Graph E-ABC. Both the qualities are in turn determined by a feedback signal that reflect the performance of the model in which they appear in.

The remainder of this section concerns the discussion of the Graph E-ABC scheme by describing in detail the main agents groups and the tasks assigned necessary for the synthesis and the optimization of the embedding spaces.

5.4.1 Granulator Agent Task Definition

The first action performed by a granulator agent a is to gather a set of subgraphs $\mathcal{S}^{(tr)}$ sampled from graphs from the training set $\mathcal{D}^{(tr)}$, where the number of subgraph $W = |\mathcal{S}^{(\mathrm{tr})}|$ is a user-defined parameter. The sampling process can be designed according to different graph traversal strategies which define the topology of the resulting subgraphs similar to those introduced in Section 4.1.

Once $\mathcal{S}^{(\mathrm{tr})}$ is ready, a can start the data mining process. The information extraction is performed according to the BSAS clustering algorithm working directly in the graph domain.

The adopted dissimilarity measure is the GED based nBMF algorithm as described in Section 3.2. Let recall $\mathbf{w} = [\mathcal{W} \quad \Pi^v \quad \Pi^e]$ that defines the overall dissimilarity parameters with $\mathcal{W} \in \{0,1\}^6$ is the tuple containing nodes and edges insertion, deletion, substitution weights, whereas Π^v, Π^e are the two tuple of binary parameters for the dissimilarity measures between nodes and edges (if applicable)[1]. The agent runs BSAS according to $d_{\mathbf{w}}$ and U, generating $\{\mathcal{P}_1, \ldots, \mathcal{P}_h\}$, i.e. a set of partitions, each of which is obtained for a given $\theta_i|_{i=1}^{h}$.

Each cluster $C \in \mathcal{P}_i$ undergoes the evaluation phase which determines the reliability $F(C)$ of the cluster according to two internal properties, namely its compactness $\Phi(C)$ and its cardinality $\Theta(C)$:

$$\Phi(C) = 1 - \frac{1}{|C|} \sum_{g \in C} d(g, g^*) \tag{5.13}$$

$$\Theta(C) = \frac{|C|}{|\mathcal{S}^{(\mathrm{tr})}|} \tag{5.14}$$

where g^* is the cluster representative of C, i.e. the MinSOD of the cluster. The reliability $F(C)$ reads as follows:

$$F(C) = \eta \cdot \Phi(C) + (1 - \eta) \cdot \Theta(C) \tag{5.15}$$

where η is a trade-off parameter that weights the importance of compactness against cardinality. According to a threshold τ_F, g^* can be promoted to be a symbol s or simply discarded as not relevant information. The evaluation phase is repeated for every cluster in every partition P_i, leading to a set of symbols $\mathcal{B} = \{s_1, \ldots, s_M\}$ discovered by agent a, where M depends on the number of symbols survived in the evaluation stage. For ease of notation, the granulation parameters are expressed in a more compact way in the vector $\mathbf{p} = [U \quad \tau_F \quad \eta]$.

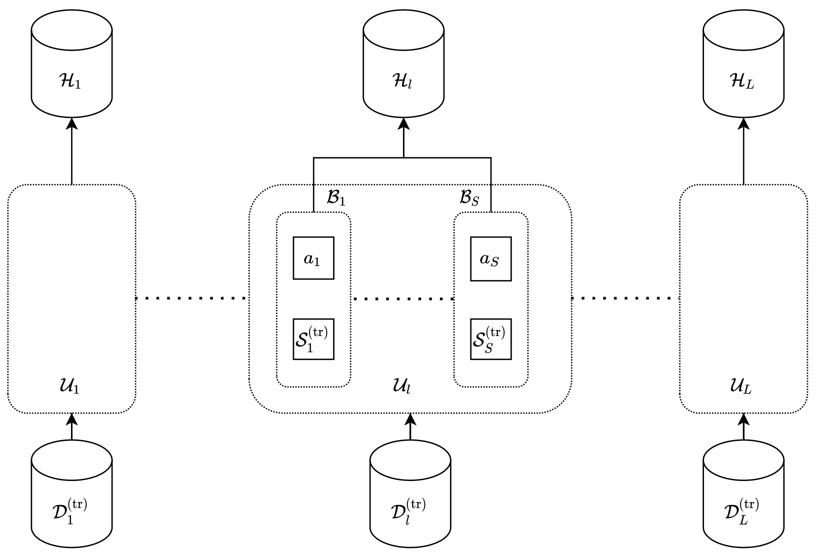

$\mathcal{H}_1$
$\mathcal{H}_l$
$\mathcal{H}_L$
$\mathcal{B}_1$
$\mathcal{B}_S$
a_1
a_S
$\mathcal{S}_1^{(\mathrm{tr})}$
$\mathcal{S}_S^{(\mathrm{tr})}$
$\mathcal{U}_1$
$\mathcal{U}_l$
$\mathcal{U}_L$
$\mathcal{D}_1^{(\mathrm{tr})}$
$\mathcal{D}_l^{(\mathrm{tr})}$
$\mathcal{D}_L^{(\mathrm{tr})}$

Figure 5.3. Schematic diagram for the granulation swarms of agent. Each swarm of granulator agents $\mathcal{U}_1, \dots, \mathcal{U}_L$ generates a class specific sets of symbols $\mathcal{H}_1, \dots, \mathcal{H}_L$ by merging all the agents symbols $\mathcal{B}_1, \dots, \mathcal{B}_S$.

5.4.2 Granulator Agent Swarm Behaviour

Individually, each swarm of agents operates on a class stratified training set of graphs $\mathcal{D}_1^{(\mathrm{tr})}, \dots, \mathcal{D}_L^{(\mathrm{tr})}$, where L is the number of classes available in the problem and $\bigcup_{l=1}^{L} \mathcal{D}_l^{(\mathrm{tr})} = \mathcal{D}^{(\mathrm{tr})}$. Then, a single agent $a \in \mathcal{U}_l$ with $l = 1, \dots, L$, receives a set of subgraphs $\mathcal{S}^{(\mathrm{tr})}$ sampled from $\mathcal{D}_l^{(\mathrm{tr})}$ which will be employed for the synthesis of information granulates as described in Section 5.4.1. It is worth noting that agents belonging to the same swarm $\mathcal{U}_l$ perform their granulation tasks relying on different subgraph sets, that is, they do not share the same view of the dataset but rather observe different aspects of the problem by drawing at random the sampled set of subgraphs $\mathcal{S}^{(\mathrm{tr})}$ with uniform distribution.

Once all the agents in a specific swarm $\mathcal{U}_l$ have completed their granulation and validation tasks, the symbols set extracted $\mathcal{B}_1, \dots, \mathcal{B}_S$ are finally collected in the class specific bucket $\mathcal{H}_l$, where S is the number of agents in the swarm. These sets will later serve as starting point for a distinct swarm of agent in charge of selecting key symbols that enable the embedding procedure by means of symbolic histograms. A schematic representation of the granulation agent swarm partitioning is given in Figure 5.3.

The agents separation in different teams combined with the definition of a single agent task (see Section 5.4.1) guarantee two fundamental aspects of a multi-agent based system: firstly, each agent is performing a very simple task from both a computational and data mining point of view; secondly, the agents have an incomplete knowledge of the other team's members. The former respects the autonomy principle typical of MAS and allows each agent to be easily implemented in a single processing

[1]Unlike the definition given in Section 3.2, in Graph E-ABC the parameters $\mathcal{W}$, Π^v and Π^e involved in the dissimilarity measure are binary valued elements in order to simplify the consensus strategy described in Section 5.4.3

unit in order to exploit the parallelism of a possible multi/many core architecture. The latter aspect is a crucial characteristic that a MAS should guarantee. Indeed, in case all agents had a clear view of the other agents' results and behaviour, they could in principle act in synchronous way, as if they were organized by a master controller [133].

5.4.3 Symbol Consensus

As discussed in the previous section, agent-based systems provide a certain degree of autonomy to the actors in play which have an incomplete view of the environment as well as the other agents results. Nonetheless, a communications mechanism is of utmost importance since it enables the cooperation between agents in order to improve the individual abilities by learning from each other relevant discoveries.

The candidate symbols collected in $\mathcal{H}_1, \ldots, \mathcal{H}_L$ and synthesized according to the discussion hold in Section 5.4.2 undergo a consensus phase which aims at exchanging information about the cluster validity and the dissimilarity parameter spaces $\mathbf{w}$ in which they are found. Initially, symbols belonging to a class specific bucket $\mathcal{H}_l$ are grouped together according to their $\mathbf{w}$. This is a mandatory step since symbols can be compared each other only if they share the same dissimilarity measure $d_{\mathbf{w}}$. Indeed, in case two symbols have been found with two different $\mathbf{w}$ parameters, this situation would arise an ambiguity in choosing the correct notion of dissimilarity to exploit for matching the select symbols. After that, the procedure moves to the next step, that consists in the definition of a matrix $\mathbf{M}$ that gives the information about the pairwise distances between the grouped symbols. Hence, $\mathbf{M}$ is an $n \times n$ matrix where n is the number of symbols in the selected group and $\mathbf{M}_{ij} = d_{\mathbf{w}}(s_i; s_j)$. By means of a threshold τ_{cns}, two symbols s_i, s_j are considered similar if the following constraint holds:

$$\mathbf{M}_{ij} \leq \tau_{cns} \tag{5.16}$$

Since in principle $d_{\mathbf{w}}$ is not a metric, the symmetry condition might not hold, leading to the situation where $\mathbf{M}_{ij} \neq \mathbf{M}_{ji}$. Clearly, this would lead inconsistent situations in case Eq. (5.16) holds for $\mathbf{M}_{ij}$ but not for $\mathbf{M}_{ji}$. For this reason, $\mathbf{M}$ is forced to be symmetric by applying the following transformation [113, 41]:

$$\mathbf{M}_{ij} = \mathbf{M}_{ji} = \frac{\mathbf{M}_{ij} + \mathbf{M}_{ji}}{2} \tag{5.17}$$

Finally, s_i and s_j are equally rewarded with a fixed value α by means of the following updating rule:

$$\begin{aligned} Q_s(s_i)^{(new)} &= Q_s(s_i)^{(old)} + \alpha \\ Q_s(s_j)^{(new)} &= Q_s(s_j)^{(old)} + \alpha \end{aligned} \tag{5.18}$$

Eventually, the procedure is repeated for those pair of symbols Eq. (5.16) holds. A complete pseudocode can be found in Algorithm 4.

5.4.4 Alphabet Selector Agent Behaviour

Once every granulator agent in the swarms $\mathcal{U}|_{l=1}^{L}$ has completed its data mining process, a class specific set of symbols is returned. The sets of symbols $\mathcal{B}$ synthesized

Algorithm 4 Reward similar symbols according to GED params

```
Input: $\mathcal{H}_l$: a set of symbols belonging to class $l$
Output: void
$\alpha$ := reward to assign to matched symbols
$\tau$ := define the minimum distance for matching two symbols
$\mathbf{w}$ := GED weights with which a symbol had been found
$d_{\mathbf{w}}$ := dissimilarity measure in graph domain parametric with $\mathbf{w}$
1: procedure CONSENSUS($\mathcal{H}_l$)
2:     groups := group symbol $\in \mathcal{H}_l$ according to $\mathbf{w}$
3:     for each group in groups do
4:         $\mathbf{M}$ = Pairwise distance matrix evaluate with $d_{\mathbf{w}}$ between symbols in group
5:         Find $i$, $j$ pairs satisfying $\mathbf{M}_{ij} \leq \tau$
6:         $Q_s(s_i)^{(new)} = Q_s(s_i)^{(old)} + \alpha$
7:         $Q_s(s_j)^{(new)} = Q_s(s_j)^{(old)} + \alpha$
8:     end for
9: end procedure
```

by each agent in a specific swarm $\mathcal{W}_l$ can be merged in a class-specific (i.e., swarm-specific) bucket of symbols $\mathcal{H}_l|_{l=1}^{L}$. In other words, $\mathcal{H}_l$ contains the collective information gathered by agents working on class l.

The procedure moves to the generation of candidate alphabets leveraging the buckets $\mathcal{H}_i$ in order to enable the graph embedding stage. This process is addressed by a separated population $\mathcal{Z}$ of N individuals, whose actions can be summarized as follows:

1. The agent $z \in \mathcal{Z}$ evaluates the maximum number of symbols per class $t = T/L$ that can be extracted according to a user-defined bound T
2. The agent explores the buckets $\mathcal{H}_l$ and extracts uniformly at random at most t symbols
3. When all the classes are explored, the selected symbols are collected into the multi-class alphabet of symbols $\mathcal{A}_z$.

The procedure is repeated for all the individuals $z \in \mathcal{Z}$ belonging to the selector agent swarm. In this way, each agent synthesizes a candidate alphabet set $\mathcal{A}_z$ that enables a graph embedding procedure. In particular, being N the number of individual in $\mathcal{Z}$, the procedure leads to the synthesis of $\mathcal{A}_1, \ldots, \mathcal{A}_N$ candidate alphabets.

5.4.5 Alphabets Evaluation

According to the symbolic histogram approach (see Section 3.3.3), the alphabets can now be exploited for building the vectorial representation of both training and validation sets ($\mathcal{D}^{(\mathrm{tr})}$ and $\mathcal{D}^{(\mathrm{vs})}$). Building the symbolic histograms of all graphs in $\mathcal{D}^{(\mathrm{tr})}$ and $\mathcal{D}^{(\mathrm{vs})}$ leads to the definition of $\mathbf{F}_i^{(\mathrm{tr})}$ and $\mathbf{F}_i^{(\mathrm{vs})}$, respectively an $|\mathcal{D}^{(\mathrm{tr})}| \times |\mathcal{A}_i|$ and an $|\mathcal{D}^{(\mathrm{vs})}| \times |\mathcal{A}_i|$ matrix, whose rows are the symbolic histograms obtained with

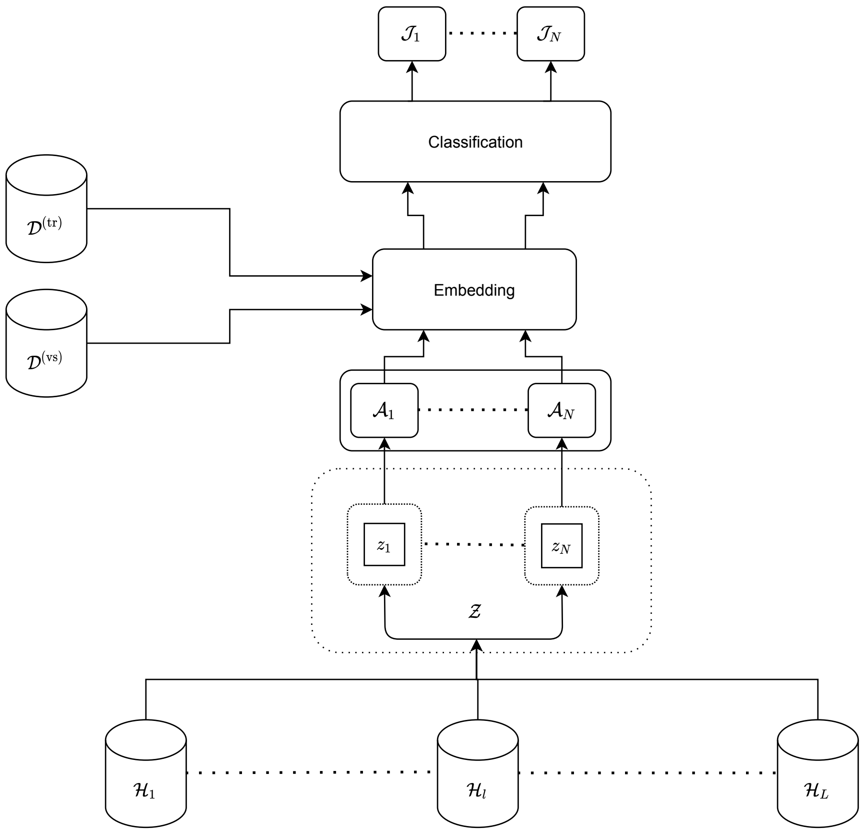


Figure 5.4. Schematic diagram for the selector agents population. Agents $z_1, \dots, z_N$ generates $\mathcal{A}_1, \dots, \mathcal{A}_N$ alphabets by sampling symbols from class-specific buckets $\mathcal{H}_1, \dots, \mathcal{H}_L$. The new alphabets are exploited for embedding training and validation set, respectively $\mathcal{D}^{(\text{tr})}$ and $\mathcal{D}^{(\text{vs})}$ into N different embedding spaces. N classifiers are trained in the corresponding vector spaces and their performances $\mathcal{J}_1, \dots, \mathcal{J}_N$ on classifying $\mathcal{D}^{(\text{vs})}$ are retained.

the i^{th} alphabet, for $i = 1, \dots, N$. A set of classification models $c_1, \dots, c_N$ can be build according to specific embedding spaces:

1. Train a classifier c_i in the respective embedding space $\mathcal{F}_i$ spanned by the symbolic histograms matrix $\mathbf{F}_i^{(\text{tr})}$
2. Test the classifier by predicting the embedded validation set $\mathbf{F}_i^{(\text{vs})}$
3. Evaluate the performance measure $\mathcal{J}_i$ of c_i in classifying $\mathbf{F}_i^{(\text{vs})}$ as the following linear convex combination:

$$\mathcal{J}_i = \sigma \cdot \omega_i + (1 - \sigma) \cdot \left(1 - \frac{|\mathcal{A}_i|}{|\mathcal{A}_{max}|}\right) \tag{5.19}$$

In Eq. (5.19) is possible to spot two terms weighted by the σ parameter that contribute in the definition of $\mathcal{J}_i$: the left-most takes into account the accuracy ω_i obtained by the c_i classifier in correctly classifying $\mathbf{F}_i^{(\text{vs})}$. The right-most term, as instead, is a correction factor that emphasizes alphabets with lower symbols. Indeed, the alphabet cardinality $|\mathcal{A}_i|$ determines the dimensionality of the embedding space where the classifier is trained, i.e. the number of features employed for the vector representation of a graph. The term $|\mathcal{A}_{max}|$ is defined instead as the cardinality of the largest alphabet being part of the elite pool $\mathcal{Z}^*$ of selector agents $\mathcal{Z}$. That is, $\mathcal{Z}^*$ is a set containing the top-Z most performative alphabets observed so far. In this way, $\mathcal{J}_i$ is simultaneously measuring the classification performance and the embedding space sparsity in which the classifier has been trained and tested. Eventually, the resulting classifiers, $c_1, \dots, c_N$ are retained together with their performance measures $\mathcal{J}_1, \dots, \mathcal{J}_N$.

5.4.6 Agents and Symbols Quality Propagation

The quality of a single symbol $Q_s(s)$ shall reflect the performances $\mathcal{J}$ of the alphabets in which the symbol s under analysis appears in. A lookup table is built in order to indicate the update value β that will be assigned to s via the following two steps:

1. Since the performance measure assumes values in range $[0, 1]$, the $[0, 1]$ range is uniformly discretized into a finite number of bins
2. each performance bin is mapped with a reward value, also uniformly discretized into the same number of bins, where the admissible range $[\beta_{\min}, \beta_{\max}]$ is user-configurable.

The quality update strategy works as follows:

1. Select the candidate alphabet $\mathcal{A}$
2. Select the symbol $s \in \mathcal{A}$
3. Find the performance bin in which $\mathcal{J}$ lies and gather the corresponding reward value β
4. Reward the symbols by applying the following update rule:
$$Q_s^{(new)}(s) = Q_s^{(old)}(s) + \beta \tag{5.20}$$
5. Repeat from step 2 for all symbols in the selected alphabet $\mathcal{A}$.
6. Repeat from step 1 for all the alphabets under evaluation.

In this approach, we interpret the performance $\mathcal{J}$ as a critic about the effectiveness of the embedding space spanned by the alphabet $\mathcal{A}$. In this way, the quality $Q_s(s)$ can be seen as a measure for determining if the symbol s is useful for building a valuable alphabet which can attain high level of performance. The complete pseudo-code can be found in Algorithm 5

The agent quality measure Q_a is defined according to the qualities $Q_s(s_1), \dots, Q_s(s_M)$ and the reliabilities $F(C_1), \dots, F(C_M)$, where $C_1, \dots, C_M$ are the clusters related to $s_1, \dots, s_M$ symbols the agent a has found (see Section 5.4.1). The overall procedure for assigning Q_a can be break down in the following steps:

1. Select the agent a from $\mathcal{U}_l$

2. According to the agent set of symbols $\mathcal{B}$, evaluate the mean quality:[2]

$$\overline{Q}_{\mathcal{B}} = \frac{1}{|\mathcal{B}|} \sum_{s \in \mathcal{B}} Q_s(s) \tag{5.21}$$

3. According to the agent set of symbols $\mathcal{B}$ evaluate the mean reliability:

$$\overline{F}_{\mathcal{B}} = \frac{1}{|\mathcal{B}|} \sum_{C \in \mathcal{B}} F(C) \tag{5.22}$$

4. Set agents quality Q_a as follow:

$$Q_a(a) = \rho \cdot \overline{Q}_{\mathcal{B}} + (1 - \rho) \cdot \overline{r}_{\mathcal{B}} \tag{5.23}$$

5. Repeat from step 1 for all agents $a \in \mathcal{U}_l$ with $l = 1, \ldots, L$.

In Eq. (5.23), ρ weights the importance between the symbols qualities and its reliabilities. In particular, in the early generations of the evolution, we give more importance to the right hand term in order to initially synthesize well-formed clusters. Next, the quality of an agent shall be better described as its ability in finding informative symbols according to Q_s, since the final scope of the whole procedure is devoted to synthesize meaningful alphabets employed for the classification problem. Indeed, this information is contained in symbols qualities Q_s whose values are actually backtracked in the agent quality Q_a.

Algorithm 5 Symbols reward routine

```
    Input: alphabets: a list of alphabets, performances: a list of performances associated to each element of alphabets
    Output: void
    $\beta_{min}$ := reward value lower bound
    $\beta_{max}$ := reward value upper bound
    $d$ := number of reward value in the table
 1: procedure REWARD_SYMBOLS(alphabets,performances)
 2:     $\mathbf{V} = [\beta_{min}, \ldots, \beta_{max}] \in \mathbb{R}^d$    ▷ Equally space $d$-length vector of rewards/penalties
 3:     $\mathbf{T} = [0, \ldots, 1] \in \mathbb{R}^d$    ▷ Equally space $d$-length vector for looking up the reward to assign
 4:     for alphabet,J in (alphabets, performances) do
 5:         for each symbol $s$ in alphabet do
 6:             Find $i$ such that $\mathbf{T}[i-1] < J \leq \mathbf{T}[i]$ with $i = 1, \ldots, d-1$
 7:             $\beta = \mathbf{V}[i]$
 8:             $Q_s^{(new)}(s) = Q_s^{(old)}(s) + \beta$
 9:         end for
10:     end for
11: end procedure
```

[2] Q_s is normalized in $[0, 1]$ according to the symbols qualities observed in $\mathcal{H}_l$.

5.4.7 Evolving Agents

The agents optimization of swarms $\mathcal{U}|_{l=1}^{L}$ is driven by a genetic evolution. As stated in Section 5.4.1, each swarm $\mathcal{U}$ must be able to synthesize a candidate set of symbols $\mathcal{H}$ that will be later employed in the formation of pivotal alphabets $\mathcal{A}$. For this reason, the genetic code a_{code} of each agent reads as follows:

$$a_{\text{code}} = [\mathbf{p} \quad \mathbf{w}] \tag{5.24}$$

which summarizes the crucial parameters for the information extraction described in Section 5.4.1. Agents in the same swarm follow a classic $(\mu + \lambda)$ selection scheme [21], where λ offsprings are generated according to common genetic operators applied to the parents population μ, i.e. mutation, crossover and random spawn of new individuals. The optimization aims at maximizing the agent fitness function f_a defined as the quality Q_a given in Eq. (5.23):

$$f_a(a) = Q_a(a) \tag{5.25}$$

The classification models optimization is an evolutionary-like procedure specifically designed according to the unconventional nature of the individual's genetic code z_{code}. Indeed, z_{code} is a direct representation of a specific alphabet $\mathcal{A}$, whose cardinality is not fixed a priori, rather depends on a uniformly distributed at random variable bounded in $R \in [L, T]$. The genetic code z_{code} reads as follow:

$$z_{\text{code}} = \mathcal{A} = [s_1, \ldots, s_R] \tag{5.26}$$

where $R = |\mathcal{A}|$. The fitness function $f_z(z)$ reflects the critic obtained by the classifier c trained in the embedding space built according to $\mathcal{A}$, i.e. the performance measure $\mathcal{J}$ the classifier c attained on the validation set:

$$f_z(z) = \mathcal{J}\left(z_{code}\right) = \mathcal{J}\left(\mathcal{A}\right) \tag{5.27}$$

Given the set of elite individuals $\mathcal{Z}^*$, i.e. top-Z individuals evaluated so far, the offspring $\hat{\mathcal{Z}}$ is generated according to custom operators defined as follows:

Crossover: two individuals z_i and z_j are selected uniformly at random from $\mathcal{Z} \cup \mathcal{Z}^*$. A cut point is determined according to the shortest code between the two and a new individual is created according to a one-point crossover operator.

Union: two individuals z_i and z_j are selected uniformly at random from $\mathcal{Z} \cup \mathcal{Z}^*$ and eventually a new individual is defined as the union set $z_i \cup z_j$.

Mutation: a single individual z_i is uniformly at random extracted from $\mathcal{Z} \cup \mathcal{Z}^*$. With a given probability p_{mut}, each symbol $s \in z_i$ has the chance to be swapped with a symbol belonging to a randomly extracted individual $z_j \in \mathcal{Z} \cup \mathcal{Z}^*$.

The recombination process is repeated until $\hat{\mathcal{Z}}$ is populated with K different individuals. The whole evolutive orchestration can be schematically described as follows:

1. Run the granulator agent swarms $\mathcal{U}|_{l=1}^{L}$ according to Section 5.4.1

2. Collect the agent symbols into $\mathcal{H}|_{l=1}^{L}$ class-specific buckets
3. Perform symbol consensus according to Section 5.4.3 for each class-specific bucket $\mathcal{H}|_{l=1}^{L}$
4. Generate the current selector agent swarm $\mathcal{Z}$ according to Section 5.4.4
5. Generate $\hat{\mathcal{Z}}$ by recombination of $\mathcal{Z}$ and $\mathcal{Z}^*$
6. Evaluate $\mathcal{Z}$ and $\hat{\mathcal{Z}}$ according to Section 5.4.5
7. Reward the symbols in $\mathcal{Z} \cup \hat{\mathcal{Z}}$ according to Section 5.4.6
8. Evaluate the agents fitnesses for the swarms $\mathcal{U}|_{l=1}^{L}$ according to Eq. (5.25)
9. Evolve the granulator agent swarms $\mathcal{U}|_{l=1}^{L}$
10. Replace $\mathcal{Z}^*$ by selecting the top-Z individuals amongst $\mathcal{Z} \cup \hat{\mathcal{Z}} \cup \mathcal{Z}^*$ according to Eq. (5.27)

The latter procedure highlights how the two different swarms $\mathcal{U}|_{l=1}^{L}$ and $\mathcal{Z}$ cooperate together. To summarize, the agents in the swarms observe new sampled data at each iteration in order to detect useful information and improve their ability thanks to genetic optimization that tries to maximize the quality of the agents' output, i.e. their symbols. On the contrary, the population $\mathcal{Z}$ explores possible combinations of agents' outputs for testing prospective embedding spaces and simultaneously provides a supervised critic about the quality of the agents' output. The recombination of $\mathcal{Z}$ with $\mathcal{Z}^*$ can be considered as an exploitation phase where the most effective individuals observed so far are mixed with just-explored ones in order to possibly generate improved alphabets. Finally, $\mathcal{Z}^*$ is intended to be created according to the selection pressure held by the limited environment space, that is, only the best Z individuals survive and will be part of the next generation.

