

# A Multi-agent Approach for Graph Classification

Luca Baldini<sup>a</sup> and Antonello Rizzi<sup>1</sup><sup>b</sup>

*Department of Information Engineering, Electronics and Telecommunications, University of Rome "La Sapienza",  
Via Eudossiana 18, 00184 Rome, Italy*

**Keywords:** Multi-agent Systems, Graph Embedding, Supervised Learning, Structural Pattern Recognition.

**Abstract:** In this paper, we propose and discuss a prototypical framework for graph classification. The proposed algorithm (Graph E-ABC) exploits a multi-agent design, where swarm of agents (orchestrated via evolutionary optimization) are in charge of finding meaningful substructures from the training data. The resulting set of substructures compose the pivotal entities for a graph embedding procedure that allows to move the pattern recognition problem from the graph domain towards the Euclidean space. In order to improve the learning capabilities, the pivotal substructures undergo an independent optimization procedure. The performances of Graph E-ABC are addressed via a sensitivity analysis over its critical parameters and compared against current approaches for graph classification. Results on five open access datasets of fully labelled graphs show interesting performances in terms of accuracy, counterbalanced by a relatively high number of pivotal substructures.


## 1 INTRODUCTION


Multi-agent systems (Panait and Luke, 2005; Stone and Veloso, 2000; Rizk et al., 2018; Dorri et al., 2018) emerged in the last years as powerful approaches in order 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 flexibility of multi-agent systems led to different research works where such paradigm has been employed for building supervised and unsupervised learning systems. For example, in (Alamgir and Von Luxburg, 2010) a multi-agent approach has been used for local graph clustering in which each agent performs a random walk on a graph with the main constraint that such agents are "tied" together by a rope, forcing them to be close to each other. In (Carvalho et al., 2016) a set of self-organising agents by means of ant colony optimisation has been applied to anomaly detection and network control. In (Chaimontree et al., 2010) each agent runs a different clustering algorithm in order to return the best one for the dataset at hand. In (Chaimontree et al., 2011) agents negotiate one another rather than being governed by a master/wrapper process (e.g. evolutive algorithm) in

order to improve the clustering solution. In (İnkaya et al., 2015) ant colony optimisation has been used in order to organise agents, where each ant "walks" on the dataset, building connections amongst points. In (Ogston et al., 2003) each agent consists in a set of data points and agents link to each other, thus leading to clustering. In (Pan and Chen, 2012) a genetic algorithm has been used where the agents' genetic code is connection-based: each agent is a clustering result whose genetic code builds a (sub)graph and, finally, such subgraphs can be interpreted as clusters. In (Park and Oh, 2006) the multi-agent approach collapses into two agents: a first agent runs a cascade of principal component analysis, self organizing maps and  $k$ -means in order to cluster data and a second agent validates such results: the two agents interactively communicate with each other.

As supervised problems are concerned, in (Bianchi et al., 2015) a multi-agent algorithm has been proposed in which agents perform a Markovian random walk on a weighted graph representation of the input dataset. Each agent builds its own graph connection matrix amongst data points, weighting the edges according to the selected distance measure parameters, and performs a random walk on such graph in order to discover clusters. This algorithm has been employed in (Bianchi et al., 2016) for an unsupervised identification of frequent behaviours of mobile network subscribers starting from a set of call data

<sup>a</sup>  <https://orcid.org/0000-0003-4391-2598>

<sup>b</sup>  <https://orcid.org/0000-0001-8244-0015>

records. In (Pan and Jiao, 2011), the authors propose the Granular Agent Evolutionary Classification algorithm, where agents are in charge of clustering patterns with similar attributes. Further, different agents lie in different class-aware subsystems and each subsystem is characterized by a set of classification rules extracted by the information collected by the granular agents. Due to the advent of large corpora, multi-agent systems have been applied for text and document classification (Mostafa et al., 2005; Ahmad et al., 2012).

The MAS paradigm has also been successfully applied in the field of Computer Vision. In (Chen et al., 2019), the authors designed an agent based system for Scene Graph generation for visual genome understanding where objects are agents that have to maximize the quality of the generated scene graph. In (González-Briones et al., 2018), the authors proposed an image recognition system for real time applications where different agents collaborate for extracting relevant features from the acquired images in order to determine the age and gender of individuals in an office building. The MAS paradigm has found application also in the field of bioinformatics (Corrêa et al., 2020) for protein structure prediction by distributing each relevant task among different agents in a decentralized way. The distributed aspect that characterizes multi-agent systems has led various authors to investigate their application in the field of Big Data. In (Lombardo et al., 2019), the authors proposed a general purpose actor-based system for distributed data mining addressing the problem of social data analysis such as topic detection, troll detection and hoax detection. Conversely in (Ding et al., 2018), the authors introduce a multi-agent co-evolutionary adaptive strategy for attribute reduction for big medical datasets concerning infant brain Magnetic Resonance Imaging. Finally, in (Modi and Shen, 2001), the authors propose a distributed multi-agent framework for classification, where the authors assume that each agent can see only a subset of features as the data is conceived to be decentralized.

In this paper, we exploit Evolutive Agent Based Clustering/Classification (E-ABC, for short) for solving classification problems. The peculiarities of E-ABC can be summarized as follows: a) each agent sees only a small random portion of the dataset; b) different swarms of agents work on class-stratified data; c) agents evolve via evolutionary optimization; d) the synthesis of classification models is orchestrated by an independent evolutionary-like optimization which allows the cooperation amongst different swarms. The rationale behind the latter point is in line with the divide-and-conquer approach which is

typical of multi-agent systems. In fact, the output of the swarms serves as input for the population of the evolutionary-like optimization, and the output of the latter refines the behaviour of the swarms, with the final (common) goal of synthesizing a performing classification system.

