SN Computer Science (2022) 3:436
https://doi.org/10.1007/s42979-022-01260-4

ORIGINAL RESEARCH

A Multi-objective Optimization Approach for the Synthesis of Granular Computing-Based Classification Systems in the Graph Domain

Luca Baldini[1] · **Alessio Martino**[2,3] · **Antonello Rizzi**[1]

Received: 4 August 2021 / Accepted: 20 June 2022 / Published online: 9 August 2022
© The Author(s), under exclusive licence to Springer Nature Singapore Pte Ltd 2022

Abstract

The synthesis of a pattern recognition system usually aims at the optimization of a given performance index. However, in many real-world scenarios, there exist other desired facets to take into account. In this regard, multi-objective optimization acts as the main tool for the optimization of different (and possibly conflicting) objective functions in order to seek for potential trade-offs among them. In this paper, we propose a three-objective optimization problem for the synthesis of a granular computing-based pattern recognition system in the graph domain. The core pattern recognition engine searches for suitable information granules (i.e., recurrent and/or meaningful subgraphs from the training data) on the top of which the graph embedding procedure towards the Euclidean space is performed. In the latter, any classification system can be employed. The optimization problem aims at jointly optimizing the performance of the classifier, the number of information granules and the structural complexity of the classification model. Furthermore, we address the problem of selecting a suitable number of solutions from the resulting Pareto Fronts in order to compose an ensemble of classifiers to be tested on previously unseen data. To perform such selection, we employed a multi-criteria decision making routine by analyzing different case studies that differ on how much each objective function weights in the ranking process. Results on five open-access datasets of fully labeled graphs show that exploiting the ensemble is effective (especially when the structural complexity of the model plays a minor role in the decision making process) if compared against the baseline solution that solely aims at maximizing the performances.

Keywords Structural pattern recognition · Supervised learning · Embedding spaces · Granular computing · Graph edit distances · Graph embedding · Multi-objective optimization · Multi-criteria decision making

Introduction

Multi-objective optimization (MOO) is an area of mathematical optimization that regards problems with more than one objective function, to be optimized simultaneously [52]. Under a decision making viewpoint, MOO is particularly suited where optimal decisions must be taken by considering trade-offs among conflicting objective functions. Notable fields in which such scenario is common include engineering [5, 45, 86], economics and finance [13, 14, 44, 79], politics [32] and mechanics [19, 76]. 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*). The

This article is part of the topical collection "Computational Intelligence" guest edited by Kurosh Madani, Kevin Warwick, Juan Julian Merelo, Thomas Bäck and Anna Kononova.

✉ Alessio Martino
amartino@luiss.it

Luca Baldini
luca.baldini@uniroma1.it

Antonello Rizzi
antonello.rizzi@uniroma1.it

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

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

[3] Institute of Cognitive Sciences and Technologies, Italian National Research Council, Via San Martino della Battaglia 44, 00185 Rome, Italy

latter can be conveniently represented as an M-dimensional plot (with M being the number of objective functions), where each point corresponds to a Pareto optimal solution and each axis corresponds to a given objective function.

With the pervasive explosion of machine learning techniques in the last two decades, MOO has also been applied for the synthesis and tuning of pattern recognition systems. In principle, a fruitful synthesis of pattern recognition systems aims at looking for models and/or hyperparameters of the models that better generalize on previously unseen validation and/or test data. In turn, this suggests that a single-objective optimization would suffice as one can simply find the models and/or hyperparameters of the models that maximize a given performance index (e.g., accuracy for classification problems, coefficient of determination for regression problems or Silhouette score for clustering problems) on some validation data. However, depending on the pattern recognition problem under analysis, a thoughtful modeling must also take into account other factors, notably the type and number of features that describe the data at hand and, eventually, the structural complexity of the model. In fact, it is well known that a large number of features might lead to undesired phenomena such as the curse of dimensionality [82], longer training times and, for some problems, deteriorated model interpretability [50]. On the other hand, a model which is utterly complex for fitting the data at hand (e.g., a Support Vector Machine with too many support vectors, an Artificial Neural Networks with too many hidden layers and neurons, a Neuro-Fuzzy Min-Max classifier with too many hyperboxes) might also lead to undesired phenomena such as overfitting [84, 85] and also longer training times.

In the late 90s, pioneer Lotfi Aliasker Zadeh in his groundbreaking paper [90] proposed the concept of *information granules* as hierarchical and (likely) fuzzy mathematical objects in order to study and analyze a set of data. Information granulation is inspired by the ways in which humans granulate information and reason with it. The theory of fuzzy information granulation proposed by Zadeh did set the formal and mathematical bases for Granular Computing (GrC) [9, 60, 62]. GrC can be described as an information processing paradigm that revolves around the processing of complex information entities (information granules) that emerge (more or less automatically) from the data in order to analyze the data itself under different levels of abstraction. In general, the soundness of information granules as finite collection of data is determined by concepts such as functional proximity, similarity, indistinguishability and coherency [61, 90]. The GrC paradigm served as the core engine for the synthesis of several pattern recognition systems, also tailored for dealing with structured domains such as graphs [6–8, 11, 23, 46, 48, 50], images [10, 68], sequences [43, 70, 71] and simplicial complexes [47]. The vast majority of the aforementioned GrC-based pattern recognition systems exploit a granulation procedure that let emerge in a data driven fashion a set of suitable information granules. Such information granules are the pivotal entities for performing the embedding procedure that allows to move the pattern recognition problem from the structured domain towards the Euclidean space, in which any classification, clustering or function approximation model can be employed without alterations.

In this paper, we address the problem of synthesizing a GrC-based pattern recognition system in the graph domain by exploiting a MOO stage in order to simultaneously minimize the miclassification error, the number of features (i.e., information granules) and the structural complexity of the classification model validated in the embedding space. The pattern recognition system at the core of this paper, GRALG, has been originally proposed in [11] and later improved in [7]. 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 always been addressed via single-objective evolutionary optimization (i.e., maximization of the accuracy of the classifier in the embedding space).

The remainder of this paper is structured as follows: in the second section, we briefly review common approaches for the synthesis of pattern recognition systems via MOO and discuss in detail the contribution of this paper; in the third section, we describe the proposed pattern recognition system (GRALG), along with the novel MOO stage. In the fourth section, we show the computational results and the fifth section concludes the paper. This paper also features an Appendix A in which we show the Pareto Fronts for all case studies in the fourth section.

Current Works on Multi-objective Optimization for Pattern Recognition and Paper Contribution

Optimization stages based on (possibly evolutive) MOO have been widely proposed in the machine learning literature. To the best of our knowledge, current works usually aim at jointly optimizing (regardless of the number of objective functions) either performances-vs-feature selection or performances-vs-structural complexity of the model, with the only exception being our previous work [50] which, as in this work, aims at jointly optimizing performances-vs-number of selected features-vs-structural complexity of the model. Here below, we give some examples of current works in MOO for the synthesis of pattern recognition systems, referring the interested reader to reviews such as [3, 34, 38, 65] for a more exhaustive overview on the subject matter.

In [75], the authors employ the NSGA-II algorithm [20] for training a MultiLayer Perceptron while jointly

minimizing the error rate and the regularization penalty in order to encourage sparse connections among neurons. Conversely, in [1], the authors test three different 2-combinations of four objective functions (error rate, size of the hidden layer, number of connections, number of features) for analyzing the well-known Breast Cancer Dataset from the UCI Machine Learning Repository [26]. Further, in [31], the authors search for the topology of a neural network in order to jointly minimize the false positive rate and maximize the true positive rate.

In [16], the authors study whether it is possible to limit the computational complexity of a Nearest Neighbor classifier[1] [18] by means of MOO in order to simultaneously optimize subset of features and subsets of training data (both to be minimized) and accuracy of the Nearest Neighbor classifier (to be maximized).

In [81], Support Vector Machines (SVMs) [17] play the role of main actors and the authors aim at minimizing the false positive rate, false negative rate and the number of support vectors. The authors also face the problem of selecting features during optimization, yet the ratio of selected features does not take part in any of the objective functions. Conversely, in [15], the authors jointly minimize false acceptance and false rejection rates for SVM classifiers, without considering feature selection and/or complexity of the trained model.

In [58], the authors propose a multi-objective genetic algorithm [30] for a two-step feature selection and building an ensemble of classifiers. A first optimization process trains candidate classifiers for the ensemble while performing feature selection with the aim of maximizing the performance while minimizing the number of features. The second optimization process, as instead, selects the candidate classifiers for building the ensemble with the aim of maximizing the accuracy of the ensemble and the diversity of the classifiers composing the ensemble itself. A similar approach has been employed in [66], where the first optimization process trains candidate classifiers while performing feature selection in order to optimize several performance indices (accuracy, true positive rate, true negative rate, F-score) along with the classifier complexity. The second optimization process is in charge of selecting suitable subset of classifiers in such a way that the four aforementioned performance indices, along with the global complexity of the ensemble, are optimized simultaneously. The problem of ensemble learning via MOO has also been addressed in [4] and [72], where the authors put major emphasis on solving class-imbalanced classification problems.

The authors in [77], as instead, propose a multi-objective framework for multi-label classification.

Under an application-oriented viewpoint, recent works that joined machine learning and MOO regarded materials/mineral engineering and chemistry [51, 57, 64, 88, 91], energy management and engineering [29, 37, 39, 42, 78, 89] and life sciences [2, 53–55, 80].

This work stems from two previous works, namely [50] and [6]. In [50], as anticipated, we formulate the same three-objective optimization problem at the basis of this work. Yet, in our previous work, the analysis only regard how the three objectives correlate each other on the Pareto front and the testing phase (i.e., how to properly choose one of the non-dominated solutions in order to assess the generalization capabilities of the classifier) has not been addressed. Further differences include the study of binary classification problems only related to bioinformatics-oriented applications (whereas in this work, we use freely available benchmark datasets for graph classification pertaining to several application fields) and in this work we also tackle the problem of dealing with fully labeled graphs by exploiting the GRALG framework proposed in [7], whereas in [50] we analyzed graphs with categorical node labels only by exploiting other embedding strategies tailored to work with such kind of graphs [47, 48].

In [6] we exploit GRALG and we test several supervised classifiers in the embedding space (namely, K-Nearest Neighbors, SVMs [74] and Neuro-Fuzzy Min-Max Classifiers [69]). Comparison among classifiers regards their performance on the test set and their structural complexity. However, a striking difference with respect to this work lies on the single-objective optimization and therefore the structural complexity emerges as a consequence of the maximization of the accuracy on some validation data and does not play any role in the optimization procedure.

Conversely, in this work, we propose the following three main novelties with respect to our previous studies:

- we employ a MOO procedure for the joint optimization of three conflicting objective functions (performance, structural complexity and dimensionality of the embedding space) within the GRALG framework;
- we address the issue of selecting a suitable subset of solutions from the Pareto front for the synthesis of the 'final' classification system;
- we investigate the usage of the top-K solutions (instead of 'the best one') in order to build an ensemble of classifiers.

[1] Notably, the Nearest Neighbor classifier scales badly with the number of data and with the number of features since, in order to classify a new test pattern, all pairwise distances must be evaluated with respect to the training data.

Proposed Classification System Equipped with Multi-objective Optimization

Building Blocks

In this section, we describe the functional blocks needed for building the proposed graph classification system for labeled graphs. Formally, a labeled graph G is defined as $G = \{\mathcal{V}, \mathcal{E}, \mathcal{L}_v, \mathcal{L}_e\}$, where $\mathcal{V}$ is the nodes set, $\mathcal{E}$ the edges set and $\mathcal{L}_v$, $\mathcal{L}_e$ are the sets of nodes' and edges' attributes, respectively. From our point of view, $\mathcal{L}_v$, $\mathcal{L}_e$ can be any data structure that describe semantically the role of nodes and edges in the network. Additionally, the system does not make any prior assumption about whether graphs are directed or not. The synthesis of the classification system relies on two disjoint sets $\mathcal{S}^{(\text{tr})}$ and $\mathcal{S}^{(\text{vs})}$, respectively, training and validation set. In brief, GRALG performs the graph embedding procedure according to the GrC paradigm, allowing the possibility to use a common SVM classifier in a comfortable geometric space $\mathcal{X} \subseteq \mathbb{R}^n$ equipped with the usual Euclidean distance and dot product between vectors.

Extractor

Starting from the training set $\mathcal{S}^{(\text{tr})}$, this block extracts a set of subgraphs $\mathcal{S}_g^{(\text{tr})}$. In the seminal work [11], the rationale behind this block was to break down each graph $G \in \mathcal{S}^{(\text{tr})}$ exhaustively in its constituent parts up until a user-defined order V, that is the maximum number of nodes a subgraph can have. These entities can later be used as candidate granules of information for the next phases. On the other hand, this approach strongly limits the chance to use GRALG only for low/medium-size graphs and small order V, due to the high cardinality of the subgraph set $\mathcal{S}_g^{(\text{tr})}$ that negatively impacts the memory footprint needed to store the data alongside the computational burden needed to process it. In [7], we proposed a stochastic procedure able to build a subgraph set with a fixed cardinality W in order to avoid an exhaustive enumeration of all subgraphs in $\mathcal{S}^{(\text{tr})}$ and relax the computational cost both in terms of time and space. Indeed, a single subgraphs g is extracted uniformly at random (replacements are allowed) from a set of graphs by exploiting either one of two well-known traversal strategies, namely Breadth First Search (BFS) and Depth First Search (DFS). The stochastic procedure can be summarized as follows:

1. A graph G is extracted uniformly at random from $\mathcal{S}^{(\text{tr})}$;
2. The selected graph G is traversed using either BFS or DFS until a fixed number of nodes V are visited, with the seed node randomly selected among the nodes in G;
3. Visited nodes and edges are stored in the resulting subgraph g and collected into $\mathcal{S}_g^{(\text{tr})}$;
4. The procedure goes back to 1. until the cardinality of $\mathcal{S}_g^{(\text{tr})}$ has reached W.

The results witnessed comparable performances in terms of accuracy with a reduced running times and memory footprint when using the stochastic extractor in lieu of the exhaustive extractor from the original GRALG classification system.

Granulator for Alphabet Synthesis

The key component that enables the graph embedding by means of GrC is the granulation procedure described in this section. The granulation process aims at synthesizing a set of highly informative entities $\mathcal{A} = \{s_1, \ldots, s_n\}$ according to the principle of justifiable granularity [61]. Indeed, each symbol $s_i \in \mathcal{A}$ is defined as a granule of information related to a specific level of abstraction at which the problem has been observed [61]. Although an information granule can be formalized in several different ways, for example as fuzzy sets, rough sets and shadowed sets [63], we follow a clustering-based approach [87] based on the Basic Sequential Algorithmic Scheme (BSAS) [83] algorithm. The procedure takes place directly on the set of subgraphs $\mathcal{S}_g^{(\text{tr})}$ and relies on four fundamental aspects that must be carefully defined: a threshold of inclusion θ, a cluster representative g^*, a dissimilarity measure $d : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ and a threshold Q of maximum allowed number of clusters. By varying θ, the clusters resolution will change accordingly, impacting on the level of granularity. Thus, we generate a set of partitions $\mathcal{P} = \{\mathcal{P}_{\theta_1}, \ldots, \mathcal{P}_{\theta_P}\}$, where partition $\mathcal{P}_{\theta_i}$ collects each cluster $\mathbf{C}$ emerged by imposing θ_i as the resolution value.