5.4.8 Test Set Evaluation in Embedding Spaces

After the evolution converges and/or reach the maximum number of generations N_{stop}, the final elite population $\mathcal{Z}^*$ can be exploited for evaluate the solutions obtained on a graph test set $\mathcal{D}^{(\text{ts})}$. Recalling that each $z^*_{code} \in \mathcal{Z}^*$ is an alphabet of symbols $\mathcal{A}$, it is possible to build the symbolic histogram matrix $\mathbf{F}^{(\text{ts})}$ of test set and train a classifier c according to $\mathbf{F}^{(\text{tr})}$, where as usual $\mathbf{F}^{(\text{tr})}$ and $\mathbf{F}^{(\text{ts})}$ are respectively an $|\mathcal{D}^{(\text{tr})}| \times |\mathcal{A}|$ and an $|\mathcal{D}^{(\text{ts})}| \times |\mathcal{A}|$ matrix. By repeating the embedding procedure for all the solutions in $\mathcal{Z}^*$, $c_1, \dots, c_Z$ classifiers are placed in ensemble in order to possibly exploit simultaneously all the information carried by the different embedding space spanned by the symbolic histograms matrices. The final performance of the system is obtained by equipping the ensemble with a winner-takes-all policy rule. That is, each classifier in ensemble emits the label for the symbolic histogram belonging to the test set under analysis. Afterwards, the most voted label is retained as the final prediction.

5.5 Limitations and Discussion with GRALG Approach

In this chapter, Graph E-ABC has been presented as a prototypical architecture following the MAS philosophy. From the discussion held in Section 5.2, it is possible to spot relevant analogies with the methodology followed by GRALG (see Chapter 3).

In the first place, it is worth noticing that both systems show a common trait of flexibility under different point of views. The most fascinating facet of both systems is their adaptability to different input domains. The approach followed by GRALG specifically for the graph domain has been revisited for dealing with heterogeneous input spaces in order to solve different problems such as sequence and text classification, time-series predictions and hypergraphs embedding [159, 110, 36, 123, 114]. Similarly, E-ABC can also deal with different data structures since it shares most of the building blocks with GRALG: substructure extractor, granulator and an embedding phase. The chance of achieving such a high level of generalization is readily explainable by the embedding strategy both systems share, i.e. the *symbolic histograms* paradigms [53]. Furthermore, both GRALG and E-ABC are highly customisable algorithms thanks to their modular design: the granulator block can be expressed with a variety of different methods not limited to clustering provided that a dissimilarity measure in the input domain and the granule type is still reasonably provided; the classification block in the vector space can be chosen freely since it has only to provide a suitable performance measure. Another important similarity that can be noticed regards the adoption of Granular Computing paradigm. Indeed, E-ABC shares with GRALG the idea that relevant information can be synthesized by exploring the data at different level of abstraction according to the Granular Computing paradigm. Hence, both E-ABC and GRALG rely on pivotal structures, i.e. the *granules* or *symbols*, that can be combined together into the *alphabet* with the ultimate goal to enable the systems with prediction capabilities. Additionally, when compared to other State of the Art methods for graph classification, Graph E-ABC and in general GrC based techniques as GRALG, can be exploited for building efficient mapping functions based on automatically-extracted information granules able to reflect the information carried by the structured data into a vector space and, at the same time, build an interpretable model for field experts. Granular Computing based techniques are able to merge the most desirable facets among current approaches for graph classification: building the embedding space is automatic (e.g., as in deep learning-based models and conversely to the feature engineering case), the embedding space is also interpretable (e.g., as in dissimilarity spaces and conversely to deep learning-based models) and the embedding space is explicit (e.g., conversely to kernel methods). Finally, it is worth stressing the main two-fold advantages of GrC-based embedding with respect to the plain dissimilarity representation one:

1. in the embedded space synthesized via GrC, meaningful dissimilarity measures can be defined even in the case it is needed to compare pairs of graphs with a large difference in their number of nodes and/or edges;

2. meaningful symbols used for embedding can greatly improve the knowledge discovery capability of the overall classification system.

Another important aspect the two methods have in common is the automatic selection of the correct dissimilarity $d_{\mathbf{w}}$ intended as a form of *metric learning*. Intuitively, this commonality is determined by noticing that both approaches share the same interpretation of GrC which extensively relies on the notion of a dissimilarity measure for determining the correct level of abstraction. On the other hand, E-ABC introduces several peculiarities mostly inherited from the MAS paradigm that in GRALG are not present:

1. Agents cooperation against individualistic behavior
2. Task Agent Separations
3. Local Metric Learning versus Global/Class Specific Metric Learning

Concerning #1, E-ABC follows a 'cooperative approach', where different swarms of agents exploit independent portions of the dataset in order to search for suitable symbols and later they join forces via another set of agents for building the classification model. This aspect clearly reflects the important MAS facet which envisages that the agents do not share the same view of the surrounding environment. On the contrary, the GRALG perspective follows a more 'individualistic approach' where all individuals start their granulation process from the same data shard sampled from the training set that does not change throughout the evolution. Furthermore, each individual independently looks for suitable granules of information, build its own model and trains the classifier accordingly. With such approach, in GRALG, the fitness function can be determined straightforwardly by the classifier performances since it is unambiguously matched with individual in the genetic algorithms population. Nevertheless, the individuals will only focus in improving their granulation abilities and model performances according to a fixed granulation bucket, limiting the exploration aspect of the evolutionary strategy. In E-ABC, the advantage of continuously exploring a possibly dynamic environment comes at the cost of a more complex procedure for establishing the alphabet effectiveness and consequently agents work evaluation. Indeed, agents need to deploy specific strategies of negotiation and cooperation between each other for grabbing additional information that can reinforce and expand the knowledge about the problem at hand. This aspect is achieved through a specific *consensus* mechanism that highlights which granules of information are valuable and affordable according to different agents opinion. Additionally, the task separation in #2 between groups of agents deployed in Graph E-ABC introduced an interesting novelty regarding the optimization scheme. Recall that GRALG is characterized by two sequential different levels of optimization, that are the alphabet synthesis (see Section 3.4.1) and the selection of relevant symbols for the classification task (see Section 3.4.2). On the contrary, Graph E-ABC leverages on a group of agents (the alphabet selector agents defined in 5.4.4) that are specialized in generating alphabets by considering simultaneously the performance in the resulting embedding space and the corresponding dimension (cf. Eq. (5.19)). Hence, the feature selection is no longer performed in a distinct phase rather it is carried on to some extent by the alphabet selector agents concurrently with the alphabet synthesis. As the #3 is concerned, the first proposed version of GRALG described in Chapter 3 and in [23] performed a global metric learning method based on a plain genetic optimization. That is, it is able to discover a single set of dissimilarity parameters valid for all the data at hand. Later on, we proposed the variant described in Section 4.3, where the system has been upgraded by enabling a *class-specific* metric learning approach. On the contrary in E-ABC, each symbol carries on the information about the dissimilarity $d_{\mathbf{w}}$ with which it has been found, moving closer to a local metric learning paradigm that allows a finer characterization of the symbols composing the final classification model.

However, it is worth to underline the prototypical aspect of the discussed implementation. Indeed, Graph E-ABC should be considered as a first attempt to condense the MAS general principles with explicit graph embedding method based on information granulation. In light of these observations, the design of some specific building blocks has been intentionally kept simple in order to evaluate the system as a whole. For example, the consensus strategy described in Section 5.4.3 is based on the mutual distances between symbols sharing the same dissimilarity parameters $\mathbf{w}$. However, the possibility of grouping symbols together according to $\mathbf{w}$ has been possible only by simplifying the parameter space from real-valued to binary weights. If on one hand this allows to easily group symbols together according to $\mathbf{w}$, on the other hand it could negatively impact on the description power of the graph dissimilarity measure. Further, another delicate aspect which should be investigated relates to how the quality of the symbols is evaluated. In fact (recall from Section 5.4.6), the quality of each symbol is evaluated independently to the one of other symbols: this approach does not allow to capture the correlation amongst symbols, that is, whether there exist 'groups of symbols' that are responsible for a fruitful embedding space. Another point which deserves attention regard the definition of the symbol reward (see Section 5.4.6). In fact, suitable reward values may change drastically from one dataset to another and a poor user-defined choice can undermine not only the symbol quality (see Eq. (5.20)), but also the agent quality (see Eq. (5.23)). A suitable countermeasure would be using adaptive strategies for populating the reward values in the lookup table. This would also limit the huge number of free-parameters to be defined by the end-user.

Chapter 6

Experiments

6.1 Dataset Description

For the experimental validation of the proposed methods, six publicly available datasets taken from the IAM repository [152] are employed. Each dataset is split in three disjoint training, validation and test sets, which have been kept unchanged with respect to the ones available in the data repository. Brief statistics about the datasets can be found in Table 6.1. Since graphs are labelled both on nodes and edges with arbitrary data structures, in the following the node and edge dissimilarities for each of the six datasets is formally described.

The AIDS dataset consists in a set of graphs describing molecular compounds that show activity or not against HIV. Node labels contain data about the chemical symbol S_{chem}, numbers of charge N_{ch} and a vector $\mathbf{v}$ of (x, y)-coordinates, whilst edges are labelled with the valence of the linkage. Node dissimilarity is thus defined as:

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

where $d_s(S_{chem}^{(a)}, S_{chem}^{(b)}) = 1$ if $S_{chem}^{(a)} = S_{chem}^{(b)}$, 0 otherwise. Conversely edge information is discarded since not useful for the classification task.

GREC represents graphs from architectural and electronics drawing symbols. Each node is equipped with the information containing the type of structures (*type*) that compose the drawing and a 2-dimensional vector $\mathbf{v}$ which assesses its corresponding positions. Conversely, edges encode the information about the connection between the structure with a pair *(type,angle)*: a string value (*line* or *arc*) and a real number,

Table 6.1. Characteristic of IAM datasets used for testing: size of Training (tr), Validation (vl) and Test (ts) set, number of classes (*# classes*), types of nodes and edges labels, average number of nodes and edges, whether the dataset is uniformly distributed amongst classes or not (*Balanced)*.

Database	size (tr, vl, ts)	# classes	node labels	edge labels	Avg # nodes	Avg # edges	Balanced
Letter-L	750, 750, 750	15	$\mathbb{R}^2$	none	4.7	3.1	Y
Letter-M	750, 750, 750	15	$\mathbb{R}^2$	none	4.7	3.2	Y
Letter-H	750, 750, 750	15	$\mathbb{R}^2$	none	4.7	4.5	Y
GREC	286, 286, 528	22	string + $\mathbb{R}^2$	tuple	11.5	12.2	Y
AIDS	250, 250, 1500	2	string + integer + $\mathbb{R}^2$	integer	15.7	16.2	N
Mutagenicity	1500, 500, 2337	2	strings	integer	30.3	30.8	N

respectively. Additionally, a frequency attribute *freq* defines the number of pairs which appear in the edge label. Custom dissimilarities on both nodes and edges for GREC are defined as follows:

$$d_v(v^{(a)}, v^{(b)}) = \begin{cases} (1-\chi_1) \cdot \frac{1}{\sqrt{2}} \|\mathbf{v}^{(a)} - \mathbf{v}^{(b)}\|_2 \\ \quad \text{if } \mathit{type}^{(a)} = \mathit{type}^{(b)} \\ \chi_1 + (1-\chi_1) \cdot \frac{1}{\sqrt{2}} \|\mathbf{v}^{(a)} - \mathbf{v}^{(b)}\|_2 \\ \quad \mathit{otherwise} \end{cases}$$

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

$$d_e(e^{(a)}, e^{(b)}) = \begin{cases} \chi_2 \cdot d^{line}(angle^{(a)}, angle^{(b)}) \\ \quad \text{if } \mathit{type}^{(a)} = \mathit{type}^{(b)} = line \\ \chi_3 \cdot d^{arc}(angle^{(a)}, angle^{(b)}) \\ \quad \text{if } \mathit{type}^{(a)} = \mathit{type}^{(b)} = arc \\ \chi_3 \quad \text{otherwise} \end{cases}$$

2. If $freq^{(a)} = freq^{(b)} = 2$

$$d_e(e^{(a)}, e^{(b)}) = \begin{cases} \frac{\chi_2}{2} \cdot d^{line}(angle_1^{(a)}, angle_1^{(b)}) + \\ + \frac{\chi_3}{2} \cdot d^{arc}(angle_2^{(a)}, angle_2^{(b)}) \\ \quad \text{if } \mathit{type}^{(a)} = \mathit{type}^{(b)} = line \\ \frac{\chi_2}{2} \cdot d^{line}(angle_2^{(a)}, angle_2^{(b)}) + \\ + \frac{\chi_3}{2} \cdot d^{arc}(angle_1^{(a)}, angle_1^{(b)}) \\ \quad \text{if } \mathit{type}^{(a)} = \mathit{type}^{(b)} = arc \\ \chi_3 \quad \text{otherwise} \end{cases}$$

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

$$d_e(e^{(a)}, e^{(b)}) = \chi_4$$

where d_{line} and d_{arc} are module distances normalized in $[-\pi, \pi]$ and $[0, arc_{max}]$ respectively. Additionally, $\Pi_v = \{\chi_1\}$ and $\Pi_e = \{\chi_2, \chi_3, \chi_3, \chi_4\}$ are the sets of real-valued parameters ranging in $[0, 1]$ for nodes and edges dissimilarity needed for evaluating the GED as described in Section 3.2.

The three LETTER datasets contain undirected graphs representing drawing of Roman capital letters with increasing amount of distortion: low (LETTER-L), medium (LETTER-M) and high (LETTER-H). Node features are 2-D real-valued vectors **v** which define the node positions in a reference system, whilst edges are not labelled. Hence, only the node dissimilarity has to be defined:

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

Finally, MUTAGENICITY is a set of graphs representing molecular compound whose classes are their mutagenicity tendencies. Nodes are chemical symbols S_{chem} and edges the valence number B_{val} of the linkage.

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

Table 6.2. Exhaustive number of subgraphs extracted from $\mathcal{D}^{(\mathrm{tr})}$.

Type	Letter-L	Letter-M	Letter-H	GREC	AIDS	Mutagenicity
Path ($o = 5$)	8193	8582	21165	27119	35208	652642
Clique	2377	2398	2493	3321	3961	48234

$$d_e(e^{(a)}, e^{(b)}) = \begin{cases} 1 & \text{if } B_{val}^{(a)} = B_{val}^{(b)} \\ 0 & \text{otherwise} \end{cases}$$

6.2 Tests and Results for GRALG Classifier

6.2.1 Stochastic Extraction Method Evaluation

In this section, the proposed lightweight extractors are evaluated and compared against the original exhaustive implementation discussed in Section 3.3.1. In particular, both BFS and DFS extractors (see Section 4.1) and clique extractor (see Section 4.1.1) are considered for the tests. The sample size W for the subgraph set $\mathcal{S}^{(\mathrm{tr})}$ is chosen as follow:

- for BFS and DFS, let $W = 10\%, 30\%, 50\%$ of the number of subgraphs extracted exhaustively
- for cliques, let $W = 40\%, 60\%, 80\%$ of the number of maximal cliques enumerated using Bron-Kerbosch

where the total numbers of subgraphs extracted exhaustively are shown in Table 6.2. From this table, it is possible to observe a clear-cut reduction in the total number of subgraphs when cliques are selected as candidate substructures. This consideration justifies the different sampling rates W used for the comparison.

It is worth remarking that when a BFS or DFS 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 clique orders are strictly topology-related rather than user-defined.

For the classification block in the embedding space (see Section 3.3.4) a k-NN classifier is selected. Due to the intrinsic randomness in the training procedures, results herein presented have been averaged across 10 runs. Other relevant parameters are chosen as follows:

- $k = 5$ (number of neighbours for k-NN)
- $o = 5$ maximum subgraph order for BFS and DFS extractors
- 20 individuals per population (both genetic algorithms)
- 20 generations (first genetic algorithm – alphabet optimization)

- 100 generations (second genetic algorithm – feature selection)
- $\sigma = 0.99$ in the fitness function for the second genetic algorithm (very minor weight to sparsity)
- $\xi = 1.1$ as tolerance value for the symbolic histograms evaluation.
- $U_{max} = 500$ upper bound for BSAS maximum number of clusters.

In order to investigate how the subsampling rates and the subgraph topologies impacts on GRALG, the analysis takes into account three different performance measures:

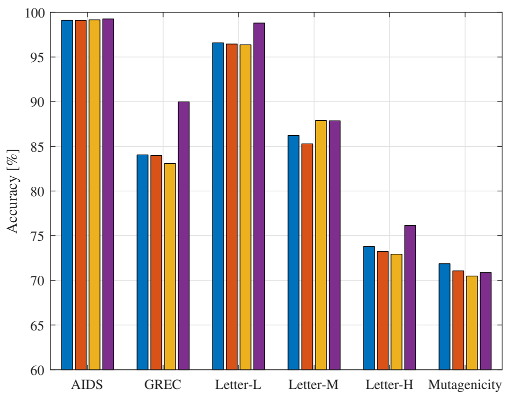
- Accuracy in correctly classifying $\mathcal{D}^{(\mathrm{ts})}$ in the optimized embedding space shown in Fig. 6.1
- Wall clock running times[1] of the overall optimization, shown in Fig. 6.2
- Optimal alphabet cardinality $|\mathcal{A}^*|$ shown in Fig. 6.3 which corresponds to the optimized embedding space dimensionality.

By matching Figures 6.1a and 6.1b, it is possible to see that BFS and DFS strategies lead to comparable results (in terms of accuracy) with those obtained by the exhaustive procedure for every value of W. The only remarkable shift can be observed for Grec (approximately 7%). It is worth remarking that the performances of the classification block are strongly influenced by the efficiency of the mapping function in preserving the graph input space properties into the embedding space. This can be achieved only if the information granules extracted are indeed meaningful representatives of the considered dataset(s). For most of the datasets, clearly some properties emerge even by performing a strong subsampling of the prospective subgraphs.

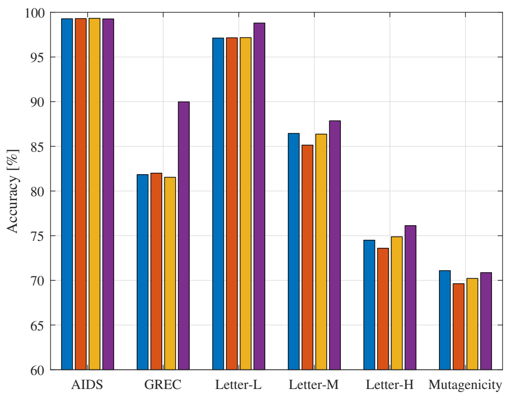
For what concerns the clique extraction strategy, in Fig. 6.1c, AIDS and Letter-L show comparable levels of accuracy with respect to both BFS and DFS extractors, respectively in Fig. 6.1a and Fig. 6.1b. Two remarkable results should be underlined: in Grec and Mutagenicity datasets, it is possible to spot a generally increased accuracy for any considered sample rate. Indeed, for Grec an improvement of about 8% is achieved by the clique extractor, whereas Mutagenicity witnessed a benefit of 4% suggesting that cliques are a relevant topology for these problems. Conversely, when the clique extractor acts on Letter-M and Letter-H, the accuracy is strongly worsen, suggesting that topologies emerging from BFS and DFS extractor 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').

From the point of view of running times, the introduction of sampling strategies show their effectiveness by observing Fig. 6.2a and Fig. 6.2b. This is due to the

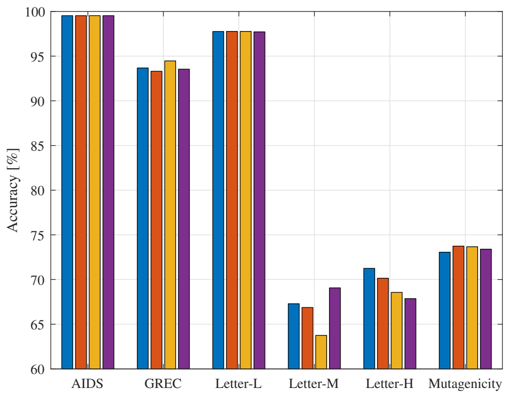
[1]In order to guarantee fair running times comparison, all tests have been performed on the same workstation running Linux Ubuntu 18.04 equipped with hyperthreaded 4-core Intel i7-3770K @3.50GHz and 32GB of RAM.



(a) BFS



(b) DFS

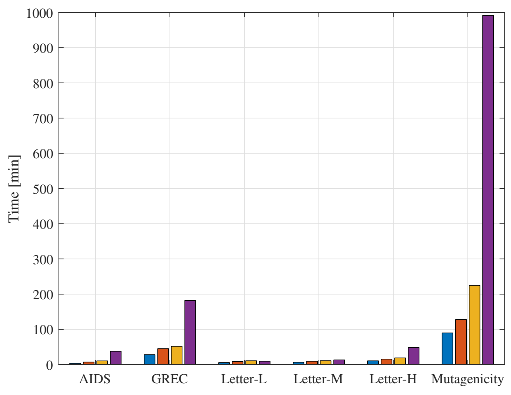


(c) Clique

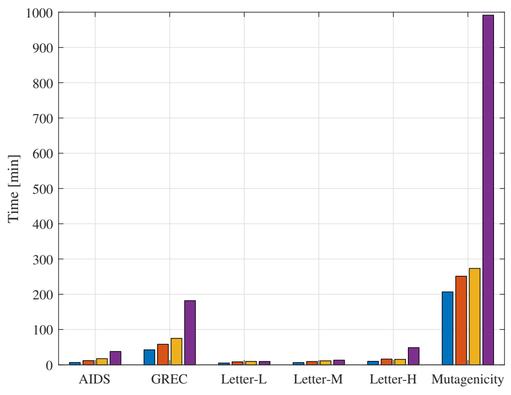
Figure 6.1. Accuracy results for different extraction strategies. Blue, red and yellow bars correspond to $W = 10\%, 30\%, 50\%$ sampling rates for BFS and DFS extractor whilst they represent $W = 40\%, 60\%, 80\%$ for clique extractor. Purple bars refer to exhaustive procedures.

lower number of subgraphs returned by the *Extractor* driving mainly the *Granulator* and due to the traverse strategy adopted by the *Embedder* before the evaluation of the symbolic histograms. Recalling Section 3.4.1, the genetic algorithm must repeat several times the entire procedure of granulation, embedding and classification in order to optimize the parameters involved. This task involves the GED computation many times, which can be very intensive and time consuming. By matching Table 6.1 and Figures 6.2a–6.2b, clearly the advantages of subsampling are more and more evident as the dataset size increases and/or in presence of complex semantic information on nodes/edges, as their dissimilarity measures impact the overall GED computational burden.

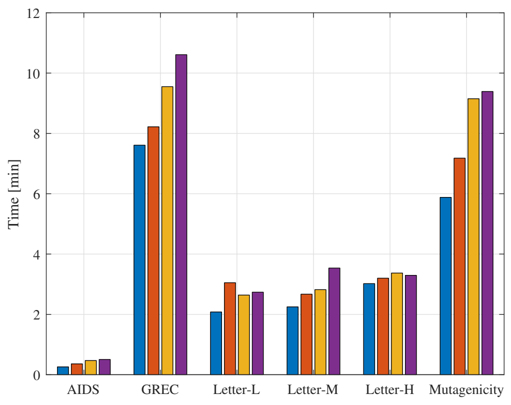
A major improvement achieved by the clique extractor can be spotted by observing results in Fig. 6.3c in all configurations (i.e., regardless of the subsampling percentage from the set of all maximal cliques). The number of symbols is significantly reduced when compared to BFS and DFS extractors respectively in Fig. 6.3a and Fig. 6.3b. Considering that every symbol in the alphabet must be matched



(a) BFS



(b) DFS



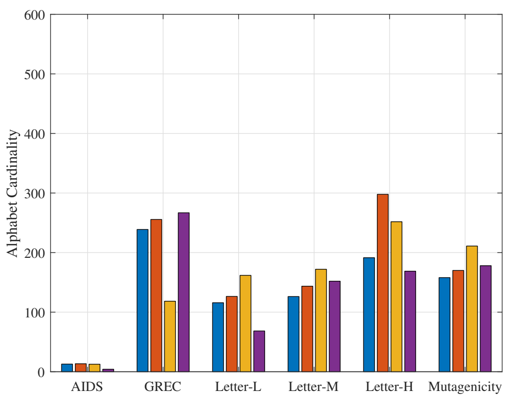
(c) Clique

Figure 6.2. Running times for different extraction strategies. Blue, red and yellow bars correspond to $W = 10\%, 30\%, 50\%$ sampling rates for BFS and DFS extractor whilst they represent $W = 40\%, 60\%, 80\%$ for clique extractor. Purple bars refer to exhaustive procedures.

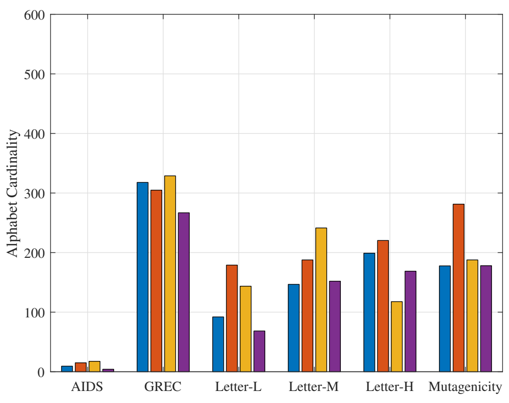
with all subgraphs that compose a graph to be embedded (see Section 3.3.3), a straightforward revenue can be observed as running times are considered: by matching Fig. 6.2c with Fig. 6.2a and 6.2b, the clique-based extractor outperforms the other strategies for each subsample size W, showing an heavy 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 further shrinks the alphabet cardinality, leading to a more explainable learning system.

6.2.2 Class-Aware Granulation Performances

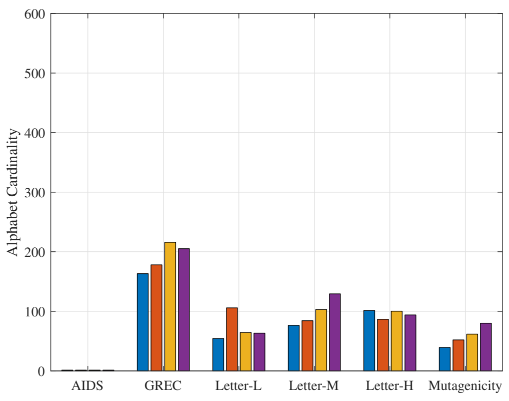
In this section, the Class Aware Granulation strategy discussed in Section 4.2 is evaluated. Specifically, the main purpose is to validate the goodness of the enhanced *Granulator* and the scaling solutions proposed for limiting the alphabet cardinality.



(a) BFS



(b) DFS



(c) Clique

Figure 6.3. Optimal alphabet cardinality for different extraction strategies. Blue, red and yellow bars correspond to $W = 10\%, 30\%, 50\%$ sampling rates for BFS and DFS extractor whilst they represent $W = 40\%, 60\%, 80\%$ for clique extractor. Purple bars refer to exhaustive procedures.

