Pattern Recognition 76 (2018) 175–190

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

A framework for dynamic classifier selection oriented by the classification problem difficulty

André L. Brun[a], Alceu S. Britto Jr.[a,b,*], Luiz S. Oliveira[c], Fabricio Enembreck[a], Robert Sabourin[d]

[a] Pontifícia Universidade Católica do Paraná (PUCPR), Curitiba, PR, Brazil
[b] Universidade Estadual de Ponta Grossa (UEPG), Ponta Grossa, PR, Brazil
[c] Universidade Federal do Paraná (UFPR), Curitiba, PR, Brazil
[d] École de technologie supérieure (ÉTS), Université du Québec, Montreal, QC, Canada

ARTICLE INFO

Article history:
Received 28 March 2017
Revised 26 August 2017
Accepted 30 October 2017
Available online 31 October 2017

Keywords:
Multiple classifier systems
Classifier pool generation
Dynamic classifiers selection
Classification problem difficulty

ABSTRACT

This paper describes a framework for Dynamic Classifier Selection (DCS) whose novelty resides in its use of features that address the difficulty posed by the classification problem in terms of orienting both pool generation and classifier selection. The classification difficulty is described by meta-features estimated from problem data using complexity measures. Firstly, these features are used to drive the classifier pool generation expecting a better coverage of the problem space, and then, a dynamic classifier selection based on similar features estimates the ability of the classifiers to deal with the test instance. The rationale here is to dynamically select a classifier trained on a subproblem (training subset) having a similar level of difficulty as that observed in the neighborhood of the test instance defined in a validation set. A robust experimental protocol based on 30 datasets, and considering 20 replications, has confirmed that a better understanding of the classification problem difficulty may positively impact the performance of a DCS. For the pool generation method, it was observed that in 126 of 180 experiments (70.0%) adopting the proposed pool generator allowed an improvement of the accuracy of the evaluated DCS methods. In addition, the main results from the proposed framework, in which pool generation and classifier selection are both based on problem difficulty features, are very promising. In 165 of 180 experiments (91.6%), it was also observed that the proposed DCS framework based on the problem difficulty achieved a better classification accuracy when compared to 6 well known DCS methods in the literature.

1. Introduction

Many researchers have focused on Dynamic Classifier Selection (DCS), and have produced interesting solutions. The main difference between the researchers' approaches lies in the criterion adopted in selecting the classifier(s) from the pool. Usually, this selection is based on the concept of classifier competence, which is most commonly estimated over a region of the feature space defined as the neighborhood of the test pattern on a validation set. In [1], a proposed taxonomy organizes the DCS methods taking into account the criterion applied to compute the classifiers' competence. In their view, we may organize them in two main groups: methods based on the sole competence of the classifiers in the pool, and methods in which the interaction between the classifiers is considered. Regardless of the large number of different criteria available to measure the competence of the classifiers in the pool, one common thread running through them is the use of accuracy-based competence analysis, which is carried out over the feature or decision space.

In such a context, it is known that the pool in which the classifier selection is executed also plays an important role in the DCS performance. However, little effort has been dedicated to investigating new strategies to create a pool well-suited for DCS-based methods. Diversity is always expected irrespective of whether a homogeneous or a heterogeneous pool is used. The most popular techniques for pool generation are Bagging [2], Boosting [3] and Random Subspaces (RSS) [4]. With the exception of Boosting, in which future weak classifiers focus more on the examples that previous weak classifiers misclassified, these techniques usually ma-

* Corresponding author at Pontifícia Universidade Católica do Paraná (PUCPR), Curitiba, PR, Brazil: .
E-mail addresses: abrun@ppgia.pucpr.br (A.L. Brun), alceu@ppgia.pucpr.br (A.S. Britto Jr.), lesoliveira@inf.ufpr.br (L.S. Oliveira), fabricio@ppgia.pucpr.br (F. Enembreck), robert.sabourin@etsmtl.ca (R. Sabourin).

https://doi.org/10.1016/j.patcog.2017.10.038
0031-3203/

nipulate the data for training weak and diverse classifiers in a random fashion.

To the best of our knowledge, there is no DCS method oriented by the classification problem properties. A DCS in which the pool is generated to provide a better compromise with the criterion used for classifier selection. More than simply classifier accuracy-based competence, we are talking here about the ability of each classifier in the pool to deal with a specific kind of problem. This idea is based on works that attempt to find the best learning method for a specific classification problem, taking into account its difficulty [6–8]. Similarly, if we consider the space of a classification problem as commonly composed of subproblems with different levels of difficulty, the best case scenario would be to have a well-suited classifier for each subproblem. Thus, the most promising classifier for a given test instance could be the one trained on a similar subproblem, i.e., a subproblem with a similar level of difficulty as that estimated in the neighborhood of the test instance. The neighborhood of the test instance could be used to specify the kind of subproblem to which it belongs. It would appear reasonable to believe that a classifier trained on a similar subproblem is able to deal with the given test instance. Nevertheless, in such a DCS-based method, the pool generated must be able to provide a better coverage of the problem complexity space, but the methods available in the literature are not suitable for creating classifiers covering different regions of this space.

To represent the classification problem difficulty, we may extract features from the problem data using complexity measures. It is worth noting that the complexity, or difficulty, here involves more than just the quantities of instances, classes and features. It considers intrinsic characteristics of a classification problem, which can be obtained by means of complexity measures applied on the problem data. For instance, there are measures of difficulty based on overlap between classes, on the behavior of the edges between classes, on the class spatial distribution, and so on.

Our first hypothesis is that DCS can be done based on the classification problem difficulty, i.e., by selecting a classifier trained on a subproblem showing a similar level of difficulty as that of the neighborhood of the test instance. In our previous work [9], we observed that the adoption of data complexity features in the process of evaluating the skill of each classifier, given a test instance, may contribute to improve the performance of the classifier selection process. Deviating from that work, here we propose a complete DCS framework to investigate the impact of using problem complexity information not only in the selection process, but also for pool generation. Thus, an important hypothesis is evaluated, which is related to a better compromise between pool generation and classifier selection in a DCS method. In fact, it is expected that a pool of classifiers covering the problem complexity space adequately, i.e., that is trained on data subsets that are diverse in terms of level of difficulty, may provide better classification performance for a DCS, mainly when the selection of classifiers is also based on the problem difficulty.

In summary, more than just proposing a new framework for DCS, we intend to answer the following research questions: (a) Could a pool generated considering the difficulty of the classification problem provide gains in terms of classification performance by covering the problem space better?; (b) What is the impact, in terms of accuracy, of using the classification problem difficulty to drive both pool generation and classifier selection of a DCS-based method? We answered these questions by means of an experimental protocol composed of 30 datasets of classification problems with different levels of difficulty. We compared the results obtained with 6 DCS-based methods of the literature. The experiments showed that the strategy of generating and selecting classifiers based on the problem difficulty is very promising. The proposed DCS provides a better compromise between pool generation and classifier selection processes. In addition, similar experiments have shown that the proposed pool generation has a positive impact on the performance of DCS methods.

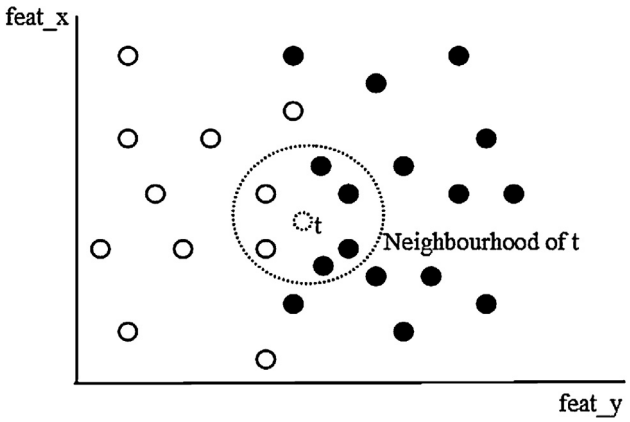

Fig. 1. Concept of competence estimated in a local region of feature space, defined as the neighborhood of the test instance in a validation set.

The remaining of this manuscript is divided into 6 sections. The Section 2 presents the main related works. Section 3 summarizes some basic concepts and definitions needed to understand the proposed DCS framework. Section 4 describes the proposed framework, detailing its generation and selection phases, while Section 5 presents the experimental protocol and corresponding results. Finally, Section 6 presents the conclusion and future work directions.

2. Related works

Various methods for dynamic selection of classifiers are available in the literature. Basically, the difference between them is at the level of the criterion used to define the competence of the classifiers for each test instance in the selection process. Fig. 1 illustrates the concept of competence estimation. A local region of the feature space, usually represented by the neighborhood of the test instance in a validation set, is used to estimate the criterion adopted.

It is common to find competence measures based on accuracy (overall or class-based) [10,11], ranking of classifiers [12], probabilistic measures [11,13], behavior of the classifiers computed on their output profiles [14], Oracle-based criteria [15,16], etc. In addition, some measures take into account group-based information such as ambiguity [19], diversity [17,18], or data handling theory like in [20].

We selected six of the preceding important contributions to the literature to implement in our experimental protocol, with 4 being single classifier selection methods, and 2 being ensemble selection methods. From [10], we have implemented 2 methods, the Overall Local Accuracy (OLA) and the Local Class Accuracy (LCA). The first calculates the classifier competence as the percentage of the correct recognition of the neighbors of the test instance in the feature space, while the second computes it as the percentage of correct classifications within the test instance neighborhood, but considering only those examples where the classifier has given the same class as the one it gives for the test instance. The other 2 single classifier selection methods were implemented from [13], the A Priori (APRI) and A Posteriori (APOS) methods. In the APRI method, a classifier is selected based on its class posterior probability estimated in the neighborhood of the test instance. This probability is weighted by the Euclidian distance between the test instance and each neighbor. Unlike in the APRI, the APOS method takes into account the class assigned by the classifier to the test instance.

The 2 ensemble selection methods are based on [16], and are named KNORA-Eliminate (KE) and KNORA-Union (KU). Both of them (K-NearestOracles) are methods that have produced very promising results by considering the neighborhood of the test pattern in a validation set as an "Oracle" which offer advice regarding the most promising classifiers to be selected. In fact, the classifiers that recognize the k-nearest neighbors are selected according to four different strategies. The two most promising in the literature are KE and KU. In the KE strategy, a classifier is selected to be part of the ensemble only if it can recognize all the neighbors of the test instance, while in the KU strategy, a classifier is selected if it recognizes at least one neighbor of the test instance. It should be noted that all these methods are described in more detail in [1].

Pool generation plays an important role in a DCS-based method, and diversity is always expected irrespective of whether the pool is homogeneous or heterogeneous. A homogeneous pool is built by using the same inducer, but its elements are trained on different subsets of data. The most popular techniques for generating such pools are Bagging [2], Boosting [3] and Random Subspaces (RSS) [4]. The first consists in randomly, and with replacement, choosing instances from the training set to form each data subset. The second one adopts a similar strategy, but considers weights for each instance at the time of the draw. The idea is to form stronger sets by defining higher weights for the most difficult instances. In the RSS technique, all the instances from the training dataset are kept, but only a subset of features are randomly selected to train each classifier to compose the pool. For the heterogeneous techniques, the idea is essentially to create classifiers supported by different concepts by varying the inducer or even its parameters.

The novelty of the proposed DCS framework lies in its consideration of information related to the problem difficulty, not only at the selection phase, but also to generate the pool of classifiers. In fact, since we expect to find a similar subproblem to use the corresponding classifier for the test instance, it could be interesting to have a pool characterized by problem difficulty diversity; in other words, generating classifiers trained on subproblems representing different levels of difficulty. The proposed framework shares the divide-and-conquer principle already explored by researchers dedicated to mixture of experts, such as the authors in [5]. In their work, the idea was to divide the problem space between the experts represented by few networks, which are supervised by a gating network that decides how to combine them for a given test instance. The subtasks in their work, here are represented using different data subsets organized taking into account complexity features, while the competence of an expert is determined by combining data complexity and accuracy based features. The next section presents some important concepts and definitions related to the main ideas that support the proposed framework.

3. Basic concepts and definitions

To estimate the level of difficulty of a classification problem, we may apply measures of complexity direct on the problem data, independent of the classifier choice. For that, we need only a set of training data consisting of points in a d-dimensional real space R^d, in which each instance is associated with a class label. Using this training set, we can compute complexity measures to define the problem difficulty. Some practical measures of data complexity with regard to classification were introduced in [6], and extended in [7,21]. They have been used for classifier evaluation [22], and recently, for meta-learning [24]. The complexity measures were classified into three categories in [6,7], as follows: (a) classes overlapping; (b) classes separability; and (c) classes geometry, topology and density. We have pre-selected one from each category to describe the problem difficulty in the proposed framework. The reason was to use low correlated measures which are supported by different concepts. By means of preliminary experiments conducted on 13 UCI databases, we estimated the correlation among the 14 measures implemented in the DCoL library [25]. The three measures described in this section presented low Pearson correlation among them, suggesting that they may work in a complementary fashion.