A cluster quality index $F(\mathbf{C}) \in [0, 1]$ (the lower is better) is then defined in order to promote only promising clusters to information granules:

$$F(\mathbf{C}) = \rho \cdot \Psi(\mathbf{C}) + (1 - \rho) \cdot \Omega(\mathbf{C}) \tag{1}$$

where $\Psi(\mathbf{C})$ and $\Omega(\mathbf{C})$ are, respectively, the compactness and cardinality for cluster $\mathbf{C}$, weighted by a trade-off parameter $\rho \in [0, 1]$:

$$\Psi(\mathbf{C}) = \frac{1}{|\mathbf{C} - 1|} \sum_{g \in \mathbf{C}} d(g^*, g) \tag{2}$$

$$\Omega(\mathbf{C}) = 1 - \frac{|\mathbf{C}|}{|\mathcal{S}_g^{(\text{tr})}|} \tag{3}$$

Being the graph domain devoid of a geometrical interpretation, some solutions for defining a cluster representative are not available if compared to Euclidean spaces (e.g., the mean

point in k-means algorithm). A valid alternative is to define g^* as the MinSoD [24, 49] element of $\mathbf{C}$, that is the graph that minimizes the pairwise sum of distances among all patterns in the cluster. Clearly, the dissimilarity measure d must be tailored to the input space under analysis (i.e., the graph domain): our choice felt on a weighted graph edit distance heuristic named node-Best Match First (nBMF)[2]. Finally, each cluster proves its validity by comparing its own quality $F(\mathbf{C})$ with a threshold $\tau \in [0, 1]$. The MinSoDs of the surviving clusters are then collected together in the alphabet $\mathcal{A}$.

GrC-Based Embedder

The block described in this section implements the embedding function $\Phi : \mathcal{G} \to \mathcal{X}$ that maps patterns from the graph domain toward a geometric Euclidean space. The output of the granulation stage described in "Granulator for Alphabet Synthesis" is exploited in the graph embedding stage, which relies on the symbolic histogram approach [23]. Indeed, this method relies on two fundamental sets: a collection of granules, i.e., the alphabet $\mathcal{A} = \{s_1, \ldots, s_n\}$ and G_{exp}, that is the set of subgraphs that compose a given graph G to be embedded. In other words, $G_{\text{exp}} = \{g_1, \ldots, g_m\}$ is a (possibly non-exhaustive) decomposition of the graph G in m atomic units that can be obtained with the same traversal strategy used in "Extractor".

The symbolic histogram $\mathbf{h}_G$ is then defined as follows:

$$\Phi^{\mathcal{A}} = \mathbf{h}_G = \left[\text{occ}\left(s_1, G_{\text{exp}}\right), \ldots, \text{occ}\left(s_n, G_{\text{exp}}\right)\right] \tag{4}$$

where $n = |\mathcal{A}|$. Intuitively, each component of the vector $\mathbf{h}_G$ representing G is obtained by counting the occurrences of all symbols $s_i \in \mathcal{A}$ in the set G_{exp}. The function $\text{occ} : \mathcal{A} \times \mathcal{G} \to \mathbb{N}$ is defined as follows:

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

In Eq. (5), Γ is a function that defines whether a symbol s_i can be matched with a subgraph $g \in G_{\text{exp}}$. Exact match between two graphs (also known as graph or subgraph isomorphism) is often an unpractical operation for its hardness from a computational complexity point of view, especially when nodes and edges are equipped with real valued vectors or even with custom data structures [28]. In order to account for inexactness in the matching procedure, Γ evaluates the degree of dissimilarity $d(s_i, g)$ between s_i and g, then it triggers the counter (i.e., a match is considered a hit) only if this value does not exceed a symbol-dependent threshold ζ_{s_i}:

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

It is worth remarking that the dissimilarity measure in Eq. (6) coincides with the nBMF distance employed in the clustering algorithm defined in "Granulator for Alphabet Synthesis".

Classification

The last block in GRALG is a classifier $\mathcal{T}$ working directly in the embedding space $\mathcal{X} \subseteq \mathbb{R}^n$ spanned by the symbolic histograms as defined in "GrC-based Embedder", being n the cardinality of the alphabet set $\mathcal{A}$. First of all, $\mathcal{S}^{(\text{tr})}$ and $\mathcal{S}^{(vs)}$ are embedded according to the symbolic histogram approach, respectively, into $\mathbf{H}^{(\text{tr})} \in \mathbb{R}^{|\mathcal{S}^{(\text{tr})}| \times n}$ and $\mathbf{H}^{(vs)} \in \mathbb{R}^{|\mathcal{S}^{(vs)}| \times n}$. In this way, the embedding space and thus the alphabet set that generated it, can be evaluated by means of a performance measure $\Pi : \mathcal{S} \to \mathbb{R}$ evaluated on the validation set $\mathbf{H}^{(vs)}$ obtained by the classifier $\mathcal{T}$ trained on $\mathbf{H}^{(\text{tr})}$. In [6], we extensively studied the behavior of five different classifiers from the point of view of Π, structural complexity of $\mathcal{T}$ and the sparsity of the embedding space by means of cardinality of $\mathcal{A}$. Since in this work we perform a joint MOO on the three aforementioned quantities, for sake of convenience we limit our analysis to SVMs and, specifically, to ν-SVMs [74], equipped with two different kernels: the linear kernel endowing the dot product between patterns $\mathcal{K}(x, x') = \langle x, x' \rangle$ and a non-linear radial basis function (RBF) kernel, i.e., $\mathcal{K}(x, x') = \exp\{-\gamma \|x - x'\|^2\}$.

Multi-objective Optimization Phase

In previous works [6, 7], we performed a sequential two-steps (single-objective) optimization phases:

1. Alphabet optimization: it aims at synthesizing an highly informative alphabet set $\mathcal{A}$ and concurrently optimize the classifier $\mathcal{T}$ hyperparameters, together with the parameters defining the adopted dissimilarity measure. Being $\mathbf{H}^{(\text{tr})}$ and $\mathbf{H}^{(vs)}$ the vector representations of $\mathcal{S}^{(\text{tr})}$ and $\mathcal{S}^{(vs)}$ obtained with $\mathcal{A}$, this optimization step produces the optimized alphabet $\overline{\mathcal{A}}$ and classifier $\overline{\mathcal{T}}$ by maximizing the performance Π obtained by $\mathcal{T}$ when classifying $\mathbf{H}^{(vs)}$;
2. Feature selection: it generates an optimal alphabet $\mathcal{A}^*$ and classifier $\mathcal{T}^*$ by removing redundant and unnecessary symbols from $\overline{\mathcal{A}}$ according to the classification performance of $\mathcal{T}^*$ whose critical parameters are simultaneously optimized.

This two-step approach has the purpose to eventually converge to a solution which is trying to synthesize the best

[2] Full details on nBMF, along with detailed pseudocodes, can be found in [7] and [46, Appendix A].

alphabet by maximizing the feedback received by $\mathcal{T}$ (see step 1.) and later reducing the dimensionality of the embedding space (see step 2.). Indeed, reducing the dimensionality of the final embedding space while maintaining affordable performances is often useful since it helps in mitigating the curse of dimensionality, the longer training/test times and the interpretability of the model. On the other hand, another critical aspect that deserves attention when designing a machine learning system is the complexity of the classifier that for a SVM can be effectively expressed by the number of support vectors emerged from the training stage[3]. Indeed, simpler models do not tend to overfit the data at hand, offering (in principle) better generalization capabilities.

After these considerations, two natural questions arise: can these three quantities be optimized simultaneously? How do these three quantities compete against each other? In order to rigorously answer these questions, we formalize a MOO problem based on the NSGA-II algorithm. The NSGA-II algorithm can be briefly outlined as follows and we refer the interested reader to [21] for a more exhaustive description:

1. population initialization: the population is initialized based on the search space range and constraints (if any);
2. evaluation: the fitness of each individual is evaluated by means of the objective functions;
3. non-dominated sorting: the population is sorted based on non-domination[4] and each individual is assigned to one of the fronts;
4. crowding distance: within each front, the crowding distance between its individuals is evaluated. The crowding distance is a measure of how close an individual is to its neighbors: large average crowding distance will result in better diversity in the population;
5. genetic operators: by jointly considering the rank (i.e., front) and the crowding distance value, parents are selected and then used to generate new offsprings. After being evaluated, the top individuals among offsprings and parents are retained;
6. repeat from step #3 until a maximum number of iterations is reached or a stopping criterion is satisfied.

For the sake of description, we give the critical parameters to be optimized 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 (deletions, insertions and substitutions of nodes and edges) for the dissimilarity measure defined in the graph domain: $\mathbf{w} = \{w^{\text{sub}}_{\text{node}}, w^{\text{sub}}_{\text{edge}}, w^{\text{ins}}_{\text{node}}, w^{\text{ins}}_{\text{edge}}, w^{\text{del}}_{\text{node}}, w^{\text{del}}_{\text{edge}}\} \in [0,1]^6$. Depending on the dataset, vertices and/or edges can require a parametric dissimilarity measure $d_v^{\pi_v} : \mathcal{L}_v \times \mathcal{L}_v \to \mathbb{R}$ and $d_e^{\pi_e} : \mathcal{L}_e \times \mathcal{L}_e \to \mathbb{R}$ with parameters $\mathbf{p} = \{\pi_v, \pi_e\}$ whose values are optimized as well;

Granulation $Q \in [1, 500]$ the maximum number of admissible cluster for the BSAS procedure; τ the quality threshold for promoting a MinSoD to a symbol; ρ trade-off parameter that weights compactness and cardinality in Eq. (1);

Classifier the set of hyperparameters $\mathbf{c}$ for the ν-SVM that depends on the considered kernel:

- Linear: only the regularization term $\nu \in (0, 1]$;
- RBF: along with $\nu \in (0, 1]$, we also tune the kernel shape $\gamma \in (0, 100]$.

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

$$\mathbf{x} = \begin{bmatrix} Q & \tau & \rho & \mathbf{w} & \mathbf{p} & \mathbf{c} \end{bmatrix} \tag{7}$$

The set of parameters collected in $\mathbf{x}$ drives the pipeline of granulation, embedding and classification described in "Building Blocks". Specifically, each individual from the NSGA-II population:

1. Runs the granulation procedure on the set of subgraphs $\mathcal{S}_g^{(\text{tr})}$ extracted by following the stochastic variant described in "Extractor". The dissimilarity measure parameters for the clustering algorithm are determined by $\mathbf{w}$ and $\mathbf{p}$. The granulator parameters (i.e., Q, τ and ρ) are employed as well for building the alphabet $\mathcal{A}$;
2. Both $\mathcal{S}^{(\text{tr})}$ and $\mathcal{S}^{(vs)}$ can now be embedded in $\mathbf{H}^{(\text{tr})}$ and $\mathbf{H}^{(vs)}$ 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. (6);
3. The set of classifier hyperparameters $\mathbf{c}$ are exploited for training $\mathcal{T}$ on $\mathbf{H}^{(\text{tr})}$. Finally, the performance $\Pi\left(\mathbf{H}^{(vs)}\right)$ is returned.

[3] Full mathematical details can be found in [50].

[4] An individual is said to *dominate* another individual if its objective functions are no worse than the other and at least one of its objective functions is better than the other.

The three-objective functions are formalized as follows:

$$f_1 = \Pi(\mathbf{H}^{vs}) \tag{8}$$

$$f_2 = \frac{SV}{|\mathcal{S}^{(\mathrm{tr})}|} \tag{9}$$

$$f_3 = \frac{|\mathcal{A}|}{|\mathcal{S}_g^{(\mathrm{tr})}|} \tag{10}$$

The first objective function $f_1 \in [0, 1]$ in Eq. (8) is defined as a negative-oriented performance index, namely the error rate of the classifier $\mathcal{T}$ in predicting $\mathbf{H}^{(vs)}$. The second objective function in Eq. (9) 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 $|\mathcal{S}^{(\mathrm{tr})}|$ 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. (10) 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 $|\mathcal{S}_g^{(\mathrm{tr})}|$ is the cardinality of the largest alphabet derivable from $\mathcal{S}_g^{(\mathrm{tr})}$ (i.e., all candidate information granules are promoted as information granules). In light of these three definitions, the optimization problem will aim at performing a joint minimization of f_1, f_2 and f_3.

Selection of Solutions from the Pareto Front

At the end of the optimization phase, NSGA-II returns a set of N non-dominated solutions $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$. More in details, in a multi-objective minimization problem a solution $\mathbf{x}_i$ is said to be dominated by $\mathbf{x}_j$ if the following hold [27]:

$$\begin{aligned} &f_q(\mathbf{x}_j) \leq f_q(\mathbf{x}_i) \quad \forall q \in \{1, \ldots, M\} \\ &\exists q \in \{1, \ldots, M\} : f_q(\mathbf{x}_j) < f_q(\mathbf{x}_i) \end{aligned} \tag{11}$$

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 cannot be known in closed-form in the context of machine learning. As introduced in "Introduction", 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. (11), a solution $\mathbf{x}_i \in X$ cannot be preferred to any other $\mathbf{x}_j \in X$ without further indications [40]. 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) [36] which has shown good reliability, low computational cost and an intuitive logical reasoning [73, 86]. Starting from the decision matrix $\mathbf{D} \in \mathbb{R}^{N \times M}$, i.e., the matrix whose i^{th} row is the vector $[f_1(\mathbf{x}_i), \ldots, f_M(\mathbf{x}_i)]$, 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 $\mathbf{u} \in \mathbb{R}^M$.

Ensemble of Classifiers for Test Phase

In light of the discussion in "Selection of Solutions from the Pareto Front", we retain from the optimization stage the top K solutions $X^* = \{\mathbf{x}_1^*, \ldots, \mathbf{x}_K^*\}$ ranked by TOPSIS according to the weight vector $\mathbf{u}$. Alongside, we hold the corresponding alphabets $\mathcal{A}^* = \{\mathcal{A}_1^*, \ldots, \mathcal{A}_K^*\}$ and the training embedded matrices $\mathcal{H}^* = \{\mathbf{H}_{\mathrm{tr},1}^*, \ldots, \mathbf{H}_{\mathrm{tr},K}^*\}$ obtained by embedding $\mathcal{S}^{(\mathrm{tr})}$ thanks to the alphabets in $\mathcal{A}^*$. In order to exploit simultaneously the information carried by the K solutions, we set up an ensemble of classifiers $\mathcal{C}$ to address the final performance on the test set $\mathcal{S}^{(\mathrm{ts})}$. The design of $\mathcal{C}$ goes as follows:

1. For each alphabet $\mathcal{A}_i^* \in \mathcal{A}^*$, we build the vectorial representation $\mathbf{H}_{\mathrm{ts},i}$ of $\mathcal{S}^{(\mathrm{ts})}$ according to the symbolic histogram paradigm as discussed in "GrC-based Embedder";
2. From each solution $\mathbf{x}_i^* \in X$, we retrieve the hyperparameters $\mathbf{c}_i^*$ of the optimized classifier;
3. The classifier $\mathcal{T}_i$ is trained with $\mathbf{H}_{\mathrm{tr},i}^* \in \mathcal{H}^*$ with hyperparameters $\mathbf{c}_i^*$;
4. All the trained classifiers $\mathcal{T}_i$ with $i = 1, \ldots, K$ are collected in the stack $\mathcal{C}$.

It is worth remarking that each classifier $\mathcal{T}_i$ operates in a specific embedding space $\mathcal{X}_i$ (to which $\mathbf{H}_{\mathrm{tr},i}$ and $\mathbf{H}_{\mathrm{ts},i}$ belong) generated by the alphabet $\mathcal{A}_i^*$ emerged from the solution $\mathbf{x}_i^*$.