Hence, the tests have been conducted by levaraging on four different configurations:

Configuration 1 (Baseline): No Class-Aware Granulation

Configuration 2 (CA): Class-Aware granulation without U scaling as described in Section 4.2

Configuration 3 (CA-US): Class-Aware granulation with Uniform U Scaling (see Section 4.2.1)

Configuration 4 (CA-FS): Class-Aware granulation with Frequency-based U Scaling (see Section 4.2.1)

In the Baseline benchmark (configuration #1), results are obtained by considering the same GRALG configuration employed in Section 6.2.1 with the sampling strategy based on BFS extractor. That is, the sampling strategy is enabled for populating the set of subgraph for the (non-stratified) clustering ensemble and the embedding

phase. Conversely the CA configurations (#2,#3,#4) follow the stratified random extraction procedure described in Section 4.2, by setting up the maximum number of allowed subgraphs W equal to a given percentage of the number of subgraphs emerging from the exhaustive extractor. In order to guarantee fair comparison, both baseline and all CA configurations have been tested with the same sets of W parameters. The traversal strategy deployed in both extraction and embedding phase is the BFS according with the Baseline configuration. This choice stems from the results obtained in previous section, where no clear winner emerged between BFS and DFS traversal strategies in terms of performances and running times for the considered datasets, hence the choice has been considered arbitrary.
The algorithm parameters are set as follows:

– $W = 10\%,\ 30\%,\ 50\%$ of the total number of subgraphs extract exhaustively with $o = 5$ as shown in Tab. 6.2

– $U_{max} = 500$ is the upper bound value for BSAS U parameter

– $o = 5$ the maximum order of the extracted subgraphs

– 20 individuals for the population of both genetic algorithms

– 20 generations for the first genetic algorithm (alphabet optimization)

– 100 generations for the second genetic algorithm (feature selection)

– $\sigma = 0.99$ in the fitness function for the second genetic algorithm (no weight to sparsity)

– $k = 5$ for the k-NN classifier

– $\xi = 1.1$ as tolerance value for the symbolic histograms evaluation

As highlighted in Figures 6.4–6.6, three different aspects has been considered in order to compare the four strategies:

1. Accuracy on the Test Set

2. Overall running times (including training and test phases) [2].

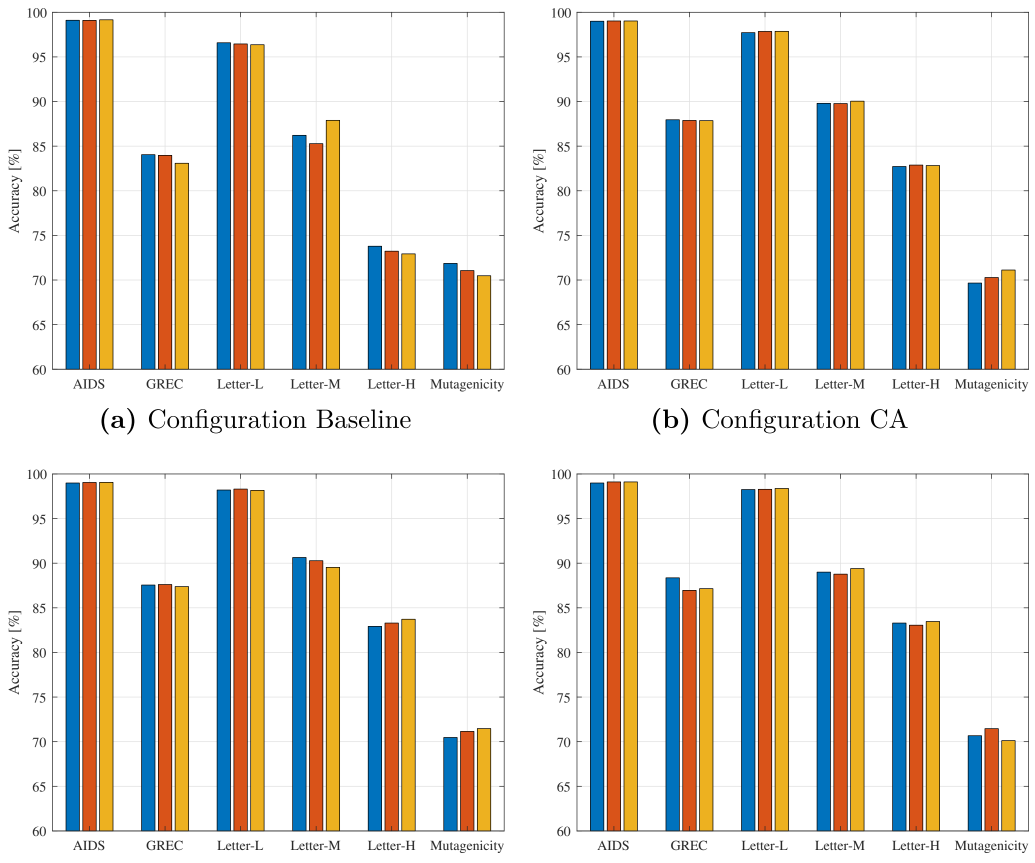
3. Total number of symbols in the alphabet at the end of the alphabet synthesis phase [3]

Due to the intrinsic randomness in the training procedures, results herein presented have been averaged across 10 different runs. The random extraction procedure has been tested with three different values of W and up to a maximum subgraph order o.

When the Class-Aware is employed, in any of its configurations, Figures 6.4b-6.4c-6.4d show improved results with respect to the non-stratified version in Fig.

[2]In order to guarantee fair running times comparison, all tests have been carried on the same workstation deployed for tests in Section 6.2.1.

[3]This number refers to the alphabet before the feature selection phase in order to have a fair comparison, free of biases from the second genetic algorithm.



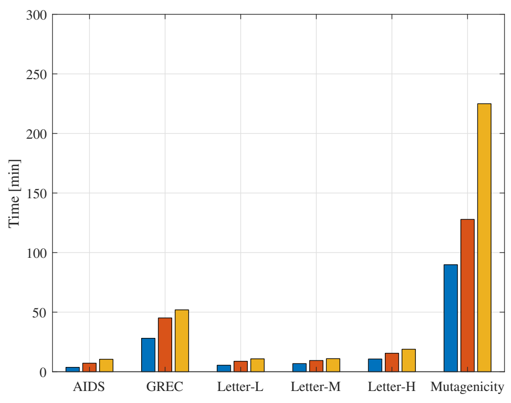
(a) Configuration Baseline **(b)** Configuration CA

(c) Configuration CA-US **(d)** Configuration CA-FS

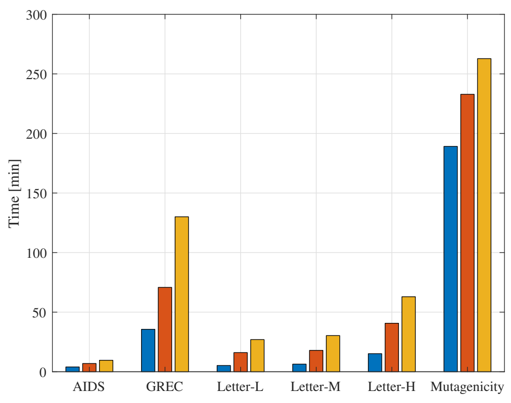
Figure 6.4. Accuracy comparison for the 4 configurations. Blue, red and yellow bars correspond to $W = 10\%, 30\%, 50\%$ sampling rates.

6.4a. Indeed, results for GREC and LETTER-H improved by 4%-5% and 8%-9% in terms of accuracy, respectively, followed by LETTER-M that gains a 3%-4% accuracy boost.

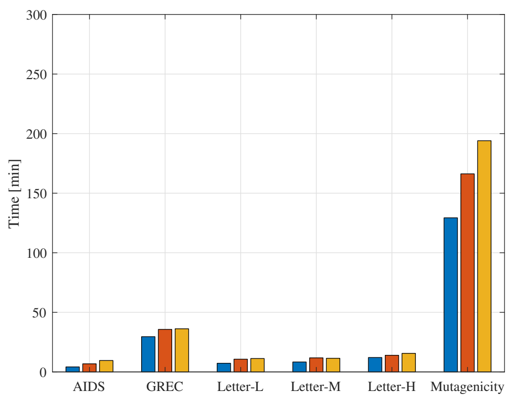
On the other hand, when no scaling method is considered, the Class-Aware procedure worsen the dimensionality of the embedding space as shown in Fig. 6.5b. An explanation of this behaviour may be found by considering the Embedding procedure in Section 3.3.3: when a graph has to be embedded in the vector space, every symbol of the alphabet $\mathcal{A}$ has to be matched with the set of subgraphs drawn from the graph itself and therefore the computational complexity can grow rapidly if both the subgraphs and the alphabet set become too large, as in the case explained in Section 4.2, where the above-mentioned set cardinality can assume values to $\mathcal{O}(L \cdot |\boldsymbol{\theta}| \cdot U_{max})$ being L and θ respectively the number of classes in the problem and the BSAS thresholds of inclusion. It is not difficult to see this phenomenon by comparing Fig. 6.6a with Fig. 6.6b, where the latter shows a clear-cut augmentation of symbols in the alphabet for almost all of the considered datasets. The results obtained when U scaling methods are employed, show that they are effective in reducing the cardinality of the alphabet $\mathcal{A}$, as it is possible to spot by comparing



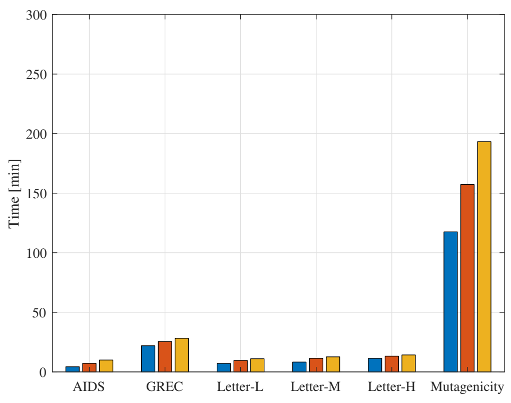
(a) Configuration Baseline



(b) Configuration CA



(c) Configuration CA-US



(d) Configuration CA-FS

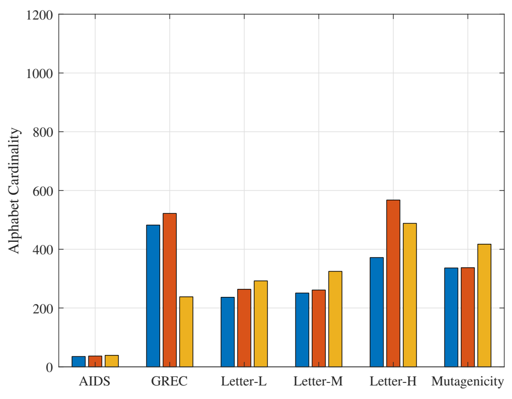
Figure 6.5. Running times comparison for the 4 configurations. Blue, red and yellow bars correspond to $W = 10\%, 30\%, 50\%$ sampling rates.

Fig. 6.6b with Fig. 6.6c-6.6d. As a consequence, in Figs. 6.5c-6.5d running times for the configuration with the aforementioned methods scale down comparably to the base configuration in Fig. 6.5a

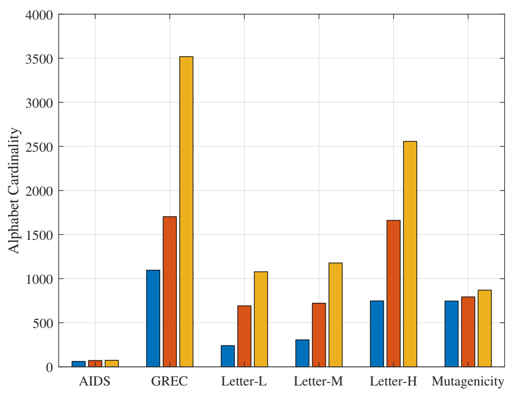
6.2.3 Class Specific Metric Learning Performances

In this section, the Class Specific Metric Learning approach introduced in Section 4.3 is analyzed. The performance of the proposed method are addressed under two different aspects: the learning performances (i.e., accuracy on the test set) and the cardinality of the alphabet after the features selection stage. Indeed, the latter is the crucial set that allows the embedding from the graph domain towards the geometric space and thus can affect both the model complexity and its interpretability.

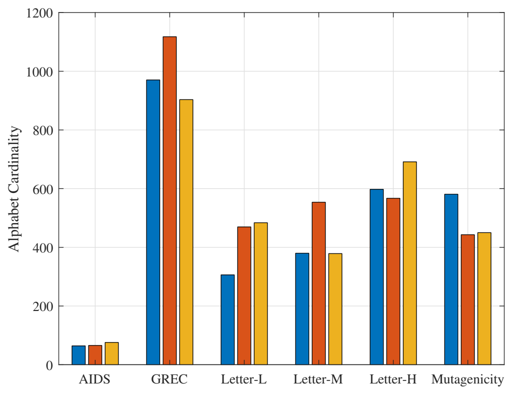
GRALG equipped with lightweight extractor as described in Section 4.1 has been chosen as baseline method for comparison and assessment of the improvements. Indeed, the selected baseline GRALG version performs a global metric learning strategy by means of a single population of individuals for its evolutionary strategy and thus avoiding all the class stratification necessary for the class-specific metric



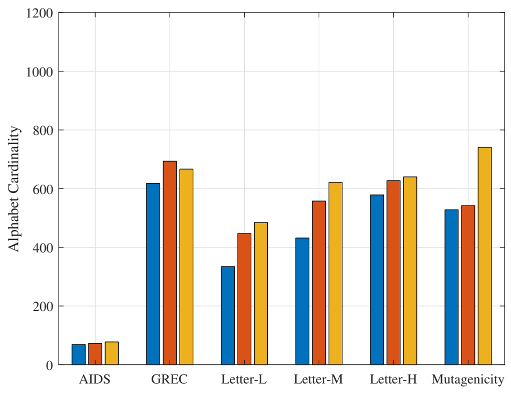
(a) Configuration Baseline



(b) Configuration CA



(c) Configuration CA-US



(d) Configuration CA-FS

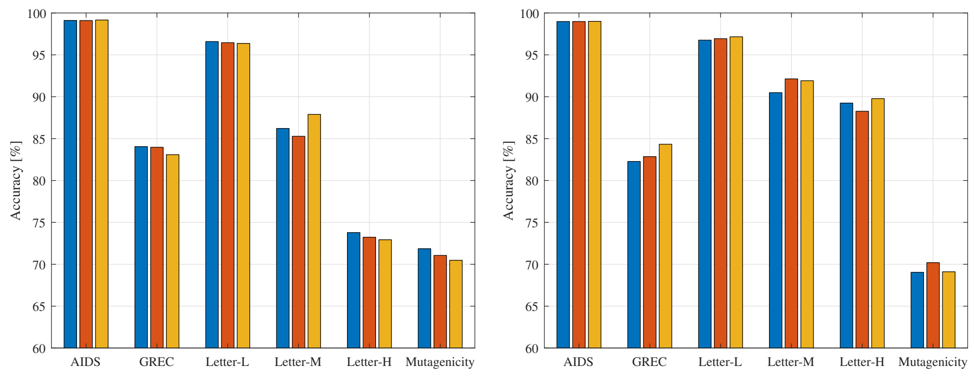
Figure 6.6. Alphabet size comparison for the 4 configurations. Blue, red and yellow bars correspond to $W = 10\%, 30\%, 50\%$ sampling rates.

learning approach. Hence, a single subgraph set $\mathcal{S}^{(\text{tr})}$ is extracted from graphs in the training set regardless of its class label. A single individual is in charge of synthesizing an alphabet $\mathcal{A}$ from $\mathcal{S}^{(\text{tr})}$ thanks to the Granulator block with parameters $\mathbf{p}$ that exploits the clustering ensemble method and consequently builds the vectorial representation of $\mathbf{F}^{(\text{tr})}$ and $\mathbf{F}^{(\text{vs})}$ thanks to the Embedder block equipped with $\mathcal{A}$. Needless to say, all blocks use the same dissimilarity measure with the same parameters $\mathbf{w}$ without class distinction.

Both systems are equipped with the stochastic subgraph Extractor (see Section 4.1) which allows to limit the set of subgraphs $\mathcal{S}^{(\text{tr})}$ with a user-defined cardinality W, whose subgraphs have a fixed order o. BFS has been selected as the traversal strategy for Extractor and Embedder block, both in the baseline GRALG and in the proposed variant. Relevant parameters have been selected as follows:

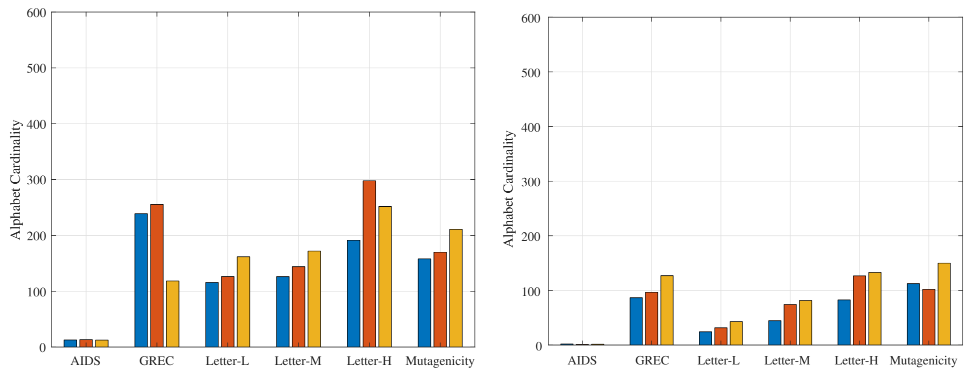
- $W = 10\%, 30\%, 50\%$ of the exhaustive number of subgraphs (see Table 6.2)
- $o = 5$ the maximum order of the extracted subgraphs
- $\sigma = 0.9$ in Eq. (4.14)

- $\xi = 1.1$ as tolerance parameter for subgraph occurrences count
- the classifier is a k-NN classifier with $k = 5$
- 20 generations in the evolutionary strategy for alphabet optimization
- 100 generations in the evolutionary strategy for feature selection.



(a) (b)

Figure 6.7. Average system accuracies on test set. Plain GRALG and proposed approach results respectively in Fig. 6.7a and Fig. 6.7b. Blue, red and yellow bars correspond to $W = 10\%, 30\%, 50\%$ sampling rates.



(a) (b)

Figure 6.8. Average number of selected symbols (after feature selection). Plain GRALG and proposed approach results respectively in Fig. 6.8a and Fig. 6.8b.Blue, red and yellow bars correspond to $W = 10\%, 30\%, 50\%$ sampling rates.

Since GRALG relies on stochastic routines, results shown in Fig. 6.7, Fig. 6.8 have been averaged across 10 different runs. By comparing Fig. 6.7a and Fig. 6.7b, it is possible to spot the major improvements in terms of learning performances (i.e., accuracy on the test set). The proposed approach outperforms the baseline method for LETTER-M and LETTER-H, whereas it holds similar accuracy on the remaining datasets. On the other hand, the class-specific variant clearly improves

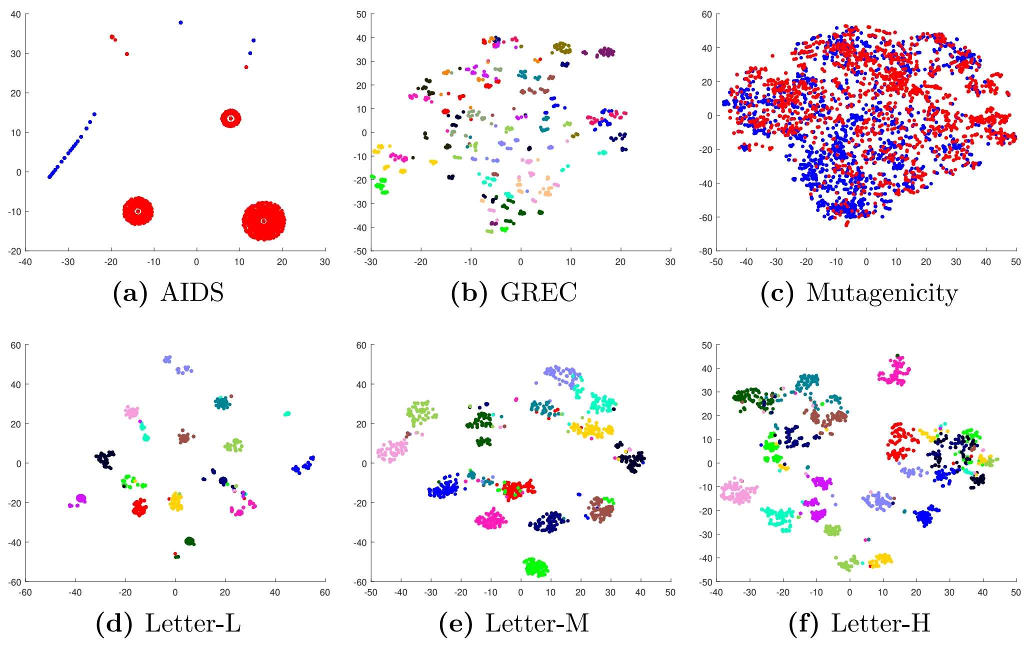
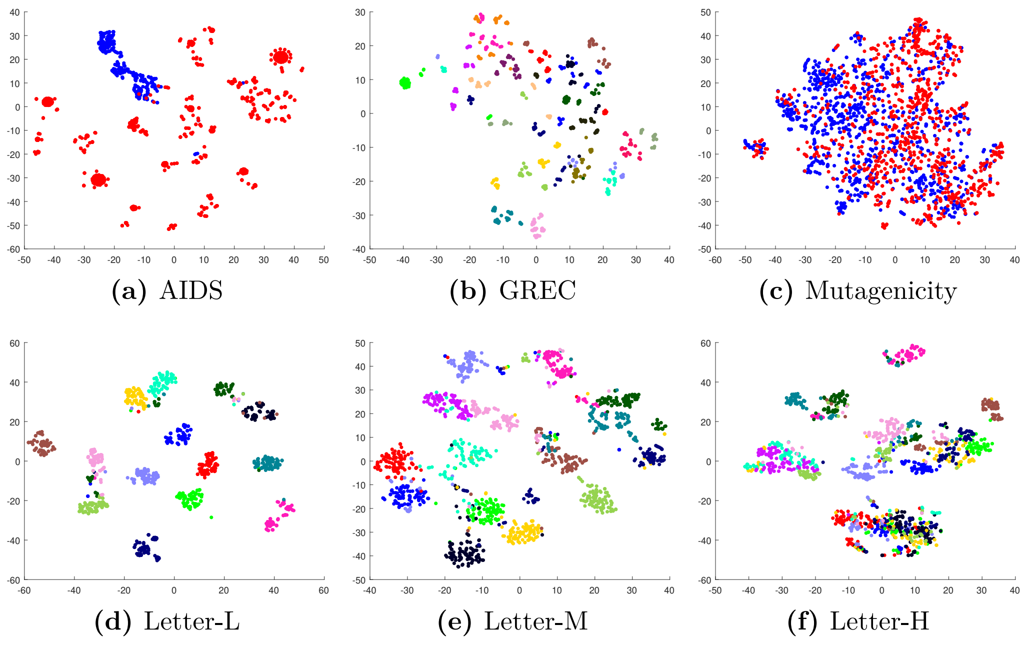


Figure 6.9. Two dimensional embedding space using t-SNE for dimensionality reduction obtained with the graph embedding method equipped with class-specific learning. For each dataset, different colours denote different classes.

the numbers of synthesized symbols, as shown when compared Fig. 6.8a against 6.8b. In particular, by matching Figs. 6.8a and 6.8b, it is possible to see that the proposed class-specific metric learning variant outperforms the plain GRALG version in building the alphabet all the datasets considered. In general, in case of alphabets with lower cardinality, the final model benefits in a two-fold fashion. First, from the interpretability point of view: the symbols contained in the alphabet are meaningful subgraphs required for the classification task that can be analyzed by field experts to figure out what are the meaningful features that characterize a particular problem-related class. Even though this can be achieved also by the plain version of GRALG (i.e., with no class-specific learning), the class specific approach aims to go even further in terms of interpretability thanks to its ability in selecting suitable metric parameters which increase the characterization among different classes. Second, the testing phase for previously-unseen test patterns is faster since there are less symbols to match the test data against (cf. Section 3.5)).

Embedding Visualization

A major benefit of graph embedding strategies for classification problems is the flexibility of the embedding space. Being vector spaces, they are naturally equipped with well-known distance measures such as the Euclidean distance, which allows the use of many machine learning tools. Additionally, this facet enables the opportunity to take advantage of a plethora of dimensionality reduction techniques which can help in visualizing the data at hand. t-Distributed Stochastic Neighbor Embedding (t-SNE) [174] is a powerful method that performs a non-linear mapping from a



(a) AIDS **(b)** GREC **(c)** Mutagenicity

(d) Letter-L **(e)** Letter-M **(f)** Letter-H

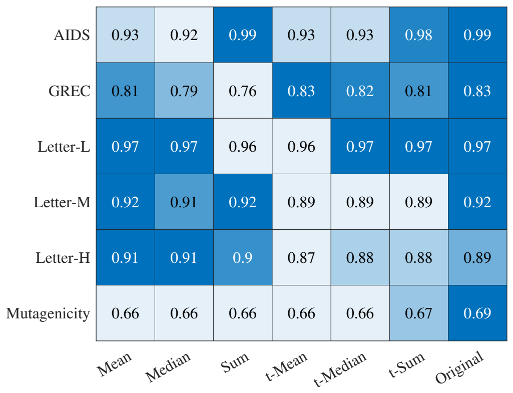
Figure 6.10. Two dimensional embedding space using t-SNE for dimensionality reduction obtained with the baseline method (GRALG). For each dataset, different colours denote different classes.

high-dimensional space into a lower dimensional one, while retaining both local and global relationships among data by preserving their mutual distances in the reduction process. The latter is possible thanks to the evaluation of the similarity between the two probability distributions (i.e., in the original and reduced space), hence solving an optimization problem which aims at minimizing the Kullback-Leibler divergence between the two probability distributions. t-SNE has been exploited in order to visualize the resulting embedding spaces in a 2-dimensional space for the six datasets. For the sake of comparison, for each dataset, the t-SNE embedding of GRALG (baseline method) and the class-specific metric learning are provided in Figures 6.10 and 6.9, respectively.

For AIDS, an interesting plot emerges: the low dimensional space depicted in Figure 6.9a features well-separated and homogeneous clusters, if compared with the baseline counterpart shown in Figure 6.10a. Since t-SNE focuses on distance preservation in the mapping process, this phenomenon can be considered as an evidence of effectiveness for the class-specific metric learning strategy. Nonetheless, the two classes are still easily separable in both cases, as confirmed by the accuracy values in Figures 6.7b and 6.7a.

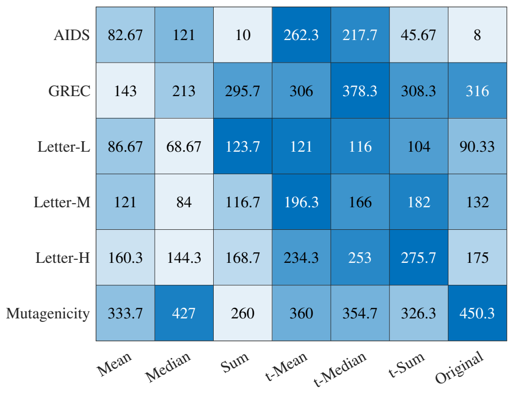
On the other hand, GREC and MUTAGENICITY do not show relevant information about data distribution in the low dimensional space build by t-SNE, whilst LETTER-L shows good separability with both approaches.

Finally, both LETTER-M and LETTER-H benefit from the use of the proposed method. Specifically, for LETTER-M, by comparing Figure 6.9e with Figure 6.10e, we can see how patterns in same classes appear more compact, with lower level of



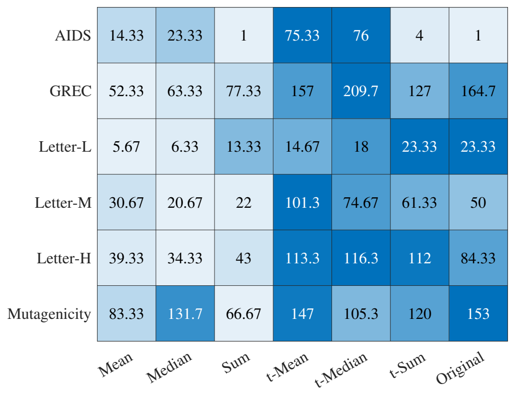
	Mean	Median	Sum	t-Mean	t-Median	t-Sum	Original
AIDS	0.93	0.92	0.99	0.93	0.93	0.98	0.99
GREC	0.81	0.79	0.76	0.83	0.82	0.81	0.83
Letter-L	0.97	0.97	0.96	0.96	0.97	0.97	0.97
Letter-M	0.92	0.91	0.92	0.89	0.89	0.89	0.92
Letter-H	0.91	0.91	0.9	0.87	0.88	0.88	0.89
Mutagenicity	0.66	0.66	0.66	0.66	0.66	0.67	0.69

(a) Accuracy on the Test Set



	Mean	Median	Sum	t-Mean	t-Median	t-Sum	Original
AIDS	82.67	121	10	262.3	217.7	45.67	8
GREC	143	213	295.7	306	378.3	308.3	316
Letter-L	86.67	68.67	123.7	121	116	104	90.33
Letter-M	121	84	116.7	196.3	166	182	132
Letter-H	160.3	144.3	168.7	234.3	253	275.7	175
Mutagenicity	333.7	427	260	360	354.7	326.3	450.3

(b) Embedding space dimensionality before Feature Selection



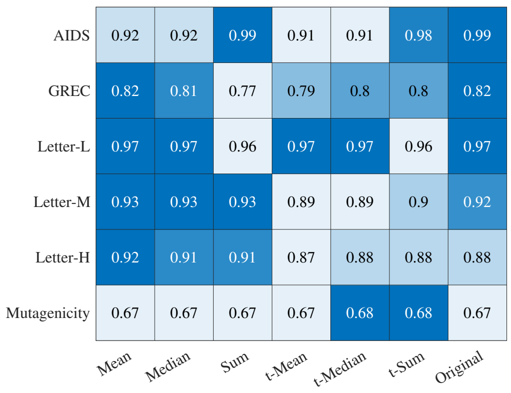
	Mean	Median	Sum	t-Mean	t-Median	t-Sum	Original
AIDS	14.33	23.33	1	75.33	76	4	1
GREC	52.33	63.33	77.33	157	209.7	127	164.7
Letter-L	5.67	6.33	13.33	14.67	18	23.33	23.33
Letter-M	30.67	20.67	22	101.3	74.67	61.33	50
Letter-H	39.33	34.33	43	113.3	116.3	112	84.33
Mutagenicity	83.33	131.7	66.67	147	105.3	120	153

(c) Embedding space dimensionality after Feature Selection

Figure 6.11. Results at 10% subsampling rate. Color maps are normalized row-wise (i.e., independently for each dataset), with a white-to-blue range mapping smallest-to-largest values.

