Neurocomputing 463 (2021) 59–76

Contents lists available at ScienceDirect

Neurocomputing

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

A lexicographic cooperative co-evolutionary approach for feature selection

Jesús González [a,*], Julio Ortega [a], Juan José Escobar [a], Miguel Damas [a]

[a] Department of Computer Architecture and Technology, CITIC, University of Granada, Granada, Spain

ARTICLE INFO

Article history:
Received 11 January 2021
Revised 1 June 2021
Accepted 8 August 2021
Available online 11 August 2021
Communicated by Zidong Wang

Keywords:
Cooperative co-evolution
Multi-objective optimization
Lexicographic optimization
Feature selection
Classification

ABSTRACT

This paper starts with two hypotheses. The first one is that the simultaneous optimization of the hyperparameters regulating the classifier within a wrapper method, while the best subset of features is being determined, should improve the results with respect to those obtained with a pre-parameterized classifier. The second one is that solving these two problems can be formulated as a lexicographic optimization problem, allowing the use of a simple single-objective evolutionary algorithm to solve this multi-objective problem.

The fitness function is of key importance for such wrapper methods. It is responsible for guiding the search towards potentially good solutions and it also consumes most of the runtime. Having these issues in mind, this paper also proposes a new lexicographic fitness function, designed to minimize the runtime of the algorithm and also to avoid over-fitting. Furthermore, the execution time and the quality of the results obtained by the wrapper procedure also depend on some algorithmic hyperparameters: the similarity thresholds used when comparing two different solutions lexicographically and the percentage of data samples used for validation during the training process. Thus, an experimental analysis has been carried out to find adequate values for these hyperparameters. Finally, the lexicographic cooperative co-evolutionary wrapper approach, using the new fitness function proposed in this paper, has been tested with several datasets belonging to the University of California, Irvine (UCI) repository and also with some real high-dimensional datasets, obtaining quite good results, compared to other state-of-the-art wrapper methods. The comparison has also been made lexicographically, with a new methodology proposed in this paper.

1. Motivation

Since datasets may contain redundant, noisy or even irrelevant features (concerning the process being observed), one of the first steps in any machine learning application is related to the selection of the subset of features that best describe the data. Feature selection makes easier the learning process in many ways: it reduces both, the dataset storage requirements and the training time, since fewer data are needed. It also helps to mitigate the curse of dimensionality [1] and to improve the prediction performance [2].

Although first attempts to the feature selection problem were proposed in the sixties [3,4], it was at the end of last century when the feature selection problem was more thoroughly studied and characterized [5,6]. Regarding the kind of processing, feature selection techniques can be divided into filter, wrapper and embedded methods. Filters are considered as a pre-processing step applied to select the best subset of features before the data mining process is applied. Thus filter methods are completely independent of the posterior learning algorithm. Wrappers are on the opposite side. They rely on the machine learning process to perform feature selection, using it as a black box to score different subsets of features proposed by a search algorithm until a termination criterion is met. Finally, embedded approaches are specific to some learning machines, since they select the best subset of features within the machine's training phase.

Wrapper methods are commonly used since they are inherently simple. They essentially consist of a machine learning procedure, a search algorithm, and a way to determine the prediction accuracy of the learning machine to guide the search towards good feature subsets [6]. Additionally, since wrapper methods select the features subset with the aid of the learning machine that will be applied later to the test set, they generally obtain better accuracy than filter methods, although they are also more computationally demanding [7,8].

* Corresponding author.
E-mail address: jesusgonzalez@ugr.es (J. González).

https://doi.org/10.1016/j.neucom.2021.08.003
0925-2312/

Regarding the learning machine, this paper is focused on classification problems. Thus, the learning machine is a classifier. Many classifiers have been tested within wrapper methods, such as ID3 [6], Naive Bayes Classifier (NBC) [6,9–11], k-Nearest Neighbors (KNN) [12–17], Fisher's Linear Discriminant Analysis (LDA) [11,18,19], or Support Vector Machines (SVM) [16,17,20–23]. However, the behavior of some of these classifiers depend on some hyperparameters that need to be fine tuned, according to the dataset, in order to achieve a good accuracy. For example, SVM [24] depends on both the regularization hyperparameter C and the set of hyperparameters determining the type of kernel used. The correct setting of these hyperparameters is a fundamental issue, mainly because the final result of the wrapper procedure will depend on them. The problem is that the values of these hyperparameters depend on the final data defined by the selected features too, which is a priori unknown.

Some methods set these hyperparameters heuristically before the wrapper procedure is applied. For instance, in [20,21] they are initialized using the whole dataset (containing all the features) before the application of the wrapper procedure. Nevertheless, the initial values obtained for the hyperparameters might not be optimal for the definitive subset of features found by the wrapper procedure. Besides, different values for the initial hyperparameters could obtain a different features subset.