- $F1$ (Fisher's Discriminant Ratio): this measure belongs to the first category, and it measures how separable are two classes considering a particular feature. Let us to consider a given feature space, and that μ_1, μ_2, σ_1, and σ_2 are the means and standard deviations for classes 1 and 2, respectively. $F1$ is computed for each feature as denoted in Eq. (1). The final value for $F1$ corresponds to the largest over the whole set of features. The larger is the $F1$ value, the easier is to separate the classes.

$$F1 = \frac{(\mu_1 - \mu_2)^2}{{\sigma_1}^2 - {\sigma_2}^2} \tag{1}$$

- $N2$ (the ratio of intra/inter class nearest neighbor distance): representing the second category, this measure estimates the separability of two classes taking into account an analyse of the border between them. To this end, it considers the distance of each sample of the problem to its nearest neighbor inside and outside of the same class. $N2$ can be computed as denoted by Eq. (2), in which $\delta(N_1^{=}(x_i), x_i)$ corresponds to the Euclidean distance between the sample i and its nearest neighbor inside the same class, $\delta(N_1^{\neq}(x_i), x_i)$ is the distance from the sample i to the nearest neighbor of different class, and n is the number of samples. The smaller is the $N2$ value, the easier is to separate the classes.

$$N2 = \frac{\sum_{i=1}^{n} \delta(N_1^{=}(x_i), x_i)}{\sum_{i=1}^{n} \delta(N_1^{\neq}(x_i), x_i)} \tag{2}$$

- $N4$ (the nonlinearity of the one-nearest neighbor classifier): as a representative of the third category was selected the $N4$ measure. It corresponds to the error rate of the 1NN classifier on a test set created by the linear interpolation between randomly drawn pairs of samples from the same class. The smaller is the $N4$ value, the easier is the problem.

Considering the preceding concepts, we can now define the problem complexity space. Given a standard training set D, and a strategy to generate M new training subsets DS_i, with each one containing just a percentage of samples obtained from D, by sampling uniformly and with replacement, for each DS_i, we can estimate the difficulty by means of complexity measures. Fig. 2 illustrates the complexity space of a given classification problem represented by two complexity measures ($F1$ and $N2$). Each element in that space corresponds to a subproblem (data subset, DS_i) with its own level of difficulty. It should be mentioned that the strategy used to generate the data subsets from D plays an important role in the problem space representation. For a clearer explanation, consider the neighborhood of a given test instance projected in the same space to find a similar subproblem. In the example of Fig. 2, the most similar subproblem is that represented by DS_i. We expect that a classifier trained on DS_i, may present the necessary skills to deal with the test instance t.

4. Proposed framework

Fig. 3 presents an overview of the proposed DCS framework. The training set of a given classification problem is the input of a sampling process that initially generates M data subsets $(DS_1^1, DS_2^1, \ldots, DS_M^1)$. Each DS_i^p corresponds to the ith individual of the pth population of a genetic algorithm (GA) which is oriented by accuracy and features related to the classification problem difficulty estimated using complexity measures. The idea is to organize

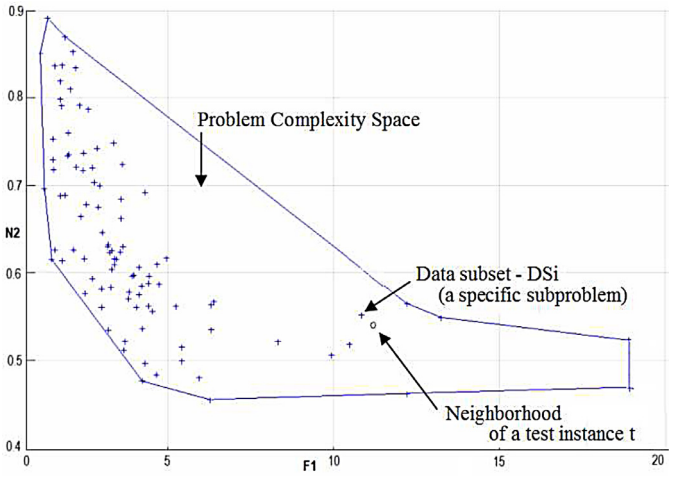

Fig. 2. Problem complexity space using complexity measures *F*1 and *N*2. Each point in the space is a training data subset (DS_i) of a given problem.

subsets of data with different levels of difficulty to train the classifiers to compose the pool.

The output of the first stage is the pool of classifiers $(C_1^N, C_2^N, \ldots, C_M^N)$, the data subset used to train each member of the pool $(DS_1^N, DS_2^N, \ldots, DS_M^N)$, and their corresponding complexity signatures $(sigDS_1^N, sigDS_2^N, \ldots, sigDS_M^N)$; in other words, a set of features describing the difficulty of each data subset DS_i^N.

During the classifier selection stage, given a test instance t, a vector containing three meta-features (f_{1i}, f_{2i}, f_{3i}) is estimated considering the complexity signature $(sigDS_i^N)$ of data subset DS_i^N used for training classifier C_i^N and the neighborhood of t in a validation set. The similarity between the complexity of the test instance neighborhood and the complexity signature of the training subset of each classifier is combined with accuracy information to estimate the competence of each classifier. A detailed description of the pool generation is presented in Section 4.1, while the particularities of the selection process are described in Section 4.2.

4.1. Pool generation

A Genetic Algorithm (GA) is used in order to evolve an initial pool of classifiers, taking into account data complexity features estimated on their training subsets, combined with their corresponding accuracies. The idea is inspired on the works [21,22], where a GA successfully allowed the generation of data subsets for a classification problem covering the problem complexity space better. However, here, we use the GA to evolve subsets of data from the original training set. The fitness function uses the difference in terms of difficulty among the generated subsets, combined with the accuracy of the corresponding trained classifiers. The base inducer is a parameter of the proposed method. Fig. 4 shows the chromosome definition of the GA developed.

Each training subset (DS_i^p) is an individual within the *p*th population. In terms of individual genotype, the genes of the chromosomes correspond to the instances used for training each classifier. The values of the *F*1 and *N*2 complexity measures, the accuracy (*Acc*) of the classifier trained on (DS_i^p), the average dispersion (*Disp*) estimated in a pairwise manner between DS_i^p and all other subsets in the population, plus the fitness values, constitute the individual phenotype.

As can be seen, we used just two of the three complexity measures pre-selected for our framework. This is because after analyzing different combinations of these three measures, we observed the best results for pool generation when only F1 and N2 were considered. In fact, it can be explained since the measure N4 uses the error rate of the 1NN classifier to estimate the problem complexity, and our fitness function already has a component based on accuracy as shown in Eq. (4). Thus, in the current version of the proposed GA method, each individual difficulty or complexity is estimated based on the *F*1 (Maximum Fisher's discriminant ratio) and *N*2 (the ratio of average intra / inter class nearest neighbor distance) metrics. They are used to compute the average dispersion of each individual in the population $(Disp_{DS_i^p})$, as denoted by Eq. (3).

$$Disp_{DS_i^p} = \frac{\sum_{j=1}^{M} \sqrt{\sum_{k=1}^{nc} (x_{i,k} - x_{j,k})^2}}{M-1} \tag{3}$$

where M corresponds to the number of classifiers in the pool, nc indicates the number of complexity measures adopted in the process, while $x_{i,k}$ and $x_{j,k}$ correspond to the value of the kth complexity measure for the elements i and j, respectively.

The dispersion in terms of complexity corresponds to the average distance of the individuals in the population (training subsets). The idea is to have training subsets better distributed over the problem complexity space. The accuracy $Acc_{C_i^p}$ of each classifier trained on the generated data subsets is combined with the corre-

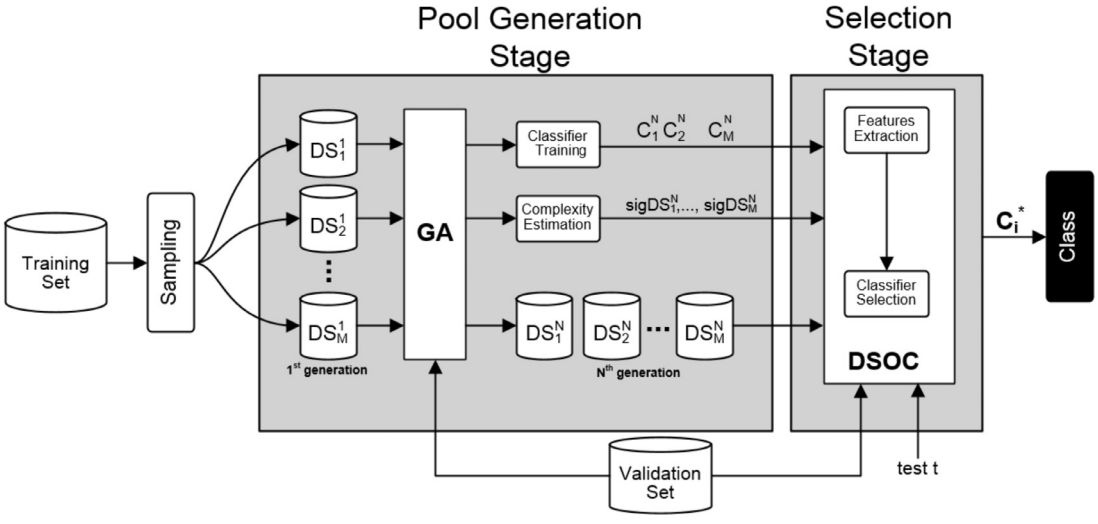

Fig. 3. An overview of the proposed DCS framework, presenting the pool generation and classifier selection stages.

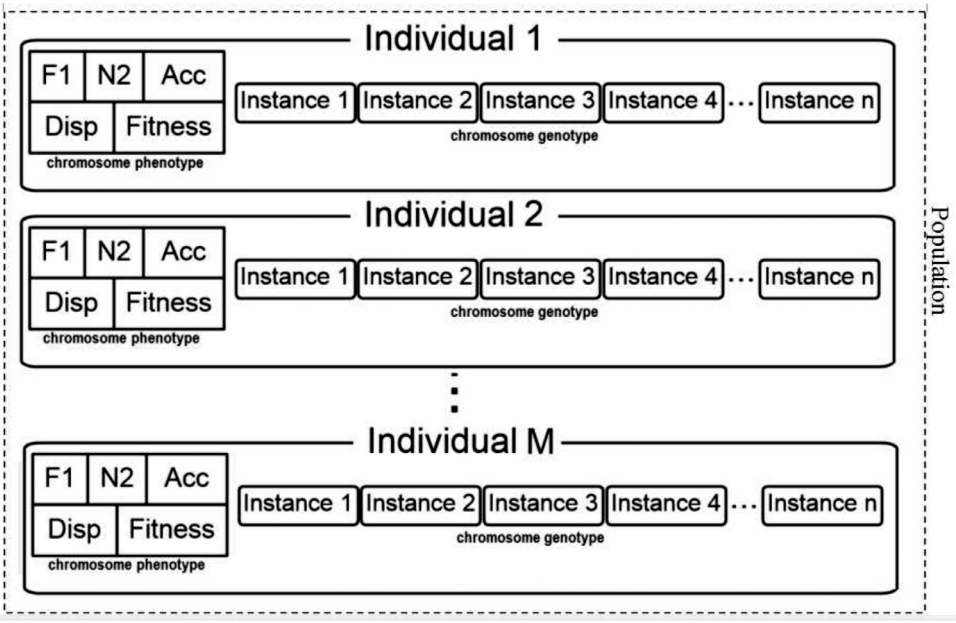

Fig. 4. Structure of the GA adopted.

sponding dispersion value $Disp_{DS_i^p}$, with a validation set used for this purpose. Thus, a classifier having higher accuracy and trained on data subset farther from concentration areas in the complexity space will have a greater fitness value. Therefore, our method favors those individuals that are accurate, but exploring the problem complexity space, as defined in the fitness denoted by the Eq. (4).

$$Fit_{DS_i^p} = Acc_{C_i^p} + Disp'_{DS_i^p} \tag{4}$$

where $Disp'_{DS_i^p}$ corresponds to the normalized metric of $Disp_{DS_i^p}$ using the MinMax method, as denoted in Eq. (5).

$$Disp'_{DS_i^p} = \frac{Disp_{DS_i^p} - Disp_{min}}{Disp_{max} - Disp_{min}} \tag{5}$$

The steps of the GA process are described in the Algorithm 1. In the first GA generation, the population is composed of M individuals (DS_1^1, DS_2^1, ... , DS_M^1), corresponding to bags of data for training potential classifiers, which can be generated using any sampling technique applied in the initial problem training dataset. After determining the fitness of each individual (performed on lines 12–14 of the Algorithm 1) an elitism process is conducted, preventing the best elements might change during crossover and mutation phases (conducted on lines 16–19). These elements are taken directly to the next generation, after which the crossover step is performed (lines 21–23). The selection process is carried out at random (roulette), and as a result, the higher the fitness of the elements, the greater their chances of being chosen to propagate their genes to the next generations. Such a selection strategy was defined after we evaluate three different selection strategies (roulette wheel, rank, and tournament). The selection based on roulette wheel has shown to be more efficient in terms of early generation and time processing. Similar behavior has been observed in [23], where the authors compare these selection strategies for timetabling problem. A two-point crossover was adopted, which are randomly selected. This is an attempt to be more aggressive than consider just one point. However, this choice was based on the evaluation of three different strategies: one, two and three-point crossover. The configuration based on two points has shown the best results. The idea is that instances (samples of the datasets) should be exchanged between the individuals, but however, maintaining the balance between classes. To ensure that there is no change in the proportions, the instances are sorted by class, and so when the exchange of "segments" between two parents occurs, there will be no change in the number of instances of each group. In our implementation, despite the randomness in the two-point selection, we impose a restriction requiring that each crossover point be positioned within different classes. Thus, the process is more aggressive, ensuring that elements from different classes are changed. This process is illustrated in Fig. 5, where each number corresponds to an instance that composes the individuals i and j. Different values indicates different classes.