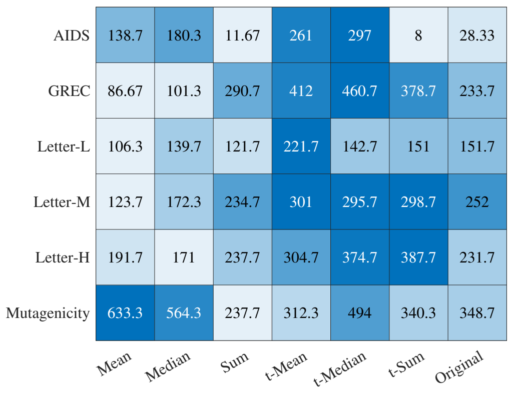
dispersion with respect to the baseline method. The same effect is also prominent for LETTER-H, as depicted in Figures 6.9f and 6.10f where, in the former case, classes are also more overlapped with respect to its counterpart. It is worth noting that LETTER-M and LETTER-H, namely the two datasets that show the most benefit in terms of t-SNE separability are the ones that show the greater accuracy boost with respect to the plain GRALG version (see Figures 6.9 and 6.10).

In conclusion, visualizing the embedding spaces with a dimensionality reduction technique such as t-SNE has confirmed the validity of the results obtained in the previous section. Indeed, performance boosts observed in terms of accuracy are graphically reflected with a clear-cut class distribution in the low dimensional space (LETTER-M and LETTER-H), whist graphics are similar in problems where no major improvements can be observed.



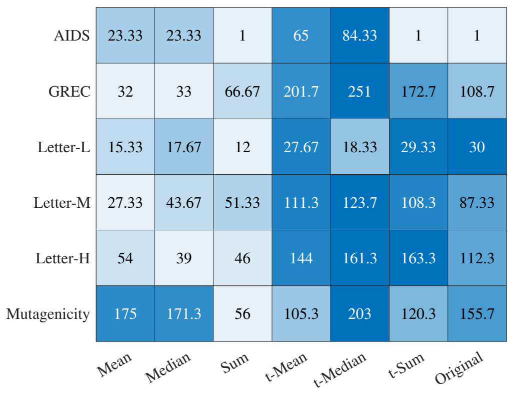
	Mean	Median	Sum	t-Mean	t-Median	t-Sum	Original
AIDS	0.92	0.92	0.99	0.91	0.91	0.98	0.99
GREC	0.82	0.81	0.77	0.79	0.8	0.8	0.82
Letter-L	0.97	0.97	0.96	0.97	0.97	0.96	0.97
Letter-M	0.93	0.93	0.93	0.89	0.89	0.9	0.92
Letter-H	0.92	0.91	0.91	0.87	0.88	0.88	0.88
Mutagenicity	0.67	0.67	0.67	0.67	0.68	0.68	0.67

(a) Accuracy on the Test Set



	Mean	Median	Sum	t-Mean	t-Median	t-Sum	Original
AIDS	138.7	180.3	11.67	261	297	8	28.33
GREC	86.67	101.3	290.7	412	460.7	378.7	233.7
Letter-L	106.3	139.7	121.7	221.7	142.7	151	151.7
Letter-M	123.7	172.3	234.7	301	295.7	298.7	252
Letter-H	191.7	171	237.7	304.7	374.7	387.7	231.7
Mutagenicity	633.3	564.3	237.7	312.3	494	340.3	348.7

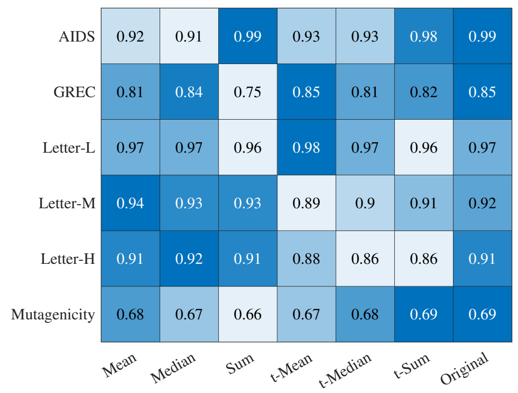
(b) Embedding space dimensionality before Feature Selection



	Mean	Median	Sum	t-Mean	t-Median	t-Sum	Original
AIDS	23.33	23.33	1	65	84.33	1	1
GREC	32	33	66.67	201.7	251	172.7	108.7
Letter-L	15.33	17.67	12	27.67	18.33	29.33	30
Letter-M	27.33	43.67	51.33	111.3	123.7	108.3	87.33
Letter-H	54	39	46	144	161.3	163.3	112.3
Mutagenicity	175	171.3	56	105.3	203	120.3	155.7

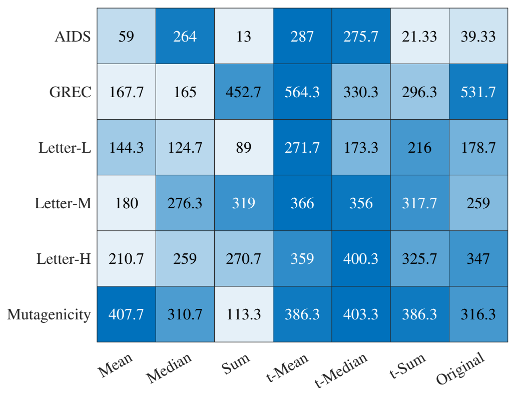
(c) Embedding space dimensionality after Feature Selection

Figure 6.12. Results at 30% subsampling rate. Color maps are normalized row-wise (i.e., independently for each dataset), with a white-to-blue range mapping smallest-to-largest values.



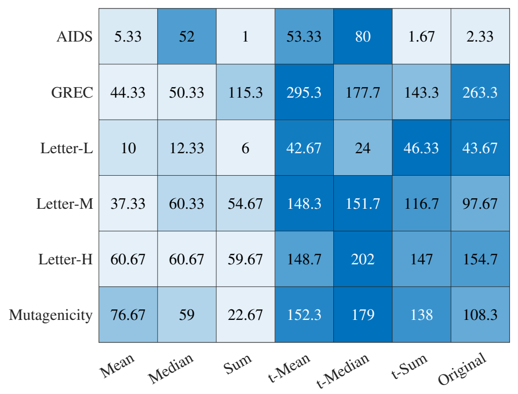
	Mean	Median	Sum	t-Mean	t-Median	t-Sum	Original
AIDS	0.92	0.91	0.99	0.93	0.93	0.98	0.99
GREC	0.81	0.84	0.75	0.85	0.81	0.82	0.85
Letter-L	0.97	0.97	0.96	0.98	0.97	0.96	0.97
Letter-M	0.94	0.93	0.93	0.89	0.9	0.91	0.92
Letter-H	0.91	0.92	0.91	0.88	0.86	0.86	0.91
Mutagenicity	0.68	0.67	0.66	0.67	0.68	0.69	0.69

(a) Accuracy on the Test Set



	Mean	Median	Sum	t-Mean	t-Median	t-Sum	Original
AIDS	59	264	13	287	275.7	21.33	39.33
GREC	167.7	165	452.7	564.3	330.3	296.3	531.7
Letter-L	144.3	124.7	89	271.7	173.3	216	178.7
Letter-M	180	276.3	319	366	356	317.7	259
Letter-H	210.7	259	270.7	359	400.3	325.7	347
Mutagenicity	407.7	310.7	113.3	386.3	403.3	386.3	316.3

(b) Embedding space dimensionality before Feature Selection



	Mean	Median	Sum	t-Mean	t-Median	t-Sum	Original
AIDS	5.33	52	1	53.33	80	1.67	2.33
GREC	44.33	50.33	115.3	295.3	177.7	143.3	263.3
Letter-L	10	12.33	6	42.67	24	46.33	43.67
Letter-M	37.33	60.33	54.67	148.3	151.7	116.7	97.67
Letter-H	60.67	60.67	59.67	148.7	202	147	154.7
Mutagenicity	76.67	59	22.67	152.3	179	138	108.3

(c) Embedding space dimensionality after Feature Selection

Figure 6.13. Results at 50% subsampling rate. Color maps are normalized row-wise (i.e., independently for each dataset), with a white-to-blue range mapping smallest-to-largest values.

6.2.4 Soft Symbolic Histogram Variants Evaluation

In this section, the symbolic histogram variants discussed in Section 4.4 are evaluated. For this purpose, the GRALG classification system has been equipped with the lightweight stochastic Extractor discussed in Section 4.1. The traversal strategy employed for the extraction stage and the graph expansion necessary for the embedding block is the BFS.

The algorithm parameters are set as follows:

- maximum order $o = 5$ for populating the subgraph set $\mathcal{S}^{(\text{tr})}$ in the extraction phase
- $W = 10\%,\ 30\%,\ 50\%$ of the exhaustive number of subgraphs (see Table 6.2)
- maximum number of 20 generations for both evolution stages (alphabet optimization and feature selection)
- 20 individuals for the population of the first evolution (alphabet optimization)
- 100 individuals for the population of the second evolution (feature selection)
- $\sigma = 0.99$ in the fitness function f_2 of the second genetic evolution (major weight to performance)
- k-NN with $k = 5$ as classification system in the embedding space
- $\xi = 1.1$ as tolerance value for the symbolic histograms evaluation.

In Figures 6.11, 6.12 and 6.13, the results obtained by equipping the GRALG classification system presented in Section 3 with the six different symbolic histograms-inspired embedding methods as described in Section 4.4. Each figure corresponds to either one of the three subsampling rates W in order to address the performances of the system as a function of the candidate number of subgraphs for the granulation phase. In order to account for the stochastic nature of the algorithm, results in the following have been averaged across 10 different runs. The efficiency of the proposed variants are evaluated under three different point of views, i.e. the classifier performance measured on the test set $\mathcal{S}^{(\text{ts})}$ and the cardinality of the alphabet before ($|\tilde{\mathcal{A}}|$) and after ($|\mathcal{A}^{*}|$) the feature selection phase (see Section 3.4.2) It is worth noting that the performances on the test set are obtained by the classifier trained with $\mathbf{F}^{*(\text{tr})}$. Recall from Section 3.5 that the vectorial representation $\mathbf{F}^{*(\text{ts})}$ of $\mathcal{S}^{(\text{ts})}$ is obtained thanks to the selected embedding procedure using the optimized alphabet $\mathcal{A}^{*}$.

By comparing Figures 6.11a, 6.12a and 6.13a it is possible to spot the differences in terms of accuracy. The results depicted in the first two columns (Mean and Medium) witnessed that the selected embedding methods are reaching comparable performances with respect to the Original symbolic histogram method equipped with the hard-limiting function regardless of the number of candidate information granules W. A note should be mentioned for AIDS, which is arguably attaining lower level of performances (6~7 %) when compared with the Original column. On the other hand, the remaining four approaches, i.e. Sum, t-Mean, t-Median and t-Sum, show (on average) worst performances when compared to Mean, Median and Original. Again an exception can observed for AIDS: when using the Sum aggregation operator, it shows the highest result in terms of accuracy.

An interesting behaviour that emerge from the tests regards the number of symbols that compose the alphabets $\mathcal{A}^*$ and $\tilde{\mathcal{A}}$. With respect to thresholded methods (t-Mean, t-Median, t-Sum and Original), Mean, Median and Sum are by far producing optimal alphabets with lower number of symbols, regardless of W. This can be spotted by observing Figures 6.11c, 6.12c, 6.13c and their counterparts before the feature selection phase (Figures 6.11b, 6.12b, 6.13b), suggesting that such reduction in terms of number of features is not due to the feature selection phase but is a proper characteristic of the aggregation function. With the exception of MUTAGENICITY, all datasets show a clear-cut difference in terms of alphabet cardinality when not-thresholded methods are exploited. It is possible to advance the hypothesis that by using hard-limiting functions in the symbolic histograms, the resulting dynamic of each vector component can be influenced by the choice of the threshold. In fact, the set of parameters (notably those related to the dissimilarity measure, i.e. $\mathbf{w}$ in Eq. 3.5) explored by the evolutionary algorithm might not be suitable enough for imposing an expressive dissimilarity measure able to fairly compare the symbols with the substructure set G'. In this particular situation, most of the symbols would not be matched with substructures in G', leading to an uninformative embedding space spanned by flat vectors possibly having many null components. On the other hand, the evolutionary algorithm in the attempt to optimize the error rate might be tempted to relieve this issue by exploring granulation parameters (i.e., $\mathbf{p}$ in Eq. 3.9) which allow larger alphabets with higher chances of scoring matches between symbols and the substructures of the graphs to be embedded. Clearly, this situation does not hold in non-thresholded methods, where each match counts (albeit proportionally to its dissimilarity degree): in turn, this means that a given symbol from the alphabet is (in very plain terms) 'always found' in the graph to be embedded which is therefore completely explored during the embedding procedure.

6.2.5 Multiobjective Optimization Method Evaluation

In this section, experiments are carried out in order to study the relationship between the three objective functions f_1, f_2 and f_3 (respectively denoted as *Performance*, *Complexity* and *Sparsity* in the following) according to the multiobjective strategy discussed in Section 4.5 for the alphabet optimization.

The algorithm parameters have been set as follows:

- 100 individuals with a maximum number of 100 generations, in order to ensure a reasonable tracking of the Pareto Frontier;
- early stop criterion if the change in the fitness values are below 0.0025 for 15 consecutive generations;
- $W = 30\%$ of the maximum number of subgraphs attainable from the training set with an exhaustive extraction

- a simple random walk is employed as graph traversing strategy for mining subgraphs of maximum order $o = 5$ for the extraction phase (i.e., for populating $\mathcal{S}^{(\mathrm{tr})}$)[4]
- a BFS traversal strategy is employed for extracting subgraphs for the embedding phase (i.e., for building G')
- parameter θ in BSAS follows a binary search in range $[0, 1]$;
- ξ is the symbol-dependent threshold for scoring a hit in the embedding procedure (cf. Eq. (3.12)).

The tests are conducted by showing the Pareto frontiers obtained for five datasets described in Section 6.1 and the pairwise 2D projections. Mutagenicity is not reported due to the high running times necessary to complete the whole optimization with the considered hyperparameters for NSGA-II.

The results are summarized individually for each dataset in Figures 6.14, 6.15, 6.16, 6.17 and 6.18, respectively for Letter-H, Letter-M, Letter-L, AIDS and GREC, for both kernels (linear and RBF) used in SVM classifiers. To ensure readability, axes have been scaled in terms of percentage.

A first immediate regular behaviour can be observed in the relation between Sparsity and Performance. Indeed, for all Letter datasets, a clear inverse proportionality between the two objective functions emerges from Figures 6.14c, 6.15c and 6.16c by using linear SVM. The same behaviour holds for its kernelized counterpart as depicted in Figures 6.14g, 6.15g and 6.16g. A similar phenomenon, but slightly less prominent, occurs for GREC dataset for both linear (Figure 6.18c) and RBF kernel (Figure 6.18g). On the other hand, Aids dataset does not show any relevant trend for the Sparsity-Performance pair due to the dense superposition of the solutions in the Pareto frontier as can be noticed in Figure 6.17a (Linear-SVM) and Figure 6.17e (RBF-SVM). These results entail a transparent relationship between the number of symbols in the alphabet $\mathcal{A}$ necessary for attaining reliable performances in f_1, i.e. low level of error-rate on validation set.

From the Complexity-vs-Performance perspective, the computational results for Letter-H show a clear inverse relation for both Linear and RBF kernels, respectively depicted in Figure 6.14b and Figure 6.14f. In this particular problem, an elbow-like behaviour for the two considered objective functions arise, suggesting that its decision boundary can be well defined with an increasing number of support vectors. For the remaining two Letter datasets, i.e. Letter-M (Figure 6.15b and Figure 6.15f) and Letter-L (Figure 6.16b and Figure 6.16f), the projected Pareto frontiers are far more chaotic limiting the possibility to draw reasonable conclusions. This is likely due to the fact that Letter-M and Letter-L are easier classification problems to solve with respect to Letter-H, so there exist solutions for which a lower number of support vectors leads to reasonable performances: this is notably not true for harder classification problems (i.e., Letter-H, in this case) for which a higher number of support vectors are needed to draw a more accurate decision boundary. On the

[4]Since no relevant differences have been witnessed between BFS or DFS in terms of performances and/or running times, the rationale behind choosing random walks is that there exist the possibility of having both star-like and path-like subgraphs in $\mathcal{S}^{(\mathrm{tr})}$.

other hand, for Grec dataset, the Linear-SVM projection in Figure 6.18b has a firm decreasingly trend, whilst its kernelized counterpart in Figure 6.18f shows a less compact behaviour but with a more defined point cloud in the upper-left region of the plot. Again, no clear trend emerges from AIDS dataset due to the former discussion about the superposition of the solutions.

The last aspect to consider regards the Complexity-vs-Sparsity projection. For almost all datasets, i.e. Letter-H (Figure 6.14d and Figure 6.14h), Letter-M (Figure 6.15d and Figure 6.15h), AIDS (Figure 6.17d and Figure 6.17h) and Grec (Figure 6.18d and Figure 6.18h), no clear trends emerge between the number of support vectors and the cardinality of the alphabets, being the Pareto fronts not well shaped for both SVM configurations. A notable exception can be observed for Letter-L, where the Complexity-vs-Sparsity projection leads to an 'L-shaped' frontier, especially for the Linear-SVM case (Figure 6.16d). This suggests that limiting the number of support vectors is counterbalanced by a higher embedding space dimensionality.

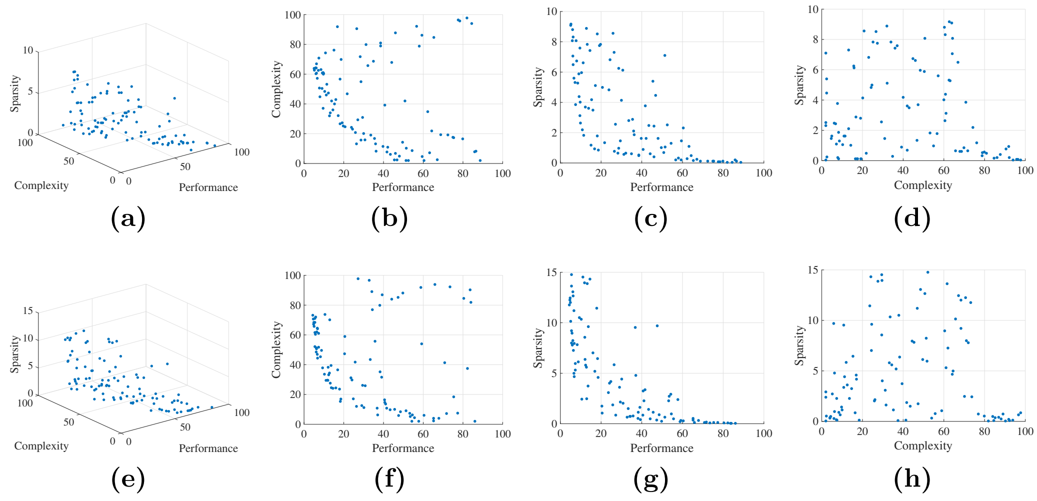


Figure 6.14. Dataset: Letter-H. 3D Pareto Front and pairwise 2D projections. Top panels correspond to the Linear-SVM, bottom panels correspond to RBF-SVM.

Baseline Cases

In this section, the computational results obtained on the test phase with the introduction of the ensemble of classifiers working on heterogeneous embedding spaces (see Section 4.5.2) are evaluated. Let recall from Section 4.5.1 that TOPSIS allows the ranking of the solutions according to a weight vector $\mathbf{u}$.

At first, the *baseline cases* (see Table 6.3) are considered as the solutions in X that achieve the best results in the three objective functions when considered separately. The selection of the baseline solutions works by sorting the decision matrix $\mathbf{D}$ according to each column (i.e., objective function) separately. After each sort, the solution in X that corresponds to the top-ranked row in $\mathbf{D}$ is retained. In particular, for each dataset and for each classifier, Table 6.3 features a 3×3

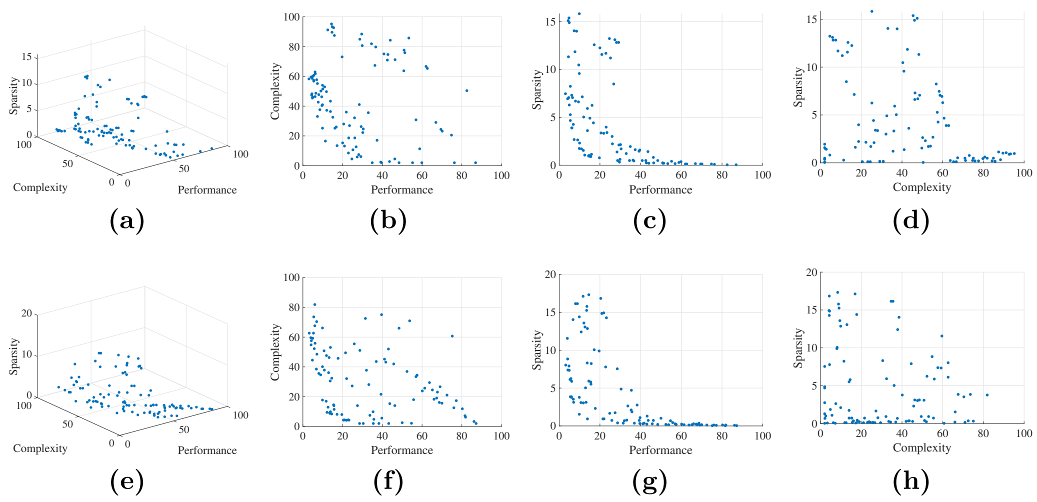


Figure 6.15. Dataset: Letter-M. 3D Pareto Front and pairwise 2D projections. Top panels correspond to the Linear-SVM, bottom panels correspond to RBF-SVM.

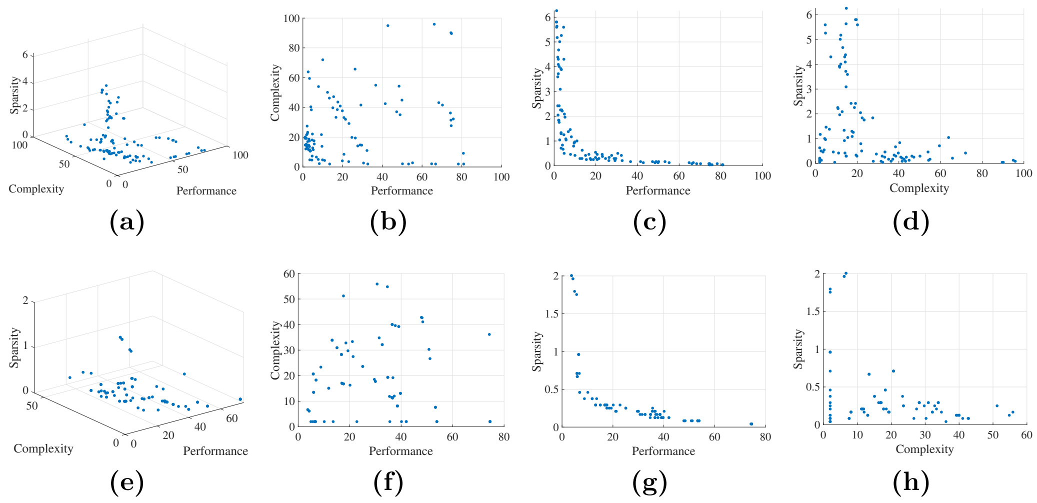


Figure 6.16. Dataset: Letter-L. 3D Pareto Front and pairwise 2D projections. Top panels correspond to the Linear-SVM, bottom panels correspond to RBF-SVM.

sub-matrix which in position (i, j) shows the i^{th} objective function value obtained when selecting the solution in X that minimizes the j^{th} objective function.

For the sake of argument, *uniform weighting* is considered as well: that is, TOPSIS method ranks the solutions in X without introducing any preferences for the three objective functions, i.e. $\mathbf{u} = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix}$. The number of classifiers and embedding spaces to consider in the ensemble is chosen according to the top-K solutions X^* returned by TOPSIS[5], with $K = 10$. In Figure 6.19, the results obtained

[5]In Appendix A, for the sake of completeness, we show the 3D Pareto fronts in which the solutions belonging to X^* are highlighted (Figures A.1a–A.1b for AIDS, Figures A.2a–A.2b for GREC, Figures A.3a–A.3b for Letter-L, Figures A.4a–A.4b for Letter-M, Figures A.5a–A.5b for

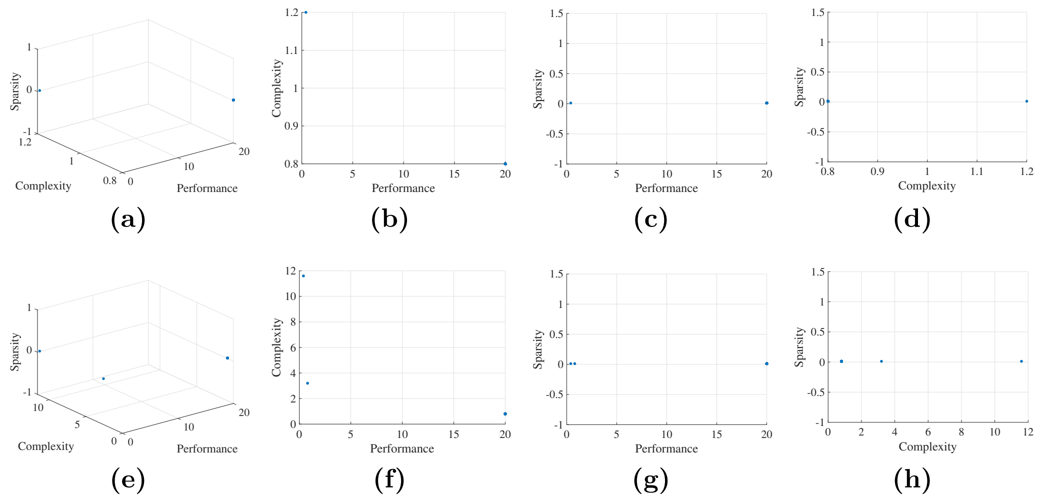


Figure 6.17. Dataset: AIDS dataset. 3D Pareto Front and pairwise 2D projections. Top panels correspond to the Linear-SVM, bottom panels correspond to RBF-SVM.

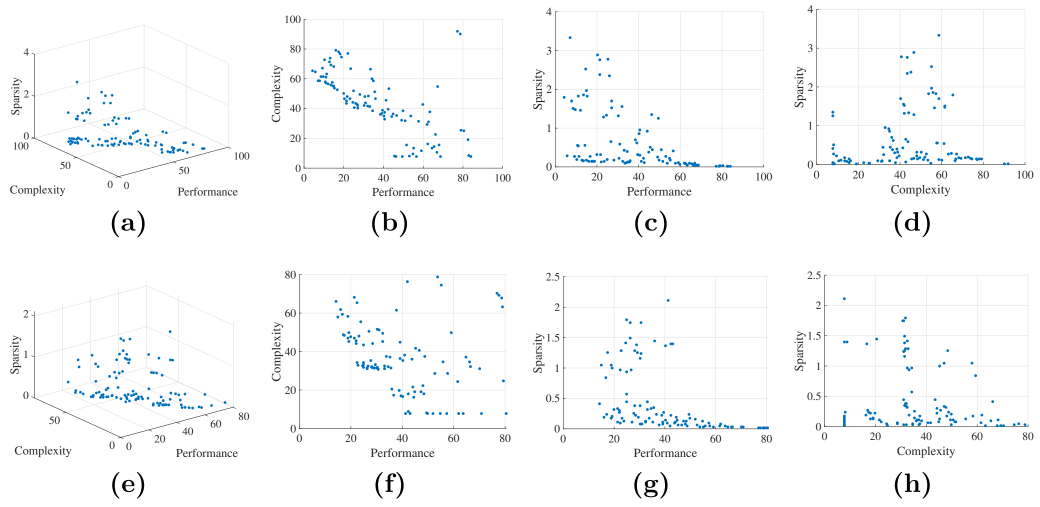


Figure 6.18. Dataset: GREC. 3D Pareto Front and pairwise 2D projections. Top panels correspond to the Linear-SVM, bottom panels correspond to RBF-SVM.

by varying $K = 1, \ldots, 10$ and observing the three main objective of interests are shown: accuracy on test set of the ensemble (Figure 6.19a), the sum of symbols in the alphabets $\mathcal{A}^*$ (Figure 6.19b) and the sum of support vectors for the classifiers in the ensemble $\mathcal{C}$ (Figure 6.19c).