This approach allows to establish a common Winner-Takes-All policy when predicting a single pattern from the test set. Let $g_k \in \mathcal{S}^{(\mathrm{ts})}$ be the kth graph in the test set; the steps that lead to the prediction of its class label by the ensemble $\mathcal{C}$ follows:

1. let $\{\mathbf{h}_k^1, \ldots, \mathbf{h}_k^K\}$ be the vectorial representations of g_k in K different embedding spaces $\mathcal{X}_1, \ldots, \mathcal{X}_K$, where in general $\mathcal{X}_i \neq \mathcal{X}_j$ for all $i \neq j$. This step corresponds in gathering the k^{th} rows of $\{\mathbf{H}_{\mathrm{ts},1}^*, \ldots, \mathbf{H}_{\mathrm{ts},K}^*\}$;

2. each classifier $\mathcal{T}_i \in \mathcal{C}$ individually predicts the label l_k^i for $\mathbf{h}_k^i$ (for $i = 1, \ldots, K$);
3. all predicted labels are collected in the output stack $L = \left(l_k^1, \ldots, l_k^K\right)$;
4. the most frequent label in L 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. We approach this problem by defining a confidence measure $s_i\left(l_t\right)$ of classifier $\mathcal{T}_i$ in predicting a specific label l_t, where $t = 1, \ldots, T$, being T the number of classes. The confidence $s_i\left(l_t\right)$ for class l_t is chosen as the (multi-class) precision score obtained in classifying the training set $\mathbf{H}_{\mathrm{tr},i}$.

Let R be the subset of the T 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.

Tests and Results

Datasets Description and Algorithmic Setup

In order to test the proposed algorithm and the different embedding strategies, we have considered five datasets from the IAM Repository [67] and two additional datasets from the TUDataset Repository [56]. A brief description of the datasets follows:

AIDS: The AIDS dataset is composed by graphs representing molecular compounds showing activity or not against HIV. Molecules are converted into graphs by representing atoms as nodes and the covalent bonds as edges. Nodes are labeled with the number of the corresponding chemical symbol and edges by the valence of the linkage.

GREC: The GREC dataset consists of graphs representing symbols from architectural and electronic drawings. Drawings have been pre-processed and ending points, circles, corners and intersections are considered as nodes labeled with their position and type. Edges connecting such nodes are equipped with a hierarchical data structure that identifies the type of connection and its characteristics.

Letter: The Letter datasets involve graphs that represent distorted letter drawings. Among the capital letters of the Roman alphabet, 15 have been selected due to them being representable by straight lines only. Drawings are then converted into graphs by considering lines as edges and ending points as nodes. Nodes are labeled with a 2-dimensional vector indicating their position, whereas edges are unlabeled. Three different amount of distortion (Low, Medium, High) account for three different datasets, hereinafter referred to as Letter-L, Letter-M and Letter-H, respectively.

MUTAG: The MUTAG dataset consists of 188 graphs corresponding to chemical compounds divided into 2 classes according to their respective mutagenic effect on a bacterium. Both nodes and edges are equipped with categorical labels: node labels identify the atom type and edge labels identify the bond type (single, double, triple or aromatic).

PTC: The PTC dataset contains compounds labeled according to carcinogenicity on male mice (MM) rodents. Each compound is converted into a graph where vertices represent atoms and edges represent chemical bonds. Explicit hydrogen atoms have been removed and vertices are labeled by atom type (categorical value) and edges by bond type (categorical value).

As the nodes and edges dissimilarities are concerned, all of them are customized according to the nodes and edges attributes for each dataset[5]. GREC is the only dataset for which the dissimilarity measures between nodes and edges are parametric themselves: such values populate $\mathbf{p}$ which shall be optimized, as described in "Multi-Objective Optimization Phase". Brief statistics for each dataset are shown in Table 1, along with the reference paper in which each dataset has been originally presented, to which we refer the interested reader for more exhaustive information.

The algorithm parameters have been set as follows:

[5] Full details on the dissimilarity measures for the IAM datasets can be found in [46, Appendix B].

Table 1 Dataset statistics and description

Dataset	# graphs	Avg. # nodes	Avg. # edges	# classes	Balanced	Domain	Repository	References
AIDS	2000	15.69	16.20	2	No	Chemoinformatics	IAM	[67]
Letter-L	2250	4.7	3.1	15	Yes	Computer vision	IAM	[67]
Letter-M	2250	4.7	3.2	15	Yes	Computer vision	IAM	[67]
Letter-H	2250	4.7	4.5	15	Yes	Computer vision	IAM	[67]
GREC	1100	11.5	12.2	22	Yes	Electronics	IAM	[25, 67]
MUTAG	188	17.93	19.79	2	No	Chemoinformatics	TUD	[22, 41]
PTC (MM)	336	13.97	14.32	2	No	Chemoinformatics	TUD	[35, 41]

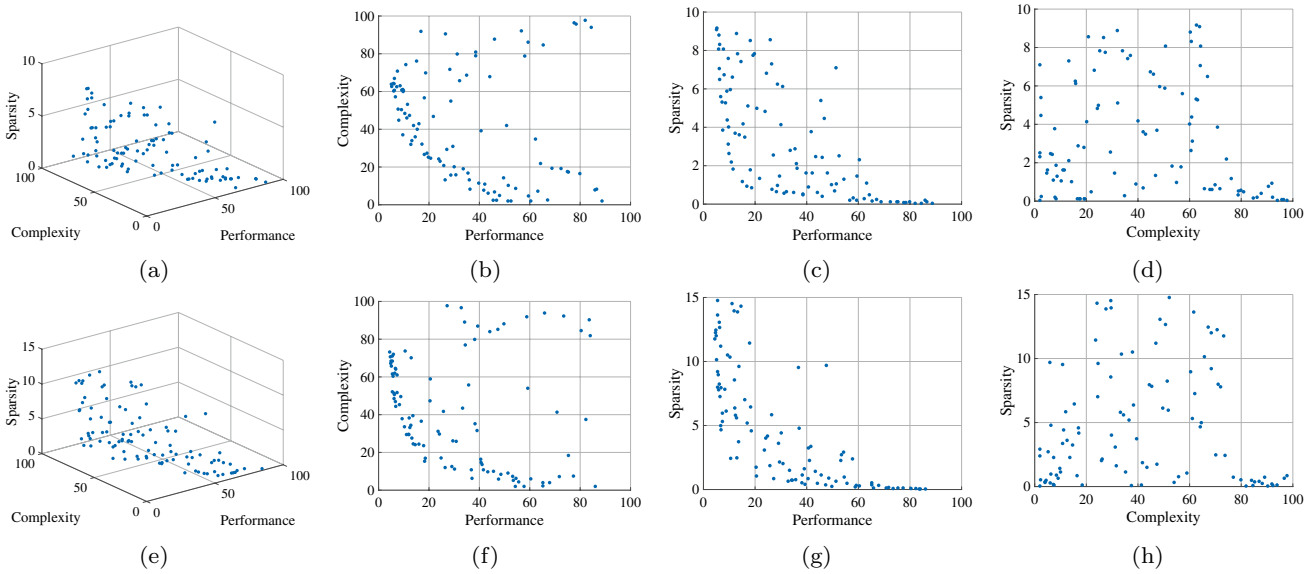


Fig. 1 Dataset: Letter-H. 3D Pareto Front and pairwise 2D projections. Top panels correspond to the Linear-SVM, bottom panels correspond to RBF-SVM

- 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 is below 0.0025 for 15 consecutive generations;
- $W = 30\%$ of the maximum number of subgraphs attainable from the training set[6];
- a simple random walk is employed as graph traversing strategy for mining subgraphs of maximum order $V = 5$ for the extraction phase (i.e., for populating $\mathcal{S}_g^{(\mathrm{tr})}$))[7]
- a modified BFS[8] is employed for extracting subgraphs for the embedding phase (i.e., for building G_{exp})
- parameter θ in BSAS follows a binary search in range [0, 1];
- $V = 5$, as the maximum number of nodes for the subgraphs;
- $\zeta_{s_i} = 1.1 \cdot \Phi(\mathbf{C}_i)$ is the symbol-dependent threshold for scoring a hit in the embedding procedure (cf. Eq. (6)).

The software has been implemented in Python with the support of the following libraries: PyMoo [12] for MOO (NSGA-II), NetworkX [33] for handling graphs and Scikit-Learn [59] for machine learning-related routines.

[6] AIDS: 27589; GREC: 21009; Letter-L: 7975; Letter-M: 8217; Letter-H: 12603; MUTAG: 15530; PTC: 16127.

[7] In our previous work [7], we witnessed that there is no clear winner between BFS or DFS in terms of performances and/or running times, so the rationale behind choosing random walks is that there exist the possibility of having both star-like and path-like subgraphs in $\mathcal{S}_g^{(\mathrm{tr})}$.

[8] As proposed in [7], we use BFS to extract subgraphs in such as way that already-visited nodes do not appear as root nodes for following traversals: this assures a complete coverage of the graph while, at the same time, keeping a low number of resulting subgraphs.

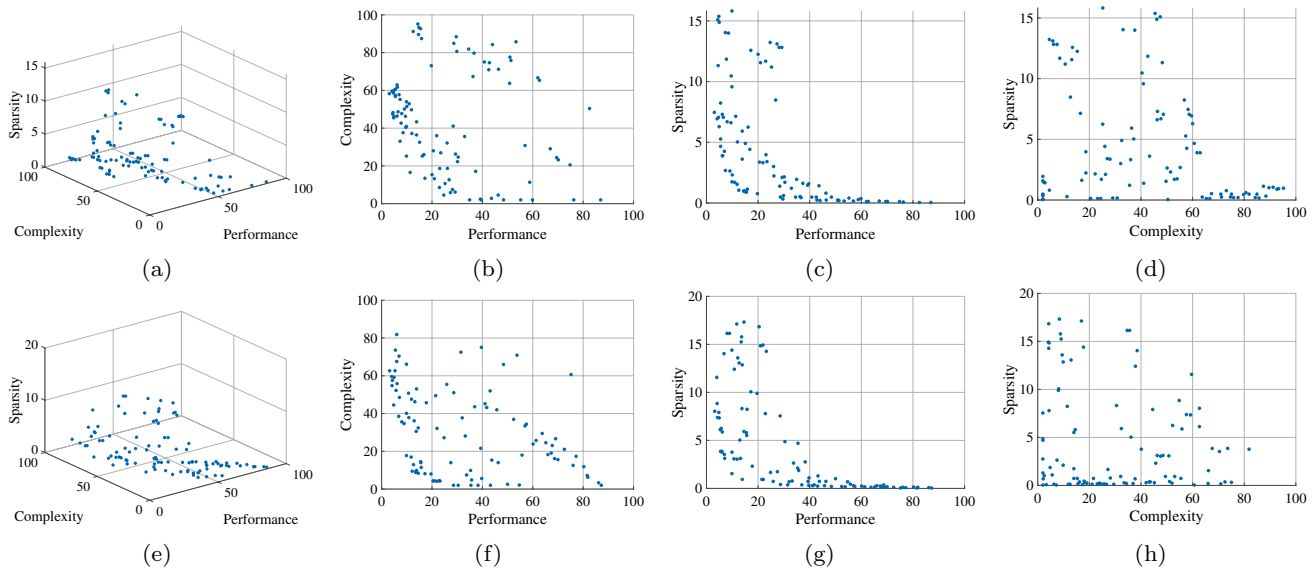


Fig. 2 Dataset: Letter-M. 3D Pareto Front and pairwise 2D projections. Top panels correspond to the Linear-SVM, bottom panels correspond to RBF-SVM

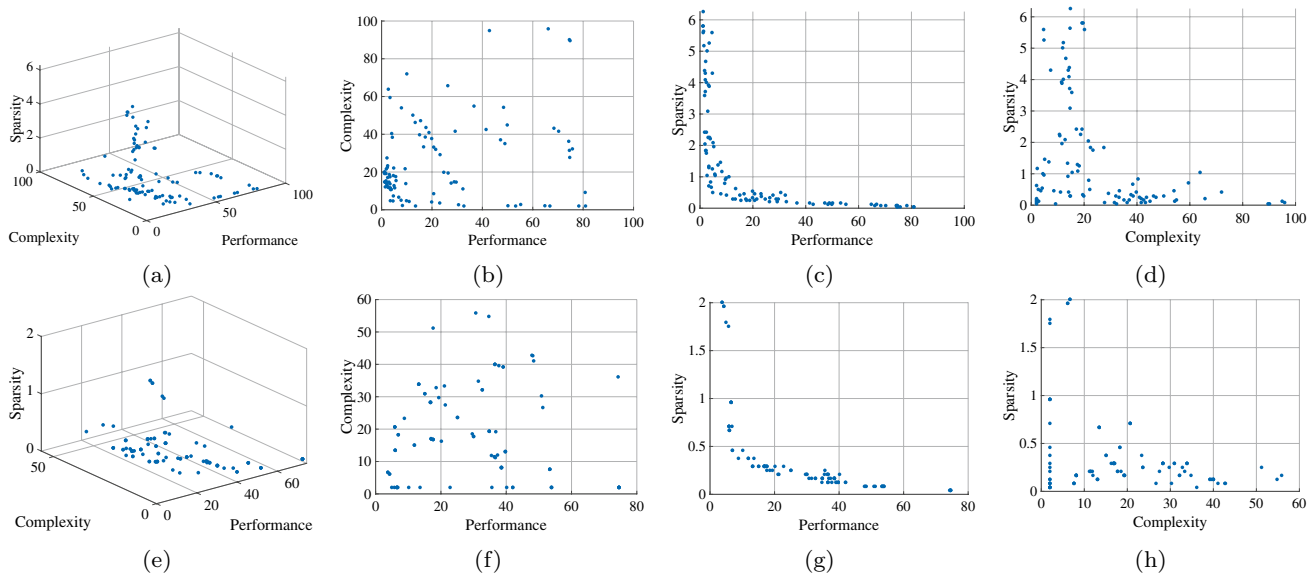


Fig. 3 Dataset: Letter-L. 3D Pareto Front and pairwise 2D projections. Top panels correspond to the Linear-SVM, bottom panels correspond to RBF-SVM

Computational Results

In this section, we assess the relation between the three-objective functions f_1, f_2 and f_3 (respectively, denoted as *Performance*, *Complexity* and *Sparsity* in the following) by showing the Pareto frontiers obtained for the five datasets described in "Datasets Description and Algorithmic Setup" and the pairwise 2D projections. The results are summarized individually for each dataset in Figs. 1, 2, 3, 4 and 5, respectively for Letter-H, Letter-M, Letter-L, AIDS, GREC, MUTAG and PTC for both kernels (linear and RBF) used in the SVM classifiers. To ensure readability, axes have been scaled in terms of percentage.