After the new elements are generated through the crossover, they are subjected to a mutation process (line 24 of the algorithm). In our scenario, each instance corresponds to a gene. When it is submitted to mutation, this instance is replaced by another one of the same class, thus maintaining the class balance. This process is represented in Fig. 6.

The choice by the new gene is made randomly. Initially, a random individual j, different from i, is selected and, within this, a random instance that belongs to the same class is chosen. The last phase consists of the removal of duplicate instances (lines 27–29 of the algorithm). When a duplicate is present, it is replaced, performing an approach similar to mutation. The process is repeated until each data set has no further duplicates. The main output of this DCS phase is the pool of generated classifiers. However, we also have the subset of data used for training each classifier, as well as the complexity signature of each data subset. These additional outputs are described in detail in the next section, since they are used in the proposed dynamic classifier selection.

4.2. Dynamic classifier selection

Our dynamic selection method, named DSOC (Dynamic Selection Based on Complexity) combines accuracy with information related to the classification problem difficulty. The main assumption is that the most promising classifier for the test instance was trained on a subset of data presenting a similar level of difficulty as that estimated in the test neighborhood.

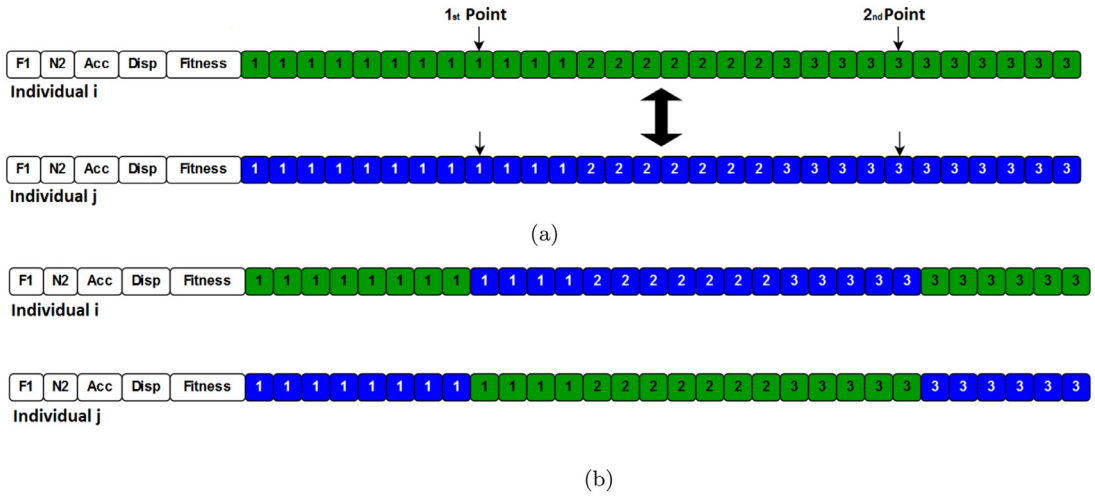

Fig. 5. Crossover process: (a) The two-point selection positioned in two different classes; (b) Swap of the segments between parents *i* and *j*.

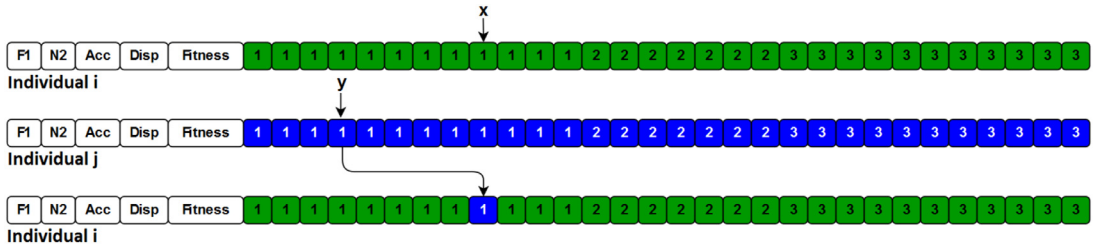

Fig. 6. Mutation process: the selected instance is swapped by that randomly chosen in a different individual, necessarily belonging to the same class.

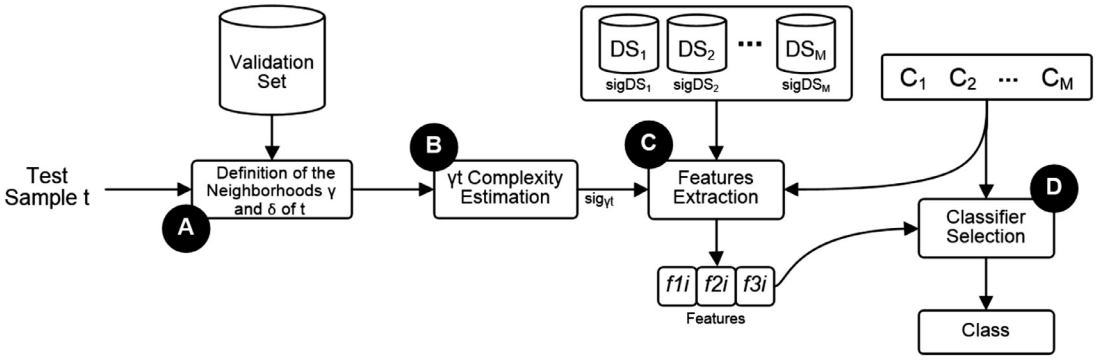

Fig. 7. DSOC - Dynamic Classifier Selection based on Complexity.

The Algorithm 2 presents each step of the DSOC method, while a complete description is presented in [9]. Here, we emphasize its main aspects using the Fig. 7 in which is possible to observe the complete method which is composed of 4 steps (A, B, C, and D). In the step (*A*), we define the neighborhoods γ and δ of the test instance t in a validation set (Algorithm 2, lines 5 and 6), which differ in terms of size, L and K, respectively. Then, in the step (*B*), sig_{γ_t} is estimated as the complexity signature of γ_t based on the complexity measures *F*1, *N*2 and *N*4 (Algorithm 2, line 7). The similarity between sig_{γ_t} and the complexity signature sig_{DS_i} of each training set DS_i is used to compute the meta-feature f_{1i} (step *C*), which represents the similarity in terms of difficulty between the neighborhood of t and each training set (DS_i) used for learning the pool of classifiers. Such a similarity is computed considering the class α predicted by the classifier C_i, since all complexity measures are usually computed in a one-against-all fashion, showing how difficult is to classify each class of the given problem. The second meta-feature f_{2i} (step *C*) considers the distance of t to the centroid of the predicted class (α) in each training set DS_i. Since distinct data subsets with different distributions in the feature space may present similar complexity values, we have defined an additional criterion to better describe the complexity of each data subset under analysis, i.e. a criterion to distinguish them. This is the purpose of the meta-feature f_{2i}. Although not defined in the literature as a measure of complexity, we may consider f_{2i} as a measure to indicate how "difficult" is for a specific data subset to provide a classifier suitable to classify the test sample under analysis. Finally, the meta-feature f_{3i} is computed in the step (*C*), which is related to the local class accuracy of each classifier estimated on δ_t considering the predicted class (α). These meta-features are computed for each classifier in the pool (Algorithm 2, lines 10–12). In the last step (*D*), the computed meta-features are combined (Algorithm 2, line 14). The best combination strategy was the sum as presented in Eq. (6).

$$Comb\_{C_i} = (1 - f'_{1i}) + (1 - f'_{2i}) + f_{3i} \tag{6}$$

Algorithm 1: Classifier pool generation based on accuracy and complexity features.

Input: training set Tr; validation set Va; number of classifiers M; size of bags Ba; size of elitism El; the base inducer BI

Output: the final pool C of M classifiers; the complexity signature of all elements in DS; the final M bags DS

1 $DS = \{\}$;
2 **for** $i \leftarrow 1$ **to** M **do**
3 Generate bag DS_i^1 with Ba size based on Tr;
4 $DS = DS \cup DS_i^1$;
5 C_i^1 = Train BI on DS_i^1 bag;
6 Compute the difficulty signature of DS_i^1;
7 **end**
8 **for** $p \leftarrow 1$ **to** $NumGenerations$ **do**
9 $DS_{temp} = \{\}$;
10 $E = \{\}$;
11 **for** $i \leftarrow 1$ **to** M **do**
12 Calculate the average distance $Disp_{DS_i^p}$;
13 Compute the accuracy of C_i^p on Va;
14 Compute the fitness $Fit_{C_i^p}$;
15 **end**
16 **for** $i \leftarrow 1$ **to** El **do**
17 Select the i-th best individual $DS_i^p \ni E$;
18 $E = E \cup DS_i^p$;
19 **end**
20 **while** $size(DS_{temp}) < (M - El)$ **do**
21 Select parents DS_{p1} and DS_{p2};
22 DS_{new1} = two-point crossover of DS_{p1} and DS_{p2};
23 DS_{new2} = two-point crossover of DS_{p1} and DS_{p2};
24 Apply mutation on DS_{new1} and DS_{new2};
25 $DS_{temp} = DS_{temp} \cup DS_{new1} \cup DS_{new2}$;
26 **end**
27 **for** *each bag* $DS_i^p \in DS_{temp}$ **do**
28 Remove duplicated genes;
29 **end**
30 $DS = DS_{temp} \cup E$;
31 **for** $i \leftarrow 1$ **to** M **do**
32 C_i^{p+1} = Train BI on DS_i^{p+1} bag;
33 Compute the complexity signature of DS_i^{p+1};
34 **end**
35 **end**

Algorithm 2: DSOC - DS on complexity.

Input: the pool C of M classifiers; training, validation and testing sets, Tr, Va and Te; and the neighborhood sizes L and K

Output: C^*, the most promising classifier for each testing sample t in Te

1 **for** *each classifier* C_i *in the pool* **do**
2 Compute the complexity signature sig_{DS_i} from data subset DS_i;
3 **end**
4 **for** *each test* t *in Te* **do**
5 Find the γ_t as the L-nearest-neighboors of t in Va;
6 Find the δ_t as the K-nearest-neighboors of t in Va;
7 Compute the complexity signature of γ_t;
8 **for** *each classifier* C_i *in the pool* **do**
9 $\alpha = C_i(t)$;
10 Compute meta-feature f_{1i} using γ_t, α and sig_{DS_i};
11 Compute meta-feature f_{2i} using DS_i and α;
12 Compute meta-feature f_{3i} using δ_t and α;
13 Normalize f_{1i} and f_{2i};
14 $Comp_{-C_i} = (1 - f'_{1i}) + (1 - f'_{2i}) + f_{3i}$;
15 **end**
16 $C^* = argmax(Comp_{-C_i})$;
17 Use the classifier C^* to classify t;
18 **end**

where f'_{1i} and f'_{2i} correspond to the normalized f_{1i} and f_{2i}, respectively. They were normalized using the MinMax scaling (Algorithm 2, line 13). Finally, the best classifier C^* is selected as described in Eq. (7) (Algorithm 2, line 16).

$$C^* = argmax(Comb_{-C_i}) \tag{7}$$

It is important to mention that to define this combination strategy we have evaluated each meta-feature individually, and also the different possibilities of combining them (using product and sum rules). The most promising results were achieved when the three meta-features were combined using sum rule. In addition, we have also investigated different weights for each meta-feature. However, the results were not promising.

5. Experiments

The experimental protocol used to evaluate the proposed framework considers 30 datasets previously used in our research group [9,30,31]. 28 datasets come from different repositories:

Table 1
Datasets, their main features and corresponding repositories.

Datasets	Repository	# of instances	# of features	# of classes
Adult	UCI	690	14	2
Banana	PRTools	2000	2	2
Blood	UCI	748	4	2
CTG	UCI	2126	21	3
Diabetes	UCI	766	8	2
Ecoli	UCI	336	7	8
Faults	UCI	1941	27	7
German	STATLOG	1000	24	2
Glass	UCI	214	9	6
Haberman	UCI	306	3	2
Heart	STATLOG	270	13	2
ILPD	UCI	583	10	2
Image	UCI	2310	19	7
Ionosphere	UCI	350	34	2
Laryngeal1	LKC	213	16	2
Laryngeal3	LKC	353	16	3
Lithuanian	PRTools	2000	2	2
Liver	UCI	345	6	2
Magic	KEEL	19,020	10	2
Mammo	KEEL	830	5	2
Monk	KEEL	432	6	2
Phoneme	ELENA	5404	5	2
Sonar	UCI	208	60	2
Thyroid	LKC	692	16	2
Vehicle	STATLOG	847	18	4
Vertebral	UCI	300	6	2
WBC	UCI	569	30	2
WDVG	UCI	5000	21	3
Weaning	LKC	302	17	2
Wine	UCI	178	13	3

UCI machine learning repository [26], KEEL (Knowledge Extraction based on Evolutionary Learning) repository [27], Ludmila Kuncheva Collection [28], and the STATLOG project [29]. The other 2 datasets were artificially generated with the Matlab PRTools. These datasets were selected taking into account classification problems with different levels of difficulty. In addition, we have considered problems composed of numeric attributes and no missing data. Thus,

Table 2
Comparison of the proposed GA-based method and the Bagging technique both used as pool generators for 4 different DCS methods (OLA, LCA, APRI and APOS). The average and corresponding standard deviations of 20 replications with the best results in boldface.