E-ABC has been originally proposed in (Martino et al., 2019a) as a clustering algorithm based on the multi-agent paradigm conceived to work in Euclidean spaces. In (Giampieri et al., 2018), it has been extended towards supervised problems by letting the resulting clusters to compose a decision cluster classifier (Di Noia et al., 2020) and applied to the classification of faulty states in a real-world smart grid. In (Giampieri et al., 2020), E-ABC has been upgraded with a multimodal optimization approach (Wong, 2015; Preuss, 2015) in order to foster the exploration of different subspaces, hence foster its local metric learning capabilities. Conversely to all previous works, in which E-ABC has been designed and employed to process data described by vectors lying in a Euclidean space, in this paper we propose an extension of E-ABC tailored to work in the graph domain. The transition from a structured domain such as the graph one to a metric space such as the Euclidean one is motivated by a well-established embedding approach already explored in works such as (Martino and Rizzi, 2021) and (Baldini et al., 2021). As will be thoroughly explained later, agents are in charge of searching for meaningful subgraphs amongst the training data and these subgraphs will act as the pivotal substructures for moving the classification problem from the graph domain towards the Euclidean space, where any classifier can freely be used.

The remainder of the paper is structured as follows: in Section 2 we describe the proposed algorithm, in Section 3 we show the computational results, in terms of sensitivity analysis to critical parameters and comparison against similar approaches for graph classification. Finally, in Section 4 we conclude the paper, remarking future directions.

## 2 GRAPH E-ABC

Graph E-ABC is a graph classification system based on multi-agent evolutive strategy. The system is intended to be multi-agent by means of a collaborative approach established between the individuals belonging to different swarms  $\Sigma = \{S_1, \dots, S_S\}$ , where  $S$  is the number of classes in the classification problem. In particular, each swarm  $S_i \in \Sigma$  is a team of agents that individually performs a simple data mining task exploiting graphs belonging to the  $i^{\text{th}}$  class

(i.e., each swarm is devoted to extract meaningful information by observing class-stratified data). The results of this process are later examined and collected together in order to be exploited for a final collaborative scope, i.e. the synthesis of candidate alphabets set  $\mathcal{A}_1, \dots, \mathcal{A}_K$ , which enable to generate different embedding spaces  $\mathcal{F}_1, \dots, \mathcal{F}_K$  exploiting the symbolic histograms approach. In the resulting embedding spaces, will finally be possible to set up common classification systems, being the generic embedding space  $\mathcal{F}_i$  an Euclidean space. In order to improve the quality of the alphabets, we designed a genetic algorithm-inspired evolutive strategy which combines and mutates the promising alphabets with custom genetic operators via an independent set of agents  $\mathcal{Z}$ .

### 2.1 Agent Behaviour

The first action performed by an agent  $a$  belonging to a swarm  $\mathcal{S}_i$  is to gather a set of class-stratified subgraphs  $\mathcal{D}_g^r$  sampled from graphs in  $\mathcal{D}_r$  belonging to class  $i$ , where the number of subgraph  $W = |\mathcal{D}_g^r|$  is a user-defined parameter. The sampling process can be designed accordingly to different graph traversal strategies which define the topology of the resulting subgraphs (Baldini. et al., 2019; Baldini et al., 2020; Baldini et al., 2021).

Once  $\mathcal{D}_g^r$  is ready,  $a$  can start the data mining process. The information extraction is performed according to the Basic Sequential Algorithmic Scheme (BSAS) clustering algorithm (Theodoridis and Koutroumbas, 2008) working directly in the graph domain. BSAS relies on three key factors: a suitable dissimilarity measure  $d : \mathcal{G} \times \mathcal{G} \rightarrow \mathbb{R}$  between patterns; a resolution parameter  $\theta \in [0, 1]$  that defines the threshold on  $d$  for including patterns into a specific cluster; the maximum number of allowed clusters  $Q$ . The adopted dissimilarity measure is a heuristic based on a Graph Edit Distance named weighted node Best Match First (nBMF) (Baldini et al., 2021). In the nBMF procedure,  $\mathbf{w} = [w_{ins}^{node}, w_{del}^{node}, w_{sub}^{node}, w_{ins}^{edge}, w_{del}^{edge}, w_{sub}^{edge}] \in \{0, 1\}^6$ , is a set of nodes and edges insertion, deletion, substitution weights for the weighted nBMF and  $\boldsymbol{\gamma}$  is the set of parameters for the dissimilarity measures between nodes and edges (if applicable). Full details on nBMF, along with detailed pseudocodes, can be found in (Baldini. et al., 2019; Martino and Rizzi, 2021). Clearly, the choice of  $\mathbf{w}$  and  $\boldsymbol{\gamma}$  is critical since wrong values could undermine the ability of  $d$  to correctly capture the proximity between semantically close graphs. The agent runs BSAS according to  $d$  and  $Q$ , generating  $\Pi = \{\mathcal{P}_1, \dots, \mathcal{P}_P\}$ , i.e. a set of partitions, each of which is obtained for a given  $\theta_{i|_{i=1}}^P$ .