When the Linear-SVM is considered, an interesting result for Letter-L emerge: the best solution in terms of accuracy can be achieved by considering 4 classifiers in the ensemble with a lower number of support vectors and symbols. Indeed, it is possible to see that if the solution that minimizes the error rate are selected regardless of the other two objective functions, the accuracy is 98% (see Table 6.3)

Letter-H).

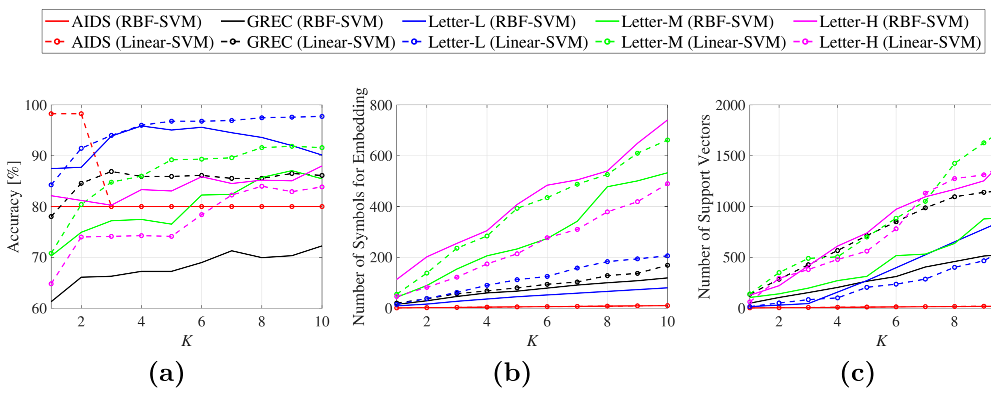


Figure 6.19. Uniform weighting results according to K number of classifiers in ensemble

whilst 4 classifiers in ensemble can obtain 96% of correct predictions on test set. With this small difference of accuracy ($\approx 2\%$), the ensemble requires 101 support vectors and 90 symbols against 147 and 139, respectively, obtained with the best solution in terms of error rate. Similarly, for GREC with Linear-SVM, comparable accuracy performance can be obtained between the baseline solution that minimizes the error rate and 3 classifiers in ensemble, respectively 92% and 87%. Despite 5% of shift, the ensemble relies only on 57 symbols against 113 needed by the baseline solution. On the other hand, the ensemble requires a considerably higher number of support vectors (427 against 185). The same discussion does not hold for the other datasets/configurations, where the shift in terms of accuracy is not sufficiently counterbalanced by improvements in terms of alphabet cardinality and/or structural complexity of the model.

Case Studies

In this section, three different case studies are considered in order to analyze the behaviour of the ensemble of classifiers under different configurations for TOPSIS: computational results are shown by assigning different priorities to each specific function in the decision making process. At this purpose, three different choices for the weighting vector $\mathbf{u}$ are considered:

a) totally neglect the structural complexity, giving equal weights to the other two objectives;

b) totally neglect the number of symbols, giving equal weights to the other two objectives;

c) give more importance to accuracy with respect to the other two objectives.

The rationale behind these three case studies stems from the observation that (in real-world pattern recognition applications) the generalization capability is always the first index to be considered when judging the synthesis of a classification model. In plain terms, low accuracy undermines the trustworthiness of the model (regardless of the features and regardless of the complexity of the model itself) and therefore must always be preferred (or at least comparable to) other objectives. In light of

Table 6.3. Results of the Baseline cases.

	AIDS					
	Linear-SVM			RBF-SVM		
	min Error Rate	min Complexity	min Sparsity	min Error Rate	min Complexity	min Sparsity
Accuracy	98%	80%	80%	99%	80%	80%
Complexity	3	2	2	29	2	2
Sparsity	1	1	1	1	1	1
	GREC					
	Linear-SVM			RBF-SVM		
	min Error Rate	min Complexity	min Sparsity	min Error Rate	min Complexity	min Sparsity
Accuracy	92%	15%	15%	85%	50%	20%
Complexity	185	22	22	187	22	22
Sparsity	113	1	1	26	11	1
	Letter-L					
	Linear-SVM			RBF-SVM		
	min Error Rate	min Complexity	min Sparsity	min Error Rate	min Complexity	min Sparsity
Accuracy	98%	68%	25%	97%	92%	24%
Complexity	147	15	676	50	15	15
Sparsity	139	5	1	48	11	1
	Letter-M					
	Linear-SVM			RBF-SVM		
	min Error Rate	min Complexity	min Sparsity	min Error Rate	min Complexity	min Sparsity
Accuracy	94%	53%	11%	95%	70%	13%
Complexity	437	15	15	470	15	15
Sparsity	184	12	1	198	120	1
	Letter-H					
	Linear-SVM			RBF-SVM		
	min Error Rate	min Complexity	min Sparsity	min Error Rate	min Complexity	min Sparsity
Accuracy	93%	50%	17%	92%	14%	14%
Complexity	479	15	733	549	15	15
Sparsity	224	175	1	290	1	1

these observations, any case study in which the weight given to the performance of the classifier is smaller than the weight given to the other two objective functions are deliberately not considered: in real-world applications, the machine learning engineer will likely not accept solutions on the Pareto Frontier which feature a low accuracy for any arbitrary low or high number of support vectors and/or number of features.

Case Study A: No weight to model structural complexity. In this setting, TOPSIS prioritizes the solutions that are showing simultaneously low error rate and small number of symbols in the alphabets; that is, the objective functions f_1 and f_3 as described in Eq. (4.25) and Eq. (4.27), respectively. By letting $\mathbf{u} = \begin{bmatrix} \frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix}$, the ranking process will be not influenced by the objective function f_2 in Eq. (4.26); hence, no importance will be given to the model structural complexity. The impact of the imposed condition can be observed by inspecting the positions of the marked solutions on the Pareto Frontiers for both kernels: for Letter-H (Figures A.5c-A.5d), Letter-M (Figures A.4c-A.4d), Letter-L (Figures A.3c-A.3d) and Grec (Figures A.2c-A.2d), the retained solution by TOPSIS are mostly distributed in the objective function space with minimal sparsity and accuracy performance with a considerably large number of support vectors. On the other hand, for Aids dataset (Figures A.1c-A.1d), all solutions are overlapped in few clusters, making less evident the selected solution positions. As already noticed in Section 6.2.5, the introduction of the ensemble of classifiers implies some benefits when compared to the baseline solutions shown in Table 6.3. In this setting, the compromise between Performance and Sparsity introduced by weighting accordingly the different objective functions, gives a clearer advantages in terms of number of symbols for a larger variety of problems. The computational results by varying K are shown in Figures 6.20a, 6.20b and 6.20c (accuracy on test set, number of symbols and number of support vectors, respectively).

For both SVMs, Grec achieves the same performance on the test set if compared to the baseline results ($\approx$ 92% and $\approx$ 85% for Linear-SVMs and RBF-SVMs, respectively) by using $K = 4$ classifiers in the ensemble. The major benefit regards the number of symbols on which the ensemble leverages: for Linear-SVMs, the ensemble outperforms the best accuracy solution using 43 against 113 symbols, whilst for RBF-SVMs the results are comparable: 43 symbols in the ensemble and 26 for the baseline. Notably, the best ensemble configuration ($K = 6$) ensures better performance accuracies for Linear-SVM with respect to the baseline, keeping at the same time a lower sum of alphabet cardinalities, i.e. 78 symbols in alphabets. Despite this beneficial effect, the structural complexity in the ensemble increases rapidly, requiring already at $K = 3$ a way larger number of support vectors, i.e. 617 (Linear) and 543 (RBF) against 185 (Linear) and 187 (RBF) needed for the baseline case.

The same discussion holds for Letter-L: when equipped with the linear kernel, $K = 2$ classifiers already reach the same accuracy on the test set obtained by the baseline ($\approx$ 97%) involving 23 symbols against 50 using, on the other hand, a more complex model.

A similar situation occurs for RBF kernel in Letter-M: $K = 3$ classifier in ensemble yield a minor negative shift of 3% in accuracy, which is counterbalanced by improved results under the number of symbols perspective. Indeed, the baseline solution in accuracy attains 198 symbols in its alphabet, whereas the considered ensemble of classifiers relies in total on 138 symbols.

Finally, with $K = 4$ classifiers, Letter-H achieves comparable performances with respect to the baseline with a small accuracy shift (3%) for both linear and RBF kernels. Again, a considerable lower number of symbols, 118 (Linear) and 190 (RBF) against 224 (Linear) and 290 (RBF) can be observed if compared to the best solution. Nonetheless, for both Letter-H and Letter-M, the reduced amount of symbols is still compensated by a considerable higher number of support vectors. These patterns of behaviour perfectly fit with the priority given to performance accuracy and minimization of the alphabet cardinality fed to TOPSIS via the weighting vector $\mathbf{u}$.

Case Study B: No weight to sparsity. In this case study, the ensemble behaviour is investigated by choosing $\mathbf{u} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix}$. In this way, TOPSIS will drive its preferences towards solutions with low error rate and minimal model structural complexity, namely the objective functions f_1 and f_2 in Eqs. (4.25) and (4.26). The weighting vector will give no importance to the objective function f_3 (i.e., sparsity). The solutions selected by TOPSIS are marked on the Pareto Frontiers for both kernel configurations: Letter-H (Figures A.5e-A.5f), Letter-M (Figures A.4e-A.4f), Letter-L (Figures A.3e-A.3f), Grec (Figures A.2e-A.2f) and Aids

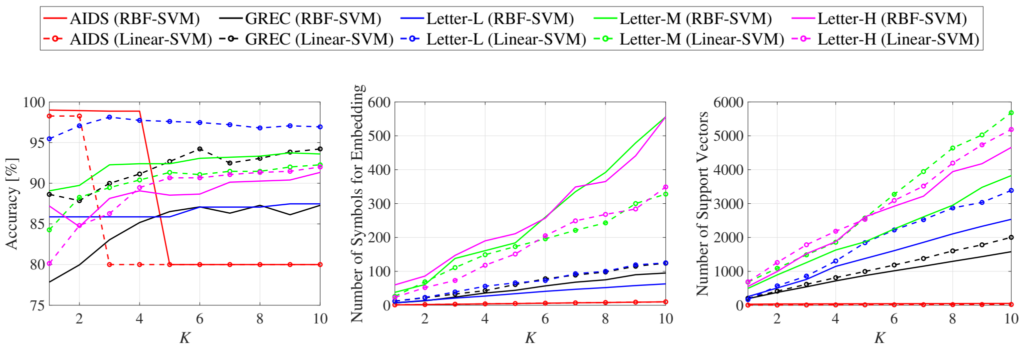


Figure 6.20. Case Study A (No weight to model structural complexity) results according to K number of classifiers in ensemble

(Figures A.1e-A.1f). Since the sparsity is not considered in the decision making process, the plots underline how the selected solutions are distributed in the region with high level of sparsity, conversely to the minimal values of performance and complexity. The performance on the test set, model structural complexity and sparsity of the alphabet with respect to the number of classifier K in the ensemble are shown respectively in Figures 6.21a, 6.21c and 6.21b.

For Grec, using the ensemble of classifiers does not introduce beneficial effects since the baseline performances are way better (more than 10%) when compared to the best ensemble configuration for both kernels, that is $K = 10$ for RBF and $K = 6$ for Linear.

For Letter-L, the ensemble with $K = 3$ classifiers attains the same level of accuracy on test set with respect to the baseline in Table 6.3: Linear-SVMs reaches 97% of accuracy (very close to the baseline solution, that scores 98%) by using 126 support vectors (ensemble) against 147 (baseline); similarly, RBF-SVM attains 98% with a similar complexity with respect to the baseline, that is 45 support vectors (ensemble) against 50 (baseline).

In Letter-M, a major improvement can be spotted by equipping the ensemble with $K = 4$ classifiers and RBF kernel. In this configuration, with a small difference in accuracy (4%), the ensemble drastically reduces the structural complexity by using 268 support vectors against 549 employed by the best solution in terms of accuracy.

For Letter-H, RBF kernel with $K = 3$ classifiers attained similar performances with respect to the baseline, i.e. 89% (ensemble) against 93% (baseline). This result come with a light improvement in terms of support vectors, i.e. 488 (ensemble) against 549 (baseline).

Finally, Aids can be solved as efficiently as the baseline with Linear-SVMs. Indeed, the same accuracy, i.e. 98%, can be attained by using $K = 2$ and 5 support vectors. Conversely, the baseline scores a similar number of support vectors, i.e. 3.

As in the previous case study, no beneficial effects can be spot from the neglected objective function point of view, that in this case coincides with the sparsity of the alphabet. Indeed, all the aforementioned results come at cost of a considerable higher number of symbols in the ensemble of classifiers.

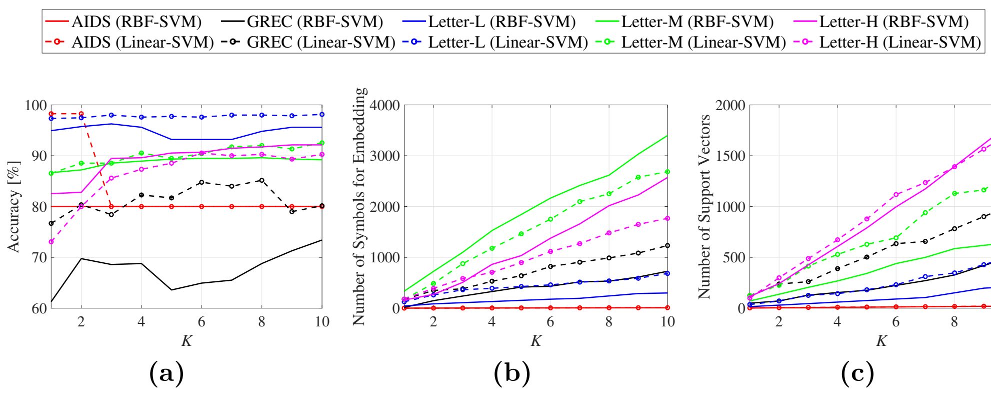


Figure 6.21. Case Study B (No weight to sparsity) results according to K number of classifiers in ensemble.

Case Study C: Priority to accuracy performance. In the last case study, TOPSIS exploits a weighting vector $\mathbf{u} = \begin{bmatrix} \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \end{bmatrix}$. This configuration drives the TOPSIS decision making process mainly toward solutions featured by low error rate, that is, more priority is given to the minimization of f_1 in the ranking process. Nonetheless, structural complexity and sparsity of the alphabet are not totally neglected as in the two previous case studies. As instead, f_2 and f_3 have the same importance, being equally weighted in the selection phase. The Pareto Frontiers with the highlighted selected solutions are shown for both Linear-SVM and RBF-SVM: Letter-H (Figures A.5g-A.5h), Letter-M (Figures A.4g-A.4h), Letter-L (Figures A.3g-A.3h), Grec (Figures A.2g-A.2h) and Aids (Figures A.1g-A.1h). The results of the ensemble of classifiers in terms of accuracy on test set, model structural complexity and sparsity of the alphabets are shown respectively in Figures 6.22a, 6.22c and 6.22b accordingly to the number of K of classifiers in the ensemble.

For Grec, already with $K = 2$ classifiers in ensemble, the same accuracy of the baseline result can be achieved by using Linear SVM classifiers ($\approx 92\%$). On the other hand, the ensemble relies only on 35 symbols against 113 employed with best solution in terms of accuracy. From the model complexity point of view, the ensemble is not able to improve the baseline since it requires only 185 support vectors against 332 needed for the ensemble. Notably, the ensemble outperforms the baseline when more than $K = 3$ classifier are employed, where the best result, i.e. $\approx 95\%$, is obtained with $K = 4$ classifiers. Even though this improvement comes at the cost of an higher structural complexity, as long as the number of classifiers is limited below $K = 8$, the ensemble still exploits a lower number of symbols. Nonetheless, in the considered problem, the SVM with RBF kernel is able to reach the performance of the baseline ($\approx 85\%$ at $K = 4$) without improving any of the two other objective functions.

For Letter-L, the benefit of the ensemble can be summarized as follows: in case of linear kernel, $K = 3$ classifiers are able to perform equally to the baseline ($\approx 98\%$) with improvements in terms of sparsity, i.e. 85 symbols for ensemble against 139 for the baseline. Conversely, no major benefits are witnessed under the model structural complexity point of view. When equipped with RBF kernel, the ensemble with $K = 3$ classifiers achieved similar results to the baseline, respectively 96% and

97% of accuracy. Similarly to the linear case, the only improvement regards the sparsity, where the classifier shows a minimal boost, i.e. 29 symbols against 48 in the baseline.

For Letter-M, the results are quite similar to Letter-L. If Linear-SVMs are considered, the ensemble attained the same baseline results in terms of accuracy with $K = 3$ classifiers, i.e. $\approx 94\%$, only improving the sparsity (160 symbols in alphabet against 184 of the baseline). On the other hand, the kernelized SVM is able to attain the best solution in terms of accuracy with $K = 5$ without improving the other two objective functions.

For Letter-H, the linear kernel achieved similar performances with respect to the baseline using $K = 3$ classifiers, respectively 91% and 93%. The sparsity of the alphabet with such ensemble is comparable with the baseline, i.e ≈ 225 symbols, whilst the structural complexity of the ensemble is considerably worsen. With $K = 4$, the RBF kernel counterpart attained the baseline solution ($\approx 91\%$) but no major improvement can be observed in terms of number of support vectors and sparsity.

Finally, Aids dataset behave as the baseline solution with Linear-SVMs, approaching 98% of accuracy with same number of symbols and support vectors. Conversely, when SVMs are equipped with the RBF kernel, the ensemble is considerably less accurate of the best solution, showing more than 18% of performance shift.

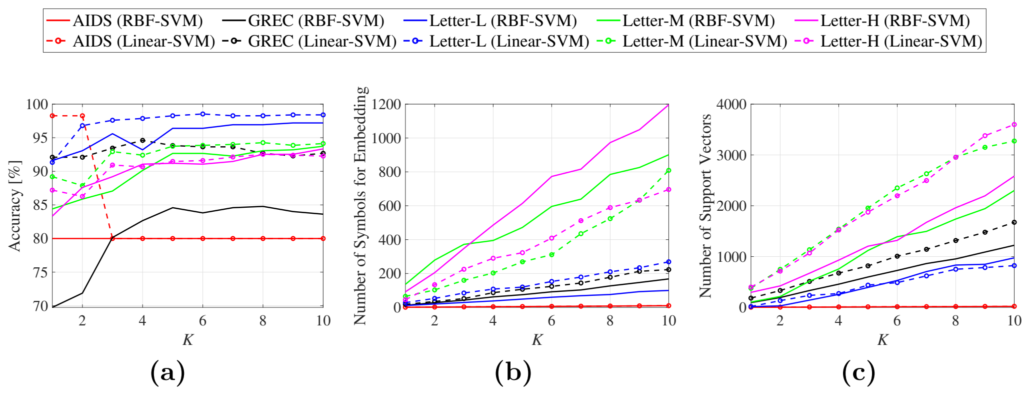


Figure 6.22. Case Study C (Priority to accuracy performance) results according to K number of classifiers in ensemble.

6.2.6 Comparison Against State of the Art Classifiers

In this section, state-of-the-art graph classification systems are compared with different GRALG variants discussed. In Table 6.4, the accuracies on test set of the considered methods are shown.

For sake of comparison, GRALG based variants in Table 6.4 are all equipped with BFS lightweight extractor unless expressively stated otherwise. The results are reported according to the following criteria:

a) Best result among sampling rate W for variants with sampling strategy

b) Best result among different U scaling strategies for class-aware granulation methods

Table 6.4. Comparison against state-of-the-art graph classification system in terms of accuracy. Asterisks indicate that results refers to cross-validation rather than a separate test set. Best accuracy among all subsampling values have been reported for methods based on GRALG.

Technique	**AIDS**	**GREC**	**Letter-L**	**Letter-M**	**Letter-H**	**Mutagenicity**	**Reference**
Bipartite Graph Matching + k-NN	-	86.3	91.1	77.6	61.6	-	[153]
Lipschitz Embedding + SVM	98.3	96.8	99.3	95.9	92.5	74.9	[154]
Graph Edit Distance + k-NN	97.3	95.5	99.6	94	90	71.5	[152]
Graph of Words + k-NN	-	97.5	98.8	-	-	-	[69]
Graph of Words + kPCA + k-NN	-	97.1	97.6	-	-	-	[69]
Graph of Words + ICA + k-NN	-	58.9	82.8	-	-	-	[69]
Topological embedding	99.4	-	-	-	-	77.2	[45, 168]
FMGE	99.0	-	-	-	-	76.5	[45, 101]
Attribute Statistics	99.6	-	-	-	-	76.5	[45, 70]
ODD ST_+ kernel	82.06*	-	-	-	-	-	[48]
ODD ST_+^{TANH} kernel	82.54*	-	-	-	-	-	[48]
Laplacian kernel	92.6	-	-	-	-	70.2	[45, 30]
Treelet kernel	99.1	-	-	-	-	77.1	[45, 64]
Treelet kernel with MKL	99.7	-	-	-	-	77.6	[45, 66]
Weighted Jaccard Hypergraph kernel + SVM	99.5*	-	-	-	-	82*	[117]
CGMM + linear SVM	84.16*	-	-	-	-	-	[6]
G-L-Perceptron	-	70	95	64	70	-	[109]
G-M-Perceptron	-	75	98	87	81	-	[109]
C-1NN	-	-	96	93	84	-	[109]
C-M-1NN	-	-	98	81	71	-	[109]
EigenGCN-1	-	-	-	-	-	80.1	[102]
EigenGCN-2	-	-	-	-	-	78.9	[102]
EigenGCN-3	-	-	-	-	-	79.5	[102]
GCN with logical descriptors	-	96.93	96.64	85.27	79.91	-	[80]
MPNN	-	89.5	91.3	81.2	64.24	-	[149]
MPNN (no set2set)	-	92.98	94.8	86.1	75.7	-	[149]
Hypergraph Embedding + SVM	99.3	-	-	-	-	77.0	[114]
RECTIFIER + k-NN	99.07	95.57	97.12	92.16	91.60	-	[118]
Dual RECTIFIER + k-NN	99.13	96.61	96.40	93.04	91.31	-	[118]
GRALG (exhaustive extraction)	99.44	92.23	98.05	84.83	76.13	70.87	Fig. 6.1a
GRALG (clique extraction)	99.53	94.47	97.77	69.07	67.87	73.40	Fig. 6.1c
GRALG (stochastic extractor)	99.16	84.04	96.58	87.89	73.78	71.86	Fig. 6.1a
GRALG (class-aware granulation)	99.06	87.61	98.30	90.64	83.72	71.47	Fig. 6.4c
GRALG (mean symbolic histogram)	92	81	97	94	91	68	Fig 6.13a
GRALG (class-specific metric learning)	99.0	84.32	97.15	92.13	89.77	70.19	Fig. 6.7b)

c) Best result among all the configurations for soft symbolic histogram embedding

Specifically in b), uniform U scaling has been selected as the best class-aware granulation option since it proved a good trade-off between accuracy, running times and genetic code complexity according to the discussion hold in Section 6.2.2. For what concern c), the selection of *mean* strategy is driven by the good results attained in both accuracy on test sets and cardinality of the final alphabet $\mathcal{A}^*$ as discussed in Section 6.2.4.

Competitors span a variety of techniques including classifiers working on the top of GEDs [152, 153], kernel methods [45, 48, 117], and several explicit embedding techniques [45, 154, 69], including GrC-based [114, 118] and those based on neural networks and deep learning [102, 6, 109, 80, 149].

Among the methods based on GRALG, the class-specific metric learning variant emerges as an interesting choice. Indeed, this approach attained very good results in almost all datasets when compared with other GRALG systems with the exception of Grec and Mutagenicity. For the latter datasets, the clique extractor attained the best results probably due to the fact that the selected topology better characterizes the problems. On the other hand when compared to the other methods, the class-specific metric learning strategy deployed offer considerable higher perspectives for analytical investigation by field experts thanks to its interpretability advantages given by the ability in building class-specific sets of information granules, each of which characterized by a proper class-specific dissimilarity measure parameters.

By inspecting Table 6.4, it is possible to spot the GRALG performances with respect to the main competitors. For AIDS dataset, the vast majority of the solutions reported can attain good accuracy values on the tests ($\approx 99\%$). The same hold for all GRALG variants with the exception of *mean* symbolic histogram configuration. When GREC dataset is considered, GRALG equipped with the clique extractor is able to attain similar results compared with the best choice in the State-of-the-Art (*Graph of Words + k-NN*), respectively 94.47% and 97.5%. Concerning LETTER-L dataset, State-of-the-Art methods and GRALG have similar high performances in the range of 98% – 99% correctly identified patterns. In the immediately harder LETTER problem, i.e. LETTER-M, GRALG equipped with *mean* symbolic histogram embedding strategy emerges as the second best option with 94% of accuracy whereas the top solution (*Lipschitz Embedding + SVM*) attains 95.9%. Concerning the hardest LETTER dataset, i.e. LETTER-H, *Lipschitz Embedding + SVM* is confirmed as being the best solution with 92.5% of accuracy whereas GRALG with *mean* symbolic histogram embedding strategy still provide comparable performances, that is 91% of accuracy, similarly to the class-specific metric learning variant which attained $\approx 90\%$ of correctly identified pattern on test sets. For MUTAGENICITY dataset, all GRALG variants are not able to compete with *EigenGCN-1* method which is the top solution obtaining $\approx 80\%$ of accuracy[6].

6.3 Tests and Results for Graph E-ABC Classifier

In this section, the performances of the proposed classifier Graph E-ABC (see Chapter 5) are discussed. Amonst the large number of hyper-parameters, the first test described in Section 6.3.1 aims at determining the most suitable values for a pair of parameters that has been considered critical under the performance point of view: the Granulator Agent sample size W (see Section 5.4.1) and T, the Alphabet Selector Agents upper bound considered in the generation of new candidate alphabet (see Section 5.4.4). In Section 6.3.2, the classifier performances are compared against a selection of techniques that share with Graph E-ABC the graph embedding approach, that is all methods involved in the comparison are based on graph embedding via Information Granulation. Finally in Section 6.3.3, the proposed classification system is compared with State-of-the-Art approaches.

In all the tests, the algorithm parameters are set as follows. For the orchestration of agents in $\mathcal{A}$:

- The sets of sampled subgraphs $\mathcal{D}^{(\mathrm{tr})}$ is built accordingly to a random walk extraction

[6]In Table 6.4, *Weighted Jaccard Hypergraph kernel + SVM* appears as the best solution. Nonetheless, the result is obtained with a cross-validation method, hence it has not been considered in the comparison.

- The BSAS resolution θ assumes linearly spaced values in range $[0, 1]$ with step size 0.1
- $\mu = 5$ and $\lambda = 15$, the parents and offspring population sizes (respectively)
- $\eta = 0.5$, weight between compactness Φ and cardinality Θ in Eq. (5.4.1)
- $\sigma = 0.9$ weight between the accuracy of classifier and the sparsity of the alphabet in Eq. (5.19)
- The maximum number of generation $N_{stop} = 20$.

For the orchestration of the model evolution:

- $N = 10$, the number of individuals in $\mathscr{Z}$ generated in each generation
- $Z = 10$, the number of best individuals in elite pool of agents $\mathscr{Z}^*$
- $K = 20$, the number of recombined individuals populating $\hat{\mathscr{Z}}$ in each generation

Other parameters include:

- $\beta_{\min} = -10$ and $\beta_{\max} = 10$, respectively the maximum penalty and the maximum reward values for updating the symbol quality Q_s
- $\tau_{cns} = 0.1$ as threshold of dissimilarity between symbols in Eq. (5.16) for the consensus phase
- ρ assumes equally spaced values in the range $[0, 1]$ with step size $\frac{1}{N_{stop}}$ in Eq. (5.23)
- The number of bins for the lookup table in Section 5.4.6 equals the number of classes, so it changes in a dataset-dependent fashion
- $\xi = 1.1$ as tolerance value for the symbolic histograms evaluation in the embedding phase.

The remaining two parameters (T and W) are subject to a sensitivity analysis, as detailed in the following.