	OLA		LCA		APRI		APOS	
Dataset	Bagging	GA	Bagging	GA	Bagging	GA	Bagging	GA
Adult	84.04 (2.87)	**84.77 (2.80)**	83.26 (2.52)	**83.52 (2.87)**	84.01 (2.34)	**85.73 (2.78)***	83.55 (2.60)	**85.20 (2.91)***
Banana	**85.03 (1.73)**	84.87 (1.68)	**84.88 (1.78)***	84.68 (1.79)	84.52 (1.64)	**84.57 (1.87)**	83.82 (1.69)	**83.83 (1.76)**
Blood	**76.12 (0.26)**	**76.12 (0.26)**	**75.86 (1.23)**	**76.12 (0.26)**	**76.12 (0.26)**	**76.12 (0.26)**	**76.12 (0.26)**	**76.12 (0.26)**
CTG	**86.99 (0.89)**	86.70 (1.04)	80.94 (0.95)	**81.33 (0.82)***	86.45 (1.38)	**86.76 (0.76)**	86.28 (1.29)	**86.30 (0.85)**
Diabetes	64.92 (2.85)	**65.00 (2.27)**	62.37 (2.64)	**63.75 (2.89)***	**65.00 (2.03)**	64.51 (2.46)	64.38 (2.44)	**64.41 (2.76)**
Ecoli	**64.94 (4.74)**	62.38 (3.09)	52.56 (5.52)	**52.72 (5.91)**	64.05 (5.05)	**64.46 (4.83)**	**63.39 (4.00)**	62.38 (3.94)
Faults	**58.35 (2.31)***	56.82 (2.99)	37.96 (10.5)	**38.84 (10.1)***	**56.08 (2.15)**	55.70 (3.42)	54.63 (2.25)	**54.73 (3.66)**
German	71.90 (2.35)	**73.48 (2.42)***	64.22 (3.31)	**65.88 (2.98)***	71.74 (2.54)	**72.80 (3.08)**	69.80 (2.39)	**71.52 (2.85)**
Glass	**53.77 (4.79)**	51.89 (7.46)	24.15 (9.04)	**25.47 (10.2)**	47.92 (7.41)	**53.11 (6.09)**	24.72 (19.5)	**30.75 (16.9)**
Haberman	73.68 (0.59)	**74.14 (0.75)**	73.62 (0.88)	**74.41 (1.74)***	73.82 (0.39)	**74.41 (1.35)**	73.68 (0.59)	**73.29 (3.00)**
Heart	78.58 (4.04)	**80.37 (4.12)**	70.15 (4.58)	**72.84 (5.31)***	78.81 (4.02)	**80.67 (3.73)**	72.24 (8.64)	**77.31 (4.10)***
ILPD	**69.72 (3.38)**	68.62 (2.91)	61.76 (3.67)	**63.79 (3.82)***	69.24 (3.22)	**70.34 (2.97)**	67.52 (4.16)	**68.41 (3.68)**
Image	**42.03 (2.30)**	41.25 (2.12)	32.18 (4.11)	**32.82 (3.87)***	40.68 (1.82)	**41.33 (1.99)**	40.29 (2.35)	**41.42 (2.34)**
Ionosphere	80.91 (3.88)	**81.48 (4.56)**	69.72 (4.21)	**72.10 (4.14)***	80.63 (4.72)	**80.72 (4.14)**	77.84 (4.62)	**81.02 (4.36)***
Laryngeal1	**80.09 (4.40)**	78.87 (5.08)	70.94 (6.24)	**73.49 (5.04)***	**81.13 (5.57)**	80.00 (3.89)	**78.11 (6.92)**	77.08 (5.11)
Laryngeal3	66.02 (4.19)	**67.22 (4.90)**	55.34 (5.77)	**58.81 (6.25)***	66.02 (4.70)	**66.14 (3.23)**	62.84 (6.18)	**63.92 (4.43)**
Lithuanian	**68.30 (2.49)**	67.06 (3.19)	**70.38 (1.84)**	69.03 (2.40)	67.06 (2.42)	**67.49 (3.21)**	64.50 (3.36)	**65.15 (3.16)**
Liver	**60.99 (3.63)**	60.17 (3.14)	50.06 (5.30)	**51.34 (5.15)**	**58.90 (5.51)**	57.38 (4.96)	**55.17 (5.47)**	54.24 (8.20)
Magic	**79.22 (0.70)**	79.00 (0.56)	78.17 (0.55)	**78.23 (0.60)**	78.73 (0.81)	**78.78 (0.53)**	**78.59 (0.83)**	78.53 (0.63)
Mammo	79.78 (2.29)	**80.56 (2.96)**	76.23 (3.11)	**77.97 (3.08)***	80.10 (3.17)	**80.51 (3.44)**	79.08 (2.74)	**79.15 (3.44)**
Monk	81.99 (3.56)	**82.59 (3.22)**	69.26 (3.98)	**72.73 (3.81)***	79.58 (4.07)	**81.48 (4.86)***	66.76 (12.8)	**68.19 (14.5)**
Phoneme	77.18 (1.25)	**77.22 (0.82)**	76.54 (0.96)	**76.77 (0.86)**	76.17 (1.47)	**77.14 (0.98)***	76.08 (1.55)	**77.05 (0.96)***
Sonar	**61.73 (6.32)**	60.29 (5.38)	**46.83 (7.28)**	45.96 (5.73)	58.65 (6.10)	**61.44 (5.94)**	41.63 (22.5)	**45.96 (20.8)**
Thyroid	93.67 (1.25)	**93.73 (1.45)**	91.88 (1.97)	**91.99 (1.71)**	93.29 (1.71)	**93.41 (1.92)**	**92.51 (2.79)**	92.02 (1.97)
Vehicle	32.16 (3.87)	**32.44 (2.65)**	**24.53 (4.66)**	23.96 (4.74)	30.28 (3.92)	**32.04 (3.33)**	29.55 (3.85)	**31.11 (4.01)**
Vertebral	**81.53 (4.35)**	80.73 (4.70)	71.67 (5.13)	**73.20 (5.20)**	**82.53 (3.32)**	81.67 (5.07)	**78.00 (6.06)**	77.87 (8.13)
WBC	77.36 (14.6)	**77.57 (13.5)**	84.86 (3.70)	**84.89 (3.25)**	77.75 (14.1)	**79.79 (12.18)**	70.32 (17.3)	**79.01 (10.6)**
WDVG	79.92 (1.13)	**80.24 (0.88)**	67.81 (3.77)	**69.91 (3.41)***	79.18 (1.21)	**80.08 (1.19)***	78.37 (1.35)	**79.49 (1.33)**
Weaning	76.87 (4.14)	**77.27 (4.43)**	60.40 (5.50)	**63.33 (6.57)***	75.67 (5.21)	**78.40 (4.19)***	59.13 (16.8)	**64.93 (19.9)**
Wine	**33.52 (3.44)**	**33.52 (2.95)**	45.23 (14.1)	**43.86 (13.3)**	34.32 (4.71)	**35.68 (7.58)**	32.61 (2.41)	**33.41 (2.88)**

Table 3
Comparison of the proposed GA-based method and the Bagging technique both used as pool generators for 2 ensemble selection methods (KE and KU), the combination of all classifiers in the pool (ALL), and the use of the best classifier in the pool (SB). The average and corresponding standard deviations of 20 replications with the best results in boldface.

	SB		ALL		KNORA-U		KNORA-E	
Dataset	Bagging	GA	Bagging	GA	Bagging	GA	Bagging	GA
Adult	84.74 (2.92)	**85.29 (2.93)**	**86.83 (2.58)**	86.66 (2.67)	83.08 (2.03)	**84.24 (1.85)***	81.92 (1.85)	**83.72 (1.92)***
Banana	**84.49 (1.43)**	84.10 (1.61)	84.16 (1.59)	**84.40 (1.76)**	**87.59 (1.24)**	87.42 (1.12)	**85.06 (1.46)**	**85.06 (1.45)**
Blood	**76.12 (0.26)**	**76.12 (0.26)**	**76.12 (0.26)**	**76.12 (0.26)**	**76.12 (0.26)**	**76.12 (0.26)**	**76.12 (0.26)**	**76.12 (0.26)**
CTG	86.62 (1.24)	**86.85 (1.54)**	**88.14 (1.12)**	**88.14 (1.14)**	84.67 (0.96)	**85.01 (0.93)***	83.95 (0.89)	**84.21 (0.89)***
Diabetes	64.95 (2.06)	**65.09 (2.38)**	64.48 (1.31)	**65.26 (2.54)**	64.48 (0.95)	**64.64 (0.87)**	64.56 (1.09)	**64.79 (1.11)**
Ecoli	63.63 (7.54)	**65.06 (4.65)**	53.39 (12.5)	**53.75 (13.7)**	54.40 (2.77)	**54.44 (2.77)**	52.02 (2.10)	**52.26 (1.99)**
Faults	55.76 (1.94)	**55.85 (3.11)**	**44.16 (11.7)**	43.98 (11.8)	43.65 (2.49)	**44.96 (2.80)***	37.41 (1.96)	**39.25 (2.27)***
German	**72.74 (2.99)**	72.28 (3.21)	**76.38 (2.28)**	75.70 (2.01)	71.88 (1.02)	**72.94 (1.36)***	71.52 (0.88)	**72.28 (1.15)***
Glass	**52.26 (7.98)**	50.00 (4.64)	48.77 (6.76)	**49.15 (8.23)**	51.13 (5.37)	**51.98 (5.15)**	45.28 (5.13)	**47.45 (4.71)***
Haberman	**74.01 (1.24)**	73.75 (0.65)	73.75 (0.29)	**73.95 (0.79)**	**73.68 (0.00)**	**73.68 (0.00)**	**73.68 (0.00)**	**73.68 (0.00)**
Heart	79.40 (4.54)	**79.93 (4.49)**	**84.33 (2.52)**	83.36 (3.48)	75.67 (4.77)	**77.16 (4.50)**	74.55 (4.21)	**76.19 (4.31)**
ILPD	69.48 (3.94)	**69.76 (3.88)**	**71.86 (3.89)**	71.83 (3.83)	**71.90 (0.61)**	**71.90 (0.75)**	**71.83 (0.45)**	**71.83 (0.45)**
Image	40.75 (3.10)	**41.05 (3.53)**	25.60 (7.83)	**25.76 (7.81)**	**36.39 (1.28)**	36.35 (1.52)	**35.86 (1.19)**	**35.88 (1.48)**
Ionosphere	81.59 (3.74)	**82.39 (4.03)**	82.61 (2.62)	**83.24 (3.33)**	**89.38 (3.54)**	87.73 (3.40)	**88.69 (3.83)**	87.50 (3.17)
Laryngeal1	**80.47 (4.40)**	80.19 (4.94)	**81.89 (4.60)**	81.42 (4.55)	73.58 (4.77)	**74.25 (3.79)**	72.08 (4.57)	**72.74 (4.07)**
Laryngeal3	67.05 (4.07)	**67.10 (3.42)**	**70.68 (2.80)**	69.94 (3.62)	**61.14 (3.48)**	61.02 (3.43)	58.64 (3.86)	**59.37 (3.25)**
Lithuanian	63.67 (2.97)	**64.43 (4.02)**	65.50 (2.26)	**66.39 (2.46)**	58.43 (0.98)	**59.87 (1.40)***	56.69 (0.80)	**58.04 (1.21)***
Liver	**63.02 (6.00)**	60.17 (5.25)	61.45 (4.05)	**62.56 (4.97)**	56.63 (5.59)	**57.09 (6.77)**	52.21 (3.51)	**55.17 (4.58)***
Magic	**78.74 (0.76)**	**78.74 (0.57)**	78.85 (0.58)	**78.87 (0.57)**	78.90 (0.58)	**78.95 (0.58)**	**78.80 (0.55)**	**78.80 (0.58)**
Mammo	80.58 (3.06)	**80.66 (3.57)**	81.76 (2.15)	**81.86 (2.41)**	78.50 (2.66)	**79.13 (2.93)**	77.85 (2.83)	**78.77 (2.76)***
Monk	**79.58 (3.98)**	79.21 (3.43)	79.86 (1.75)	**82.64 (5.43)***	**79.31 (3.26)**	78.84 (3.67)	78.15 (3.72)	**79.86 (3.08)**
Phoneme	77.08 (1.06)	**77.15 (0.86)**	76.59 (0.63)	**77.11 (0.80)***	74.77 (0.74)	**75.74 (0.81)***	74.07 (0.64)	**75.17 (0.74)***
Sonar	61.06 (5.40)	**61.63 (4.65)**	61.54 (5.27)	**64.62 (6.21)**	53.37 (1.03)	**55.00 (1.86)***	53.17 (0.92)	**53.85 (1.36)***
Thyroid	**93.61 (1.97)**	93.21 (2.86)	**96.42 (1.06)***	95.61 (1.32)	**89.22 (2.73)**	88.70 (4.11)	**88.79 (2.82)**	88.38 (4.05)
Vehicle	**31.30 (3.68)**	30.52 (4.08)	32.39 (5.21)	**33.65 (5.24)**	27.51 (1.69)	**28.22 (2.57)**	26.30 (1.23)	**26.75 (1.31)**
Vertebral	**81.33 (4.02)**	80.93 (5.64)	**83.53 (3.16)**	83.13 (4.22)	79.27 (4.47)	**80.47 (4.64)**	78.47 (3.80)	**79.67 (4.58)**
WBC	67.78 (20.2)	**81.73 (15.7)***	85.92 (3.06)	**90.18 (2.16)***	**89.15 (3.16)**	88.49 (2.96)	**88.59 (3.72)**	88.27 (3.47)
WDVG	**79.86 (1.14)**	79.54 (0.97)	81.50 (0.79)	**81.54 (1.02)**	71.07 (1.12)	**72.71 (1.15)***	69.66 (1.05)	**71.6 (1.18)***
Weaning	75.60 (5.45)	**76.60 (4.82)**	81.47 (4.56)	**83.18 (3.42)**	66.93 (4.98)	**71.07 (3.96)***	64.47 (4.45)	**68.47 (3.58)***
Wine	34.43 (6.32)	**36.59 (9.00)**	32.84 (1.13)	**38.30 (10.9)***	**32.84 (1.13)**	**32.84 (1.13)**	**32.84 (1.13)**	**32.84 (1.13)**