A first immediate regular behavior 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 Figs. 1c, 2c and 3c using linear SVM. The same behavior holds for its

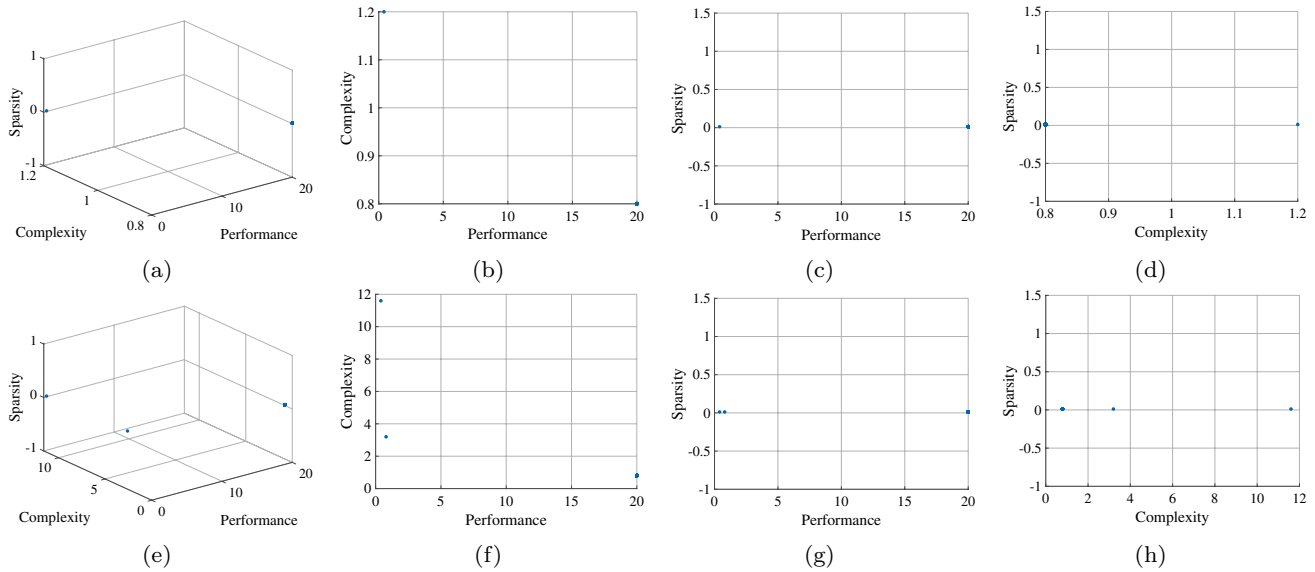


Fig. 4 Dataset: AIDS dataset. 3D Pareto Front and pairwise 2D projections. Top panels correspond to the Linear-SVM, bottom panels correspond to RBF-SVM

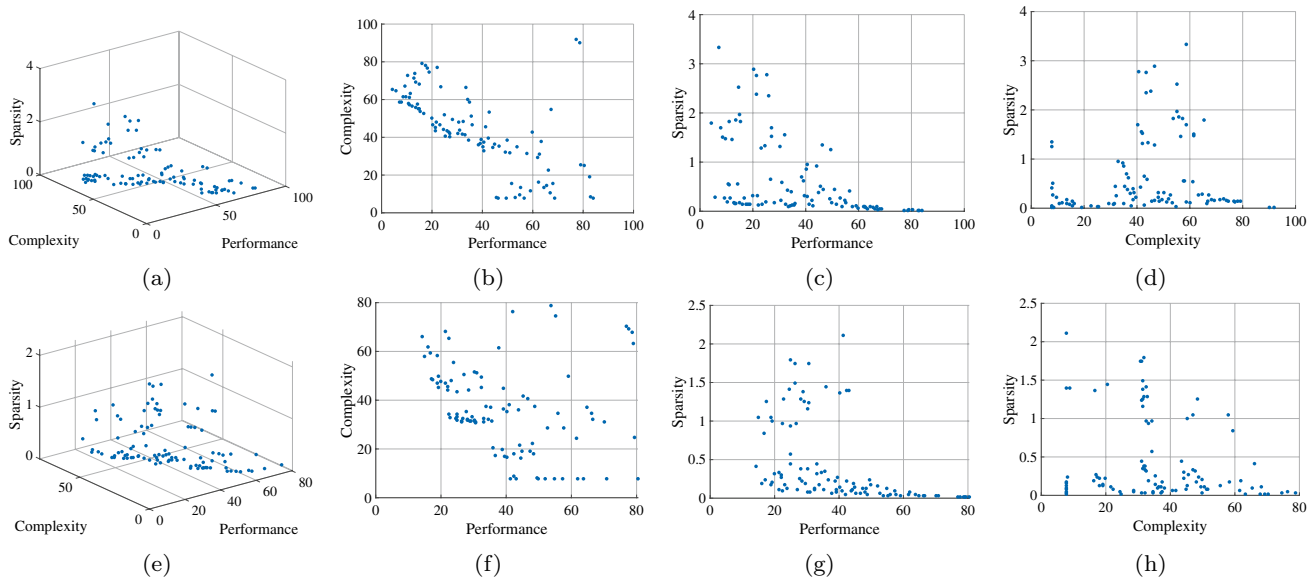


Fig. 5 Dataset: GREC. 3D Pareto Front and pairwise 2D projections. Top panels correspond to the Linear-SVM, bottom panels correspond to RBF-SVM

kernelized counterpart as depicted in Figs. 1g, 2g and 3g. A similar phenomenon, but slightly less prominent, occurs for GREC dataset for both linear (Fig. 5c) and RBF kernel (Fig. 5g). On the other hand, AIDS, MUTAG and PTC datasets do not show any relevant trend for the Sparsity-Performance pair due to the dense superposition of the solutions in the Pareto frontiers as can be noticed in Figs. 4a, 6a, and 7a (Linear-SVM) and Figs. 4 e, 6e, and 7e (RBF-SVM). These results entail a transparent relationship between the number of symbols in the alphabet $\mathcal{A}$ necessary for attaining reliable performances Π, 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 Fig. 1b, f. In this particular problem, an elbow-like behavior for the two considered objective functions arise, suggesting that its decision boundary can be well defined with an increasing number of support vectors. For

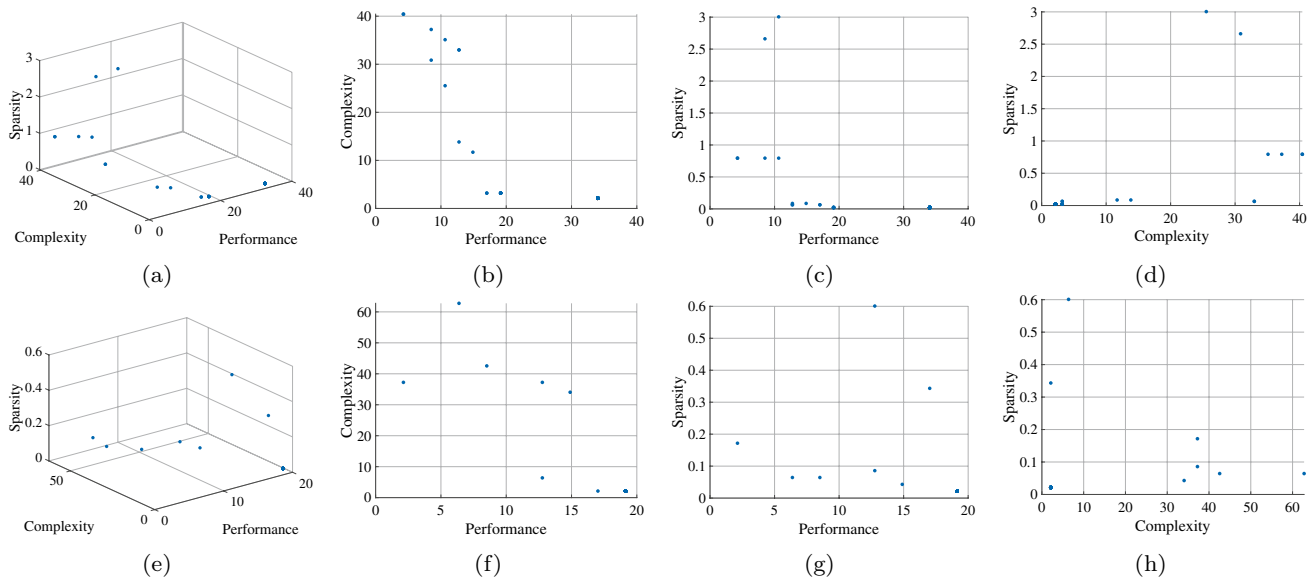


Fig. 6 Dataset: MUTAG. 3D Pareto Front and pairwise 2D projections. Top panels correspond to the Linear-SVM, bottom panels correspond to RBF-SVM

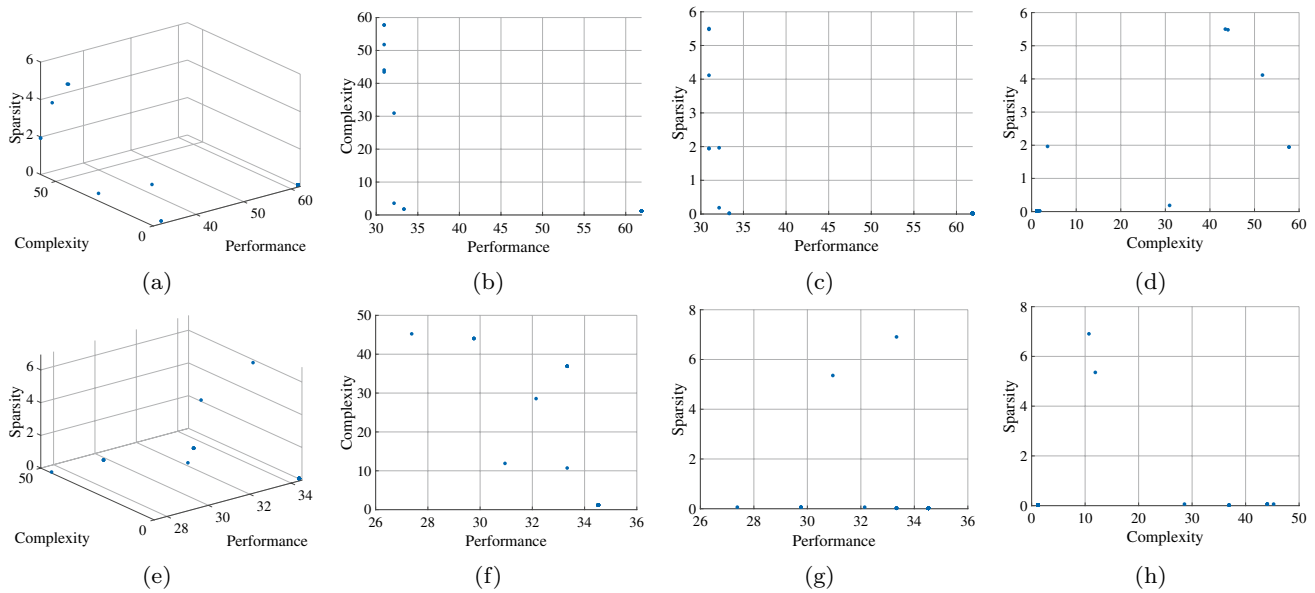


Fig. 7 Dataset: PTC. 3D Pareto Front and pairwise 2D projections. Top panels correspond to the Linear-SVM, bottom panels correspond to RBF-SVM

the remaining two Letter datasets, i.e., Letter-M (Fig. 2b, f) and Letter-L (Fig. 3b, f), 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 other hand, for GREC dataset, the Linear-SVM projection in Fig. 5b has a firm decreasingly trend, while its kernelized counterpart shows a less compact behavior but with a more defined point cloud in the upper-left region of the plot. Again, no clear trend emerges from AIDS, MUTAG and PTC datasets due to the former discussion about the superposition of the solutions.

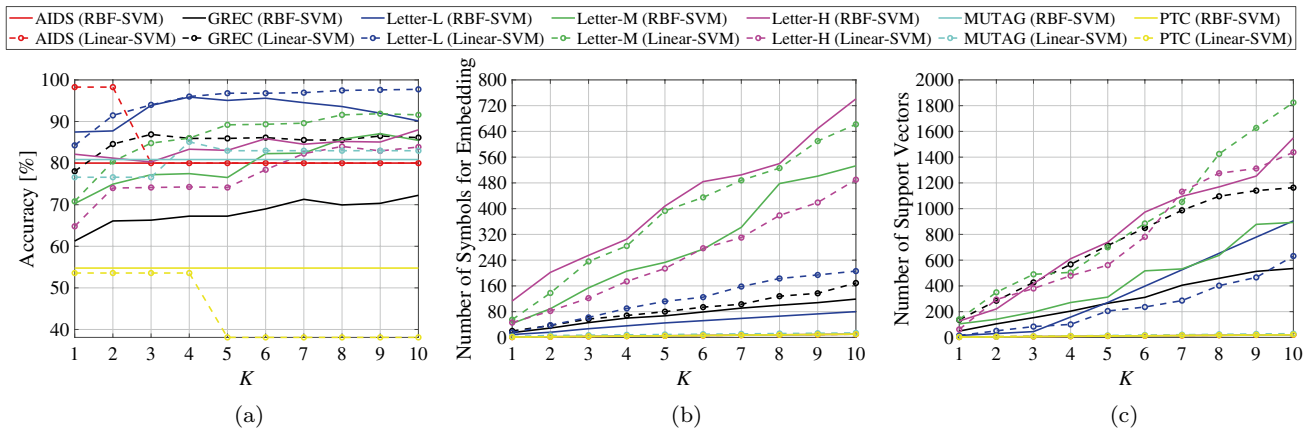


Fig. 8 Uniform weighting results

The last aspect to consider regards the Complexity-vs-Sparsity projection. For almost all datasets, i.e., Letter-H (Fig. 1d, h), Letter-M (Fig. 2d, h), AIDS (Fig. 4d, h), GREC (Fig. 5d), MUTAG (Fig. 5d, h) and PTC (Fig. 7d, h) 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 (Fig. 3d). This suggests that limiting the number of support vectors is counterbalanced by a higher embedding space dimensionality.

Baseline Cases

In this section, we show the computational results obtained on the test phase with the introduction of the ensemble of classifiers working on heterogeneous embedding spaces, as described in "Ensemble of Classifiers for Test Phase". Let us recall from "Selection of Solutions from the Pareto Front" that TOPSIS allows the ranking of the solutions according to a weight vector $\mathbf{u}$.

At first, we consider the *baseline cases*, collected in Table 2, in which we show the solutions in X that achieve the best results in the three-objective functions, if 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, we retain the solution in X that corresponds to the top-ranked row in $\mathbf{D}$. In particular, for each dataset and for each classifier, Table 2 features a 3×3 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. Hereinafter, for the sake of discussion, the Performance objective function will be referred to as the accuracy of the classifier (i.e., the complement of the error rate in Eq. (8)).

For the sake of argument, we also consider a *uniform weighting*: that is, we let TOPSIS to rank solutions in X without introducing any preferences for the three-objective functions, i.e., $\mathbf{u} = \left[\frac{1}{3}\ \frac{1}{3}\ \frac{1}{3}\right]$. The number of classifiers and embedding spaces to consider in the ensemble is chosen according to the top-K solutions X^* returned by TOPSIS[9], with $K = 10$. In Fig. 8, we show the results obtained by varying $K = 1, \ldots, 10$ and observing the three main objective of interests: accuracy on test set of the ensemble (Fig. 8a), the sum of symbols in the alphabets $\mathcal{A}^*$ (Fig. 8b) and the sum of support vectors for the classifiers in the ensemble $\mathcal{C}$ (Fig. 8c).

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, we can see that if we select the solution that minimizes the error rate regardless of the other two objective functions, the accuracy is 98% (see Table 2) while 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. For GREC with Linear-SVM, we can obtain comparable accuracy performance 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.

[9] In Appendix A, for the sake of completeness, we show the 3D Pareto fronts in which the solutions belonging to X^* are highlighted (Fig. 12a, b for AIDS, Fig. 13a, b for GREC, Fig. 14a, b for Letter-L, Fig. 15a, b for Letter-M, Fig. 16a, b for Letter-H, Fig. 17a, b for MUTAG and Fig. 18a, b for PTC.