6.3.1 Sensitivity Analysis

Amongst the parameters that characterize Graph E-ABC, two of them has been identified as critical for performances and need a careful tuning: the number of subgraphs that Granulator agents need to extract before running BSAS (W) and the maximum number of symbols Alphabet Selector Agent can include in a new alphabet (T). In order to address how these two parameters affect the performances of the overall system, a sensitivity analysis is performed via grid searching. In particular, a set of candidate values are chosen for both T and W, namely $T = W = [10, 50, 100, 200]$. For each $\langle T, W\rangle$-pair, the accuracy on test set and the number of symbols are recorded. In order to take into account the stochastic nature of the algorithm results are average on 10 different runs.

In Figure 6.23, the sensitivity analysis of Graph E-ABC is shown with respect to the grid over $T \times W$. From the sensitivity analysis emerges that T is the most critical parameter affecting the performances: in particular, for Aids, Letter-L and Mutagenicity, performances tend to deteriorate as $T \to 0$, whereas for harder classification problems such as Letter-M and Letter-H, performances tend to deteriorate for larger T. As W is concerned, Letter-M is the only dataset which shows an increasing trend in performances as W grows, whereas for the remaining five datasets a clear trend does not emerge.

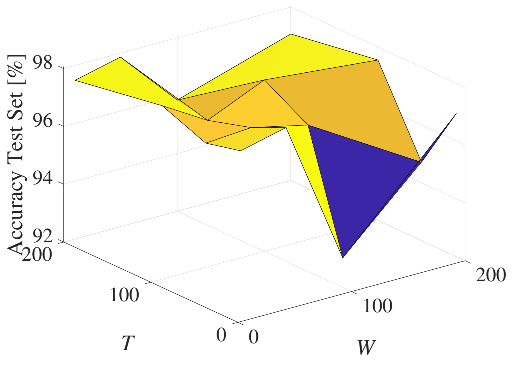
6.3.2 Comparison Against Current Granular Approaches for Graph Classification

In order to compare the performances of Graph E-ABC, four suitable competitors are selected. Like Graph E-ABC, the selected methods exploit the steps on information granulation and graph embedding via symbolic histograms: the class-aware version of GRALG (see Section 4.2), the Class Specific Metric Learning variant of GRALG (see Section 4.3, the RECTIFIER and Dual-RECTIFIER classifiers proposed in [118]. The major difference between Graph E-ABC and the four competitors is that Graph E-ABC follows a 'cooperative approach', where different agents exploit independent portions of the dataset in order to search for suitable symbols and later they join forces (via another set of agents) for building the classification model. GRALG, RECTIFIER and Dual-RECTIFIER, as instead, follow an 'individualistic approach', where all individuals start from the same set of subgraphs extracted from the training data that does not change throughout the evolution and each individual independently looks for suitable granules of information, performs the

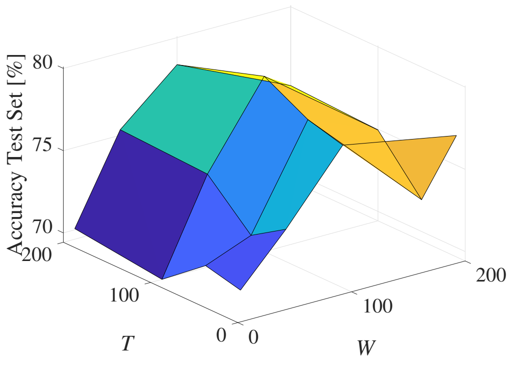
Table 6.5. Comparison against current approaches for graph classification system in terms of accuracy on the test set and, in brackets, size of the embedding space (i.e., number of symbols). For each dataset, we highlight in bold the most performing technique in terms of accuracy (in case of ties, the size of the embedding space acts as a tiebreaker).

Technique	Aids	Grec	Letter-L	Letter-M	Letter-H	Mutagenicity
Graph E-ABC (best point)	98.07 (34)	80.09 (1284)	96.89 (2260)	86.27 (4407)	79.78 (2517)	66.13 (370)
Graph E-ABC (average)	96.95 (222)	74.56 (1748)	92.20 (1460)	81.79 (3879)	74.76 (4592)	64.12 (654)
RECTIFIER (5%)*	98.12 (6)	92.64 (87)	93.33 (23)	86.36 (52)	84.57 (90)	-
RECTIFIER (10%)*	98.41 (8)	92.91 (169)	94.58 (32)	86.01 (71)	85.62 (145)	-
RECTIFIER (30%)*	98.30 (9)	92.78 (415)	95.99 (79)	87.94 (173)	86.89 (300)	-
RECTIFIER (50%)*	98.40 (16)	92.65 (439)	95.57 (81)	89.37 (232)	86.31 (431)	-
RECTIFIER (80%)*	98.57 (16)	93.09 (741)	94.80 (157)	87.21 (294)	88.92 (668)	-
Dual-RECTIFIER (5%)*	**99.11** (6)	94.37 (67)	94.77 (19)	85.74 (27)	82.59 (49)	-
Dual-RECTIFIER (10%)*	98.97 (9)	95.25 (77)	94.37 (22)	86.50 (42)	84.84 (86)	-
Dual-RECTIFIER (30%)*	98.95 (11)	**95.59** (173)	94.19 (29)	90.14 (77)	87.58 (162)	-
Dual-RECTIFIER (50%)*	98.81 (13)	95.22 (264)	95.11 (35)	89.88 (104)	86.67 (177)	-
Dual-RECTIFIER (80%)*	98.94 (37)	95.57 (215)	95.37 (50)	90.31 (136)	86.83 (275)	-
Class-Aware GRALG (10%) †	98.98 (10)	87.55 (467)	98.20 (129)	90.64 (184)	82.92 (298)	70.47 (284)
Class-Aware GRALG (30%) †	99.04 (9)	87.61 (545)	98.30 (207)	90.28 (267)	83.29 (284)	71.15 (212)
Class-Aware GRALG (50%) †	99.06 (10)	87.37 (434)	98.16 (219)	89.53 (183)	83.72 (344)	71.40 (213)
Class-Aware GRALG (10%)‡	98.99 (9)	88.36 (288)	98.25 (146)	89.00 (214)	83.29 (287)	70.67 (249)
Class-Aware GRALG (30%)‡	99.11 (11)	86.96 (335)	98.28 (205)	88.77 (277)	83.05 (319)	71.46 (259)
Class-Aware GRALG (50%)‡	99.11 (12)	87.15 (322)	**98.37** (218)	89.40 (311)	83.47 (313)	70.20 (346)
Class-Specif Metric Learning GRALG (10%)	98.98 (2)	82.27 (87)	96.76 (24)	90.49 (45)	89.24 (83)	69.04 (113)
Class-Specif Metric Learning GRALG (30%)	98.98 (1)	82.85 (97)	96.93 (32)	**92.13** (74)	88.27 (127)	70.20 (102)
Class-Specif Metric Learning GRALG (30%)	99.0 (2)	84.33 (127)	97.16 (43)	91.91 (82)	**89.78** (133)	69.11 (150)

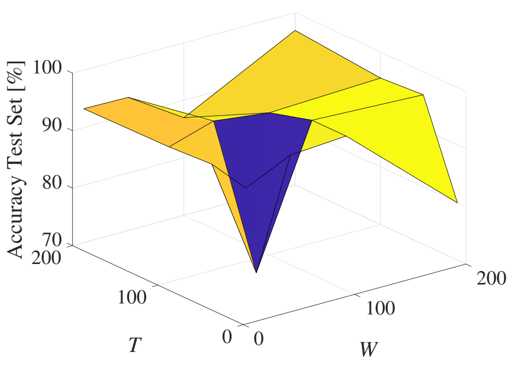
* Averaged across different trade-off values in the objective function.
† Class-Aware Uniform Scaling heuristic with BFS extractor for driving the granulation procedure.
‡ Class-Aware Frequency Scaling heuristic with BFS extractor for driving the granulation procedure.



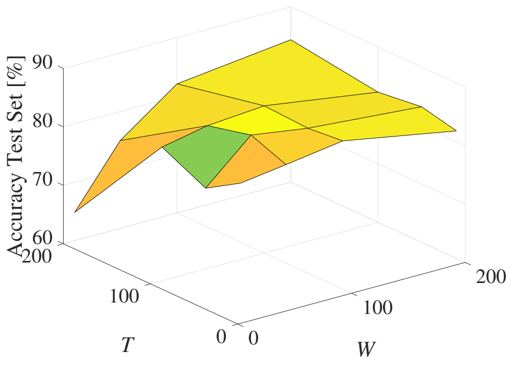
(a) AIDS



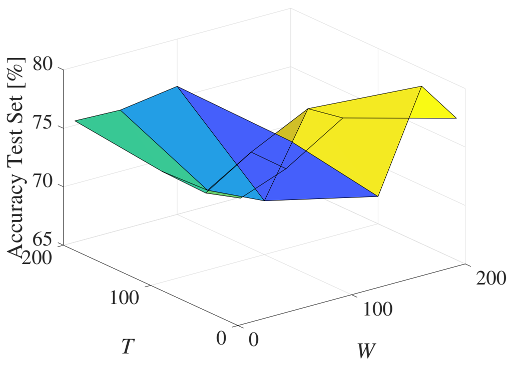
(b) GREC



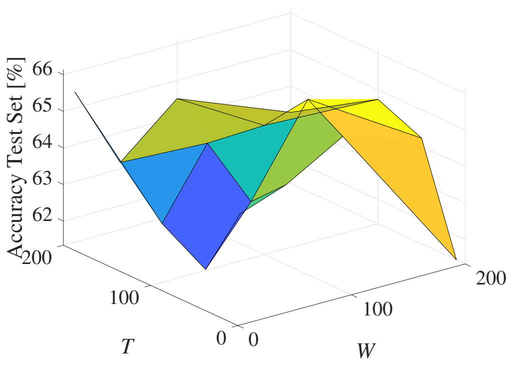
(c) Letter-L



(d) Letter-M



(e) Letter-H



(f) Mutagenicity

Figure 6.23. Graph E-ABC sensitivity analysis (average accuracy on the test set).

embedding procedure and trains the classifier in the embedding space. GRALG variants, Dual-Rectifier and Rectifier are driven by a single-objective unimodal evolutionary procedure, hence the best individual is retained for the synthesis of the final classification system, to be validated on the test set.

In Table 6.5 are shown the results of the comparison amongst Graph E-ABC, GRALG, RECTIFIER and Dual-RECTIFIER in terms of accuracy on the test set and resulting number of symbols (i.e., size of the embedding space). Results for GRALG, RECTIFIER and Dual-RECTIFIER are reported as function of visited symbols, expressed in percentage with respect to the maximum attainable number of subgraphs that can be drawn from the training set. For Graph E-ABC, as instead, the best point is selected on the grids in Figure 6.23 (that is, the $\langle T, W\rangle$-pair) leading to the maximum accuracy on the test set) and the average across all points in the grid. In terms of accuracy, Graph E-ABC does not rank amongst the most performing methods for any of the datasets. However, the following observations arise: for easy classification problems such as Aids and Letter-L, the shift of Graph E-ABC (best point) with respect to the best performing algorithm is negligible ($\approx 1\%$ and $< 2\%$, respectively). The same is not true for harder problems such as Grec and Letter-H, where the accuracy shift with respect to the best point is approximately 15% (Grec), 9% (Letter-H). For Letter-M, a mid-hardness classification problem, the accuracy shift with respect to the best performing algorithm is approximately 4%.

In terms of alphabet size, Graph E-ABC ranks as the least suitable algorithm. This is due to the ensemble-like nature of the synthesis of the model. In fact, recall from Section 5.4.8, K different classifiers are in charge of classifying the data in K different embedding spaces. Hence, if the size of each the K embedding spaces is summed together, this inevitably leads to a drastically higher number of symbols. However, if the average number of symbols that each model has to exploit is considered (i.e., the sum of symbols divided by K models), it is possible to see that the results are rather in line with the competitors for low subsampling rates (i.e., more than 50%).

6.3.3 Comparison Against State of the Art Graph Classifiers

In this section, state-of-the-art graph classification systems are compared with Graph E-ABC classifier. In Table 6.6, the accuracies on test set are shown with competitors spanning different methodology for the graph classification. For Aids dataset, most of the solutions reported achieved good accuracy values on the test ($\approx 99\%$). Similarly, Graph E-ABC achieved competitive performance on the test, $\approx 98\%$ when the best point is considered.

Concerning the simplest case of Letter dataset, i.e. Letter-L, Graph E-ABC with the best combination of W and T parameters has comparable accuracy ($\approx 97\%$) with respect to the State-of-the-Art classifiers which span in the range of 98% – 99% correctly identified patterns. On the other hand, when the problems get harder the performances slightly decreased when compared to the other methods. In Letter-M, the proposed system attained approximately 86% whereas the best solution reached by *Lipschitz Embedding + SVM* is able to identify correctly 95.9% of the test data. Concerning Letter-H which is the dataset with the highest amount of

Table 6.6. Comparison against state-of-the-art graph classification system 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**	**Mutagenicity**	**Reference**
Bipartite Graph Matching + k-NN	-	86.3	91.1	77.6	61.6	-	[153]
Lipschitz Embedding + SVM	98.3	96.8	99.3	95.9	92.5	74.9	[154]
Graph Edit Distance + k-NN	97.3	95.5	99.6	94	90	71.5	[152]
Graph of Words + k-NN	-	97.5	98.8	-	-	-	[69]
Graph of Words + kPCA + k-NN	-	97.1	97.6	-	-	-	[69]
Graph of Words + ICA + k-NN	-	58.9	82.8	-	-	-	[69]
Topological embedding	99.4	-	-	-	-	77.2	[45, 168]
FMGE	99.0	-	-	-	-	76.5	[45, 101]
Attribute Statistics	99.6	-	-	-	-	76.5	[45, 70]
ODD ST_+ kernel	82.06*	-	-	-	-	-	[48]
ODD ST_+^{TANH} kernel	82.54*	-	-	-	-	-	[48]
Laplacian kernel	92.6	-	-	-	-	70.2	[45, 30]
Treelet kernel	99.1	-	-	-	-	77.1	[45, 64]
Treelet kernel with MKL	99.7	-	-	-	-	77.6	[45, 66]
Weighted Jaccard Hypergraph kernel + SVM	99.5*	-	-	-	-	82*	[117]
CGMM + linear SVM	84.16*	-	-	-	-	-	[6]
G-L-Perceptron	-	70	95	64	70	-	[109]
G-M-Perceptron	-	75	98	87	81	-	[109]
C-1NN	-	-	96	93	84	-	[109]
C-M-1NN	-	-	98	81	71	-	[109]
EigenGCN-1	-	-	-	-	-	80.1	[102]
EigenGCN-2	-	-	-	-	-	78.9	[102]
EigenGCN-3	-	-	-	-	-	79.5	[102]
GCN with logical descriptors	-	96.93	96.64	85.27	79.91	-	[80]
MPNN	-	89.5	91.3	81.2	64.24	-	[149]
MPNN (no set2set)	-	92.98	94.8	86.1	75.7	-	[149]
Hypergraph Embedding + SVM	99.3	-	-	-	-	77.0	[114]
RECTIFIER + k-NN	99.07	95.57	97.12	92.16	91.60	-	[118]
Dual RECTIFIER + k-NN	99.13	96.61	96.40	93.04	91.31	-	[118]
Graph E-ABC (best point)	98.07	80.09	96.89	86.27	79.78	66.13	This work

distortion, Graph E-ABC has a negative shift in performance of approximately 12% with respect to *Lipschitz Embedding + SVM* method, that it is confirmed as the best solution for Letter problems with 92.5% of accuracy against $\approx$ 79% obtained by Graph E-ABC.

For Mutagenicity dataset, the best solution is achieved by *EigenGCN-1* (Graph Convolutional Neural Network method) which show 80% of accuracy, whereas Graph E-ABC is able to correctly recognize $\approx$ 66% of the test set data witnessing a negative shift of 14%.

Finally for Grec dataset, Graph E-ABC performance is approximately 80% of accuracy, with a negative shift of $\approx$ 17% when compared with the best State of the Art method for this problem (*Graph of Words + k-NN*)) which achieved 97.5% of correct identified patterns in test set.

The discussed results, where Graph E-ABC shows comparable performance with respect to two datasets out of six, underline the prototypical facet of the proposed method. As discussed in Section 5.5, different algorithmic aspect characterizing Graph E-ABC, such as a more accurate method for defining the symbols/agents qualities and an adaptive rewarding strategy, need to be investigate in order to possibly improve the general performance.

Chapter 7

Conclusions

In this thesis, Granular Computing paradigm has been investigated for graph embedding purposes in order to enable pattern recognition techniques in the graph domain. GRALG has been introduced as a graph classification system leveraging on the Information Granulation principles. The modular design of GRALG allows exploring new possible solutions enhancing the system under different points of view as the learning performances, the interpretability and computational aspects as well. The GRALG framework and the novelties introduced have been used as starting points for a novel graph classification system based on Multi-Agent System.

The first proposed technique presented in Section 4.1 addressed the possibility of designing a Granular Computing-based classification system for labelled graphs, i.e. GRALG, by performing stochastic extraction procedures on the training data in order to improve the information granulation procedure in terms of running time and memory footprint. The extractor is able to provide different subgraph topologies according to the traversal strategies employed, namely Breadth First Search, Depth First Search and a maximal clique enumeration method, i.e. Bron-Kerbosh algorithm. The hypothesis behind a stochastic granulation procedure is that the information (regularities), whether present in the dataset, can still be observed if only a randomly chosen subset of the original training data is considered. Results show that sampling subgraphs according to the proposed extraction methods drastically outperforms an exhaustive procedure in terms of running times and, at the same time, keeps mostly unaltered the classification performances achievable on the test set. Attained results somehow prove our hypothesis, at least for the considered datasets, showing that clustering techniques may be promising for synthesizing information granules even with random subsampling.

Then in Section 4.2, a novel Class-Aware Granulator able to synthesize class-specific symbols is proposed. The class-aware (stratified) granulation method leverages on an enhanced extractor which operates in a class-aware fashion as well. The hypothesis behind this method is that by composing different alphabets whose symbols represent relevant and specific substructures for a given class, the embedding space spanned by the symbolic histograms should exhibit a better separation amongst the problem-related classes. Computational results show that the class-aware method is effective (at least on the six considered datasets) by improving the classification performances for most of the datasets when compared with the implementation equipped with the stochastic extractor. Additionally, relevant improvements in terms of accuracy can also be witnessed when compared with the baseline method equipped with an exhaustive subgraphs extraction. If on one hand this method shows interesting accuracy boosts, on the other hand rapid growth of the alphabet cardinality is witnessed with respect to number of classes for the problem at hand. To address this problem, two different approaches are investigated for bounding the maximum number of clusters for each clustering ensemble procedure in the Class-Aware strategy: a first method tries to limit the number of resulting clusters by uniformly scaling the genetic algorithm upper bound for the maximum number of clusters, where this scaled value is common to all granulation instances, regardless of the class under analysis. A second method scales the upper bound in a class-aware fashion as well by weighting the maximum number of clusters according to the frequency of each class. Despite the latter case seems smarter and more suitable for unbalanced classes, the resulting genetic code grows with respect to the former case as each granulation instance will have its own bounds, increasing the search space. The achieved results proved that both strategies are effective in synthesizing meaningful embedding spaces while, at the same time, reducing the alphabet cardinality and the computational time.

A novel optimization scheme is later introduced in Section 4.3 in order to enable a class-specific metric learning for GRALG based classifier. Indeed, the procedure relies on the same building blocks, yet it considers class-specific sets of information granules, each of which is characterized by a proper class-specific dissimilarity measure parameters as opposed to GRALG where a global optimization strategy is deployed. Computational results show that while improving classification performances with respect to plain GRALG equipped with a global metric learning strategy, the proposed class specific metric learning variant provides a significant reduction of pivotal information granules in the optimized alphabet, hence a more interpretable model and a lower dimensional embedding space. The embedding spaces returned by the proposed system have also been visually compared with the previous work via t-SNE dimensionality reduction, thanks to which it is possible to see that, on average, the class-specific metric learning variant leads to embedding spaces with wider separation among patterns belonging to different classes, hence with decision boundaries easier to approximate.

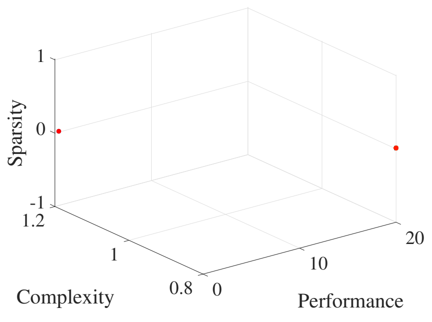
The dissertation then moved to Section 4.4, where six 'relaxed' variants of symbolic histograms for graph classification purposes are proposed. Inspired by the dissimilarity space embedding, the 'relaxed' variants take into consideration the proper magnitude of the dissimilarities when matching the pivotal granules of information against the constituent parts of the graphs to be embedded. Three operators (mean, sum and median of distances) have been proposed to aggregate such dissimilarity values, with additional three variants which include a thresholding stage in order to account only for similarities that are 'close enough' with respect to the information granule under analysis. The computational results corroborate the effectiveness of the 'relaxed' variants. Especially when the thresholding stage is not employed, the resulting set of pivotal symbols is drastically reduced with respect to the thresholded variants (including the original symbolic histogram). In conclusion, the non-thresholded mean and median emerged as the most interesting operators in order to populate the (relaxed) symbolic histogram since they led to very small embedding spaces while, at the same time, maintaining interesting performances in terms of accuracy on the test set.

In Section 4.5, the synthesis of the optimal alphabet for the information granulation graph embedding procedure has been approached as a multi-objective optimization problem. Three independent objective functions drive the evolutionary algorithm taking into account the performance of the classifier, the number of symbols necessary for building the symbolic histograms and the structural complexity of the SVM classifier. Furthermore, the possibility of selecting a small and finite number of K Pareto-efficient solutions for building an ensemble of classifiers has been explored. The peculiarity of the ensemble is that each classifier composing the ensemble is a different solution drawn from the Pareto Frontier, hence operates in a different embedding space and exploits different SVM hyperparameters, along with different parameters for the dissimilarity measure. In order to study the behavior of the ensemble (as a function of the number of classifiers), a uniform weighting case alongside three different case studies (that differ on how much each objective function weights in the selection of the solutions from the Pareto Frontier) and three baseline cases (where three best solutions from the Pareto Frontier are the one that minimizes each objective function, independently) has been investigated. According to the computational results, Linear-SVMs are more suitable for being stacked in the ensemble: in fact, RBF-SVMs require more classifiers in the ensemble to reach a similar accuracy with respect to the baseline case. This, in turn, yields a higher number of support vectors and a higher number of symbols. Another interesting aspect that can be spotted from the attained results is that taking into account the number of symbols rather than the number of support vectors has a more beneficial impact on the performance of the classifiers: in fact, it has been observed that joint optimization of performance and sparsity yields ensemble models that improve against the baseline case in terms of the number of symbols and accuracy on the test set. The same does not hold when TOPSIS selects Pareto optimal solutions by jointly considering the number of support vectors and performance. Finally, results show that there seem to exist a cut-off K value for which the ensemble is able to reach the same performance of the baseline case that maximizes the accuracy: in many cases, this yields an ensemble that shows a favorable behavior for another objective function (be it the number of symbols or the number of support vectors). However, once this cut-off K has passed, the growth of the number of support vectors and/or symbols does not justify any possible improvements in terms of performance. Finally, in Chapter 5, Graph E-ABC has been introduced as a graph classification system inspired by GRALG framework. As opposed to GRALG, Graph E-ABC is designed by following Multi-Agent System principles where two agent swarms collaborate together by performing individually a two-fold optimization on the granulation phase and alphabet synthesis. In this approach the division of tasks aimed at removing the a-posteriori feature selection stage that in GRALG has been designed to run after the alphabet synthesis phase. The granulation agents extract symbols from a continuously different bucket of subgraphs, evaluating the results by addressing also the opinion of other agents according to the consensus phase, as opposed to the GRALG approach, where each individual leverage on a fixed set of candidate granules. Furthermore, the alphabet of symbols in the final models, i.e. a set of clusters, are individually extracted with a suitable parametric dissimilarity measure pushing Graph E-ABC toward a local metric learning approach. Computational results on open-access datasets show interesting results in terms of accuracy for 3 out of 6 datasets. However, these interesting accuracy results are counterbalanced by a high dimensionality of the feature space. Despite promising, this prototypical implementation has some drawbacks that might affect the behavior of Graph E-ABC. Future works can be directed toward the investigation of good trade-off in the number of model in the ensemble, that is, how the performance change as a function of Z. In fact, while a higher number of models in the ensemble may lead to higher accuracy, it also leads to a higher number of symbols (since, trivially, more embedding spaces have to be tested). Another potential future work relates to the evaluation of symbol qualities. In fact, the quality of each symbol is evaluated independently from other selected symbols belonging to the same alphabet for embedding: this approach does not allow to capture the correlation amongst symbols, that is, whether there exist 'groups of symbols' that are responsible for a fruitful embedding space.

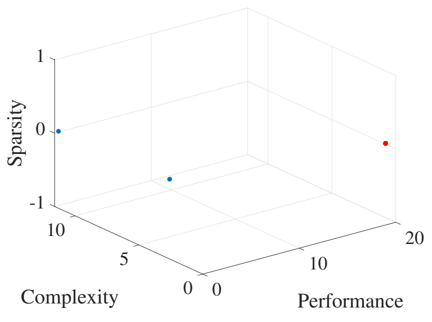
Appendix A

Pareto Frontiers with TOPSIS Selection

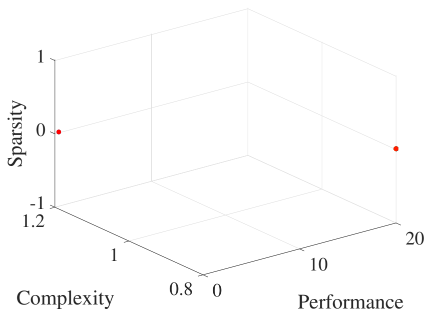
In this Appendix, we show the 3D Pareto Fronts (pairwise 2D projections are omitted for brevity) highlighting the top $K = 10$ solutions returned by TOPSIS under different weighting schemes. In particular, Figures A.1, A.2, A.3, A.4 and A.5 depict the Pareto Fronts for AIDS, GREC, LETTER-L, LETTER-M and LETTER-H, respectively. Each Figure features a 4×2 layout, where each column correspond to a classifier (Linear-SVM or RBF-SVM) and each row corresponds to a different weighting scheme, according to the different scenarios presented in Section 6.2.5. Selected solutions are highlighted via a color scale ranging from red (top ranked solutions) to yellow (least ranked solutions), with the remaining solutions featuring the 'standard' blue color (already in Figures 6.14–6.18). Worthy of attention is Figure A.1 (dataset AIDS): in fact, since the N solutions are overlapping into two or three points in the Pareto Front, selected solutions are overlapping to non-selected solutions.