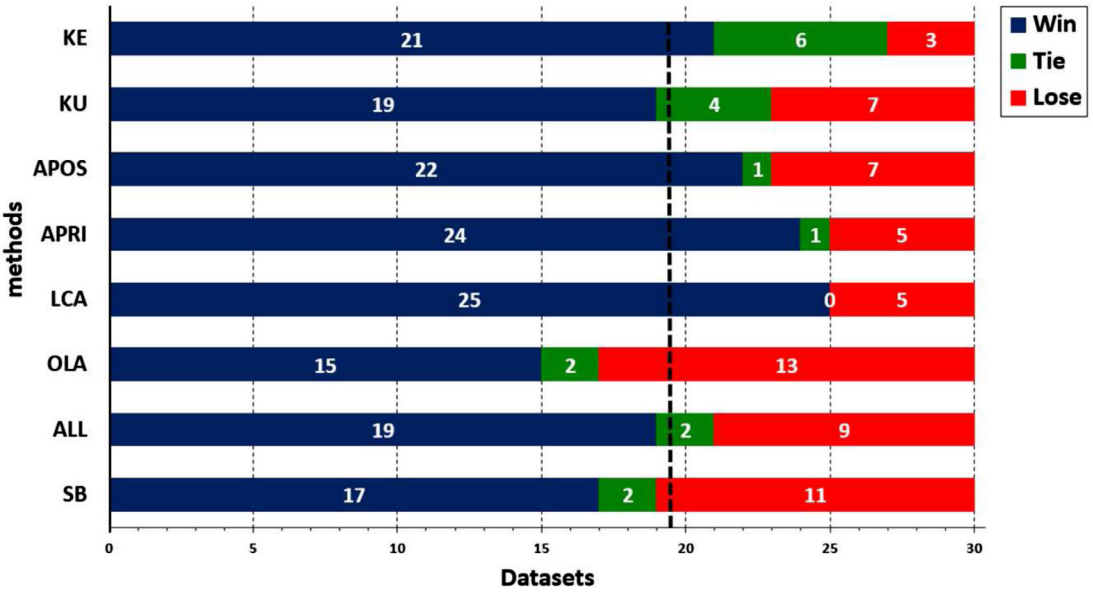

Fig. 8. Comparison between both pool generation methods (GA-based and Bagging) in terms of wins, ties and losses, considering different DCS methods (KE, KU, APOS, APRI, LCA and OLA), the combination of all classifiers in the pool (ALL) and the single best classifier in the pool (SB). In blue, green, and red, we have, respectively, the number of wins, ties and losses related to the GA-based method. The dashed line illustrates the critical value (19.5). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 4
Average dispersion in the complexity space of the generated training subsets.

	Dispersion	
Dataset	Bagging	GA
Adult	0.53 (0.06)	**1.26 (0.26)***
Banana	0.20 (0.02)	**0.25 (0.03)***
Blood	0.14 (0.01)	**0.22 (0.04)***
CTG	0.24 (0.04)	**0.25 (0.04)***
Diabetes	0.18 (0.02)	**0.28 (0.04)***
Ecoli	20.4 (4.11)	**21.1 (3.99)***
Faults	**0.74 (0.05)**	0.73 (0.06)
German	0.12 (0.02)	**0.31 (0.14)**
Glass	2.62 (1.30)	**7.9 (10.4)***
Haberman	0.17 (0.02)	**0.36 (0.08)***
Heart	0.40 (0.09)	**2.19 (1.30)***
ILPD	0.08 (0.01)	**0.12 (0.01)***
Image	12.7 (2.10)	**13.1 (1.99)**
Ionosphere	0.21 (0.03)	**0.47 (0.13)***
Laryngeal1	0.68 (0.10)	**1.59 (0.44)***
Laryngeal3	1.77 (0.25)	**1.93 (0.33)***
Lithuanian	0.17 (0.01)	**0.24 (0.03)***
Liver	0.10 (0.01)	**0.17 (0.03)***
Magic	0.03 (0.00)	**0.04 (0.00)**
Mammo	0.26 (0.03)	**0.30 (0.07)**
Monk	0.28 (0.04)	**0.60 (0.12)***
Phoneme	0.04 (0.00)	**0.06 (0.01)***
Sonar	0.28 (0.04)	**0.72 (0.20)***
Thyroid	0.83 (0.09)	**1.62 (0.26)***
Vehicle	0.37 (0.03)	**0.42 (0.04)***
Vertebral	0.30 (0.04)	**0.52 (0.13)***
WBC	0.55 (0.06)	**0.86 (0.11)***
WDVG	0.12 (0.01)	**0.13 (0.02)**
Weaning	0.38 (0.05)	**0.65 (0.13)***
Wine	1.90 (0.20)	**2.16 (0.49)***

the final set of problems has 2-class and 3-class problems showing different levels of complexity, and also one 4-class, one 6-class, two 7-class and one 8-class problem. Table 1 presents a summary with the main details of the used datasets. In addition, it is important to mention that, for each of 20 replications, these datasets were divided in a random fashion into 50% for training, 25% for validation, 25% for testing. It is worth mentioning that all parameters empirically estimated were defined using the validation set. The parameters of the DCS methods compared to our approach, basically the size of the neighborhood used to estimated the classifiers competence, were defined based on previous studies [9,16,30,31].

In the first set of experiments, we evaluated the proposed pool generation method. The pool generated for each problem was compared to that generated using the Bagging method. The second set of experiments evaluated the proposed DCS method, considering the problem difficulty properties in both phases: pool generation and classifier selection. In both sets of experiments, 6 dynamic selection (DCS) methods already established in the literature were considered, being 4 methods for selection of a single classifier (LCA, OLA, APRI and APOS), and 2 methods for selection of ensembles (KE and KU). The only parameter of these DCS methods is the size of the test instance neighborhood (K). Such a neighborhood is used to define the competence of each classifier. In our experiments, K was defined as 7, since this value had been proved to be the most appropriate in previous studies [9,16,30,31]. The same size of neighborhood ($K = 7$) was used to estimate the accuracy based meta-feature f_{3i} in the DSOC method. On the other hand, the size of the neighborhood (L) used to estimate the complexity based meta-feature (f_{1i}) (Algorithm 2, line 6) for each test instance was defined as 30. By adopting a larger neighborhood in this step, we ensure the presence of at least two distinct classes among the selected instances, making it possible to calculate the complexity measures. However, it is important to say that the adoption of large neighborhoods would mean that, for small datasets, different instances may have the same neighborhood. To drive this choice, we evaluated neighborhoods with sizes ranging from 20 to 50 (varying from 5 to 5).

5.1. Experiments on pool generation

Two pools composed of 100 perceptrons were generated for each classification problem. One of the pools was created using the Bagging method [2], and the other using the proposed GA-based method. In both pools, the size of the bags corresponded to 50% of the training set. The perceptron was used as the base classifier since it represents a weak and unstable learner. In cases of prob-

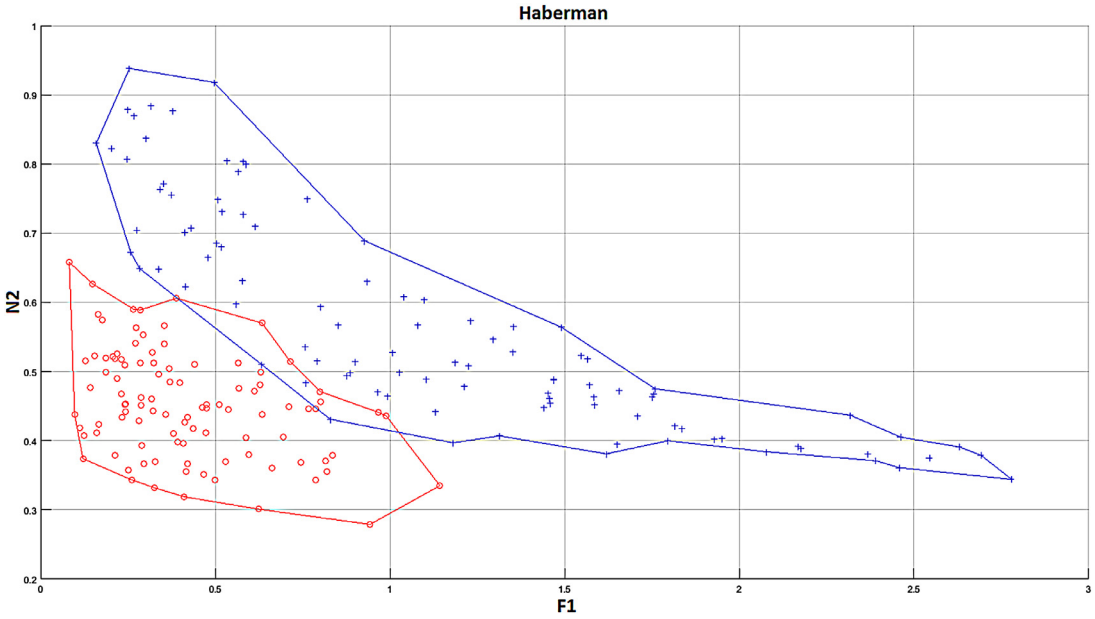

Fig. 9. Dispersion of the data subsets generated for the Haberman problem in the complexity space. The elements built by Bagging are shown in red, while those in blue show the pool obtained by the GA. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

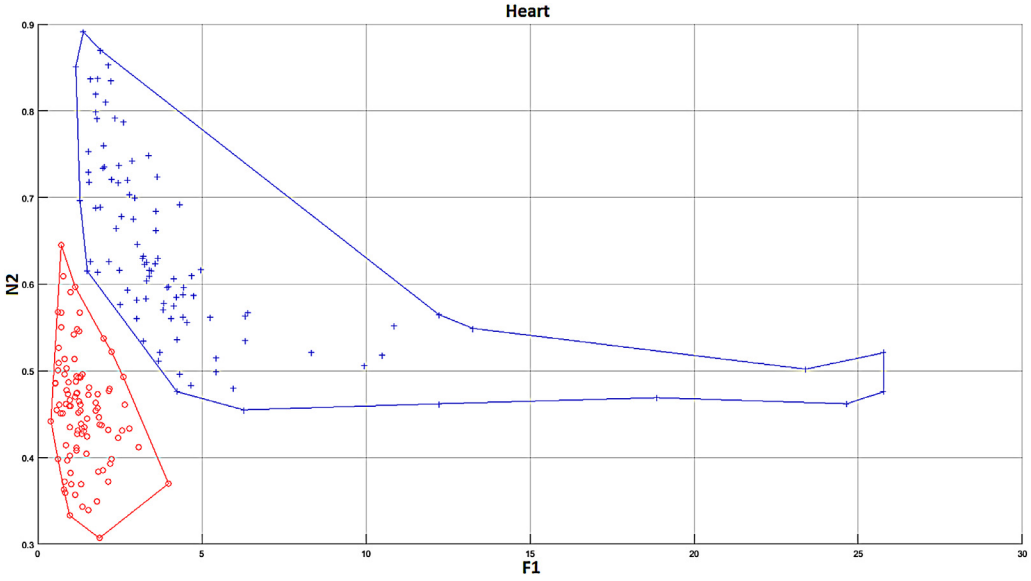

Fig. 10. Dispersion of the data subsets generated for the Heart problem in the complexity space. The elements built by Bagging are shown in red, while those in blue show the pool obtained by the GA. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

lems composed of more than 2 classes, the One-Against-All (OAA) strategy was adopted. In the GA-method, the elitism size was defined as 4 chromosomes. This value was determined empirically, once it makes possible to maintain high fitness elements without leading to premature convergence. The crossover probability was 0.8, meaning that the algorithm would evolve rather quickly, and that the process for substituting old members with new ones would be of moderate intensity, thereby limiting any chance of losing high fitness elements. The mutation rate applied was 5%, a value which can prevent the stagnation of the process, and increases the possibility of exploration of new areas of the solution space. 30 generations in all were evolved.

To analyze how adopting the proposed GA contributes to the literature, we compared its accuracy to that of 6 aforementioned DCS methods. Each method was tested using the sets generated by Bagging method and those built by the proposed GA-based method. Furthermore, we considered a combination of all classifiers (here, named ALL) and the single best classifier (SB) in the pool. The SB for each problem were defined using the validation set. In the combination process, the Majority Vote was used to combine the classifiers generated by Bagging, while for those built by the GA, we used a sigmoid function, as defined by Eq. (8), to estimate weights for each classifier based on their fitness.

$$f(x, a, c) = \frac{1}{1 + e^{-a(x-c)}} \tag{8}$$

where x corresponds to the fitness of each element, a refers to the curve inclination while c is the curve inflection point. The values of a and c were empirically defined as 2 and 1, respectively.