Table 2 Results of the baseline cases

	Linear-SVM			RBF-SVM		
	min f_1	min f_2	min f_3	min f_1	min f_2	min f_3
AIDS						
Accuracy	98%	80%	80%	99%	80%	80%
Complexity	3	2	2	29	2	2
Sparsity	1	1	1	1	1	1
GREC						
Accuracy	92%	15%	15%	85%	50%	20%
Complexity	185	22	22	187	22	22
Sparsity	113	1	1	26	11	1
Letter-L						
Accuracy	98%	68%	25%	97%	92%	24%
Complexity	147	15	676	50	15	15
Sparsity	139	5	1	48	11	1
Letter-M						
Accuracy	94%	53%	11%	95%	70%	13%
Complexity	437	15	15	470	15	15
Sparsity	184	12	1	198	120	1
Letter-H						
Accuracy	93%	50%	17%	92%	14%	14%
Complexity	479	15	733	549	15	15
Sparsity	224	175	1	290	1	1
MUTAG						
Accuracy	87%	66%	66%	85%	81%	81%
Complexity	38	2	2	35	2	2
Sparsity	37	1	1	8	1	1
PTC						
Accuracy	55%	38%	38%	64%	55%	55%
Complexity	97	2	2	76	2	2
Sparsity	94	1	1	3	1	1

On the other hand, the ensemble requires a considerably higher number of support vectors (427 against 185). If we consider the MUTAG dataset, the baseline result in terms of accuracy can be reached with $K = 4$ classifiers using a Linear kernel: the baseline method achieves 87% of accuracy score with 38 support vectors and 37 symbols in the alphabet, whereas the ensemble method is able to correctly classify 85% of the patterns in test set by relying only on 12 support vectors and 8 symbols. 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, we present three different case studies in order to analyze the behavior of the ensemble of classifiers under different configurations for TOPSIS. Indeed, we explore how the computational results change as a function of higher priorities given to specific objective functions in the decision making process. In the following, we investigate three different choices for $\mathbf{u}$, that, respectively:

(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)

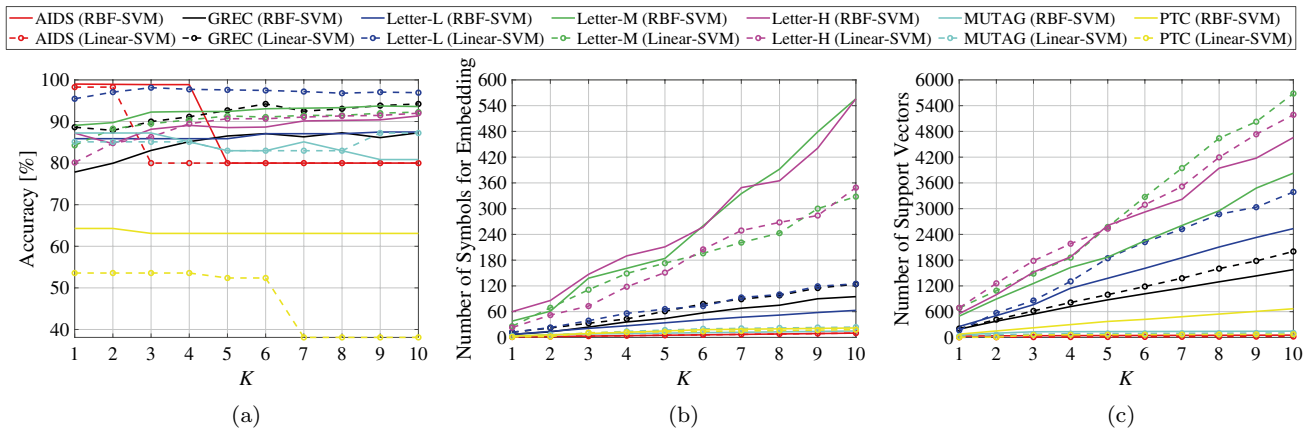


Fig. 9 Case Study A (no weight to model structural complexity) results

other objectives. In light of these observations, we deliberately did not consider 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: 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, we inform TOPSIS to prioritize 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. (8) and Eq. (10), respectively. By letting $\mathbf{u} = \left[\frac{1}{2}\ 0\ \frac{1}{2}\right]$, the ranking process will be not influenced by the objective function f_2 in Eq. (9); 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 (Fig. 16c, d), Letter-M (Fig. 15c, d), Letter-L (Fig. 13c, d) and GREC (Fig. 13c, d), 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, MUTAG (Fig. 12c, d) and PTC (Fig. 17c, d) datasets (Fig. 17c, d), all solutions are overlapped in few clusters, making less evident the selected solution positions. As already noticed in "Baseline Cases", the introduction of the ensemble of classifiers implies some benefits when compared to the baseline solutions shown in Table 2. 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 Fig. 9a–c (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) 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, while 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 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.

Similarly, the same behavior can be noticed for MUTAG dataset for both kernel choices: with only one classifier ($K = 1$), it is possible to achieve very similar performance in terms of accuracy when compared to the baseline method, that is 85% with Linear-SVM (baseline is 87%) and 87% with RBF kernel (baseline is 85%) using only 3 symbols in the alphabet, whereas the baseline leverages, respectively 37 and 8 symbols.

The same discussion holds for Letter-L: whether 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 while 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,

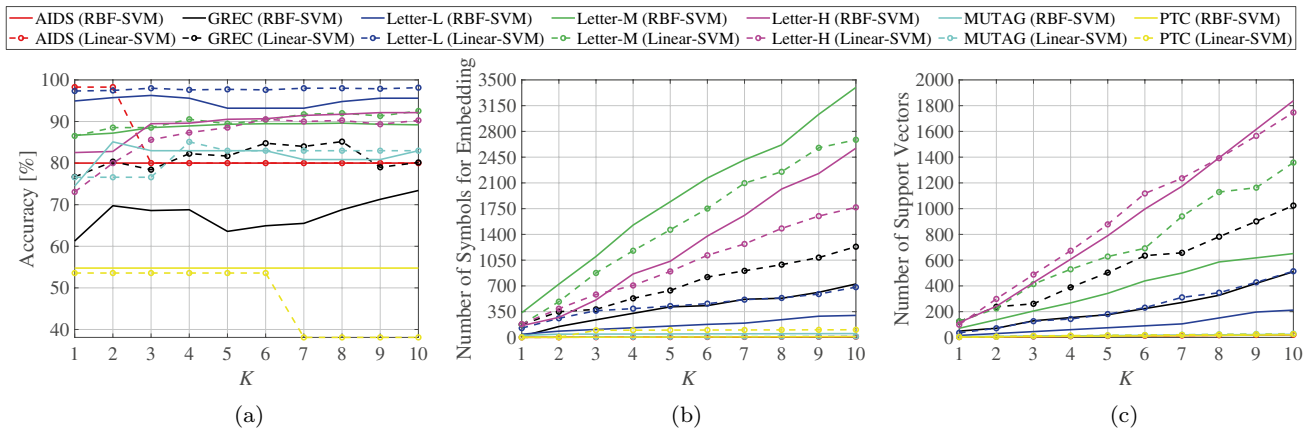


Fig. 10 Case Study B (no weight to sparsity) results

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, we witnessed a considerable lower number of symbols, 118 (Linear) and 190 (RBF) against 224 (Linear) and 290 (RBF) if compared to the best solution in terms of performance. 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 behavior 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, we investigate the ensemble behavior by choosing $\mathbf{u} = \left[\frac{1}{2}\ \frac{1}{2}\ 0\right]$. 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. (8) and (9). 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 (Fig. 16e–h), Letter-M (Fig. 15e, f), Letter-L (Fig. 13e, f), GREC (Fig. 13e, f), AIDS (Fig. 12e, f), MUTAG (Fig. 17e, f) and PTC (Fig. 17e, f). 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. By varying the number of classifier K in the ensemble, we report the performance on the test set, model structural complexity and sparsity of the alphabet, respectively, in Fig. 10a–c.

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 2: Linear-SVMs reaches 97% of accuracy (very close to the baseline solution, that scores 98%) 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 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).

Also for MUTAG, the RBF kernel with $K = 2$ classifiers approaches the same performances of the baseline, i.e., 85%. Notably, the number of support vectors is slightly improved, i.e., 8 (ensemble) against 35 (baseline). Similarly, with Linear-SVM and $K = 4$, the classifier relies on 12 support vectors obtaining 85% of accuracy, while the baseline achieves a similar result, i.e., 85% on the test set with 38 support vectors.

Finally, AIDS can be solved as efficiently as the baseline with Linear-SVMs. Indeed, the same accuracy, i.e., 98%, can be attained using $K = 2$ and 5 support vectors.

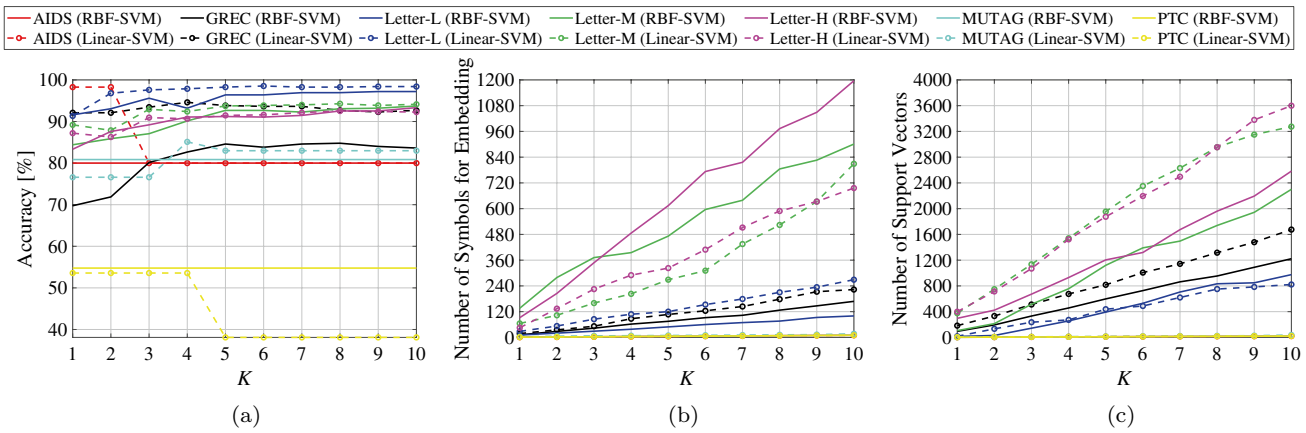


Fig. 11 Case Study C (priority to accuracy performance) results

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.

Case Study C: Priority to Accuracy Performance

In the last case study, we choose the TOPSIS weighting vector $\mathbf{u} = \left[\frac{1}{2}\ \frac{1}{4}\ \frac{1}{4}\right]$. This configuration drives the TOPSIS decision making process mainly toward solutions featured by low error rate, that is, we are prioritizing 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 (Fig. 16g, h), Letter-M (Fig. 15g, h), Letter-L (Fig. 13g, h), GREC (Fig. 13g, h) AIDS (Fig. 12e, f, MUTAG (Fig. 17g, h) and PTC (Fig. 17g, h). 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 Fig. 11a–c 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 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 results comparable with the baseline, i.e., ≈ 225 symbols, while 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. For MUTAG, when the Linear kernel is considered, the ensemble with $K = 4$ classifiers achieves performances (85%) comparable with the baseline method in terms of accuracy (87%) with a major improvement in terms of both complexity and sparsity: 8 support vectors for the ensemble against 38 in the baseline and 12 symbols (ensemble) versus 37 (baseline). On the other hand, the RBF kernel attained same accuracy of the baseline (85%) with $K = 2$ classifiers without improving the other two objective functions.

For PTC, the linear kernel SVM with $K = 1$ obtains the same performances of the baseline simultaneously improving both complexity (1 support vector in ensemble against 97 for the baseline) and sparsity (3 total symbols in alphabets against 94 for the baseline). Conversely, with the RBF kernel, it is not possible to achieve comparable results in terms of accuracy, regardless of the number of classifiers in the ensemble since in all configurations the maximum accuracy score obtained is less than 54%.

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.

Conclusions

In this paper, we have addressed the possibility of synthesizing a GrC-based pattern recognition system in the graph domain via MOO. Conversely to the vast majority of the current works in the literature, our approach aims at simultaneously minimizing:

1. an error term on the validation set;
2. the number of symbols (i.e., features) for embedding the graphs into a geometric space;
3. the structural complexity of the classification model.

The variable space of the MOO optimizer (NSGA-II) includes the parameters of the granulation module, the weights of the dissimilarity measure and the hyperparameters of the classifiers. As the latter point is concerned, we tested a linear SVM and a non-linear SVM equipped with the RBF kernel. Finally, we explored the possibility of selecting a small and finite number of K Pareto-efficient solutions for building an ensemble of classifiers. 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 dissimilarity measure parameters.

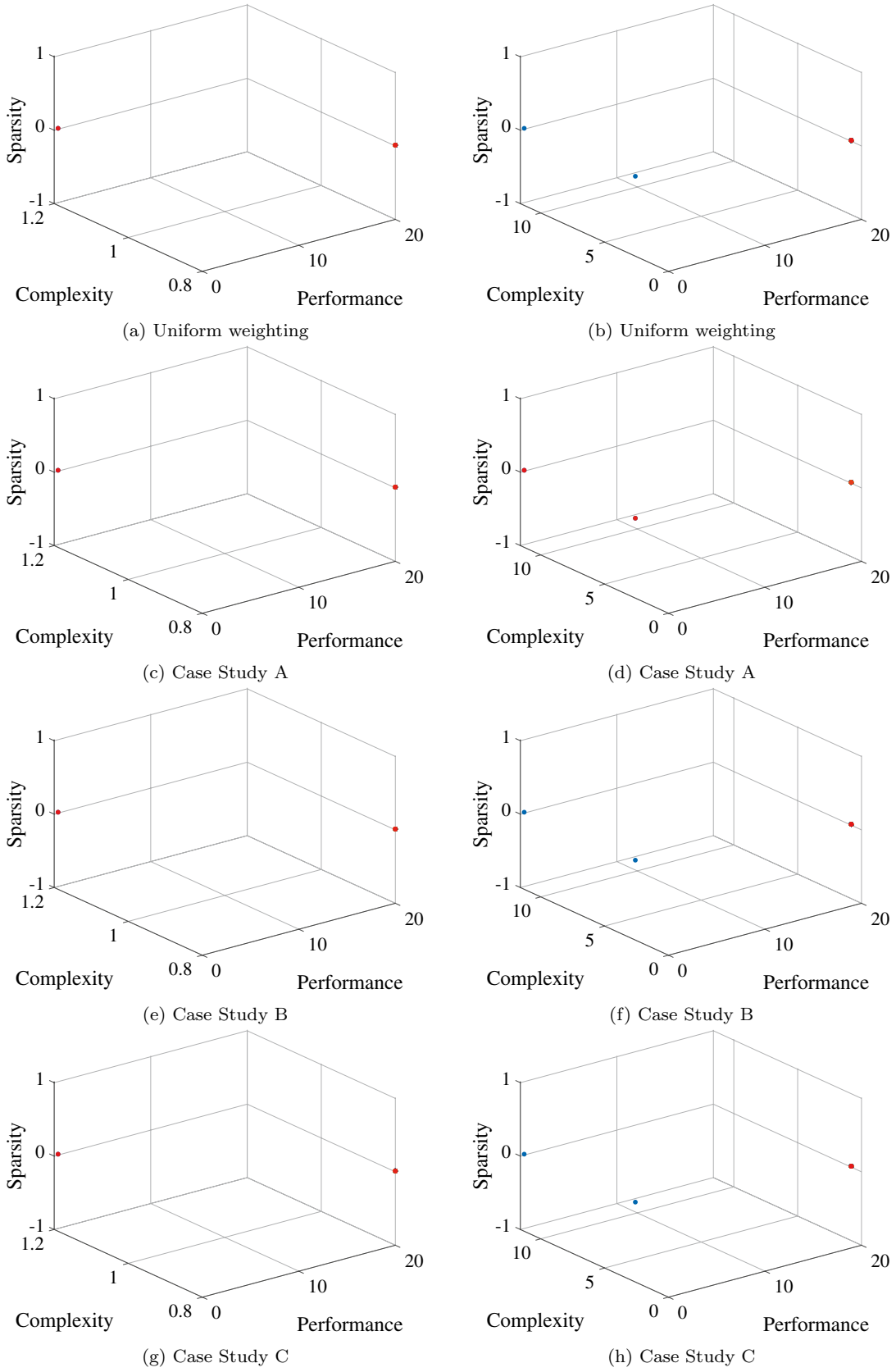
In order to study the behavior of the ensemble (as a function of the number of classifiers), we also considered a uniform weighting case alongside three different case studies (that differ on how much each objective function weights in the selection of the solutions from the Pareto Frontier) and three baseline cases (where three best solutions from the Pareto Frontier are the one that minimize each objective function, independently).