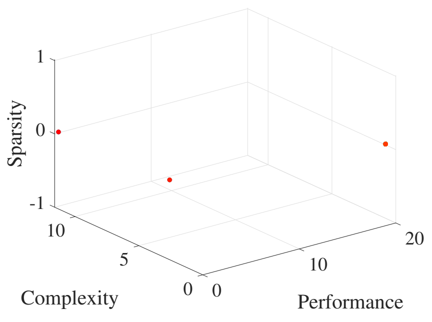
(a) Uniform weighting



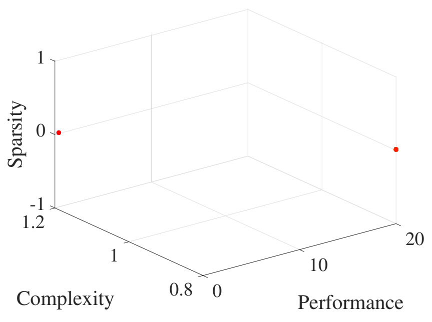
(b) Uniform weighting



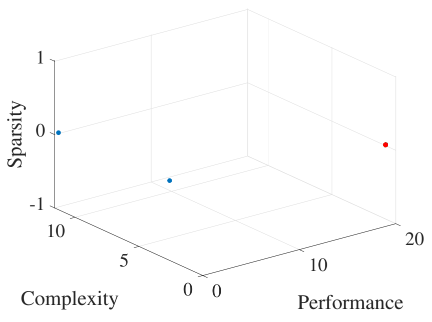
(c) Case Study A



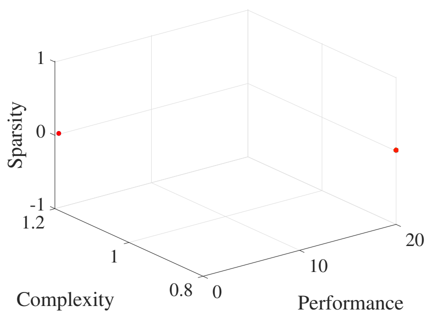
(d) Case Study A



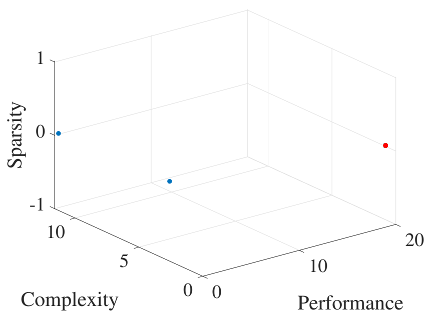
(e) Case Study B



(f) Case Study B

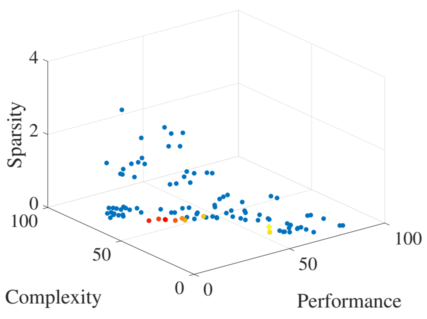


(g) Case Study C

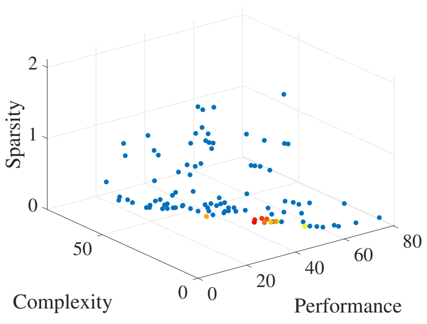


(h) Case Study C

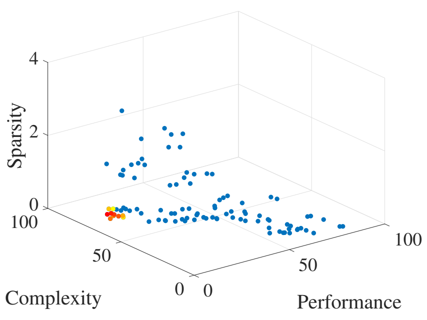
Figure A.1. Dataset: AIDS. Left panels correspond to Linear-SVM, right panels correspond to RBF-SVM



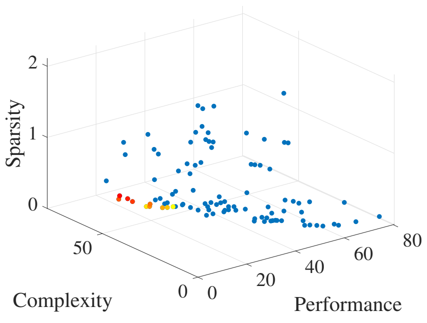
(a) Uniform weighting



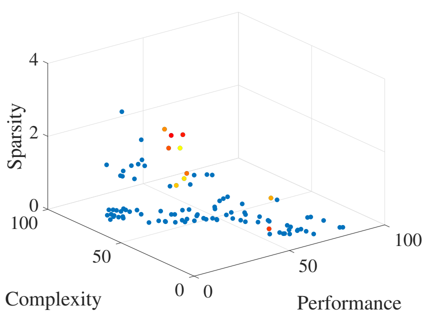
(b) Uniform weighting



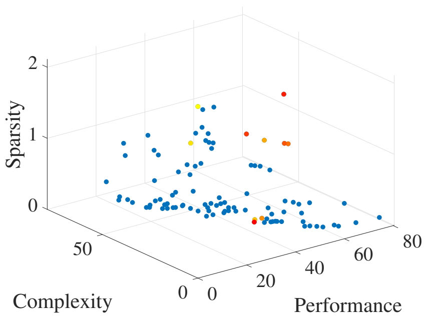
(c) Case Study A



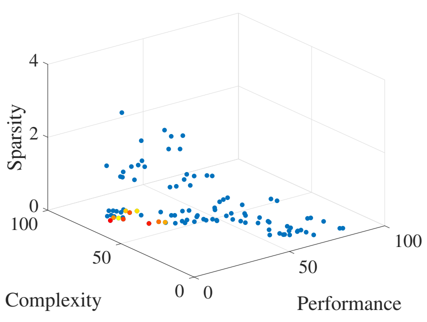
(d) Case Study A



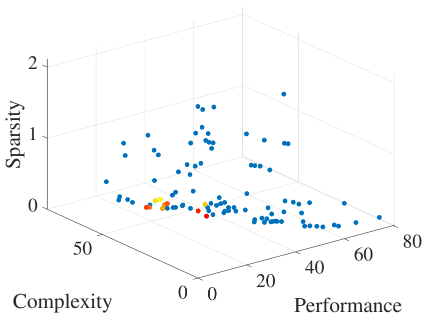
(e) Case Study B



(f) Case Study B

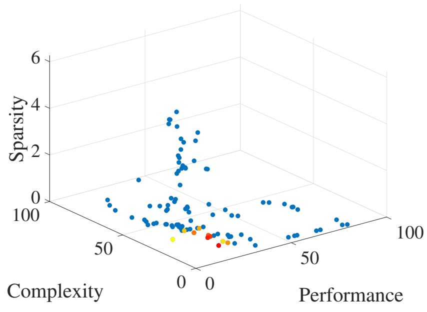


(g) Case Study C

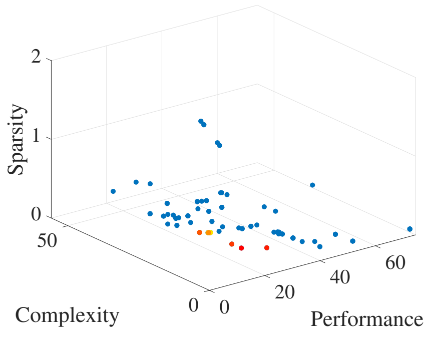


(h) Case Study C

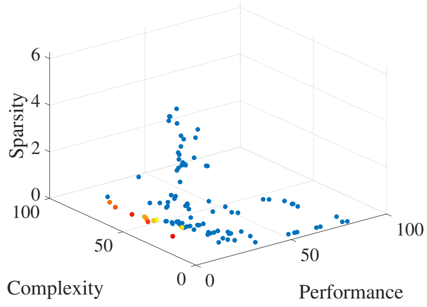
Figure A.2. Dataset: GREC. Left panels correspond to Linear-SVM, right panels correspond to RBF-SVM



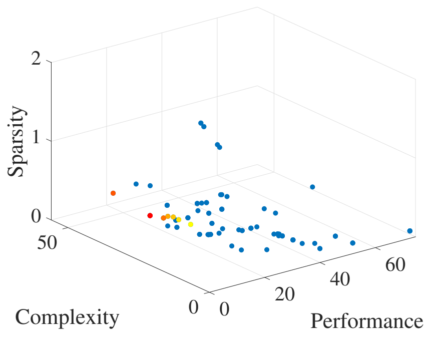
(a) Uniform weighting



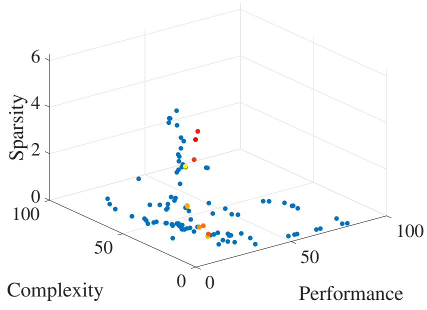
(b) Uniform weighting



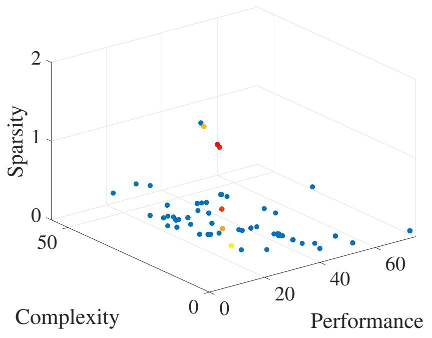
(c) Case Study A



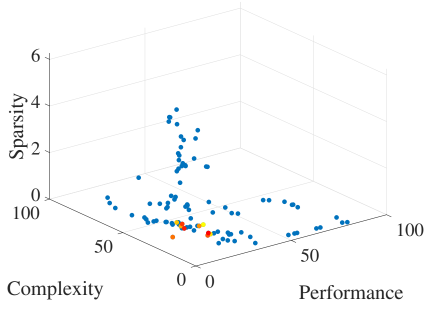
(d) Case Study A



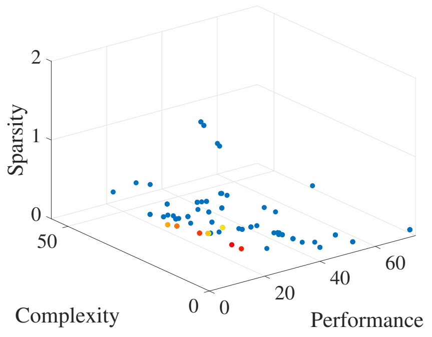
(e) Case Study B



(f) Case Study B

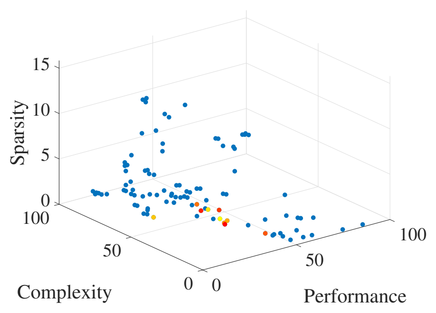


(g) Case Study C

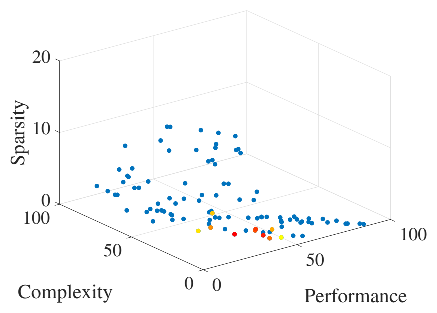


(h) Case Study C

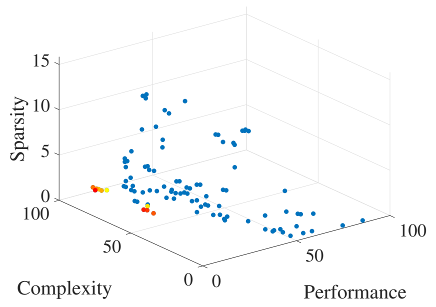
Figure A.3. Dataset: Letter-L. Left panels correspond to Linear-SVM, right panels correspond to RBF-SVM



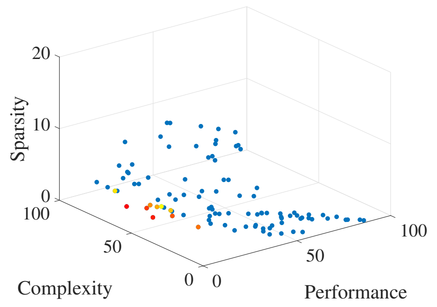
(a) Uniform weighting



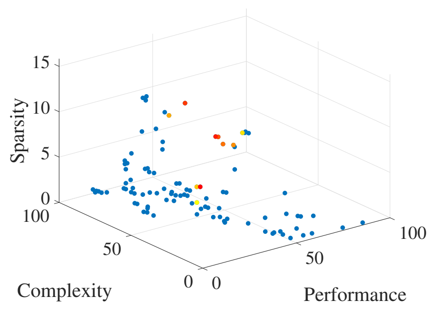
(b) Uniform weighting



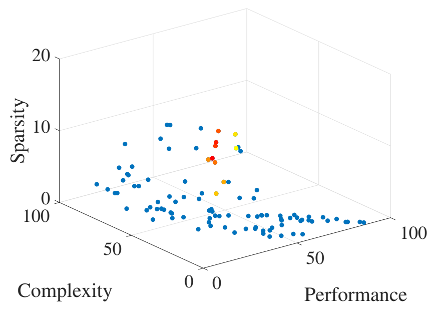
(c) Case Study A



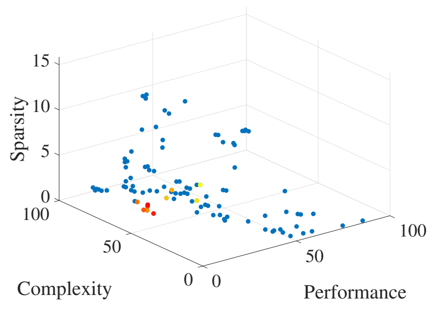
(d) Case Study A



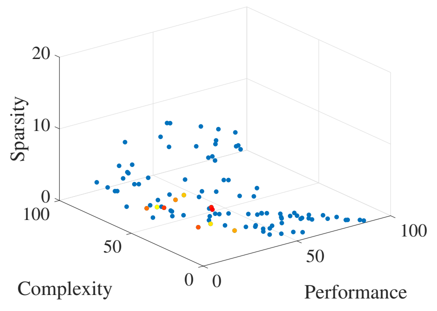
(e) Case Study B



(f) Case Study B

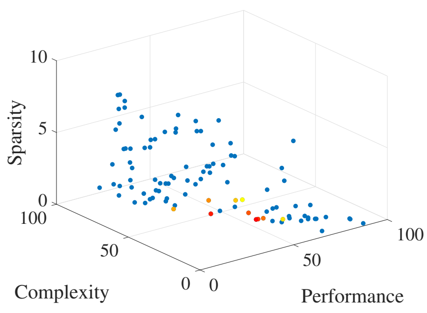


(g) Case Study C

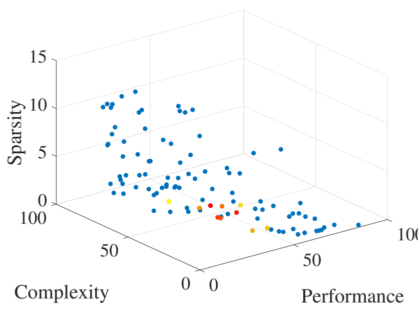


(h) Case Study C

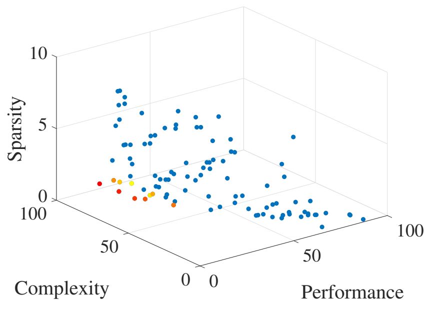
Figure A.4. Dataset: Letter-M. Left panels correspond to Linear-SVM, right panels correspond to RBF-SVM



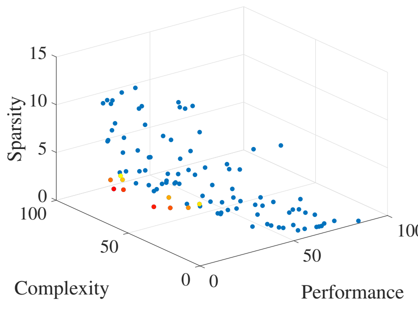
(a) Uniform weighting



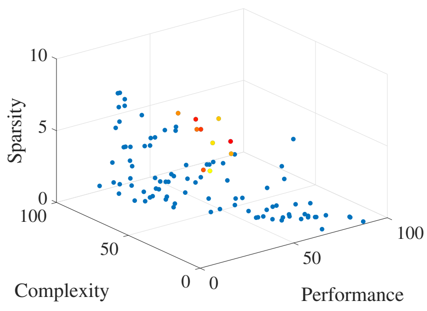
(b) Uniform weighting



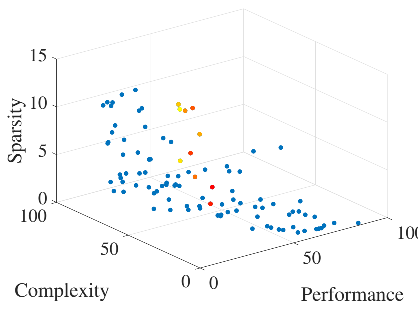
(c) Case Study A



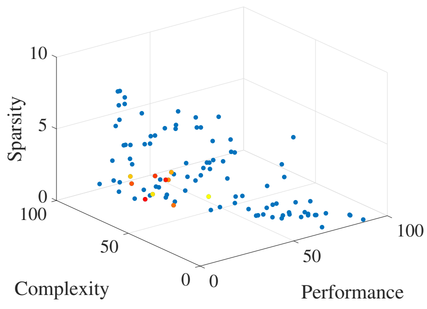
(d) Case Study A



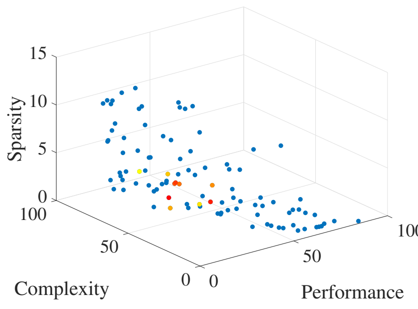
(e) Case Study B



(f) Case Study B



(g) Case Study C



(h) Case Study C

Figure A.5. Dataset: Letter-H. Left panels correspond to Linear-SVM, right panels correspond to RBF-SVM

Bibliography

[1] Abdullah, M. F. A., Sayeed, M. S., Muthu, K. S., Bashier, H. K., Azman, A., and Ibrahim, S. Z. Face recognition with symmetric local graph structure (slgs). *Expert Systems with Applications*, **41** (2014), 6131.

[2] Aggarwal, C. C., Wolf, J. L., Yu, P. S., Procopiuc, C., and Park, J. S. Fast algorithms for projected clustering. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, SIGMOD '99, p. 61–72. Association for Computing Machinery, New York, NY, USA (1999). ISBN 1581130848. doi:10.1145/304182.304188.

[3] Aittokallio, T. and Schwikowski, B. Graph-based methods for analysing networks in cell biology. *Briefings in Bioinformatics*, **7** (2006), 243.

[4] Akoglu, L., Tong, H., and Koutra, D. Graph based anomaly detection and description: a survey. *Data Mining and Knowledge Discovery*, **29** (2015), 626.

[5] Annamdas, K. K. and Rao, S. S. Multi-objective optimization of engineering systems using game theory and particle swarm optimization. *Engineering Optimization*, **41** (2009), 737. doi:10.1080/03052150902822141.

[6] Bacciu, D., Errica, F., and 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).

[7] Bacciu, D., Errica, F., Micheli, A., and Podda, M. A gentle introduction to deep learning for graphs. *Neural Networks*, **129** (2020), 203. Available from: https://www.sciencedirect.com/science/article/pii/S0893608020302197, doi:https://doi.org/10.1016/j.neunet.2020.06.006.

[8] Baldini., L., Martino., A., and Rizzi., A. Stochastic information granules extraction for graph embedding and classification. In *Proceedings of the 11th International Joint Conference on Computational Intelligence - NCTA, (IJCCI 2019)*, pp. 391–402. INSTICC, SciTePress (2019). ISBN 978-989-758-384-1. doi:10.5220/0008149403910402.

[9] Baldini., L., Martino., A., and Rizzi., A. Complexity vs. performance in granular embedding spaces for graph classification. In *Proceedings of the 12th International Joint Conference on Computational Intelligence - NCTA,*,

pp. 338–349. INSTICC, SciTePress (2020). ISBN 978-989-758-475-6. doi:10.5220/0010109503380349.

[10] Baldini, L., Martino, A., and Rizzi, A. Exploiting cliques for granular computing-based graph classification. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9 (2020). doi:10.1109/IJCNN48605.2020.9206690.

[11] Baldini, L., Martino, A., and Rizzi, A. A class-specific metric learning approach for graph embedding by information granulation. *Applied Soft Computing*, (2021), 108199.

[12] Baldini., L., Martino., A., and Rizzi., A. A multi-objective optimization approach for the synthesis of granular computing-based classification systems in the graph domain. *SN Computer Science*, (2021). Submitted for pubblication.

[13] Baldini., L., Martino., A., and Rizzi., A. Relaxed dissimilarity-based symbolic histogram variants for granular graph embedding. In *Proceedings of the 13th International Joint Conference on Computational Intelligence - NCTA,*. INSTICC, SciTePress (2021).

[14] Baldini, L., Martino, A., and Rizzi, A. *Towards a Class-Aware Information Granulation for Graph Embedding and Classification*, pp. 263–290. Springer International Publishing, Cham (2021). ISBN 978-3-030-70594-7. Available from: https://doi.org/10.1007/978-3-030-70594-7_11, doi:10.1007/978-3-030-70594-7_11.

[15] Baldini., L. and Rizzi., A. A multi-agent approach for graph classification. In *Proceedings of the 13th International Joint Conference on Computational Intelligence - NCTA,*. INSTICC, SciTePress (2021).

[16] Bargiela, A. and Pedrycz, W. Toward a theory of granular computing for human-centered information processing. *IEEE Transactions on Fuzzy Systems*, **16** (2008), 320. doi:10.1109/TFUZZ.2007.905912.

[17] Bargiela, A. and Pedrycz, W. Granular computing. In *Handbook on Computational Intelligence: Volume 1: Fuzzy Logic, Systems, Artificial Neural Networks, and Learning Systems*, pp. 43–66. World Scientific (2016).

[18] Belghache, E., Georgé, J.-P., and Gleizes, M.-P. Towards an adaptive multi-agent system for dynamic big data analytics. In *2016 Intl IEEE Conferences on Ubiquitous Intelligence Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, pp. 753–758 (2016). doi:10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0121.

[19] Bellet, A., Habrard, A., and Sebban, M. Metric learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, **9** (2015), 1.

[20] Bereta, M., Pedrycz, W., and Reformat, M. Local descriptors and similarity measures for frontal face recognition: A comparative analysis. *Journal of Visual Communication and Image Representation*, **24** (2013), 1213 . `doi:10.1016/j.jvcir.2013.08.004`.

[21] Beyer, H.-G. and Schwefel, H.-P. Evolution strategies – a comprehensive introduction. *Natural Computing*, **1** (2002), 3. Available from: `https://doi.org/10.1023/A:1015059928466`, `doi:10.1023/A:1015059928466`.

[22] Bianchi, F. M., Grattarola, D., Livi, L., and Alippi, C. Graph neural networks with convolutional arma filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2021), 1. `doi:10.1109/TPAMI.2021.3054830`.

[23] Bianchi, F. M., Livi, L., Rizzi, A., and Sadeghian, A. A granular computing approach to the design of optimized graph classification systems. *Soft Computing*, **18** (2014), 393.

[24] Bishop, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg (2006). ISBN 0387310738.

[25] Bollobás, B. *Modern graph theory*, vol. 184. Springer Science & Business Media (2013).

[26] Borgwardt, K. M. *Graph kernels*. Ph.D. thesis, lmu (2007).

[27] Borowska, K. and Stepaniuk, J. A rough-granular approach to the imbalanced data classification problem. *Applied Soft Computing*, **83** (2019), 105607. `doi:10.1016/j.asoc.2019.105607`.

[28] Bron, C. and Kerbosch, J. Algorithm 457: Finding all cliques of an undirected graph. *Commun. ACM*, **16** (1973), 575. `doi:10.1145/362342.362367`.

[29] Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, **34** (2017), 18.

[30] Brun, L., Conte, D., Foggia, P., and Vento, M. A graph-kernel method for re-identification. In *Image Analysis and Recognition* (edited by M. Kamel and A. Campilho), pp. 173–182. Springer Berlin Heidelberg, Berlin, Heidelberg (2011). ISBN 978-3-642-21593-3.

[31] Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, (2013).

[32] Bunke, H. Graph-based tools for data mining and machine learning. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pp. 7–19. Springer (2003).

[33] Bunke, H. and Riesen, K. Recent advances in graph-based pattern recognition with applications in document analysis. *Pattern Recognition*, **44** (2011), 1057. Available from: `https://www.sciencedirect.com/science/article/pii/S003132031000542X`, `doi:https://doi.org/10.1016/j.patcog.2010.11.015`.

[34] Bunke, H. and Riesen, K. Towards the unification of structural and statistical pattern recognition. *Pattern Recognition Letters*, **33** (2012), 811. Special Issue on Awards from ICPR 2010. Available from: `https://www.sciencedirect.com/science/article/pii/S0167865511001309`, `doi:https://doi.org/10.1016/j.patrec.2011.04.017`.

[35] Cai, H., Zheng, V. W., and Chang, K. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge & Data Engineering*, **30** (2018), 1616. `doi:10.1109/TKDE.2018.2807452`.

[36] Capillo, A., De Santis, E., Mascioli, F. M. F., and Rizzi, A. Mining m-grams by a granular computing approach for text classification. In *IJCCI*, pp. 350–360 (2020).

[37] Cavallari, S., Cambria, E., Cai, H., Chang, K. C.-C., and Zheng, V. W. Embedding both finite and infinite communities on graphs [application notes]. *IEEE Computational Intelligence Magazine*, **14** (2019), 39. `doi:10.1109/MCI.2019.2919396`.

[38] Chang, C.-C. A boosting approach for supervised mahalanobis distance metric learning. *Pattern Recognition*, **45** (2012), 844 . `doi:10.1016/j.patcog.2011.07.026`.

[39] Chapelle, O., Scholkopf, B., and Zien, A. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, **20** (2009), 542.

[40] Chen, X. Understanding spectral graph neural network. *arXiv preprint arXiv:2012.06660*, (2020).

[41] Chen, Y., Garcia, E. K., Gupta, M. R., Rahimi, A., and Cazzanti, L. Similarity-based classification: Concepts and algorithms. *Journal of Machine Learning Research*, **10** (2009).

[42] Chiaselotti, G., Ciucci, D., and Gentile, T. Simple graphs in granular computing. *Information Sciences*, **340-341** (2016), 279 . `doi:10.1016/j.ins.2015.12.042`.

[43] Cios, K. J., Pedrycz, W., and Swiniarski, R. W. *Data mining methods for knowledge discovery*, vol. 458. Springer Science & Business Media (2012).

[44] Conte, D., Foggia, P., Sansone, C., and Vento, M. Thirty years of graph matching in pattern recognition. *International journal of pattern recognition and artificial intelligence*, **18** (2004), 265.

[45] Conte, D., Ramel, J.-Y., Sidère, N., Luqman, M. M., Gaüzère, B., Gibert, J., Brun, L., and Vento, M. A comparison of explicit and implicit graph embedding methods for pattern recognition. In *Graph-Based Representations in Pattern Recognition* (edited by W. G. Kropatsch, N. M. Artner, Y. Haxhimusa, and X. Jiang), pp. 81–90. Springer Berlin Heidelberg, Berlin, Heidelberg (2013). ISBN 978-3-642-38221-5. `doi:10.1007/978-3-642-38221-5_9`.

[46] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to algorithms*. MIT press (2009).

[47] Cover, T. and Hart, P. Nearest neighbor pattern classification. *IEEE transactions on information theory*, **13** (1967), 21.

[48] Da San Martino, G., Navarin, N., and Sperduti, A. Ordered decompositional dag kernels enhancements. *Neurocomputing*, **192** (2016), 92 .

[49] Darwen, P. and Yao, X. Every niching method has its niche: Fitness sharing and implicit sharing compared. In *International Conference on Parallel Problem Solving from Nature*, pp. 398–407. Springer (1996).

[50] De Santis, E., Martino, A., and Rizzi, A. An infoveillance system for detecting and tracking relevant topics from italian tweets during the covid-19 event. *IEEE Access*, **8** (2020), 132527.

[51] Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, **29** (2016), 3844.

[52] Del Vescovo, G., Livi, L., Mascioli, F. M. F., and Rizzi, A. On the problem of modeling structured data with the minsod representative. *International Journal of Computer Theory and Engineering*, **6** (2014), 9.

[53] Del Vescovo, G. and Rizzi, A. Automatic classification of graphs by symbolic histograms. In *2007 IEEE International Conference on Granular Computing (GRC 2007)*, pp. 410–410. IEEE (2007).

[54] Dey, A., Broumi, S., Son, L. H., Bakali, A., Talea, M., and Smarandache, F. A new algorithm for finding minimum spanning trees with undirected neutrosophic graphs. *Granular Computing*, **4** (2019), 63. `doi:10.1007/s41066-018-0084-7`.

[55] Di Paola, L., De Ruvo, M., Paci, P., Santoni, D., and Giuliani, A. Protein contact networks: an emerging paradigm in chemistry. *Chemical Reviews*, **113** (2013), 1598.

[56] Ding, S., Du, M., and Zhu, H. Survey on granularity clustering. *Cognitive neurodynamics*, **9** (2015), 561.

[57] Dipert, R. R. The mathematical structure of the world: The world as graph. *The journal of philosophy*, **94** (1997), 329.