The average performance of each approach for all classification problems is shown in Tables 2 and 3. The first one presents the results obtained by methods in which a single classifier is selected,

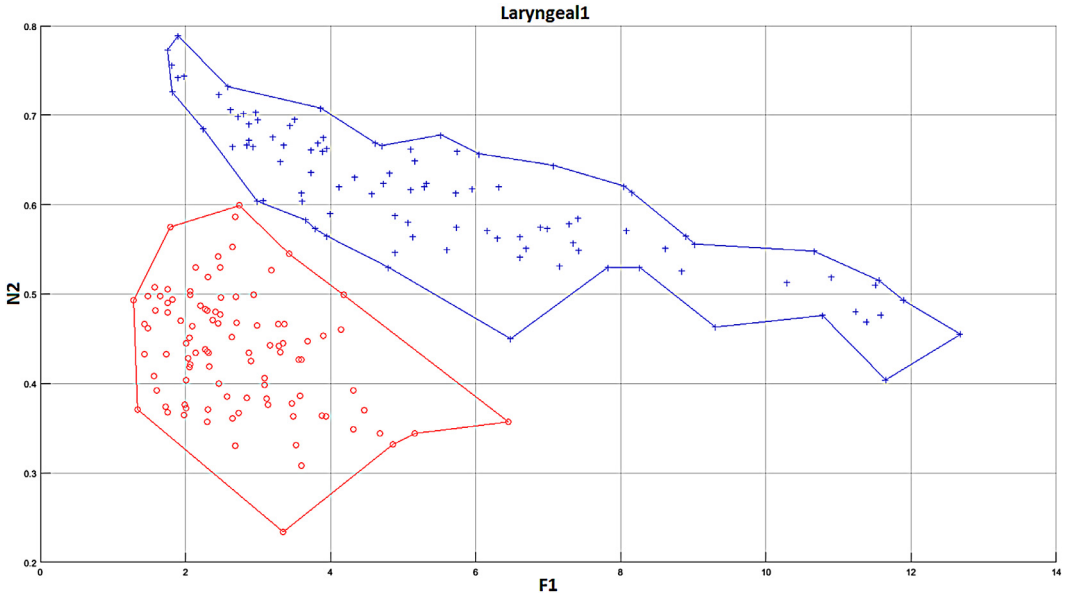

Fig. 11. Dispersion of the data subsets generated for the Laryngeal1 problem in the complexity space. The elements built by Bagging are shown in red, while those in blue show the pool obtained by the GA. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 5
Comparison of the proposed methods (DSOC Bagging and DSOC GA) with the dynamic selection methods, OLA, LCA, A Priori (APRI), A Posteriori (APOS), Knora-U (KU) and Knora-E (KE). The average accuracy and corresponding standard deviation computed from 20 replications. The values in boldface are the best results.

	OLA Bagg	LCA Bagg	APRI Bagg	APOS Bagg	KU Bagg	KE Bagg	DSOC Bagg	DSOC GA
Adult	84.04 (2.87)	83.26 (2.52)	84.01 (2.34)	83.55 (2.60)	83.08 (2.03)	81.92 (1.85)	86.77 (2.65)*	**86.80 (1.85)***
Banana	85.03 (1.73)	84.88 (1.78)	84.52 (1.64)	83.82 (1.69)	**87.59 (1.24)**	85.06 (1.46)	82.17 (1.64)	87.20 (1.62)◇
Blood	**76.12 (0.26)**	75.86 (1.23)	**76.12 (0.26)**	**76.12 (0.26)**	**76.12 (0.26)**	**76.12 (0.26)**	73.98 (2.80)	**76.12 (0.26)** ◇
CTG	86.99 (0.89)	80.94 (0.95)	86.45 (1.38)	86.28 (1.29)	84.67 (0.96)	83.95 (0.89)	85.68 (1.14)	**87.50 (1.25)** ◇
Diabetes	64.92 (2.85)	62.37 (2.64)	65.00 (2.03)	64.38 (2.44)	64.48 (0.95)	64.56 (1.09)	66.69 (3.29)	**68.41 (3.93)*** ◇
Ecoli	64.94 (4.74)	52.56 (5.52)	64.05 (5.05)	63.39 (4.00)	54.4 0(2.77)	52.02 (2.10)	69.29 (3.23)*	**75.00 (2.44)*** ◇
Faults	58.35 (2.31)	37.96 (10.5)	56.08 (2.15)	54.63 (2.25)	43.65 (2.49)	37.41 (1.96)	48.68 (3.17)*	**65.02 (1.58)*** ◇
German	71.90 (2.35)	64.22 (3.31)	71.74 (2.54)	69.80 (2.39)	71.88 (1.02)	71.52 (0.88)	71.88 (2.66)	**73.98 (3.28)*** ◇
Glass	53.77 (4.79)	24.15 (9.04)	47.92 (7.41)	24.72 (19.5)	51.13 (5.37)	45.28 (5.13)	53.11 (8.03)	**59.25 (5.22)*** ◇
Haberman	73.68 (0.59)	73.62 (0.88)	73.82 (0.39)	73.68 (0.59)	73.68 (0.00)	73.68 (0.00)	74.14 (3.18)	**74.61 (2.43)**
Heart	78.58 (4.04)	70.15 (4.58)	78.81 (4.02)	72.24 (8.64)	75.67 (4.77)	74.55 (4.21)	83.43 (2.79)*	**83.58 (3.24)***
ILPD	69.72 (3.38)	61.76 (3.67)	69.24 (3.22)	67.52 (4.16)	**71.90 (0.61)**	71.83 (0.45)	64.97 (4.10)	66.86 (2.73)
Image	42.03 (2.30)	32.18 (4.11)	40.68 (1.82)	40.29 (2.35)	36.39 (1.28)	35.86 (1.19)	38.30 (1.58)	**51.03 (1.11)*** ◇
Ionosphere	80.91 (3.88)	69.72 (4.21)	80.63 (4.72)	77.84 (4.62)	**89.38 (3.54)**	88.69 (3.83)	86.08 (5.12)	86.53 (4.23)
Laryngeal1	80.09 (4.40)	70.94 (6.24)	81.13 (5.57)	78.11 (6.92)	73.58 (4.77)	72.08 (4.57)	80.66 (4.81)	**82.45 (4.51)**
Laryngeal3	66.02 (4.19)	55.34 (5.77)	66.02 (4.70)	62.84 (6.18)	61.14 (3.48)	58.64 (3.86)	65.45 (6.68)	**68.75 (5.04)**◇
Lithuanian	68.30 (2.49)	70.38 (1.84)	67.06 (2.42)	64.50 (3.36)	58.43 (0.98)	56.69 (0.80)	74.86 (3.00)*	**82.47 (2.55)*** ◇
Liver	60.99 (3.63)	50.06 (5.30)	58.90 (5.51)	55.17 (5.47)	56.63 (5.59)	52.21 (3.51)	59.36 (5.05)	**61.86 (5.39)**◇
Magic	79.22 (0.70)	78.17 (0.55)	78.73 (0.81)	78.59 (0.83)	78.90 (0.58)	78.80 (0.55)	78.46 (0.61)	**79.99 (0.73)**
Mammo	79.78 (2.29)	76.23 (3.11)	80.10 (3.17)	79.08 (2.74)	78.50 (2.66)	77.85 (2.83)	**81.04 (2.48)**	80.99 (2.23)
Monk	81.99 (3.56)	69.26 (3.98)	79.58 (4.07)	66.76 (12.8)	79.31 (3.26)	78.15 (3.72)	82.69 (2.90)	**85.42 (3.44)*** ◇
Phoneme	77.18 (1.25)	76.54 (0.96)	76.17 (1.47)	76.08 (1.55)	74.77 (0.74)	74.07 (0.64)	76.61 (1.20)	**79.00 (1.04)*** ◇
Sonar	61.73 (6.32)	46.83 (7.28)	58.65 (6.10)	41.63 (22.5)	53.37 (1.03)	53.17 (0.92)	67.40 (7.44)	**68.17 (8.48)**
Thyroid	93.67 (1.25)	91.88 (1.97)	93.29 (1.71)	92.51 (2.79)	89.22 (2.73)	88.79 (2.82)	89.10 (2.73)	**94.02 (1.60)** ◇
Vehicle	32.16 (3.87)	24.53 (4.66)	30.28 (3.92)	29.55 (3.85)	27.51 (1.69)	26.30 (1.23)	33.25 (2.21)	**35.43 (2.28)*** ◇
Vertebral	81.53 (4.35)	71.67 (5.13)	**82.53 (3.32)**	78.00 (6.06)	79.27 (4.47)	78.47 (3.80)	77.40 (4.14)	80.33 (3.60) ◇
WBC	77.36 (14.6)	84.86 (3.70)	77.75 (14.1)	70.32 (17.3)	89.15 (3.16)	88.59 (3.72)	92.75 (1.88)*	**93.13 (2.19)***
WDVG	79.92 (1.13)	67.81 (3.77)	79.18 (1.21)	78.37 (1.35)	71.07 (1.12)	69.66 (1.05)	75.54 (2.38)	**82.32 (1.11)** ◇
Weaning	76.87 (4.14)	60.40 (5.50)	75.67 (5.21)	59.13 (16.8)	66.93 (4.98)	64.47 (4.45)	80.67 (3.83)*	**81.67 (4.48)***
Wine	33.52 (3.44)	45.23 (14.1)	34.32 (4.71)	32.61 (2.41)	32.84 (1.13)	32.84 (1.13)	49.77 (12.8)	**55.68 (11.0)***

while the second one presents the results related to the ensemble selection methods, the combination of all classifiers in the pool (ALL), and the single best classifier (SB). The boldfaced values in these tables represent the best result for each problem. In order to compare the behavior of the approaches, the Wilcoxon test was performed with a confidence of 95%. The "*" symbol highlights the cases where a significant difference was observed between the approaches. When comparing the behavior of the DCS methods, we can see that in 126 of the 180 experiments (70.00%), adopting the proposed GA to generate the pool led to improved accuracy. On the other hand, in 22.22% of the scenarios (40 experiments), it was more appropriate to employ only the Bagging method. 14 cases (7.78%) demonstrated similar results.

An analysis of the combination of all classifiers available in the pool reveals that the GA prevails in 63.33% of the experiments. On the other hand, Bagging achieves a higher accuracy in 30.00%

Table 6
Comparison of the DSOC GA with the OLA, LCA, A Priori (APRI), A Posteriori (APOS), Knora-U (KU) and Knora-E (KE) DCS methods using the same pool generated by the proposed method. The average accuracy and corresponding standard deviation computed from 20 replications. The values in boldface are the best results.