Our findings can be summarized as follows:

1. 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.
2. 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, we observed that a joint optimization of performance and sparsity yields ensemble models that improve against the baseline case in terms of 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.
3. There seem to exist a cutoff 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 cutoff K has past, the growth of the number of support vectors and/or symbols does not justify any possible improvements in terms of performances.

A Pareto Fronts with Selected Solutions

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, Figs. 12, 13, 14, 15, 16, 17 and 18 depict the Pareto Fronts for AIDS, GREC, Letter-L, Letter-M, Letter-H, MUTAG and PTC, 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 "Computational Results". 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'



(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

Fig. 12 Dataset: AIDS. Left panels correspond to Linear-SVM, right panels correspond to RBF-SVM

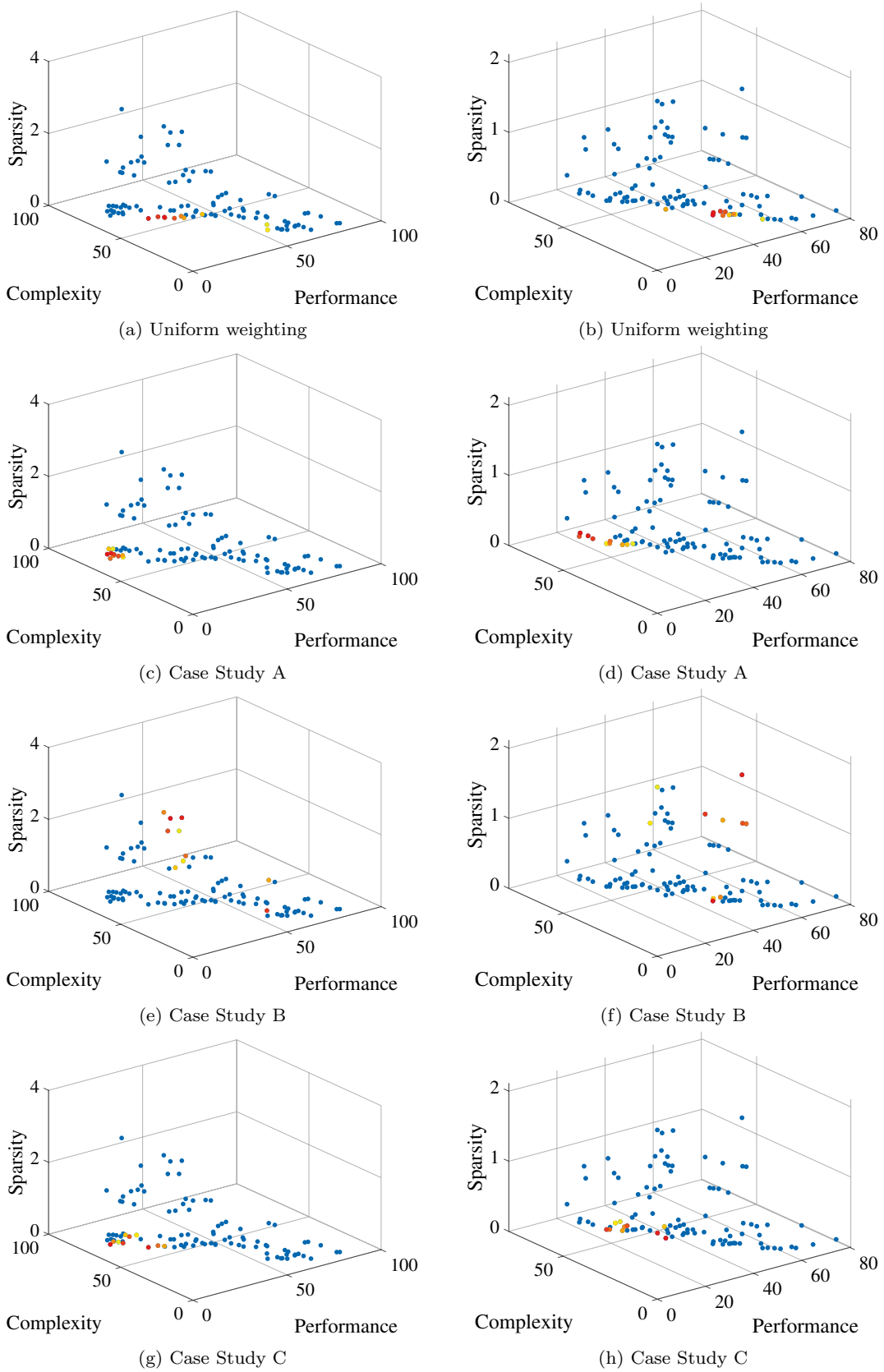


Fig. 13 Dataset: GREC. Left panels correspond to Linear-SVM, right panels correspond to RBF-SVM

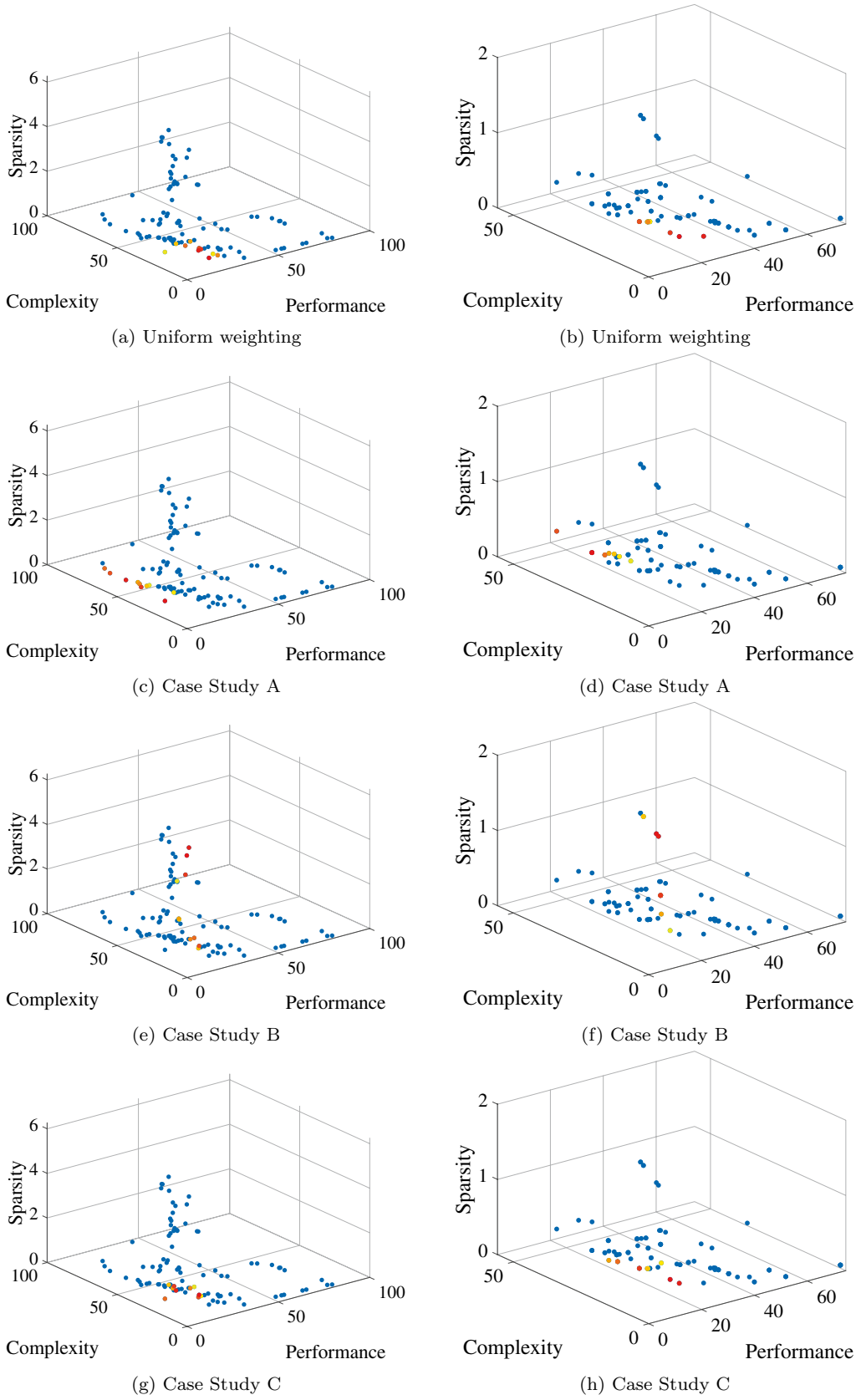


Fig. 14 Dataset: Letter-L. Left panels correspond to Linear-SVM, right panels correspond to RBF-SVM

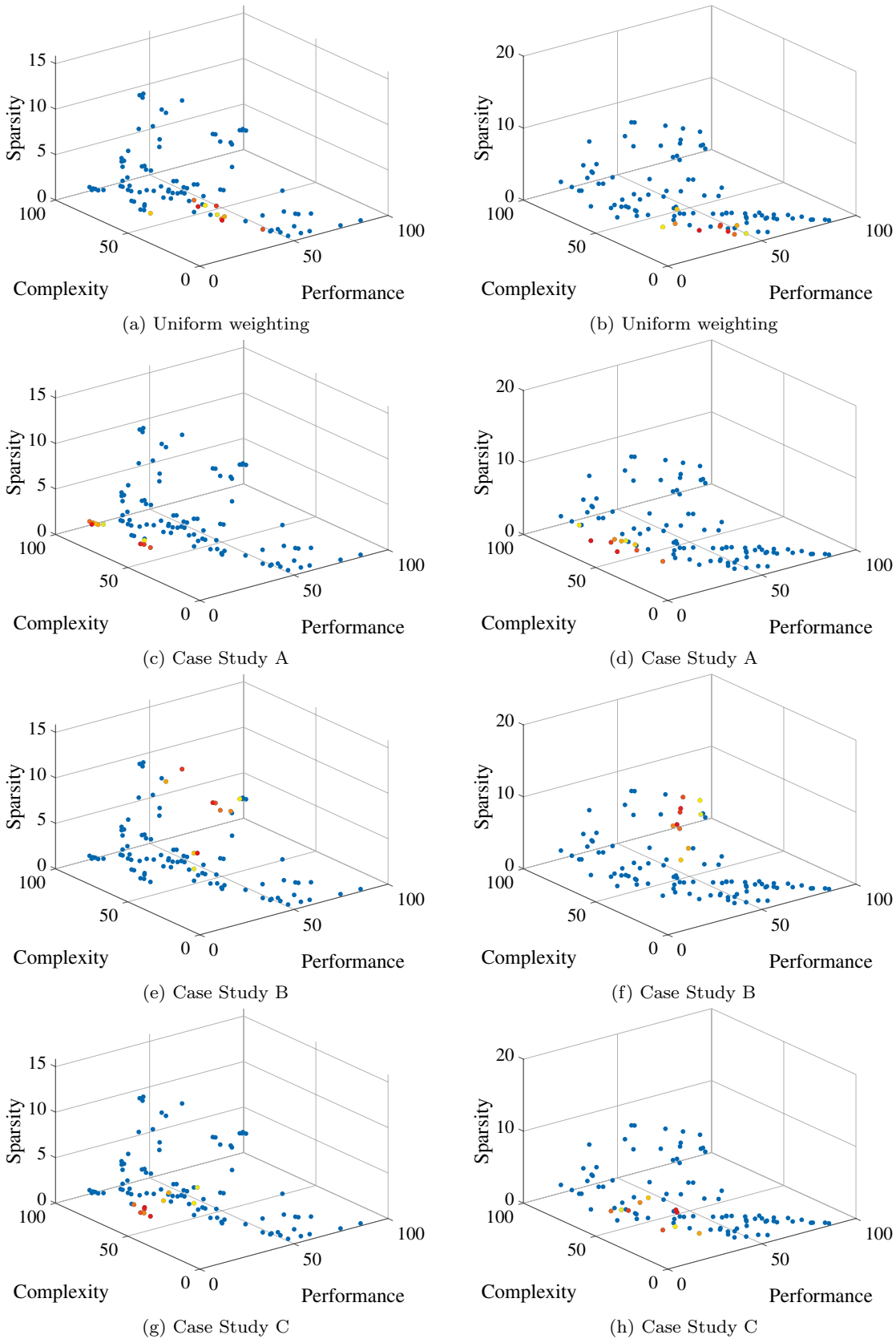


Fig. 15 Dataset: Letter-M. Left panels correspond to Linear-SVM, right panels correspond to RBF-SVM