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

$$f_{co}(\mathbf{C}) = 1 - \frac{1}{|\mathbf{C}| - 1} \sum_{g \in \mathbf{C}} d(g, g^*) \quad (1)$$

$$f_{ca}(\mathbf{C}) = \frac{|\mathbf{C}|}{|\mathcal{D}_g^r|} \quad (2)$$

where  $g^*$  is the cluster representative of  $\mathbf{C}$ , i.e. the MinSOD of the cluster (Martino et al., 2019b). The reliability  $r(\mathbf{C})$  reads as follows:

$$r(\mathbf{C}) = \eta \cdot f_{co} + (1 - \eta) \cdot f_{ca} \quad (3)$$

where  $\eta$  is a trade-off parameter that weights the importance of compactness against cardinality. According to a threshold  $\tau$ ,  $g^*$  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  $P_i \in \Pi$ , leading to a set of symbols  $\mathcal{B} = \{s_1, \dots, s_M\}$  discovered by agent  $a$ , where  $M$  depends on the number of symbols survived in the evaluation stage.

### 2.2 Synthesis of Classification Models

Once every agent in every swarm in  $\Sigma$  has completed its data mining process, a class specific set of symbols is returned. The sets of symbols  $\mathcal{B}$  synthesized by each agent in a specific swarm  $\mathcal{S}_i$  can be merged in a class-specific (i.e., swarm-specific) bucket of symbols  $\mathcal{H}_i|_{i=1}^S$ . In other words,  $\mathcal{H}_i$  contains the collective information gathered by agents working on class  $i$ .

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  $K$  individuals, whose actions can be summarized as follows:

1. The agent  $z \in \mathcal{Z}$  evaluates the maximum number of symbols per class  $t = T/S$  that can be extracted according to a user-defined bound  $T$
2. The agent explores the buckets  $\mathcal{H}_i$  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 continues for all  $z \in \mathcal{Z}$ , leading to the synthesis of  $\mathcal{A}_1, \dots, \mathcal{A}_K$ .

According to the symbolic histogram approach (Del Vescovo and Rizzi, 2007a; Del Vescovo and Rizzi, 2007b), the alphabets can now be exploited for building the vectorial representation of both training

and validation sets ( $\mathcal{D}^{tr}$  and  $\mathcal{D}^{vs}$ ). In short, the symbolic histogram for a given graph  $G$  can be evaluated via the following two-steps procedure:

1. the graph  $G$  is decomposed in  $k$  atomic units, i.e.  $G_{exp} = \{g_1, \dots, g_k\}$
2. the symbolic histogram  $\mathbf{h}_G^{\mathcal{A}}$  is defined as:

$$\mathbf{h}_G^{\mathcal{A}} = [\text{occ}(s_1, G_{exp}), \dots, \text{occ}(s_n, G_{exp})] \quad (4)$$

where the function  $\text{occ} : \mathcal{A} \times \mathcal{G} \rightarrow \mathbb{N}$  performs the process of counting the occurrences of a symbol  $s_i \in \mathcal{A}$  in  $G_{exp}$ :

$$\text{occ}(s_i, G_{exp}) = \sum_{g \in G_{exp}} \Gamma(s_i, g) \quad (5)$$

where

$$\Gamma(s_i, g) = \begin{cases} 1 & \text{if } d(s_i, g) \leq \zeta_{s_i} \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

and  $d(\cdot, \cdot)$  is the nBMF dissimilarity measure introduced in Section 2.1.

Building the symbolic histograms of all graphs in  $\mathcal{D}_{tr}$  and  $\mathcal{D}_{vs}$  leads to the definition of  $\mathbf{F}_i^{tr}$  and  $\mathbf{F}_i^{vs}$ , respectively an  $|\mathcal{D}_{tr}| \times |\mathcal{A}_i|$  and an  $|\mathcal{D}_{vs}| \times |\mathcal{A}_i|$  matrix, whose rows are the symbolic histograms obtained with the  $i^{\text{th}}$  alphabet, for  $i = 1, \dots, K$ . A set of classification models  $c_1, \dots, c_K$  can be build accordingly to specific embedding spaces:

1. Train a classifier  $c_i$  in the respective embedding space  $\mathcal{F}_i$  spanned by the symbolic histograms matrix  $\mathbf{F}_i^{tr}$
2. Test the classifier by predicting the embedded validation set  $\mathbf{F}_i^{vs}$
3. Evaluate the performance measure  $\omega_i$  as the accuracy of  $c_i$  in correctly classify  $\mathbf{F}_i^{vs}$ .

The resulting classifiers,  $c_1, \dots, c_K$  can be retained together with their performance measure  $\omega_1, \dots, \omega_K$ .

## 2.3 Driving Evolutions

In Sections 2.1 and 2.2, we described the goal of the main actors: the swarms  $\Sigma$  must be successful in finding relevant information for building informative alphabet sets; conversely, the population  $\mathcal{Z}$  must be able to select and collect effective symbols together in order to build meaningful embedding spaces whose validity can be assessed by the classifier performances  $\omega$ . Hence, the evolutive process must take into account the different roles played by  $\Sigma$  and  $\mathcal{Z}$  in order to converge to suitable solutions. For sake of clearness, before diving into the design of the evolutionary strategy, we give a formal description of the fundamental quantities involved in the optimization phase.