	OLA GA	LCA GA	APRI GA	APOS GA	KU GA	KE GA	DSOC GA
Adult	84.77 (2.80)	83.52 (2.87)	85.73 (2.78)	85.20 (2.91)	84.24 (1.85)	83.72 (1.92)	**86.80 (2.24)**
Banana	84.87 (1.68)	84.68 (1.79)	84.57 (1.87)	83.83 (1.76)	**87.42 (1.12)**	85.06 (1.45)	87.20 (1.62)
Blood	**76.12 (0.26)**	**76.12 (0.26)**	**76.12 (0.26)**	**76.12 (0.26)**	**76.12 (0.26)**	**76.12 (0.26)**	**76.12 (0.26)**
CTG	86.70 (1.04)	81.33 (0.82)	86.76 (0.76)	86.30 (0.85)	85.01 (0.93)	84.21 (0.89)	**87.50 (1.25)***
Diabetes	65.00 (2.27)	63.75 (2.89)	64.51 (2.46)	64.41 (2.76)	64.64 (0.87)	64.79 (1.11)	**68.41 (3.93)***
Ecoli	62.38 (3.09)	52.72 (5.91)	64.46 (4.83)	62.38 (3.94)	54.44 (2.77)	52.26 (1.99)	**75.00 (2.44)***
Faults	56.82 (2.99)	38.84 (10.1)	55.70 (3.42)	54.73 (3.66)	44.96 (2.80)	39.25 (2.27)	**65.02 (1.58)***
German	73.48 (2.42)	65.88 (2.98)	72.80 (3.08)	71.52 (2.85)	72.94 (1.36)	72.28 (1.15)	**73.98 (3.28)**
Glass	51.89 (7.46)	25.47 (10.2)	53.11 (6.09)	30.75 (16.9)	51.98 (5.15)	47.45 (4.71)	**59.25 (5.22)***
Haberman	74.14 (0.75)	74.41 (1.74)	74.41 (1.35)	73.29 (3.00)	73.68 (0.00)	73.68 (0.00)	**74.61 (2.43)**
Heart	80.37 (4.12)	72.84 (5.31)	80.67 (3.73)	77.31 (4.10)	77.16 (4.50)	76.19 (4.31)	**83.58 (3.24)***
ILPD	68.62 (2.91)	63.79 (3.82)	70.34 (2.97)	68.41 (3.68)	**71.9 (0.75)**	71.83 (0.45)	66.86 (2.73)
Image	41.25 (2.12)	32.82 (3.87)	41.33 (1.99)	41.42 (2.34)	36.35 (1.52)	35.88 (1.48)	**51.03 (1.11)***
Ionosphere	81.48 (4.56)	72.10 (4.14)	80.72 (4.14)	81.02 (4.36)	**87.73 (3.40)**	87.50 (3.17)	86.53 (4.23)
Laryngeal1	78.87 (5.08)	73.49 (5.04)	80.00 (3.89)	77.08 (5.11)	74.25 (3.79)	72.74 (4.07)	**82.45 (4.51)**
Laryngeal3	67.22 (4.90)	58.81 (6.25)	66.14 (3.23)	63.92 (4.43)	61.02 (3.43)	59.37 (3.25)	**68.75 (5.04)**
Lithuanian	67.06 (3.19)	69.03 (2.40)	67.49 (3.21)	65.15 (3.16)	59.87 (1.40)	58.04 (1.21)	**82.47 (2.55)***
Liver	60.17 (3.14)	51.34 (5.15)	57.38 (4.96)	54.24 (8.20)	57.09 (6.77)	55.17 (4.58)	**61.86 (5.39)**
Magic	79.00 (0.56)	78.23 (0.60)	78.78 (0.53)	78.53 (0.63)	78.92 (0.58)	78.80 (0.58)	**79.99 (0.73)***
Mammo	80.56 (2.96)	77.97 (3.08)	80.51 (3.44)	79.15 (3.44)	79.13 (2.93)	78.77 (2.76)	**80.99 (2.23)**
Monk	82.59 (3.22)	72.73 (3.81)	81.48 (4.86)	68.19 (14.5)	78.84 (3.67)	79.86 (3.08)	**85.42 (3.44)***
Phoneme	77.22 (0.82)	76.77 (0.86)	77.14 (0.98)	77.05 (0.96)	75.74 (0.81)	75.17 (0.74)	**79.00 (1.04)***
Sonar	60.29 (5.38)	45.96 (5.73)	61.44 (5.94)	45.96 (20.8)	55.00 (1.86)	53.85 (1.36)	**68.17 (8.48)***
Thyroid	93.73 (1.45)	91.99 (1.71)	93.41 (1.92)	92.02 (1.97)	88.70 (4.11)	88.38 (4.05)	**94.02 (1.60)**
Vehicle	32.44 (2.65)	23.96 (4.74)	32.04 (3.33)	31.11 (4.01)	28.22 (2.57)	26.75 (1.31)	**35.43 (2.28)***
Vertebral	80.73 (4.70)	73.20 (5.20)	**81.67 (5.07)**	77.87 (8.13)	80.47 (4.64)	79.67 (4.58)	80.33 (3.60)
WBC	77.57 (13.5)	84.89 (3.25)	79.79 (12.2)	79.01 (10.6)	88.49 (2.96)	88.27 (3.47)	**93.13 (2.19)***
WDVG	80.24 (0.88)	69.91 (3.41)	80.08 (1.19)	79.49 (1.33)	72.71 (1.15)	71.60 (1.18)	**82.32 (1.11)***
Weaning	77.27 (4.43)	63.33 (6.57)	78.40 (4.19)	64.93 (19.9)	71.07 (3.96)	68.47 (3.58)	**81.67 (4.48)**
Wine	33.52 (2.95)	43.86 (13.3)	35.68 (7.58)	33.41 (2.88)	32.84 (1.13)	32.84 (1.13)	**55.68 (10.98)***

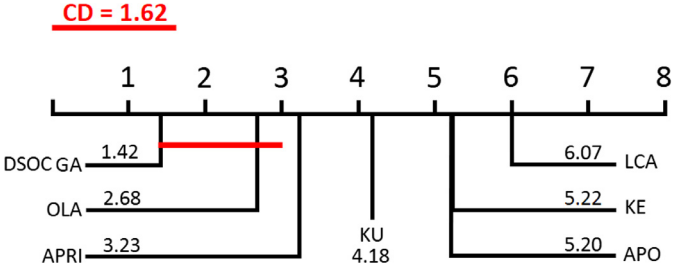

Fig. 12. Ranking produced by the Nemenyi test considering DSOC GA and the other DCS methods using a pool generated with Bagging.

of the cases. 2 ties were observed among the strategies (6.67%). For the single best classifier, an improvement was observed in 17 of the 30 tested cases (56.67%), while a decrease in accuracy was noted in 11 scenarios (36.67%). As well, 2 ties were observed among the solutions, corresponding to 6.67%.

Fig. 8 shows the pairwise comparison between Bagging and the proposed GA-based method for all 6 DCS-based approaches, a combination of all classifiers (ALL), and the single best classifier (SB). The red columns correspond to the scenarios in which the use of the Bagging method to generate subsets represents the best option, while the blue columns correspond to the cases in which the GA-based method was able to obtain the highest accuracy. The representation in green indicates where there was a tie between the approaches. The dashed line illustrates the critical value ($cv = 19.5$). This cv value was obtained from the wins, ties and losses by computing the test sign [32] with a significance level $\alpha = 0.05$. If we consider half the ties added to the wins and the other half to the losses, the GA-based method shows a significant improvement in most of scenarios, except when OLA and SB were adopted. The best contribution was observed when using the LCA as DCS method ($25 > 19.5$). On the other hand, when OLA was adopted as DCS, we did not notice a significant contribution ($16 < 19.5$).

We observed a positive impact on the classification performance when information related to problem complexity was used to orient the classifier pool generation. However, additional analysis is a must in order to obtain a full answer to our first research question. To check the coverage of the problem complexity space for each classification problem, we calculate the average complexity dispersion of both pools, namely, the one generated by the GA method and the other generated by the Bagging method.

Table 4 shows the average value of dispersion along the 20 replications, as defined in the Eq. (9). The bold values corresponds to the highest coverage space for each problem.

$$\overline{Dispersion} = \frac{\sum_1^r \frac{\sum_{j=1}^{M} \sqrt{\sum_{k=1}^{nc} (x_{i,k} - x_{j,k})^2}}{M-1}}{r} \tag{9}$$

where M corresponds to the number of classifiers in the pool, nc refers to the number of complexity measures adopted in the process, r represents the number of repetitions, while $x_{i,k}$ and $x_{j,k}$ correspond to the value of the k^{th} complexity measure for the elements i and j, respectively. As we can see, there is a clear increase in the dispersion among the subsets when the proposed GA-based pool generator is adopted. In order to compare the behavior of the pools, the Wilcoxon test was applied to compare these results with a 5% significance level. Significant differences are marked with a "*".

The occupation of the data complexity space is illustrated in Figs. 9–11, which show changes in the space defined by $F1$ and $N2$ metrics. In the representations, the red circles correspond to the pool generated by Bagging, and the blue markers to the final set obtained by the GA pool generation. These figures reflect a common behavior observed in the studied problems: the complexity

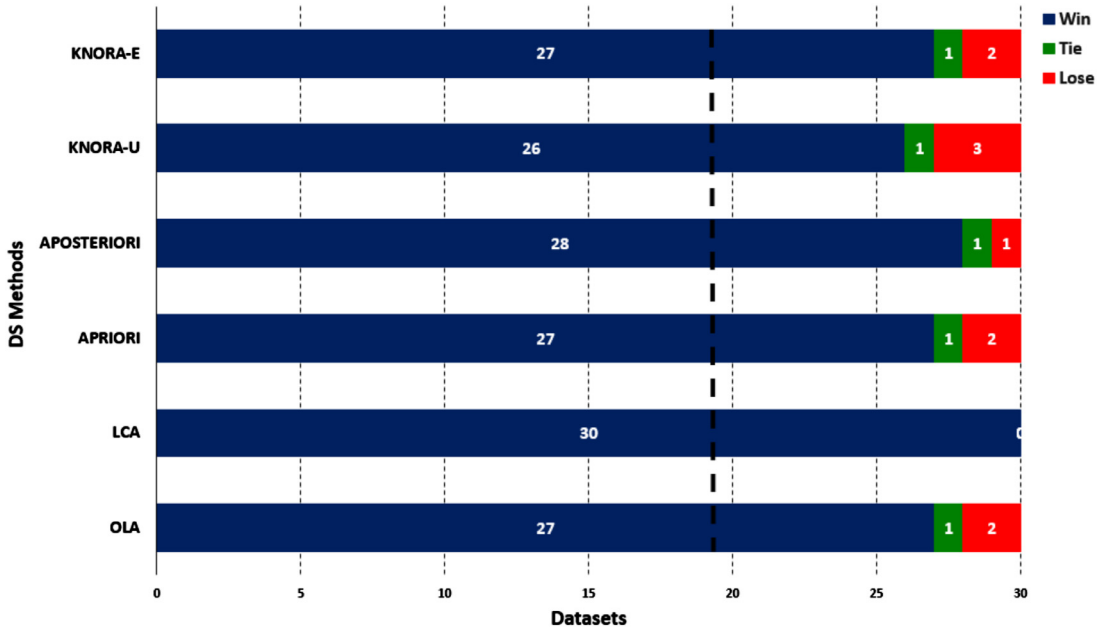

Fig. 13. Pairwise comparison of the proposed DSOC GA with 6 DCS methods appearing in the literature using Bagging. The blue bars represent the number of problems in which DSOC GA prevailed, the red bars refer to its losses, while green bars are related to ties. The dashed line illustrates the critical value (19.5). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

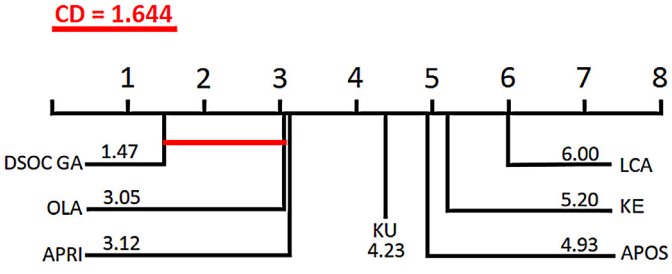

Fig. 14. Ranking produced by the Nemenyi test. DSOC GA and the other DCS methods also using the proposed GA-based method.

distribution of the pool generated by the GA allows a better exploration of the $F1 \times N2$ complexity space. However, the variation in $F1$ is more evident, while the fluctuation in the $N2$ axis is more discrete, and is characterized for a shift of the set. This behavior is caused by the search for the space coverage, once the built sets present farther centroids (higher $F1$) and, at the same time, border regions more intricate, evidenced by the increasing on $N2$. We can now therefore complete the answer of our first research question: the adopted strategy for pool generation can provide better coverage of the problem complexity space. In addition, it has been used successfully with different dynamic classifiers selection methods.

5.2. Experiments on classifier selection

The objective of this set of experiments was to evaluate the proposed DCS framework. To that end, we have considered two possible approaches, namely, DSOC Bagging and DSOC GA. In the former, the pool generation is performed using the Bagging method, while in the latter, the GA-based pool generator is used. The average performance of each approach for each classification problem is shown in Table 5. The boldfaced values in the table represent the highest accuracy for each problem.

The penultimate column of Table 5 shows the results of the DSOC Bagging. When compared with the methods appearing in the literature, this approach prevailed in 114 out of 180 experiments (63.3%), and lagged in 66 experiments (36.6%). The Kruskal Wallis statistical test was performed to compare the DSOC Bagging against the 6 DCS methods in the literature. A confidence of 95% and a degree of freedom of 6 were used. We observed a significant difference in 7 of 30 classification problems, and the 7 carry a "*" in the penultimate column of Table 5.

On the other hand, our main results are in the last column of Table 5, and are related to the DSOC GA approach. By using the GA-based pool generator in combination with the proposed dynamic selection method, this DCS prevailed in 165 out of 180 experiments (91.67%), lost in 10 experiments (5.56%), and saw a tied in 5 others (2.78%). A statistical analysis of significance in this case showed a significant difference in 15 out of 30 experiments. The 15 cases carry a "*" in the last column of Table 5. For these experiments, Fig. 12 presents a ranking produced by means of the Nemenyi post-hoc test.

We could note that our DSOC GA obtained the best position in the ranking when compared to the 6 DCS methods using Bagging. Except for the OLA method, the distances between DSOC GA and the other methods are greater than the critical distance. In addition, Fig. 13 shows the pairwise comparison between the DCS strategies in the literature using the pool built by Bagging and those built using the proposed DSOC GA. The red columns represent the scenarios in which the use of Bagging for pool generation combined with the DCS methods of the literature was the best option, while the columns in blue indicate the cases in which DSOC GA achieved the highest accuracy. The representations in green indicate where there was a tie between the approaches. The dashed line illustrates the critical value ($cv = 19.5$). As mentioned before, the cv value was obtained from the number of wins, ties and losses by computing the test sign [32] with a significance level $\alpha = 0.05$. As we can see, the DSOC GA is significantly superior in all scenarios.

Still in Table 5, we also compared the performance of DSOC Bagging against DSOC GA. The Wilcoxon test with a 5% significance was applied to compare the results of these strategies. Sig-

nificant differences appears with a "◇" marker in the last column of Table 5. For 18 out of 30 classification problems, there is a significant increase in performance when the DSOC method is used in a pool generated by the proposed GA-based generator.

Finally, in the last set of experiments, we compared all DCS methods using the same pool generated by the GA-based method, and Table 6 summarizes the results. The proposed DCS prevailed in 158 out of 180 experiments (87.78%), lost in 16 experiments (8.89%), and was tied in 6 others (3.33%). The statistical analysis of significance shows a significant difference in this case in 16 out of 30 experiments. Fig. 14 shows a ranking produced by the Nemenyi post-hoc test.