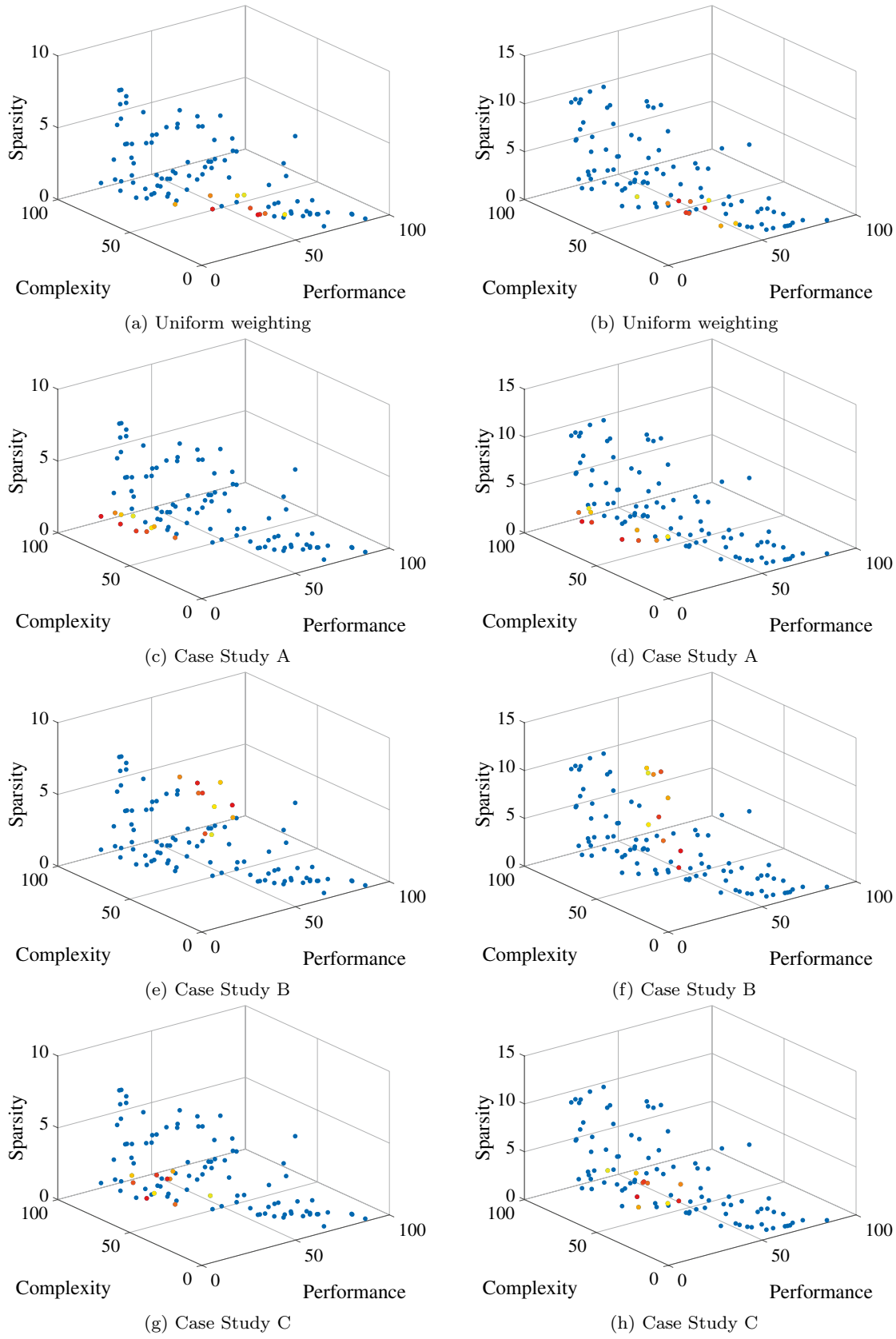


Fig. 16 Dataset: Letter-H. Left panels correspond to Linear-SVM, right panels correspond to RBF-SVM

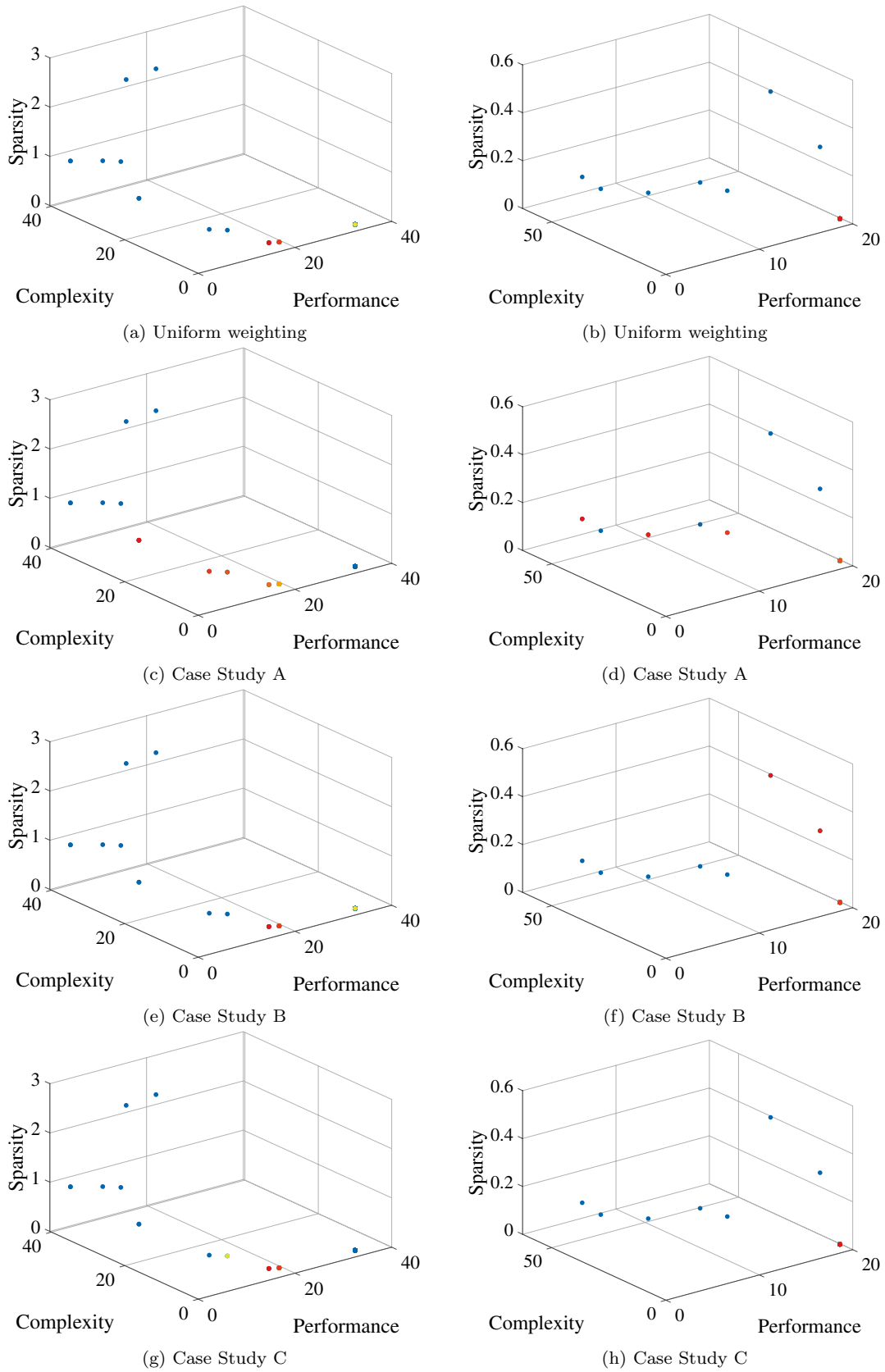


Fig. 17 Dataset: MUTAG. Left panels correspond to Linear-SVM, right panels correspond to RBF-SVM

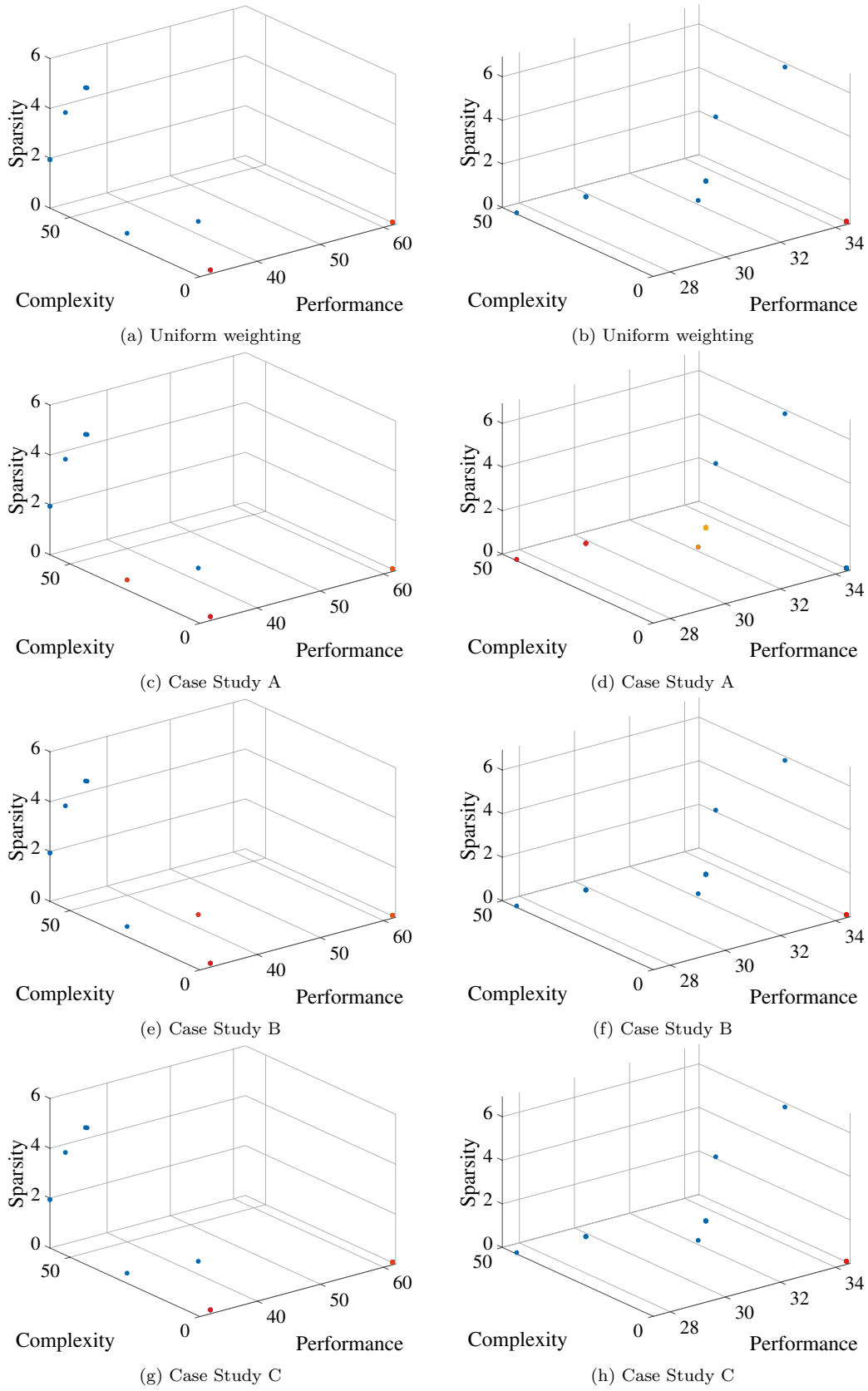


Fig. 18 Dataset: PTC. Left panels correspond to Linear-SVM, right panels correspond to RBF-SVM

blue color (already in Figs. 1, 2, 3, 4, 5). Worthy of attention are Figs. 12, 17 and 18 (AIDS, MUTAG and PTC): in fact, since the *N* solutions are overlapping into few points on the Pareto Front, selected solutions are overlapping to non-selected solutions.

Funding This research received no external funding.

Declarations

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

References

1. Ahmad F, Mat Isa NA, Hussain Z, Sulaiman SN. A genetic algorithm-based multi-objective optimization of an artificial neural network classifier for breast cancer diagnosis. Neural Comput Appl. 2013;23(5):1427–35. https://doi.org/10.1007/s00521-012-1092-1.
2. Ahmed ST, Sankar S, Sandhya M. Multi-objective optimal medical data informatics standardization and processing technique for telemedicine via machine learning approach. J Ambient Intell Humaniz Comput. 2021;12(5):5349–58. https://doi.org/10.1007/s12652-020-02016-9.
3. Al-Tashi Q, Abdulkadir SJ, Rais HM, Mirjalili S, Alhussian H. Approaches to multi-objective feature selection: a systematic literature review. IEEE Access. 2020;8:125076–96. https://doi.org/10.1109/ACCESS.2020.3007291.
4. Alves Ribeiro VH, Reynoso-Meza G. Ensemble learning by means of a multi-objective optimization design approach for dealing with imbalanced data sets. Expert Syst Appl. 2020;147: 113232. https://doi.org/10.1016/j.eswa.2020.113232.
5. Annamdas KK, Rao SS. Multi-objective optimization of engineering systems using game theory and particle swarm optimization. Eng Optim. 2009;41(8):737–52. https://doi.org/10.1080/03052150902822141.
6. Baldini L, Martino A, Rizzi A. Complexity vs. performance in granular embedding spaces for graph classification. In: Proceedings of the 12th International Joint Conference on Computational Intelligence—NCTA. INSTICC, SciTePress; 2020. pp. 338–349. https://doi.org/10.5220/0010109503380349
7. Baldini L, Martino A, Rizzi A. Stochastic information granules extraction for graph embedding and classification. In: Proceedings of the 11th international joint conference on computational intelligence—NCTA, (IJCCI 2019). INSTICC, SciTePress; 2019. pp. 391–402. https://doi.org/10.5220/0008149403910402
8. Baldini L, Martino A, Rizzi A. Towards a class-aware information granulation for graph embedding and classification. In: Merelo JJ, Garibaldi J, Linares-Barranco A, Warwick K, Madani K, editors. Computational intelligence: 11th international joint conference, IJCCI 2019 Vienna, Austria, September 17–19, 2019, Revised Selected Papers. Berlin: Springer; 2021.
9. Bargiela A, Pedrycz W. Granular computing: an introduction. In: The Springer International Series in Engineering and Computer Science, vol. 717, 1 edn. Berlin: Springer; 2003.
10. Bianchi FM, Scardapane S, Livi L, Uncini A, Rizzi A. An interpretable graph-based image classifier. In: 2014 International joint conference on neural networks (IJCNN); 2014. pp. 2339–2346. https://doi.org/10.1109/IJCNN.2014.6889601
11. Bianchi FM, Livi L, Rizzi A, Sadeghian A. A granular computing approach to the design of optimized graph classification systems. Soft Comput. 2014;18(2):393–412. https://doi.org/10.1007/s00500-013-1065-z.
12. Blank J, Deb K. Pymoo: multi-objective optimization in python. IEEE Access. 2020;8:89497–509.
13. Brauers WK. Optimization methods for a stakeholder society: a revolution in economic thinking by multi-objective optimization. In: Nonconvex Optimization and Its Applications, vol. 73, 1 edn. Berlin: Springer; 2004.
14. Castillo Tapia MG, Coello Coello CA. Applications of multi-objective evolutionary algorithms in economics and finance: a survey. In: 2007 IEEE congress on evolutionary computation; 2007. pp. 532–539. https://doi.org/10.1109/CEC.2007.4424516.
15. Chatelain C, Adam S, Lecourtier Y, Heutte L, Paquet T. Multi-objective optimization for svm model selection. In: Ninth international conference on document analysis and recognition (ICDAR 2007), vol. 1; 2007. pp. 427–431. https://doi.org/10.1109/ICDAR.2007.4378745.
16. Chen JH, Chen HM, Ho SY. Design of nearest neighbor classifiers: multi-objective approach. Int J Approx Reason. 2005;40(1):3–22. https://doi.org/10.1016/j.ijar.2004.11.009.
17. Cortes C, Vapnik V. Support-vector networks. Mach Learn. 1995;20(3):273–97. https://doi.org/10.1007/BF00994018.
18. Cover T, Hart P. Nearest neighbor pattern classification. IEEE Trans Inf Theory. 1967;13(1):21–7.
19. Deb K, Datta R. Hybrid evolutionary multi-objective optimization and analysis of machining operations. Eng Optim. 2012;44(6):685–706. https://doi.org/10.1080/0305215X.2011.604316.
20. Deb K, Agrawal S, Pratap A, Meyarivan T. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. In: Schoenauer M, Deb K, Rudolph G, Yao X, Lutton E, Merelo JJ, Schwefel HP, editors. Parallel problem solving from nature PPSN VI. Berlin: Springer; 2000. p. 849–58.
21. Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Trans Evol Comput. 2002;6(2):182–97. https://doi.org/10.1109/4235.996017.
22. Debnath AK, de Compadre RLL, Debnath G, Shusterman AJ, Hansch C. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds correlation with molecular orbital energies and hydrophobicity. J Med Chem. 1991;34(2):786–97. https://doi.org/10.1021/jm00106a046.
23. Del Vescovo G, Rizzi A. Online handwriting recognition by the symbolic histograms approach. In: 2007 IEEE international conference on granular computing (GRC 2007); 2007. p. 686. IEEE.
24. Del Vescovo G, Livi L, Frattale Mascioli FM, Rizzi A. On the problem of modeling structured data with the minsod representative. Int J Comput Theory Eng. 2014;6(1):9.
25. Dosch P, Valveny E. Report on the second symbol recognition contest. In: Liu W, Lladós J, editors. Graphics recognition. Ten years review and future perspectives. Berlin: Springer; 2006. p. 381–97.
26. Dua D, Graff C. UCI machine learning repository; 2017. http://archive.ics.uci.edu/ml.
27. Emmerich MT, Deutz AH. A tutorial on multiobjective optimization: fundamentals and evolutionary methods. Nat Comput. 2018;17(3):585–609.
28. Emmert-Streib F, Dehmer M, Shi Y. Fifty years of graph matching, network alignment and network comparison. Inf Sci. 2016;346:180–97.
29. Farsi A, Dincer I, Naterer GF. Multi-objective optimization of an experimental integrated thermochemical cycle of hydrogen