### 2.3.1 Symbol Quality

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

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

The quality update strategy works as follow:

1. Select the candidate alphabet  $\mathcal{A}$
2. Select the symbol  $s \in \mathcal{A}$
3. Find the accuracy bin in which  $\omega$  lies and gather the corresponding reward value  $t$
4. Reward the symbols by applying the following update rule:

$$Q_s^{(new)} = Q_s^{(old)} + t \quad (7)$$

5. Repeat from step 2 for all symbols in the selected alphabet  $\mathcal{A}$ .

The procedure repeats from step 1 for all the alphabets under evaluation. In this approach, we interpret the performance  $\omega$  as a critic about the effectiveness of the embedding space  $\mathcal{F}$  built accordingly to the alphabet  $\mathcal{A}$ . In this way, the quality  $Q_s$  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.

### 2.3.2 Agent Quality

The agent quality measure  $Q_a$  is defined according to the qualities  $Q_{s_1}, \dots, Q_{s_M}$  and the reliabilities  $r(\mathbf{C}_1), \dots, r(\mathbf{C}_M)$ , where  $\mathbf{C}_1, \dots, \mathbf{C}_M$  are the clusters related to  $s_1, \dots, s_M$  symbols the agent  $a$  has found (see Section 2.1). The overall procedure for assigning  $Q_a$  can be break down in the following steps:

1. Select the agent  $a$  from  $S_i \in \Sigma$
2. According to the agent set of symbols  $\mathcal{B}$ , evaluate the mean qualities<sup>1</sup>  $\bar{Q}_{\mathcal{B}} = \frac{1}{|\mathcal{B}|} \sum_{s \in \mathcal{B}} Q_s$
3. According to the agent set of symbols  $\mathcal{B}$  evaluate the mean reliability  $\bar{r}_{\mathcal{B}} = \frac{1}{|\mathcal{B}|} \sum_{s \in \mathcal{B}} r_s$

<sup>1</sup> $Q_s$  is normalized in  $[0, 1]$  according to the symbols qualities observed in  $\mathcal{H}_i$ .



4. Set agents quality  $Q_a$

$$Q_a = \rho \cdot \bar{Q}_B + (1 - \rho) \cdot \bar{r}_B \quad (8)$$

5. Repeat from step 1 for all agents  $a \in \mathcal{S}_i$  with  $i = 1, \dots, S$ .

In Eq. (8),  $\rho$  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  $Q_a$ .

### 2.3.3 Evolutive Orchestration

The optimization of the agents in the swarms  $\Sigma$  is driven by a genetic evolution. As stated in Section 2.1, each swarm  $\mathcal{S}$  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  $a_{\text{code}}$  of each agent reads as follow:

$$a_{\text{code}} = [Q \quad \mathbf{w} \quad \boldsymbol{\gamma} \quad \tau] \quad (9)$$

which summarize the crucial parameters for the information extraction described in Section 2.1. Agents in the same swarm follow a classic  $(\mu + \lambda)$  selection scheme (Beyer and Schwefel, 2002), where  $\lambda$  offspring is generated according to common genetic operators applied to the parents population  $\mu$ , i.e. mutation, crossover and random spawn of new individuals. The optimization aims at maximizing the agent fitness function defined as the quality  $Q_a$  given in Eq. (8).

The classification models optimization is an evolutionary-like procedure specifically designed according to the unconventional nature of the individual's genetic code  $z_{\text{code}}$ . Indeed,  $z_{\text{code}}$  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  $[S, T]$ . The genetic code  $z_{\text{code}}$  reads as follow:

$$z_{\text{code}} = \mathcal{A} = [s_1, \dots, s_J] \quad (10)$$

where  $J = |\mathcal{A}|$ . The fitness function  $f_z$  reflects the critic obtained by the classifier  $c$  trained in the embedding space built according to  $\mathcal{A}$ , i.e. the accuracy  $\omega$  the classifier  $c$  attained on the validation set. Given the set of elite individuals  $Z^*$ , i.e. top- $Z$  individuals evaluated so far, the offspring  $\hat{Z}$  is generated according to custom operators defined as follows:

**Crossover:** two individuals  $z_i$  and  $z_j$  are selected uniformly at random from  $Z \cup Z^*$ . 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  $Z \cup Z^*$  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  $Z \cup Z^*$ . With a given probability  $\alpha_{\text{mut}}$ , each symbol  $s \in z_i$  has the chance to be swapped with a symbol belonging to a randomly extracted individual  $z_j \in Z \cup Z^*$ .

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

1. Run the agent swarms  $\Sigma$
2. Collect the agent symbols into  $\mathcal{H}_1, \dots, \mathcal{H}_S$  class-specific buckets
3. Generate  $Z$
4. Generate  $\hat{Z}$  by recombination of  $Z$  and  $Z^*$
5. Evaluate  $Z$  and  $\hat{Z}$
6. Reward the symbols in  $Z \cup \hat{Z}$  according to the classification performance
7. Evaluate the agents fitnesses for the swarms in  $\Sigma$
8. Evolve the agent swarms  $\Sigma$
9. Replace  $Z^*$  by selecting the top- $Z$  individuals amongst  $Z \cup \hat{Z} \cup Z^*$ .

The latter procedure highlights how the two different swarms  $\Sigma$  and  $Z$  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  $Z$  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  $Z$  with  $Z^*$  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,  $Z^*$  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.