[58] Dubois, D. and Prade, H. Bridging gaps between several forms of granular computing. *Granular Computing*, **1** (2016), 115.

[59] Duin, R. P. and Pękalska, E. The dissimilarity space: Bridging structural and statistical pattern recognition. *Pattern Recognition Letters*, **33** (2012), 826.

[60] Emmerich, M. T. and Deutz, A. H. A tutorial on multiobjective optimization: fundamentals and evolutionary methods. *Natural computing*, **17** (2018), 585.

[61] Entringer, R. and Erdös, P. On the number of unique subgraphs of a graph. *Journal of Combinatorial Theory, Series B*, **13** (1972), 112. Available from: `https://www.sciencedirect.com/science/article/pii/0095895672900470`, `doi:https://doi.org/10.1016/0095-8956(72)90047-0`.

[62] Foggia, P., Genna, R., and Vento, M. Symbolic vs. connectionist learning: an experimental comparison in a structured domain. *IEEE Transactions on Knowledge and Data Engineering*, **13** (2001), 176. `doi:10.1109/69.917559`.

[63] Gao, X., Xiao, B., Tao, D., and Li, X. A survey of graph edit distance. *Pattern Analysis and applications*, **13** (2010), 113.

[64] Gaüzère, B., Brun, L., and Villemin, D. Two new graphs kernels in chemoinformatics. *Pattern Recogn. Lett.*, **33** (2012), 2038–2047. `doi:10.1016/j.patrec.2012.03.020`.

[65] Gaüzère, B., Grenier, P.-A., Brun, L., and Villemin, D. Treelet kernel incorporating cyclic, stereo and inter pattern information in chemoinformatics. *Pattern Recognition*, **48** (2015), 356.

[66] Gaüzère, B., Brun, L., Villemin, D., and Brun, M. Graph kernels based on relevant patterns and cycle information for chemoinformatics. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pp. 1775–1778 (2012).

[67] Giampieri, M., Baldini, L., De Santis, E., and Rizzi, A. Facing big data by an agent-based multimodal evolutionary approach to classification. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8 (2020). `doi:10.1109/IJCNN48605.2020.9206966`.

[68] Giampieri, M., De Santis, E., Rizzi, A., and Mascioli, F. M. F. A supervised classification system based on evolutive multi-agent clustering for smart grids faults prediction. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8 (2018). `doi:10.1109/IJCNN.2018.8489145`.

[69] Gibert, J., Valveny, E., and Bunke, H. Dimensionality reduction for graph of words embedding. In *Graph-Based Representations in Pattern Recognition* (edited by X. Jiang, M. Ferrer, and A. Torsello), pp. 22–31. Springer Berlin Heidelberg, Berlin, Heidelberg (2011). ISBN 978-3-642-20844-7.

[70] Gibert, J., Valveny, E., and Bunke, H. Graph embedding in vector spaces by node attribute statistics. *Pattern Recognition*, **45** (2012), 3072.

[71] Goldberg, D. E., Richardson, J., et al. Genetic algorithms with sharing for multimodal function optimization. In *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms*, pp. 41–49. Hillsdale, NJ: Lawrence Erlbaum (1987).

[72] Gorodetskii, V. Self-organization and multiagent systems: I. models of multiagent self-organization. *Journal of Computer and Systems Sciences International*, **51** (2012), 256.

[73] Guo, Y., Dibeklioglu, H., and Van der Maaten, L. Graph-based kinship recognition. In *2014 22nd International Conference on Pattern Recognition*, pp. 4287–4292. IEEE (2014).

[74] Henni, K., Mezghani, N., and Mitiche, A. Cluster density properties define a graph for effective pattern feature selection. *IEEE Access*, **8** (2020), 62841. doi:10.1109/ACCESS.2020.2981265.

[75] Hofmann, T., Schölkopf, B., and Smola, A. J. Kernel methods in machine learning. *The annals of statistics*, **36** (2008), 1171.

[76] Horgan, J. From complexity to perplexity. *Scientific American*, **272** (1995), 104.

[77] Howard, N. and Lieberman, H. Brainspace: Relating neuroscience to knowledge about everyday life. *Cognitive Computation*, **6** (2014), 35. doi:10.1007/s12559-012-9171-2.

[78] Huan, J., Bandyopadhyay, D., Wang, W., Snoeyink, J., Prins, J., and Tropsha, A. Comparing graph representations of protein structure for mining family-specific residue-based packing motifs. *Journal of Computational Biology*, **12** (2005), 657.

[79] Hwang, C.-L., Lai, Y.-J., and Liu, T.-Y. A new approach for multiple objective decision making. *Computers & Operations Research*, **20** (1993), 889. doi:10.1016/0305-0548(93)90109-V.

[80] Kajla, N. I., Missen, M. M. S., Luqman, M. M., and Coustaty, M. Graph neural networks using local descriptions in attributed graphs: An application to symbol recognition and hand written character recognition. *IEEE Access*, **9** (2021), 99103. doi:10.1109/ACCESS.2021.3096845.

[81] Kaski, S. and Peltonen, J. Dimensionality reduction for data visualization [applications corner]. *IEEE signal processing magazine*, **28** (2011), 100.

[82] Katina, P. F., Keating, C. B., Gheorghe, A. V., and Masera, M. Complex system governance for critical cyber-physical systems. *International Journal of Critical Infrastructures*, **13** (2017), 168.

[83] Kiani-Moghaddam, M., Shivaie, M., and Weinsier, P. D. *Introduction to Multi-objective Optimization and Decision-Making Analysis*, pp. 21–45. Springer International Publishing, Cham (2019). ISBN 978-3-030-12044-3. doi:10.1007/978-3-030-12044-3_2.

[84] Kim, J. and Wilhelm, T. What is a complex graph? *Physica A: Statistical Mechanics and its Applications*, **387** (2008), 2637.

[85] Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, (2016).

[86] Kolesnikova, O. Complex system view on natural language. *POLIBITS*, **62** (2020), 21.

[87] Kormushev, P., Calinon, S., and Caldwell, D. G. Reinforcement learning in robotics: Applications and real-world challenges. *Robotics*, **2** (2013), 122.

[88] Kung, S. Y. *Kernel methods and machine learning.* Cambridge University Press (2014).

[89] Kyriakopoulou, A. and Kalamboukis, T. Text classification using clustering. In *Proceedings of the Discovery Challenge Workshop at ECML/PKDD 2006*, pp. 28–38 (2006).

[90] Ladyman, J., Lambert, J., and Wiesner, K. What is a complex system? *European Journal for Philosophy of Science*, **3** (2013), 33.

[91] Li, X., Epitropakis, M. G., Deb, K., and Engelbrecht, A. Seeking multiple solutions: An updated survey on niching methods and their applications. *IEEE Transactions on Evolutionary Computation*, **21** (2016), 518.

[92] Lin, T. Y., Yao, Y. Y., and Zadeh, L. A. *Data mining, rough sets and granular computing*, vol. 95. Physica (2013).

[93] Liu, H. and Cocea, M. *Granular computing based machine learning: a big data processing approach.* Springer (2018).

[94] Liu, Y. and Kirchhoff, K. Graph-based semisupervised learning for acoustic modeling in automatic speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, **24** (2016), 1946.

[95] Livi, L. and Rizzi, A. The graph matching problem. *Pattern Analysis and Applications*, **16** (2013), 253.

[96] Livi, L., Rizzi, A., and Sadeghian, A. Granular modeling and computing approaches for intelligent analysis of non-geometric data. *Applied Soft Computing*, **27** (2015), 567.

[97] Livi, L. and Sadeghian, A. Data granulation by the principles of uncertainty. *Pattern Recognition Letters*, **67** (2015), 113. Granular Mining and Knowledge Discovery. Available from: https://

www.sciencedirect.com/science/article/pii/S0167865515001257, doi: https://doi.org/10.1016/j.patrec.2015.04.008.

[98] Lo, Y.-C., Rensi, S. E., Torng, W., and Altman, R. B. Machine learning in chemoinformatics and drug discovery. *Drug Discovery Today*, **23** (2018), 1538.

[99] Luce, R. D. and Perry, A. D. A method of matrix analysis of group structure. *Psychometrika*, **14** (1949), 95. doi:10.1007/BF02289146.

[100] Luong, N. C., Hoang, D. T., Gong, S., Niyato, D., Wang, P., Liang, Y.-C., and Kim, D. I. Applications of deep reinforcement learning in communications and networking: A survey. *IEEE Communications Surveys & Tutorials*, **21** (2019), 3133.

[101] Luqman, M. M., Ramel, J.-Y., Lladós, J., and Brouard, T. Fuzzy multilevel graph embedding. *Pattern Recognition*, **46** (2013), 551. doi: 10.1016/j.patcog.2012.07.029.

[102] Ma, Y., Wang, S., Aggarwal, C. C., and Tang, J. Graph convolutional networks with eigenpooling. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, p. 723–731. Association for Computing Machinery, New York, NY, USA (2019). ISBN 9781450362016. doi:10.1145/3292500.3330982.

[103] Mahmud, M., Kaiser, M. S., Hussain, A., and Vassanelli, S. Applications of deep learning and reinforcement learning to biological data. *IEEE transactions on neural networks and learning systems*, **29** (2018), 2063.

[104] Maiorino, E., Possemato, F., Modugno, V., and Rizzi, A. Information granules filtering for inexact sequential pattern mining by evolutionary computation. In *Proceedings of the International Joint Conference on Computational Intelligence - Volume 1*, IJCCI 2014, p. 104–111. SCITEPRESS - Science and Technology Publications, Lda, Setubal, PRT (2014). ISBN 9789897580529. doi:10.5220/0005124901040111.

[105] Maiorino, E., Possemato, F., Modugno, V., and Rizzi, A. Noise sensitivity of an information granules filtering procedure by genetic optimization for inexact sequential pattern mining. In *Computational Intelligence* (edited by J. J. Merelo, A. Rosa, J. M. Cadenas, A. Dourado, K. Madani, and J. Filipe), pp. 131–150. Springer International Publishing, Cham (2016). ISBN 978-3-319-26393-9. doi:10.1007/978-3-319-26393-9_9.

[106] Mairal, J. and Vert, J.-P. Machine learning with kernel methods. *Lecture Notes, January*, **10** (2018).

[107] Marino, K., Salakhutdinov, R., and Gupta, A. The more you know: Using knowledge graphs for image classification. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20–28 (2017). doi:10.1109/CVPR.2017.10.

[108] Marler, R. T. and Arora, J. S. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, **26** (2004), 369. doi:10.1007/s00158-003-0368-6.

[109] Martineau, M., Raveaux, R., Conte, D., and Venturini, G. Learning error-correcting graph matching with a multiclass neural network. *Pattern Recognition Letters*, **134** (2020), 68 . Applications of Graph-based Techniques to Pattern Recognition. doi:10.1016/j.patrec.2018.03.031.

[110] Martino, A., De Santis, E., and Rizzi, A. An ecology-based index for text embedding and classification. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE (2020).

[111] Martino, A., Frattale Mascioli, F. M., and Rizzi, A. On the optimization of embedding spaces via information granulation for pattern recognition. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8 (2020). doi:10.1109/IJCNN48605.2020.9206830.

[112] Martino, A., Giampieri, M., Luzi, M., and Rizzi, A. *Data Mining by Evolving Agents for Clusters Discovery and Metric Learning*, pp. 23–35. Springer International Publishing, Cham (2019). ISBN 978-3-319-95098-3. Available from: https://doi.org/10.1007/978-3-319-95098-3_3, doi:10.1007/978-3-319-95098-3_3.

[113] Martino, A., Giuliani, A., and Rizzi, A. Granular computing techniques for bioinformatics pattern recognition problems in non-metric spaces. In *Computational Intelligence for Pattern Recognition*, pp. 53–81. Springer (2018).

[114] Martino, A., Giuliani, A., and Rizzi, A. (hyper) graph embedding and classification via simplicial complexes. *Algorithms*, **12** (2019), 223.

[115] Martino, A., Giuliani, A., Todde, V., Bizzarri, M., and Rizzi, A. Metabolic networks classification and knowledge discovery by information granulation. *Computational Biology and Chemistry*, (2019), 107187. doi:10.1016/j.compbiolchem.2019.107187.

[116] Martino, A., Maiorino, E., Giuliani, A., Giampieri, M., and Rizzi, A. Supervised approaches for function prediction of proteins contact networks from topological structure information. In *Image Analysis* (edited by P. Sharma and F. M. Bianchi), pp. 285–296. Springer International Publishing, Cham (2017). ISBN 978-3-319-59126-1. doi:10.1007/978-3-319-59126-1_24.

[117] Martino, A. and Rizzi, A. (hyper)graph kernels over simplicial complexes. *Entropy*, **22** (2020). doi:10.3390/e22101155.

[118] Martino, A. and Rizzi, A. An enhanced filtering-based information granulation procedure for graph embedding and classification. *IEEE Access*, **9** (2021), 15426. doi:10.1109/ACCESS.2021.3053085.

[119] Martino, A., Rizzi, A., and Frattale Mascioli, F. M. Supervised approaches for protein function prediction by topological data analysis. In

2018 International Joint Conference on Neural Networks (IJCNN), pp. 1–8 (2018). doi:10.1109/IJCNN.2018.8489307.

[120] Miettinen, K. M. *Nonlinear Multiobjective Optimization*, vol. 12 of *International Series in Operations Research & Management Science*. Springer Science & Business Media, 1 edn. (1998).

[121] Mihalcea, R. and Radev, D. *Graph-based Natural Language Processing and Information Retrieval*. Cambridge university press (2011).

[122] Mitchell, M. Complex systems: Network thinking. *Artificial Intelligence*, **170** (2006), 1194. Special Review Issue. Available from: https://www.sciencedirect.com/science/article/pii/S000437020600083X, doi:https://doi.org/10.1016/j.artint.2006.10.002.

[123] Modugno, V., Possemato, F., and Rizzi, A. Combining piecewise linear regression and a granular computing framework for financial time series classification. In *IJCCI (ECTA)*, pp. 281–288 (2014).

[124] Molina-Lozano, H. A new fast fuzzy cocke–younger–kasami algorithm for dna strings analysis. *International Journal of Machine Learning and Cybernetics*, **2** (2011), 209.

[125] Monostori, L., Váncza, J., and Kumara, S. R. Agent-based systems for manufacturing. *CIRP annals*, **55** (2006), 697.

[126] Moon, J. W. and Moser, L. On cliques in graphs. *Israel Journal of Mathematics*, **3** (1965), 23. doi:10.1007/BF02760024.

[127] Mozer, M. C. A focused back-propagation algorithm for temporal pattern recognition. *Complex systems*, **3** (1989), 349.

[128] Mu, Y., Ding, W., and Tao, D. Local discriminative distance metrics ensemble learning. *Pattern Recognition*, **46** (2013), 2337 . doi:10.1016/j.patcog.2013.01.010.

[129] Ngatchou, P., Zarei, A., and El-Sharkawi, A. Pareto multi objective optimization. In *Proceedings of the 13th International Conference on, Intelligent Systems Application to Power Systems*, pp. 84–91. IEEE (2005).

[130] Niepert, M., Ahmed, M., and Kutzkov, K. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pp. 2014–2023. PMLR (2016).

[131] Nowozin, S. and Lampert, C. H. *Structured Learning and Prediction in Computer Vision*. Now publishers Inc (2011).

[132] Oliveto, P. S., Sudholt, D., and Zarges, C. On the benefits and risks of using fitness sharing for multimodal optimisation. *Theoretical Computer Science*, **773** (2019), 53.

[133] Panait, L. and Luke, S. Cooperative multi-agent learning: The state of the art. *Autonomous agents and multi-agent systems*, **11** (2005), 387.

[134] Paulsen, V. I. and Raghupathi, M. *An introduction to the theory of reproducing kernel Hilbert spaces*, vol. 152. Cambridge university press (2016).

[135] Pedrycz, A., Hirota, K., Pedrycz, W., and Dong, F. Granular representation and granular computing with fuzzy sets. *Fuzzy Sets and Systems*, **203** (2012), 17.

[136] Pedrycz, W. *Knowledge-based clustering: from data to information granules*. John Wiley & Sons (2005).

[137] Pedrycz, W. Human centricity in computing with fuzzy sets: an interpretability quest for higher order granular constructs. *Journal of Ambient Intelligence and Humanized Computing*, **1** (2010), 65.

[138] Pedrycz, W. Proximity-based clustering: a search for structural consistency in data with semantic blocks of features. *IEEE Transactions on Fuzzy Systems*, **21** (2013), 978.

[139] Pedrycz, W. *Granular computing: analysis and design of intelligent systems*. CRC press (2016).

[140] Pedrycz, W. and Homenda, W. Building the fundamentals of granular computing: A principle of justifiable granularity. *Applied Soft Computing*, **13** (2013), 4209 . `doi:10.1016/j.asoc.2013.06.017`.

[141] Pedrycz, W., Succi, G., Sillitti, A., and Iljazi, J. Data description: A general framework of information granules. *Knowledge-Based Systems*, **80** (2015), 98.

[142] Perrot, M., Habrard, A., Muselet, D., and Sebban, M. Modeling perceptual color differences by local metric learning. In *European Conference on Computer Vision*, pp. 96–111. Springer (2014).

[143] Peters, G. and Weber, R. Dcc: a framework for dynamic granular clustering. *Granular Computing*, **1** (2016), 1. `doi:10.1007/s41066-015-0012-z`.

[144] Possemato, F., Paschero, M., Livi, L., Rizzi, A., and Sadeghian, A. On the impact of topological properties of smart grids in power losses optimization problems. *International Journal of Electrical Power & Energy Systems*, **78** (2016), 755 . `doi:10.1016/j.ijepes.2015.12.022`.

[145] Possemato, F. and Rizzi, A. Automatic text categorization by a granular computing approach: Facing unbalanced data sets. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8 (2013). `doi:10.1109/IJCNN.2013.6707082`.

[146] Powers, D. M. W. Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation. *Journal of Machine Learning Technologies*, **2** (2011), 37.

[147] Qi, S., Wang, W., Jia, B., Shen, J., and Zhu, S.-C. Learning human-object interactions by graph parsing neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 401–417 (2018).

[148] Reddy, G. T., Reddy, M. P. K., Lakshmanna, K., Kaluri, R., Rajput, D. S., Srivastava, G., and Baker, T. Analysis of dimensionality reduction techniques on big data. *IEEE Access*, **8** (2020), 54776.

[149] Riba, P., Dutta, A., Lladós, J., and Fornés, A. Graph-based deep learning for graphics classification. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 02, pp. 29–30 (2017). doi:10.1109/ICDAR.2017.262.

[150] Ribeiro, P., Paredes, P., Silva, M. E. P., Aparicio, D., and Silva, F. A survey on subgraph counting: Concepts, algorithms, and applications to network motifs and graphlets. *ACM Comput. Surv.*, **54** (2021). Available from: https://doi.org/10.1145/3433652, doi:10.1145/3433652.

[151] Riesen, K. Structural pattern recognition with graph edit distance. In *Advances in computer vision and pattern recognition.* Springer (2015).

[152] Riesen, K. and Bunke, H. 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)*, pp. 287–297. Springer (2008).

[153] Riesen, K. and Bunke, H. Approximate graph edit distance computation by means of bipartite graph matching. *Image and Vision Computing*, **27** (2009), 950 . 7th IAPR-TC15 Workshop on Graph-based Representations (GbR 2007).

[154] Riesen, K. and Bunke, H. Graph classification by means of lipschitz embedding. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, **39** (2009), 1472.

[155] Riesen, K. and Bunke, H. *Graph classification and clustering based on vector space embedding*, vol. 77. World Scientific (2010).

[156] Rizzi, A. and 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*, pp. 33–38 (2006). doi:10.1109/MLSP.2006.275517.

[157] Rizzi, A., Del Vescovo, G., Livi, L., and Frattale Mascioli, F. M. A new granular computing approach for sequences representation and classification. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8 (2012). doi:10.1109/IJCNN.2012.6252680.

[158] Rizzi, A., Panella, M., and Frattale Mascioli, F. M. Adaptive resolution min-max classifiers. *IEEE Transactions on Neural Networks*, **13** (2002), 402. doi:10.1109/72.991426.

[159] Rizzi, A., Possemato, F., Livi, L., Sebastiani, A., Giuliani, A., and Mascioli, F. M. F. A dissimilarity-based classifier for generalized sequences by a granular computing approach. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8 (2013). doi:10.1109/IJCNN.2013.6707041.

[160] Roszkowska, E. Multi-criteria decision making models by applying the topsis method to crisp and interval data. *Multiple Criteria Decision Making*, **6** (2011), 200.

[161] Rylatt, R. M., et al. Exploring smart grid possibilities: A complex systems modelling approach. (2015).

[162] Salkind, N. J. *Encyclopedia of educational psychology.* SAGE publications (2008).

[163] Sareni, B. and Krahenbuhl, L. Fitness sharing and niching methods revisited. *IEEE transactions on Evolutionary Computation*, **2** (1998), 97.

[164] Schlichtkrull, M., Kipf, T. N., Bloem, P., Van Den Berg, R., Titov, I., and Welling, M. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pp. 593–607. Springer (2018).

[165] Schölkopf, B., Smola, A. J., Williamson, R. C., and Bartlett, P. L. New support vector algorithms. *Neural computation*, **12** (2000), 1207.

[166] Serratosa, F. Computation of graph edit distance: Reasoning about optimality and speed-up. *Image and Vision Computing*, **40** (2015), 38.

[167] Shen, C., Kim, J., Liu, F., Wang, L., and van den Hengel, A. Efficient dual approach to distance metric learning. *IEEE Transactions on Neural Networks and Learning Systems*, **25** (2014), 394.

[168] Sidère, N., Héroux, P., and 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*, pp. 681–685 (2009). doi:10.1109/ICDAR.2009.218.

[169] Silver, D., et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, **362** (2018), 1140.

[170] Soman, K., Loganathan, R., and Ajay, V. *Machine learning with SVM and other kernel methods.* PHI Learning Pvt. Ltd. (2009).

[171] Takigawa, I. and Mamitsuka, H. Graph mining: procedure, application to drug discovery and recent advances. *Drug Discovery Today*, **18** (2013), 50.

[172] Theodoridis, S. and Koutroumbas, K. *Pattern Recognition, Fourth Edition.* Academic Press, Inc., USA, 4th edn. (2008). ISBN 1597492728.

[173] Tichy, N. An analysis of clique formation and structure in organizations. *Administrative Science Quarterly*, **18** (1973), 194.

[174] Van der Maaten, L. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, **9** (2008).

[175] Van Engelen, J. E. and Hoos, H. H. A survey on semi-supervised learning. *Machine Learning*, **109** (2020), 373.

[176] Varnek, A. and Baskin, I. I. Chemoinformatics as a theoretical chemistry discipline. *Molecular Informatics*, **30** (2011), 20.

[177] Wang, F. and Sun, J. Survey on distance metric learning and dimensionality reduction in data mining. *Data Mining and Knowledge Discovery*, **29** (2015), 534. doi:10.1007/s10618-014-0356-z.

[178] Wang, G., Yang, J., and Xu, J. Granular computing: from granularity optimization to multi-granularity joint problem solving. *Granular Computing*, **2** (2017), 105. doi:10.1007/s41066-016-0032-3.

[179] Wang, S., Wan, J., Zhang, D., Li, D., and Zhang, C. Towards smart factory for industry 4.0: a self-organized multi-agent system with big data based feedback and coordination. *Computer networks*, **101** (2016), 158.

[180] Wang, X. and Gupta, A. Videos as space-time region graphs. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 399–417 (2018).

[181] Wang, X., Pedrycz, W., Gacek, A., and Liu, X. From numeric data to information granules: A design through clustering and the principle of justifiable granularity. *Knowledge-Based Systems*, **101** (2016), 100.

[182] Wang, Z. and Rangaiah, G. P. Application and analysis of methods for selecting an optimal solution from the pareto-optimal front obtained by multiobjective optimization. *Industrial & Engineering Chemistry Research*, **56** (2017), 560. doi:10.1021/acs.iecr.6b03453.

[183] Wasserman, L. Topological data analysis. *Annual Review of Statistics and Its Application*, **5** (2018), 501. Available from: https://doi.org/10.1146/annurev-statistics-031017-100045, arXiv:https://doi.org/10.1146/annurev-statistics-031017-100045, doi:10.1146/annurev-statistics-031017-100045.

[184] Weiss, G. and Dillenbourg, P. What is' multi'in multiagent learning. *Collaborative learning. Cognitive and computational approaches*, (1999), 64.

[185] William-West, T. O. and Singh, D. Information granulation for rough fuzzy hypergraphs. *Granular Computing*, **3** (2018), 75. doi:10.1007/s41066-017-0057-2.

[186] Wong, K.-C. *Evolutionary multimodal optimization: A short survey*, pp. 1–15. Nova Science Publishers, Inc. (2015). ISBN 9781634826945.

[187] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, (2020).

[188] Xu, F., Uszkoreit, H., Du, Y., Fan, W., Zhao, D., and Zhu, J. Explainable ai: A brief survey on history, research areas, approaches and challenges. In *Natural Language Processing and Chinese Computing* (edited by J. Tang, M.-Y. Kan, D. Zhao, S. Li, and H. Zan), pp. 563–574. Springer International Publishing, Cham (2019). ISBN 978-3-030-32236-6.

[189] Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations*, pp. 1–17 (2019).

[190] Yang, J., Wang, G., and Zhang, Q. Knowledge distance measure in multigranulation spaces of fuzzy equivalence relations. *Information Sciences*, **448** (2018), 18. doi:10.1016/j.ins.2018.03.026.

[191] Yang, L., Jin, R., Mummert, L., Sukthankar, R., Goode, A., Zheng, B., Hoi, S. C. H., and Satyanarayanan, M. A boosting framework for visuality-preserving distance metric learning and its application to medical image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **32** (2010), 30.

[192] Yao, Y. Perspectives of granular computing. In *2005 IEEE international conference on granular computing*, vol. 1, pp. 85–90. IEEE (2005).

[193] Yao, Y. Three perspectives of granular computing. *Journal of Nanchang Institute of Technology*, **25** (2006), 16.

[194] Yao, Y. A triarchic theory of granular computing. *Granular Computing*, **1** (2016), 145. doi:10.1007/s41066-015-0011-0.

[195] Yao, Y. and Zhao, L. A measurement theory view on the granularity of partitions. *Information Sciences*, **213** (2012), 1. doi:10.1016/j.ins.2012.05.021.

[196] Yao, Y.-Y. The rise of granular computing. *Journal of Chongqing University of Posts and Telecommunications (Natural Science Edition)*, **20** (2008), 299.

[197] Yin, X., Shu, T., and Huang, Q. Semi-supervised fuzzy clustering with metric learning and entropy regularization. *Knowledge-Based Systems*, **35** (2012), 304 . doi:10.1016/j.knosys.2012.05.016.

[198] Yusoff, Y., Ngadiman, M. S., and Zain, A. M. Overview of nsga-ii for optimizing machining process parameters. *Procedia Engineering*, **15** (2011), 3978. CEIS 2011. Available from: https://www.sciencedirect.com/science/article/pii/S1877705811022466, doi:https://doi.org/10.1016/j.proeng.2011.08.745.

[199] Zadeh, L. A. Soft computing and fuzzy logic. In *Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems: Selected Papers by Lotfi a Zadeh*, pp. 796–804. World Scientific (1996).

[200] Zadeh, L. A. Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic. *Fuzzy sets and systems*, **90** (1997), 111.

[201] Zeng, Z., Tung, A. K., Wang, J., Feng, J., and Zhou, L. Comparing stars: On approximating graph edit distance. *Proceedings of the VLDB Endowment*, **2** (2009), 25.

[202] Zhang, B. and Srihari, S. N. Fast k-nearest neighbor classification using cluster-based trees. *IEEE Transactions on Pattern analysis and machine intelligence*, **26** (2004), 525.

[203] Zhang, Q., Zhang, Q., and Wang, G. The uncertainty of probabilistic rough sets in multi-granulation spaces. *International Journal of Approximate Reasoning*, **77** (2016), 38.

[204] Zhang, S., Tong, H., Xu, J., and Maciejewski, R. Graph convolutional networks: a comprehensive review. *Computational Social Networks*, **6** (2019), 1.

[205] Zhang, Z., Cui, P., and Zhu, W. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, (2020), 1. doi: 10.1109/TKDE.2020.2981333.

[206] Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. Graph neural networks: A review of methods and applications. *AI Open*, **1** (2020), 57. Available from: https://www.sciencedirect.com/science/article/pii/S2666651021000012, doi: https://doi.org/10.1016/j.aiopen.2021.01.001.