production with an artificial neural network. Int J Hydrogen Energy. 2020;45(46):24355–69. https://doi.org/10.1016/j.ijhydene.2020.06.262.
30. Fonseca C, Fleming P. Multiobjective genetic algorithms. In: IEE Colloquium on Genetic Algorithms for Control Systems Engineering; 1993. pp. 6/1–6/5.
31. Graning L, Jin Y, Sendhoff B. Generalization improvement in multi-objective learning. In: The 2006 IEEE International Joint Conference on Neural Network Proceedings; 2006. pp. 4839–4846. https://doi.org/10.1109/IJCNN.2006.247162
32. Gunasekara RC, Mehrotra K, Mohan CK. Multi-objective optimization to identify key players in social networks. In: 2014 IEEE/ACM international conference on advances in social networks analysis and mining (ASONAM 2014); 2017. pp. 443–450. https://doi.org/10.1109/ASONAM.2014.6921623.
33. Hagberg AA, Schult DA, Swart PJ. Exploring network structure, dynamics, and function using networkx. In: Varoquaux G, Vaught T, Millman J, editors. Proceedings of the 7th Python in Science Conference. Pasadena, CA USA; 2008. pp. 11–15.
34. Handl J, Kell DB, Knowles J. Multiobjective optimization in bioinformatics and computational biology. IEEE/ACM Trans Comput Biol Bioinf. 2007;4(2):279–92. https://doi.org/10.1109/TCBB.2007.070203.
35. Helma C, King RD, Kramer S, Srinivasan A. The predictive toxicology challenge 2000–2001. Bioinformatics. 2001;17(1):107–8. https://doi.org/10.1093/bioinformatics/17.1.107.
36. Hwang CL, Lai YJ, Liu TY. A new approach for multiple objective decision making. Comput Oper Res. 1993;20(8):889–99. https://doi.org/10.1016/0305-0548(93)90109-V.
37. Jiang P, Liu Z. Variable weights combined model based on multi-objective optimization for short-term wind speed forecasting. Appl Soft Comput. 2019;82: 105587. https://doi.org/10.1016/j.asoc.2019.105587.
38. Jin Y, Sendhoff B. Pareto-based multiobjective machine learning: an overview and case studies. IEEE Trans Syst Man Cybern Part C (Applications and Reviews). 2008;38(3):397–415. https://doi.org/10.1109/TSMCC.2008.919172.
39. Karasu S, Altan A, Bekiros S, Ahmad W. A new forecasting model with wrapper-based feature selection approach using multi-objective optimization technique for chaotic crude oil time series. Energy. 2020;212: 118750. https://doi.org/10.1016/j.energy.2020.118750.
40. Kiani-Moghaddam M, Shivaie M, Weinsier PD. Introduction to multi-objective optimization and decision-making analysis. Cham: Springer; 2019. p. 21–45. https://doi.org/10.1007/978-3-030-12044-3_2.
41. Kriege N, Mutzel P. Subgraph matching kernels for attributed graphs. In: Proceedings of the 29th international conference on international conference on machine learning, ICML'12. Madison: Omnipress; 2012. pp. 291–298.
42. Liu H, Li Y, Duan Z, Chen C. A review on multi-objective optimization framework in wind energy forecasting techniques and applications. Energy Convers Manage. 2020;224: 113324. https://doi.org/10.1016/j.enconman.2020.113324.
43. Maiorino E, Possemato F, Modugno V, Rizzi A. Noise sensitivity of an information granules filtering procedure by genetic optimization for inexact sequential pattern mining. In: Merelo JJ, Rosa A, Cadenas JM, Dourado A, Madani K, Filipe J, editors. Computational intelligence. Cham: Springer; 2016. p. 131–50.
44. Mardle S, Pascoe S, Tamiz M. An investigation of genetic algorithms for the optimization of multi-objective fisheries bioeconomic models. Int Trans Oper Res. 2000;7(1):33–49. https://doi.org/10.1016/S0969-6016(99)00027-1.
45. Marler RT, Arora JS. Survey of multi-objective optimization methods for engineering. Struct Multidiscip Optim. 2004;26(6):369–95. https://doi.org/10.1007/s00158-003-0368-6.
46. Martino A, Rizzi A. An enhanced filtering-based information granulation procedure for graph embedding and classification. IEEE Access. 2021;9:15426–40. https://doi.org/10.1109/ACCESS.2021.3053085.
47. Martino A, Giuliani A, Rizzi A. (hyper)graph embedding and classification via simplicial complexes. Algorithms. 2019. https://doi.org/10.3390/a12110223.
48. Martino A, Giuliani A, Todde V, Bizzarri M, Rizzi A. Metabolic networks classification and knowledge discovery by information granulation. Comput Biol Chem. 2020;84: 107187. https://doi.org/10.1016/j.compbiolchem.2019.107187.
49. Martino A, Rizzi A, Frattale Mascioli FM. Efficient approaches for solving the large-scale k-medoids problem: Towards structured data. In: Sabourin C, Merelo JJ, Madani K, Warwick K, editors. Computational intelligence: 9th international joint conference, IJCCI 2017 Funchal-Madeira, Portugal, November 1–3, 2017 Revised Selected Papers. Springer, Cham; 2019. pp. 199–219. https://doi.org/10.1007/978-3-030-16469-0_11.
50. Martino A, Frattale Mascioli FM, Rizzi A. On the optimization of embedding spaces via information granulation for pattern recognition. In: 2020 International joint conference on neural networks (IJCNN); 2020. pp. 1–8. https://doi.org/10.1109/IJCNN48605.2020.9206830.
51. Meng Y, Rajagopal M, Kuntumalla G, Toro R, Zhao H, Chang HC, Sundar S, Salapaka S, Miljkovic N, Ferreira P, Sinha S, Shao C. Multi-objective optimization of peel and shear strengths in ultrasonic metal welding using machine learning-based response surface methodology. Math Biosci Eng. 2020;17(6):7411–27. https://doi.org/10.3934/mbe.2020379.
52. Miettinen KM. Nonlinear multiobjective optimization. In: International series in operations research & management science, vol. 12, 1 edn. Berlin: Springer; 1998.
53. Mitra R, Bandyopadhyay S. Multimitar: a novel multi objective optimization based mirna-target prediction method. PLoS One. 2011;6(9):1–13. https://doi.org/10.1371/journal.pone.0024583.
54. Moctezuma LA, Molinas M. Eeg channel-selection method for epileptic-seizure classification based on multi-objective optimization. Front Neurosci. 2020;14:593. https://doi.org/10.3389/fnins.2020.00593.
55. Moctezuma LA, Molinas M. Multi-objective optimization for eeg channel selection and accurate intruder detection in an eeg-based subject identification system. Sci Rep. 2020;10(1):5850. https://doi.org/10.1038/s41598-020-62712-6.
56. Morris C, Kriege NM, Bause F, Kersting K, Mutzel P, Neumann M. Tudataset: a collection of benchmark datasets for learning with graphs. In: ICML 2020 Workshop on graph representation learning and beyond (GRL+ 2020); 2020. http://www.graphlearning.io.
57. Mukherjee R, Diwekar UM. Multi-objective optimization of the teg dehydration process for btex emission mitigation using machine-learning and metaheuristic algorithms. ACS Sustain Chem Eng. 2021;9(3):1213–28. https://doi.org/10.1021/acssuschemeng.0c06951.
58. Oliveira LS, Morita M, Sabourin R. Feature selection for ensembles using the multi-objective optimization approach. In: Jin Y, editors. Multi-objective machine learning. Berlin: Springer; 2006. pp. 49–74. https://doi.org/10.1007/3-540-33019-4_3.
59. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E. Scikit-learn: machine learning in Python. J Mach Learn Res. 2011;12:2825–30.
60. Pedrycz W. Granular computing: an introduction. In: Proceedings Joint 9th IFSA world congress and 20th NAFIPS international conference, vol. 3. IEEE; 2001. pp. 1349–1354. https://doi.org/10.1109/NAFIPS.2001.943745

61. Pedrycz W, Homenda W. Building the fundamentals of granular computing: a principle of justifiable granularity. Appl Soft Comput. 2013;13(10):4209–18.
62. Pedrycz W, Skowron A, Kreinovich V. Handbook of granular computing. Oxford: Wiley; 2008.
63. Pedrycz W, Succi G, Sillitti A, Iljazi J. Data description: a general framework of information granules. Knowl-Based Syst. 2015;80:98–108.
64. Qi C, Chen Q, Sonny Kim S. Integrated and intelligent design framework for cemented paste backfill: a combination of robust machine learning modelling and multi-objective optimization. Miner Eng. 2020. https://doi.org/10.1016/j.mineng.2020.106422.
65. Qu Y, Ma Z, Clausen A, Jørgensen BN. A comprehensive review of machine learning in multi-objective optimization. In: 2021 4th International conference on big data and artificial intelligence (BDAI); 2021 (In Press).
66. Ribeiro VHA, Reynoso-Meza G. A multi-objective optimization design framework for ensemble generation. In: Proceedings of the genetic and evolutionary computation conference companion, GECCO '18. Association for Computing Machinery, New York, NY, USA; 2018. pp. 1882–1885. https://doi.org/10.1145/3205651.3208219.
67. Riesen K, 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). Springer; 2008. pp. 287–297.
68. Rizzi A, Del Vescovo G. Automatic image classification by a granular computing approach. In: 2006 16th IEEE signal processing society workshop on machine learning for signal processing; 2006. pp. 33–38. https://doi.org/10.1109/MLSP.2006.275517.
69. Rizzi A, Panella M, Frattale MFM. Adaptive resolution min-max classifiers. IEEE Trans Neural Netw. 2002;13(2):402–14. https://doi.org/10.1109/72.991426.
70. Rizzi A, Del Vescovo G, Livi L, Frattale Mascioli FM. A new granular computing approach for sequences representation and classification. In: The 2012 International joint conference on neural networks (IJCNN); 2012. pp. 1–8. https://doi.org/10.1109/IJCNN.2012.6252680.
71. Rizzi A, Possemato F, Livi L, Sebastiani A, Giuliani A, Frattale Mascioli FM. A dissimilarity-based classifier for generalized sequences by a granular computing approach. In: The 2013 International joint conference on neural networks (IJCNN); 2013. pp. 1–8. https://doi.org/10.1109/IJCNN.2013.6707041.
72. Roshan SE, Asadi S. Improvement of bagging performance for classification of imbalanced datasets using evolutionary multi-objective optimization. Eng Appl Artif Intell. 2020;87: 103319. https://doi.org/10.1016/j.engappai.2019.103319.
73. Roszkowska E. Multi-criteria decision making models by applying the topsis method to crisp and interval data. Mult Criteria Decis Making. 2011;6(1):200–30.
74. Schölkopf B, Smola AJ, Williamson RC, Bartlett PL. New support vector algorithms. Neural Comput. 2000;12(5):1207–45.
75. Senhaji K, Ramchoun H, Ettaouil M. Training feedforward neural network via multiobjective optimization model using non-smooth l1/2 regularization. Neurocomputing. 2020;410:1–11. https://doi.org/10.1016/j.neucom.2020.05.066.
76. Sessarego M, Dixon K, Rival D, Wood D. A hybrid multi-objective evolutionary algorithm for wind-turbine blade optimization. Eng Optim. 2015;47(8):1043–62. https://doi.org/10.1080/0305215X.2014.941532.
77. Shi C, Kong X, Fu D, Yu PS, Wu B. Multi-label classification based on multi-objective optimization. ACM Trans Intell Syst Technol. 2014. https://doi.org/10.1145/2505272.
78. Singh P, Dwivedi P. A novel hybrid model based on neural network and multi-objective optimization for effective load forecast. Energy. 2019;182:606–22. https://doi.org/10.1016/j.energy.2019.06.075.
79. Sinha A, Malo P, Frantsev A, Deb K. Multi-objective stackelberg game between a regulating authority and a mining company: a case study in environmental economics. In: 2013 IEEE congress on evolutionary computation; 2013. pp. 478–485. https://doi.org/10.1109/CEC.2013.6557607.
80. Su Y, Su X, Wang Q, Zhang L. A multi-objective optimization method for identification of module biomarkers for disease diagnosis. Methods. 2020. https://doi.org/10.1016/j.ymeth.2020.09.001.
81. Suttorp T, Igel C. Multi-objective optimization of support vector machines. In: Jin Y, editors. Multi-objective machine learning. Berlin: Springer; 2006. pp. 199–220. https://doi.org/10.1007/3-540-33019-4_9.
82. Tang J, Alelyani S, Liu H. Feature selection for classification: a review. In: Data classification. CRC Press; 2014. pp. 37–64. https://doi.org/10.1201/b17320.
83. Theodoridis S, Koutroumbas K. Pattern recognition. 4th ed. New York: Academic Press; 2008.
84. Vapnik VN, Chervonenkis AY. On the uniform convergence of relative frequencies of events to their probabilities. Theory Prob Appl. 1971;16(2):264–80.
85. Vapnik VN, Chervonenkis AY. On the uniform convergence of relative frequencies of events to their probabilities. In: Vovk V, Papadopoulos H, Gammerman A, editors. Measures of complexity: festschrift for alexey chervonenkis. Cham: Springer; 2015. pp. 11–30. https://doi.org/10.1007/978-3-319-21852-6_3.
86. Wang Z, Rangaiah GP. Application and analysis of methods for selecting an optimal solution from the pareto-optimal front obtained by multiobjective optimization. Ind Eng Chem Res. 2017;56(2):560–74. https://doi.org/10.1021/acs.iecr.6b03453.
87. Wang X, Pedrycz W, Gacek A, Liu X. From numeric data to information granules: a design through clustering and the principle of justifiable granularity. Knowl-Based Syst. 2016;101:100–13.
88. You J, Ampomah W, Sun Q. Development and application of a machine learning based multi-objective optimization workflow for co2-eor projects. Fuel. 2020;264: 116758. https://doi.org/10.1016/j.fuel.2019.116758.
89. Yu X, Shen Y, Guan Z, Zhang D, Tang Z, Li W. Multi-objective optimization of ann-based psa model for hydrogen purification from steam-methane reforming gas. Int J Hydrogen Energy. 2021;46(21):11740–55. https://doi.org/10.1016/j.ijhydene.2021.01.107.
90. Zadeh LA. Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic. Fuzzy Sets Syst. 1997;90(2):111–27. https://doi.org/10.1016/S0165-0114(97)00077-8.
91. Zhang J, Huang Y, Wang Y, Ma G. Multi-objective optimization of concrete mixture proportions using machine learning and metaheuristic algorithms. Constr Build Mater. 2020;253: 119208. https://doi.org/10.1016/j.conbuildmat.2020.119208.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.