## 2.4 Test Phase

After the evolution converges and/or reach the maximum number of generations  $N_{stop}$ , the final elite population  $Z^*$  can be exploited for evaluate the solutions obtained on a graph test set  $\mathcal{D}_{ts}$ . Recalling that each  $z_{code}^* \in Z^*$  is an alphabet of symbols  $\mathcal{A}$ , we can build the symbolic histogram matrix  $\mathbf{F}^{ts}$  of test set and train a classifier  $c$  according to  $\mathbf{F}^{tr}$ , where as usual  $\mathbf{F}^{tr}$  and  $\mathbf{F}^{ts}$  are respectively an  $|\mathcal{D}_{tr}| \times |\mathcal{A}|$  and an  $|\mathcal{D}_{ts}| \times |\mathcal{A}|$  matrix. By repeating the embedding procedure for all the solutions in  $Z^*$ ,  $c_1, \dots, c_Z$  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.

## 3 TEST AND RESULTS

### 3.1 Datasets and Algorithmic Setup

In order to test Graph E-ABC, we consider the following five open-access datasets from the IAM Repository (Riesen and Bunke, 2008): AIDS, GREC, Letter-L, Letter-M and Letter-H.

As the dissimilarities between nodes' and edges' attributes are concerned, all of them are customized according to the nodes and edges attributes for each dataset. GREC is the only dataset for which the dissimilarity measures between nodes and edges are parametric themselves: such values populate  $\boldsymbol{\gamma}$  which shall be optimized, as described in Section 2.3.3. Full details on the datasets, alongside formal definitions of their dissimilarity measures between nodes and edges can be found in (Martino and Rizzi, 2021) and (Baldini et al., 2021).

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

- The sets of sampled subgraphs  $\mathcal{D}_g^{tr}$  is built accordingly to a random walk extraction
- The BSAS resolution  $\theta$  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  $f_{co}$  and cardinality  $f_{ca}$

- The maximum number of generation  $N_{stop} = 20$ .

For the orchestration of the model evolution:

- $K = 10$ , the number of individuals in  $Z$
- $Z = 10$ , the number of best individuals in  $Z^*$
- $L = 20$ , the number of recombined individuals in  $\hat{Z}$
- $\alpha_{mut} = 0.15$ , the swapping probability in Section 2.2.

Other parameters include:

- $t_{min} = -10$  and  $t_{max} = 10$ , respectively the maximum penalty and the maximum reward values for updating the symbol quality  $Q_s$
- $\rho$  assumes equally spaced values in the range  $[0, 1]$  with step size  $\frac{1}{N_{stop}}$
- The number of bins for the lookup table in Section 2.3.1 equals the number of classes, so it changes in a dataset-dependent fashion
- $\zeta_{s_i} = 1.1 \cdot f_{co}(\mathbf{C})$ , where  $\mathbf{C}$  is the cluster who generated the symbol  $s_i$ .

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

### 3.2 Sensitivity Analysis

Amongst the parameters that characterize Graph E-ABC, to the best of our judgement, two of them are critical and need a careful tuning: the number of subgraphs that agents need to extract before running BSAS ( $W$ ) and the maximum number of class-related subgraphs to be included in the alphabet ( $T$ ). In order to address how these two parameters affect the performances of the overall system, we perform a sensitivity analysis via grid search. In particular, we chose a set of candidate values for both  $T$  and  $W$ , namely  $T = W = [10, 50, 100, 200]$ , and for each  $\langle T, W \rangle$ -pair, we run Graph E-ABC and record the average accuracy on the test set and the average number of symbols across 10 different runs, in order to account the stochastic nature of the algorithm.

In Figure 1 we show the sensitivity analysis of Graph E-ABC 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 and Letter-L, performances tend to deteriorate as  $T \rightarrow 0$ , whereas for harder classification problems such as Letter-M and Letter-M, performances tend to deteriorate for larger  $T$ . GREC is the only dataset for which performances seem to be rather stable, regardless of  $T$ . As  $W$  is concerned, Letter-M is the only dataset which shows an increasing trend in performances as  $W$  grows, whereas for the remaining four datasets a clear trend does not emerge.