As can be seen, the results are quite similar to those observed when considering the other DCS methods in the literature that use Bagging. For instance, here, we observed a significant difference in 16 classification problems, while in the previous comparison of DSOC GA versus other DCS using Bagging, it was observed for 15 problems.

Based on all these experiments, we can answer our second research question, saying that the use of data complexity features for pool generation and classifier selection has shown some interesting contribution in terms of classification performance. It can be concluded that the proposed selection strategy may profit the better coverage of the problem complexity space provided by the GA-based pool generation method.

In a last analysis we have computed the average time for the classification task when using the OLA, KU and DSOC methods. From the methods of the literature, OLA presented the most promising results during our experiments, being a simple strategy that selects just one classifier, while KU has shown the best results when an ensemble of classifiers is selected. The OLA method spent in average 0.70 ms (milliseconds) to classify each test instance, the KU method takes 3.63 ms, while the proposed DSOC takes 9.08 ms. As expected, the proposed method is more time consuming. The reason is the need to compute the complexity of the neighborhood of the test instance. However, this process can be optimized since in the current version of our method, we are using an external library (DCOL Library) to compute the complexity measures.

6. Conclusion

In this paper, we proposed a DCS method whose novelty lies in its use of features related to the classification problem difficulty during pool generation and classifier selection. To represent the classification problem difficulty, we extracted features from the problem data using complexity measures. A robust experimental protocol based on 30 datasets, and considering 20 replications, confirms our two main hypotheses. A better comprehension of the classification problem difficulty may have a positively impact on the performance of a DCS method. The main results, in which pool generation and classifier selection are both based on complexity features, are very promising. In 165 out of 180 experiments (91.67%), adopting the proposed GA to generate the pool, combined with the proposed DCS, allowed an improvement of the classification accuracy. For the pool generation method, in 126 out of 180 experiments (70.00%), adopting the proposed GA to generate the pool allowed improved accuracy. In addition, we conclude that the proposed pool generation strategy could achieve a better coverage of the problem complexity space, and the proposed dynamic selection method could take advantage of this scenario. Future works could follow two main directions. First, we could look at tuning the parameters of the proposed DCS, by evaluating different base classifiers, or different optimization algorithms. Secondly, we could investigate other measures in order to better describe the classification problem complexity space.

Acknowledgment

This research has been supported by the Brazilian National Council for Scientific and Technological Development (CNPq) (grant no. 307277/2014-3).

References

[1] A.S. BrittoJr., R. Sabourin, L.E.S. Oliveira, Dynamic selection of classifiers - a comprehensive review, Pattern Recognit. 47 (11) (2014) 3665–3680, doi:10.1016/j.patcog.2014.05.003.
[2] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140, doi:10.1023/A:1018054314350.
[3] Y. Freund, R.E. Schapire, Experiments with a new boosting algorithm, in: Proceedings of the 13th International Conference on Machine Learning, 1996, pp. 148–156.
[4] T.K. Ho, The random subspace method for constructing decision forests, IEEE Trans. Pattern Anal. Mach. Intell. 20 (8) (1998) 832–844, doi:10.1109/34.709601.
[5] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, G.E. Hinton, Adaptive mixtures of local experts, Neural Comput. 3 (1) (1991) 79–87. Spring.
[6] T.K. Ho, M. Basu, Complexity measures of supervised classification problems, Pattern Anal. Mach. Intell. IEEE Trans. 24 (3) (2002) 289–300, doi:10.1109/34.990132.
[7] T. Ho, M. Basu, M. Law, Measures of geometrical complexity in classification problems, in: M. Basu, T. Ho (Eds.), Data Complexity in Pattern Recognition, Advanced Information and Knowledge Processing, Springer, London, 2006, pp. 1–23.
[8] N. Macià, E. Bernadó-Mansilla, A. Orriols-Puig, T.K. Ho, Learner excellence biased by data set selection: a case for data characterisation and artificial data sets, Pattern Recognit. 46 (3) (2013) 1054–1066, doi:10.1016/j.patcog.2012.09.022.
[9] A.L. Brun, A.S. Britto, L.S. Oliveira, F. Enembreck, R. Sabourin, Contribution of data complexity features on dynamic classifier selection, in: 2016 International Joint Conference on Neural Networks (IJCNN), Springer-Verlag, Vancouver, CA, 2016, pp. 4396–4403, doi:10.1109/IJCNN.2016.7727774.
[10] K. Woods, W.P. KegelmeyerJr., K. Bowyer, Combination of multiple classifiers using local accuracy estimates, IEEE Trans. Pattern Anal. Mach. Intell. 19 (4) (1997) 405–410, doi:10.1109/34.588027.
[11] L. Didaci, G. Giacinto, F. Roli, G.L. Marcialis, A study on the performances of dynamic classifier selection based on local accuracy estimation, Pattern Recognit. 38 (11) (2005) 2188–2191.
[12] M. Sabourin, A. Mitiche, D. Thomas, G. Nagy, Classifier combination for handprinted digit recognition, in: Document Analysis and Recognition, 1993, Proceedings of the Second International Conference on, 1993, pp. 163–166, doi:10.1109/ICDAR.1993.395758.
[13] G. Giacinto, F. Roli, Methods for dynamic classifier selection, in: Image Analysis and Processing, 1999. Proceedings, 10th International Conference on, 1999, pp. 659–664, doi:10.1109/ICIAP.1999.797670.
[14] G. Giacinto, F. Roli, G. Fumera, Selection of classifiers based on multiple classifier behaviour, in: Proceedings of the Joint IAPR International Workshops on Advances in Pattern Recognition, Springer-Verlag, London, UK, UK, 2000, pp. 87–93.
[15] L. Kuncheva, J. Rodriguez, Classifier ensembles with a random linear Oracle, Knowl. Data Eng. IEEE Trans. 19 (4) (2007) 500–508, doi:10.1109/TKDE.2007.1016.
[16] A. Ko, R. Sabourin, A. BrittoJr., From dynamic classifier selection to dynamic ensemble selection, Pattern Recognit. 41 (5) (2008) 1718–1731, doi:10.1016/j.patcog.2007.10.015.
[17] A. Santana, R. Soares, A. Canuto, M.C.P.d. Souto, A dynamic classifier selection method to build ensembles using accuracy and diversity, in: Neural Networks, 2006. SBRN '06. Ninth Brazilian Symposium on, 2006, pp. 36–41, doi:10.1109/SBRN.2006.1.
[18] Y. Yan, X.-C. Yin, Z.-B. Wang, X. Yin, C. Yang, H.-W. Hao, Sorting-based dynamic classifier ensemble selection, in: Document Analysis and Recognition (ICDAR), 2013 12th International Conference on, 2013, pp. 673–677, doi:10.1109/ICDAR.2013.138.
[19] E. dos Santos, R. Sabourin, P. Maupin, Ambiguity-guided dynamic selection of ensemble of classifiers, in: Information Fusion, 2007 10th International Conference on, 2007, pp. 1–8, doi:10.1109/ICIF.2007.4408123.
[20] J. Xiao, C. He, Dynamic classifier ensemble selection based on gmdh, in: Computational Sciences and Optimization, 2009. CSO 2009, in: International Joint Conference on, volume Vol. 1, 2009, pp. 731–734, doi:10.1109/CSO.2009.276.
[21] N. Macià, A. Orriols-Puig, E. Bernadó-Mansilla, In search of targeted-complexity problems, in: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, GECCO '10, ACM, New York, NY, USA, 2010, pp. 1055–1062, doi:10.1145/1830483.1830674.
[22] N. Macià, E. Bernadó-Mansilla, A. Orriols-Puig, T.K. Ho, Learner excellence biased by data set selection: a case for data characterisation and artificial data sets, Pattern Recognit. 46 (3) (2013) 1054–1066, doi:10.1016/j.patcog.2012.09.022.
[23] W. Chinnasri, N. Sureerattanan, Comparison of performance between different selection strategies on genetic algorithm with course timetabling problem, in:

Proceedings of IEEE International Conference on Advanced Management Science(ICAMS), vol. 2. July, 2010, pp. 105–108.
[24] R.G. Mantovani, A.L.D. Rossi, J. Vanschoren, B. Bischl, A.C.P. de Leon Ferreira, To tune or not to tune: recommending when to adjust SVM hyper-parameters via meta-learning, IJCNN, 2015.
[25] A. Orriols-Puig, N. Macià, T.K. Ho, Documentation for the data complexity library in C++, Tech. rep., 2010. Barcelona, Spain.
[26] K. Bache, M. Lichman, UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.
[27] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework, J. Mult.-Valued Logic Soft Comput. 17 (2–3) (2011) 255–287. Cited By 275.
[28] L. Kuncheva, Statlog: Comparison of classification algorithms on large real-world problems, 2004. URL http://pages.bangor.ac.uk/~mas00a/activities/real_data.htm.
[29] R.D. King, C. Feng, A. Sutherland, Statlog: comparison of classification algorithms on large real-world problems, 1995.
[30] R.M. Cruz, R. Sabourin, G.D. Cavalcanti, T.I. Ren, Meta-des: a dynamic ensemble selection framework using meta-learning, Pattern Recognit. 48 (5) (2015) 1925–1935, doi:10.1016/j.patcog.2014.12.003.
[31] R.M. Cruz, R. Sabourin, G.D. Cavalcanti, META-DES. Oracle: meta-learning and feature selection for dynamic ensemble selection, Inf. Fusi. 38 (2017) 84–103.
[32] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30. URL http://dl.acm.org/citation.cfm?id=1248547.1248548.

A. L. Brun received M.Sc. degree in Agricultural Engineering from the Universidade Estadual do Oeste do Paraná (UNIOESTE, Brazil) in 2007, and Ph.D. degree in Informatics from the Pontifícia Universidade Católica do Paraná (PUCPR, Brazil) in 2017. In 2008, he joined the Informatics Department of the Universidade Estadual do Oeste do Paraná (Unioeste, Brazil). His research interests are in the areas of Machine Learning and Pattern Recognition.

A. S. Britto Jr. received M.Sc. degree in Industrial Informatics from the Centro Federal de Educação Tecnológica do Paraná (CEFET-PR, Brazil) in 1996, and Ph.D. degree in Computer Science from the Pontifícia Universidade Católica do Paraná (PUCPR, Brazil) in 2001. In 1989, he joined the Informatics Department of the Universidade Estadual de Ponta Grossa (UEPG, Brazil). In 1995, he also joined the Computer Science Department of the Pontifícia Universidade Católica do Paraná (PUCPR) and, in 2001, the Post-graduate Program in Informatics (PPGIa). His current interests include Pattern Recognition, Machine Learning, Image Analysis, and Evolutionary Computation.

L.E.S. Oliveira received the B.S. degree in Computer Science from UnicenP, Curitiba, PR, Brazil, the M.Sc. degree in electrical engineering and industrial informatics from the Centro Federal de Educacao Tecnologica do Parana (CEFET-PR), Curitiba, PR, Brazil, and Ph.D. degree in Computer Science from Ecole de Technologie Superieure, Universite du Quebec in 1995, 1998, and 2003, respectively. From 2004 to 2009 he was professor of the Computer Science Department at Pontifical Catholic University of Parana, Curitiba, PR, Brazil. In 2009 he joined the Federal University of Parana, Curitiba, PR, Brazil, where he is professor of the Department of Informatics. His current interests include Pattern Recognition, Neural Networks, Image Analysis, and Evolutionary Computation.

F. Enembreck received his graduate degree in Computer Science from the State University of Ponta Grossa in 1997, M.Sc. in Applied Computer Science from the Pontifical Catholic University of Paraná in 1999 and his Ph.D. in Information Technologies and Systems from the Université de Technologie de Compiegne in 2003. Currently he is professor at the Pontifical Catholic University of Paraná and researcher at the Graduate Program in Informatics, directing researches of MSc and PhD students. He has experience in Computer Science with emphasis on Architecture of Computing Systems, acting on the following topics: distributed artificial intelligence, multi-agent systems, adaptive agents, information retrieval and machine learning.

R. Sabourin joined in 1977 the Physics Department of the Montreal University where he was responsible for the design, experimentation and development of scientific instrumentation for the Mont Mégantic Astronomical Observatory. His main contribution was the design and the implementation of a microprocessor-based fine tracking system combined with a low-light level CCD detector. In 1983, he joined the staff of the École de Technologie Supérieure, Université du Québec, in Montréal where he cofounded the Department of Automated Manufacturing Engineering where he is currently Full Professor and teaches Pattern Recognition, Evolutionary Algorithms, Neural Networks and Fuzzy Systems. In 1992, he joined also the Computer Science Department of the Pontifícia Universidade Católica do Paraná (Curitiba, Brazil) where he was co-responsible for the implementation in 1995 of a master program and in 1998 a Ph.D. program in applied computer science. Since 1996, he is a senior member of the Centre for Pattern Recognition and Machine Intelligence (CENPARMI, Concordia University). Dr. Sabourin is the author (and co-author) of more than 260 scientific publications including journals and conference proceedings. He was co-chair of the program committee of CIFED'98 (Conférence Internationale Francophone sur l'Écrit et Le Document, Québec, Canada) and IWFHR'04 (9th International Workshop on Frontiers in Handwriting Recognition, Tokyo, Japan). He was nominated as Conference co-chair of ICDAR'07 (9th International Conference on Document Analysis and Recognition) that has been held in Curitiba, Brazil, in 2007. His research interests are in the areas of handwriting recognition, signature verification, intelligent watermarking systems and bio-cryptography.