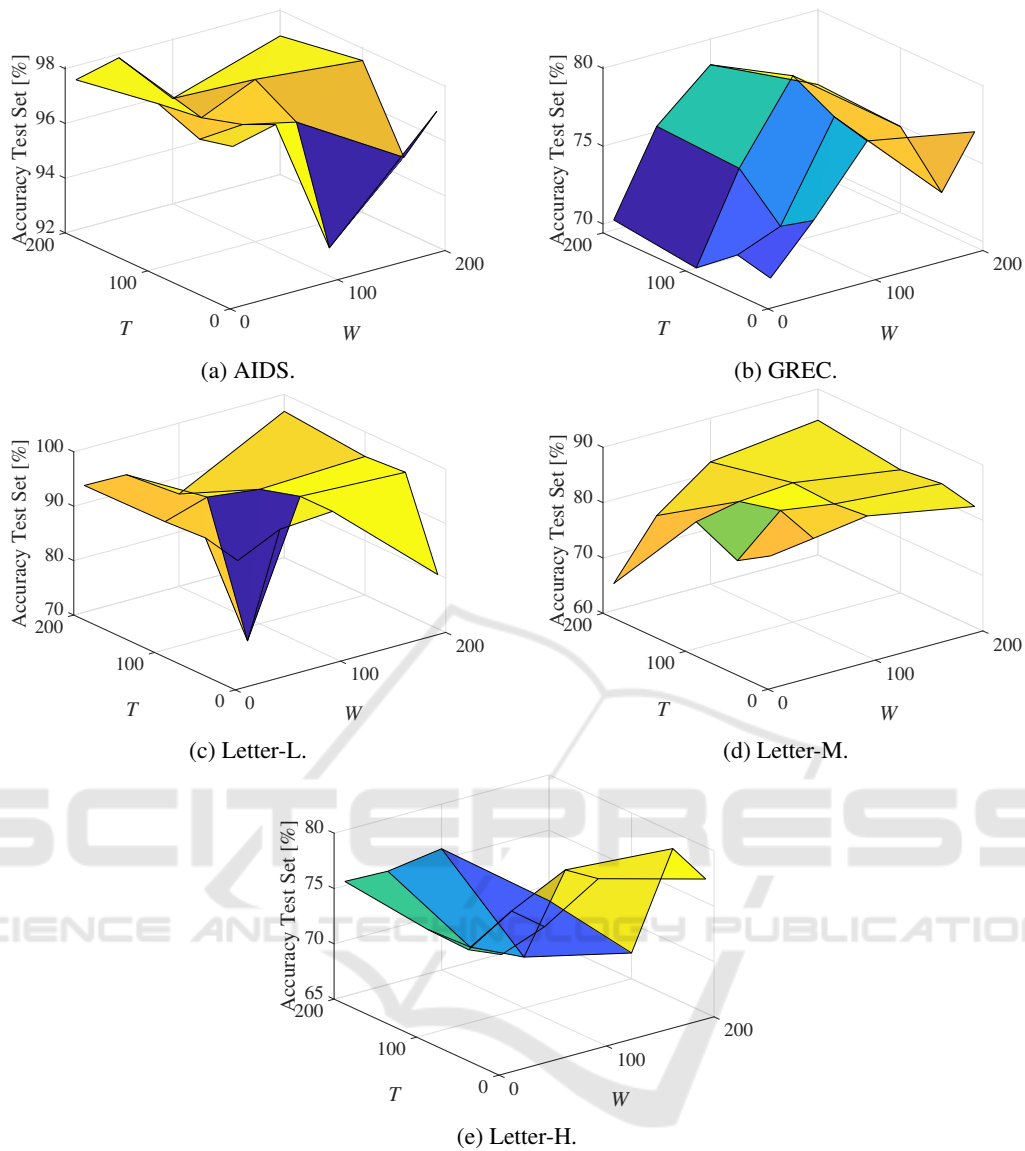


Figure 1: Graph E-ABC sensitivity analysis (average accuracy on the test set).

### 3.3 Comparison against Current Granular Approaches for Graph Classification

In order to compare the performances of Graph E-ABC, we select three suitable competitors that (like Graph E-ABC) exploit the steps on information granulation and graph embedding via symbolic histograms: the class-aware version of GRALG, proposed in (Baldini et al., 2021), and the RECTIFIER and Dual-RECTIFIER classifiers proposed in (Martino and Rizzi, 2021). The major difference between Graph E-ABC and the three 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 embedding procedure and trains the classifier in the embedding space. The three competitors 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. Amongst the three competitors, Dual-RECTIFIER is the only one that (like Graph E-ABC) exploit different optimization procedures in a class-aware fashion, yet it is worth remarking that in Dual-RECTIFIER such different optimization procedures are independent one another, whereas in Graph E-ABC there exist some sort of cooperation between different swarms.

In Table 1 we report 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 (details in (Baldini et al., 2021) and (Martino and Rizzi, 2021)). For Graph E-ABC, as instead, we report (for each dataset) the best point on the grids in Figure 1 (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 2.2,  $K$  different classifiers are in charge of classifying the data in  $K$  different embedding spaces. Hence, if we sum together the size of each of the  $K$  embedding spaces, this inevitably leads to a drastically higher number of symbols. However, if we take the average number of symbols that each model has to exploit (i.e., the sum of symbols divided by  $K$  models), we can see that the results are rather in line with the competitors for low subsampling rates (i.e., more than 50%).

## 4 CONCLUSIONS AND FUTURE DIRECTIONS

In this paper, we presented a promising multi-agent-based algorithm (Graph E-ABC) for graph classification. Graph E-ABC exploits a two-fold optimization routine in order to simultaneously improve the agents (hence their ability in searching for meaningful subgraphs) and the set of subgraphs that compose the classification model. Computational results on open

Table 1: 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
Graph E-ABC (best point)	98.07 (34)	80.09 (1284)	96.89 (2260)	86.27 (4407)	79.78 (2517)
Graph E-ABC (average)	96.95 (222)	74.56 (1748)	92.20 (1460)	81.79 (3879)	74.76 (4592)
RECTIFIER (5%) <sup>†</sup>	98.12 (6)	92.64 (87)	93.33 (23)	86.36 (52)	84.57 (90)
RECTIFIER (10%) <sup>†</sup>	98.41 (8)	92.91 (169)	94.58 (32)	86.01 (71)	85.62 (145)
RECTIFIER (30%) <sup>†</sup>	98.30 (9)	92.78 (415)	95.99 (79)	87.94 (173)	86.89 (300)
RECTIFIER (50%) <sup>†</sup>	98.40 (16)	92.65 (439)	95.57 (81)	89.37 (232)	86.31 (431)
RECTIFIER (80%) <sup>†</sup>	98.57 (16)	93.09 (741)	94.80 (157)	87.21 (294)	<b>88.92</b> (668)
Dual-RECTIFIER (5%) <sup>†</sup>	<b>99.11</b> (6)	94.37 (67)	94.77 (19)	85.74 (27)	82.59 (49)
Dual-RECTIFIER (10%) <sup>†</sup>	98.97 (9)	95.25 (77)	94.37 (22)	86.50 (42)	84.84 (86)
Dual-RECTIFIER (30%) <sup>†</sup>	98.95 (11)	<b>95.59</b> (173)	94.19 (29)	90.14 (77)	87.58 (162)
Dual-RECTIFIER (50%) <sup>†</sup>	98.81 (13)	95.22 (264)	95.11 (35)	89.88 (104)	86.67 (177)
Dual-RECTIFIER (80%) <sup>†</sup>	98.94 (37)	95.57 (215)	95.37 (50)	<b>90.31</b> (136)	86.83 (275)
Class-Aware GRALG (10%) <sup>‡</sup>	98.99 (9)	88.36 (288)	98.25 (146)	89.00 (214)	83.29 (287)
Class-Aware GRALG (30%) <sup>‡</sup>	99.11 (11)	86.96 (335)	98.28 (205)	88.77 (277)	83.05 (319)
Class-Aware GRALG (50%) <sup>‡</sup>	99.11 (12)	87.15 (322)	<b>98.37</b> (218)	89.40 (311)	83.47 (313)

<sup>†</sup> Averaged across different trade-off values in the objective function.

<sup>‡</sup> Class-Aware Frequency Scaling heuristic for driving the granulation procedure.



access datasets show interesting results in terms of accuracy for 3 out of 5 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 behaviour of Graph E-ABC. For example, we can investigate whether there exist a good trade-off when employing the ensemble, that is, how the performance change as a function of  $K$ . 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). Further, another potential drawback relates to how the quality of the symbols is evaluated. In fact (recall from Section 2.3.1), 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.

Certainly one of the most striking facets of Graph E-ABC is the highly stochastic nature of the agent fitness functions due to the observation of continuously different data shards sampled from the training set. For such reason, a repeated evaluation of the same individual can lead to a different fitness value, mostly due to the quality of the sampled data. A plain  $(\mu + \lambda)$  evolutionary scheme might not be suitable to efficiently deal with uncertain environments and future investigation will be directed to design specific evolutionary strategies able to take actions against noisy fitness functions (Bhattacharya et al., 2014; Branke et al., 2001; Merelo et al., 2016).

An additional improvement may regard the definition of the symbol quality (see Section 2.3.1). 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. (7)), but also the agent quality (see Eq. (8)). 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.

## REFERENCES

- Ahmad, R., Ali, S., and Kim, D. H. (2012). A multi-agent system for documents classification. In *2012 International Conference on Open Source Systems and Technologies*, pages 28–32.
- Alamgir, M. and Von Luxburg, U. (2010). Multi-agent random walks for local clustering on graphs. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 18–27. IEEE.
- Baldini, L., Martino, A., and Rizzi, A. (2019). Stochastic information granules extraction for graph embedding and classification. In *Proceedings of the 11th International Joint Conference on Computational Intelligence - NCTA, (IJCCI 2019)*, pages 391–402. INSTICC, SciTePress.
- Baldini, L., Martino, A., and Rizzi, A. (2020). Exploiting cliques for granular computing-based graph classification. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9.
- Baldini, L., Martino, A., and Rizzi, A. (2021). Towards a class-aware information granulation for graph embedding and classification. In Merelo, J. J., Garibaldi, J., Linares-Barranco, A., Warwick, K., and Madani, K., editors, *Computational Intelligence: 11th International Joint Conference, IJCCI 2019, Vienna, Austria, September 17–19, 2019, Revised Selected Papers*, pages 263–290. Springer International Publishing, Cham.
- Beyer, H.-G. and Schwefel, H.-P. (2002). Evolution strategies – a comprehensive introduction. *Natural Computing*, 1(1):3–52.
- Bhattacharya, M., Islam, R., and Mahmood, A. N. (2014). Uncertainty and evolutionary optimization: A novel approach. In *2014 9th IEEE Conference on Industrial Electronics and Applications*, pages 988–993.
- Bianchi, F. M., Maiorino, E., Livi, L., Rizzi, A., and Sadeghian, A. (2015). An agent-based algorithm exploiting multiple local dissimilarities for clusters mining and knowledge discovery. *Soft Computing*, 5(21):1347–1369.
- Bianchi, F. M., Rizzi, A., Sadeghian, A., and Moiso, C. (2016). Identifying user habits through data mining on call data records. *Engineering Applications of Artificial Intelligence*, 54:49–61.
- Branke, J., Schmidt, C., and Schmeck, H. (2001). Efficient fitness estimation in noisy environments. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation, GECCO’01*, page 243–250, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- Carvalho, L. F., Barbon, S., de Souza Mendes, L., and Proença, M. L. (2016). Unsupervised learning clustering and self-organized agents applied to help network management. *Expert Systems with Applications*, 54:29–47.
- Chaimontree, S., Atkinson, K., and Coenen, F. (2010). Clustering in a multi-agent data mining environment. *Agents and Data Mining Interaction*, pages 103–114.
- Chaimontree, S., Atkinson, K., and Coenen, F. (2011). A multi-agent based approach to clustering: Harnessing the power of agents. In *ADMI*, pages 16–29. Springer.
- Chen, L., Zhang, H., Xiao, J., He, X., Pu, S., and Chang, S.-F. (2019). Counterfactual critic multi-agent training for scene graph generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4613–4623.
- Corrêa, L., Arantes, L., Sens, P., Inostroza-Ponta, M., and Dorm, M. (2020). A dynamic evolutionary multi-agent system to predict the 3d structure of proteins. In *2020*

- IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE.
- Del Vescovo, G. and Rizzi, A. (2007a). Automatic classification of graphs by symbolic histograms. In *2007 IEEE International Conference on Granular Computing (GRC 2007)*, pages 410–410.
- Del Vescovo, G. and Rizzi, A. (2007b). Online handwriting recognition by the symbolic histograms approach. In *2007 IEEE International Conference on Granular Computing (GRC 2007)*, pages 686–686. IEEE.
- Di Noia, A., Martino, A., Montanari, P., and Rizzi, A. (2020). Supervised machine learning techniques and genetic optimization for occupational diseases risk prediction. *Soft Computing*, 24(6):4393–4406.
- Ding, W., Lin, C.-T., Chen, S., Zhang, X., and Hu, B. (2018). Multiagent-consensus-mapreduce-based attribute reduction using co-evolutionary quantum pso for big data applications. *Neurocomputing*, 272:136–153.
- Dorri, A., Kanhere, S. S., and Jurdak, R. (2018). Multi-agent systems: A survey. *Ieee Access*, 6:28573–28593.
- Giampieri, M., Baldini, L., De Santis, E., and Rizzi, A. (2020). Facing big data by an agent-based multimodal evolutionary approach to classification. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.
- Giampieri, M., De Santis, E., Rizzi, A., and Frattale Mascioli, F. M. (2018). A supervised classification system based on evolutive multi-agent clustering for smart grids faults prediction. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.
- González-Briones, A., Villarrubia, G., De Paz, J. F., and Corchado, J. M. (2018). A multi-agent system for the classification of gender and age from images. *Computer Vision and Image Understanding*, 172:98–106.
- İnkaya, T., Kayaligil, S., and Özdemirel, N. E. (2015). Ant colony optimization based clustering methodology. *Applied Soft Computing*, 28:301–311.
- Lombardo, G., Fornacciari, P., Mordonini, M., Tomaiuolo, M., and Poggi, A. (2019). A multi-agent architecture for data analysis. *Future Internet*, 11(2):49.
- Martino, A., Giampieri, M., Luzi, M., and Rizzi, A. (2019a). Data mining by evolving agents for clusters discovery and metric learning. In Esposito, A., Faundez-Zanuy, M., Morabito, F. C., and Pasero, E., editors, *Neural Advances in Processing Nonlinear Dynamic Signals*, pages 23–35. Springer International Publishing, Cham. Italian Workshop on Neural Nets 2017.
- Martino, A. and Rizzi, A. (2021). An enhanced filtering-based information granulation procedure for graph embedding and classification. *IEEE Access*, 9:15426–15440.
- Martino, A., Rizzi, A., and Frattale Mascioli, F. M. (2019b). Efficient approaches for solving the large-scale k-medoids problem: Towards structured data. In Sabourin, C., Merelo, J. J., Madani, K., and Warwick, K., editors, *Computational Intelligence: 9th International Joint Conference, IJCCI 2017 Funchal-Madeira, Portugal, November 1-3, 2017 Revised Selected Papers*, pages 199–219. Springer International Publishing, Cham.
- Merelo, J. J., Chelly, Z., Mora, A., Fernández-Ares, A., Esparcia-Alcázar, A. I., Cotta, C., de las Cuevas, P., and Rico, N. (2016). A statistical approach to dealing with noisy fitness in evolutionary algorithms. In Merelo, J. J., Rosa, A., Cadenas, J. M., Dourado, A., Madani, K., and Filipe, J., editors, *Computational Intelligence*, pages 79–95. Springer International Publishing, Cham.
- Modi, P. J. and Shen, W.-M. (2001). Collaborative multi-agent learning for classification tasks. In *Proceedings of the Fifth International Conference on Autonomous Agents*, AGENTS '01, page 37–38, New York, NY, USA. Association for Computing Machinery.
- Mostafa, J., Ke, W., and Fu, Y. (2005). Automated text classification using a multi-agent framework. In *Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL '05)*, pages 157–158.
- Ogston, E., Overeinder, B., Van Steen, M., and Brazier, F. (2003). A method for decentralized clustering in large multi-agent systems. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 789–796. ACM.
- Pan, X. and Chen, H. (2012). Multi-agent evolutionary clustering algorithm based on manifold distance. In *Computational Intelligence and Security (CIS), 2012 Eighth International Conference on*, pages 123–127. IEEE.
- Pan, X. and Jiao, L. (2011). A granular agent evolutionary algorithm for classification. *Applied Soft Computing*, 11(3):3093–3105.
- Panait, L. and Luke, S. (2005). Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434.
- Park, J. and Oh, K. (2006). Multi-agent systems for intelligent clustering. In *Proc. of World Academy of Science, Engineering and Technology*, volume 11, pages 97–102.
- Preuss, M. (2015). *Multimodal optimization by means of evolutionary algorithms*. Springer, 1 edition.
- Riesen, K. and Bunke, H. (2008). 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)*, pages 287–297. Springer.
- Rizk, Y., Awad, M., and Tunstel, E. W. (2018). Decision making in multiagent systems: A survey. *IEEE Transactions on Cognitive and Developmental Systems*, 10(3):514–529.
- Stone, P. and Veloso, M. (2000). Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383.
- Theodoridis, S. and Koutroumbas, K. (2008). *Pattern Recognition*. Academic Press, 4 edition.
- Wong, K. (2015). Evolutionary multimodal optimization: A short survey. *CoRR*, abs/1508.00457.