Thus, wrapper methods involve two problems that should be optimized simultaneously. Obviously, the number of features should be minimized, but since the classifier used within the search algorithm may depend on several hyperparameters, these hyperparameters should also be optimized in order to avoid a biased result [25]. This is the first hypothesis that motivates this work. The joint optimization of these two interdependent problems should improve the results. Cooperative Co-Evolutionary Algorithms (CCEAs) are particularly appropriate to this scheme, considering they were designed to co-evolve different species of solutions at the same time [26,27].

According to [28], CCEAs can be implemented at two basic levels, depending on how the problem is decomposed. Single-level CCEAs divide a large problem into smaller components or sub-problems which evolve separately, whereas two-level co-evolutionary approaches divide the problem into two species, one evolving components and another one evolving complete systems built from these components. In this case, the fitness of each component is estimated based on its contribution to the systems which it is included in.

Regarding feature selection problems, wrapper approaches based on CCEAs were formerly implemented as two-level approaches. For example, in [29,30] two species were used, one to evolve the feature subset and another to optimize the classifier. However, with the advent of big data, single-level approaches are now preferred to solve large-scale optimization problems. For such kind of problems, input variables are separated and grouped into several species, taking into account different heuristics, to leverage of the parallel computing platforms available nowadays [31–33,17]. Both approaches have advantages and drawbacks. On the one hand, two-level approaches allow the optimization of the classifier simultaneously with the feature selection process, although they cannot leverage parallel computing architectures since they use only two subpopulations. On the other hand, single-level approaches are designed to use all the available computing power, as they split the problem into many species that evolve independently. However, they are only focused on the feature selection problem, with an a priori fixed classifier, which introduces a bias in the feature selection problem.

There also exist many variants of the classical CCEA, such as [17], a single-level Particle Swarm Optimization (PSO) based approach, which implements an adaptive adjustment mechanism of subswarms to save computational cost on evaluating particles, or [34–37], which propose a completely different application of co-evolution, named Multiple Populations for Multiple Objectives (MPMO), where each subpopulation is focused on the optimization of a different single objective.

On the other hand, several objectives should be considered to guide the search towards a good combination of classifier hyperparameters and subset of features. First and foremost, the classification error and generalization capability should be optimized, since the goal of feature selection is to discover the subset of features that best characterize the original data. Depending on how these objectives are estimated, one or more objectives could be defined for this purpose. For example, in [38] only the misclassification error is used, whereas in [39] the sensitivity and the specificity are used for these purposes, and in [40] the misclassification rate and the minimization of imbalance in class sizes are applied. Another objective usually taken into account is the size of the features subset, which should also be minimized [38,40–42]. Finally, since the hyperparameters of the classifier are being optimized too, some objectives could also be defined, depending on the type of classifier used. Thus, we are dealing with a Multi-Objective Problem (MOP).

Taking into account that two problems must be solved simultaneously (the optimization of the classifier hyperparameters and the minimization of the set of most representative features), and that co-evolutionary algorithms perform much more evaluations per generation than evolutionary algorithms, since each individual in each subpopulation is usually evaluated several times to estimate its fitness, the simpler multi-objective handling scheme applied within the CCEA the better. Probably the simplest approach to solve a MOP is lexicographic optimization [43]. Lexicographic optimizers try to satisfy all the objectives in order. First, the most important objective is considered. Then, among the solutions meeting this objective, a subset of solutions is selected to satisfy the second objective, and so on until all the objectives have been processed [44]. Thus if a different priority level can be established for each objective, the MOP becomes a Lexicographic MOP (LMOP) [45]. Although it may seem a rather basic approach, there are relevant LMOPs that have been successfully solved with it, even nowadays, such as the design and optimization of integrated vehicle control systems [46] or the design of autonomous vehicles [47].

There are also more sophisticated priority handling schemes. For example, one of the first works dealing with Decision-Maker (DM) preferences about objectives in MOPs was [48], which proposes a modified Pareto-ranking procedure incorporating goals and priorities for each objective. In this approach, objectives are grouped in several priority levels and also assigned a desired goal. Then, the ranking procedure compares the objectives by groups, starting with the highest priority groups. For each group, a modified Pareto-dominance criterion is used that only takes into account those objectives not meeting their corresponding goals. Only in the case that all the goals are met, the following priority group is considered. Later on, the favor relation was introduced in [49], which relaxes the classical dominance criterion by counting the number of objectives where a solution is better than, the same as, or worse than another. Based on the favor relation, the priority-favor relation, which modifies it allowing the arbitrary assignment of priorities to each objective, was proposed in [50]. The same authors have also proposed the ϵ-preferred and prio-ϵ-preferred relations [51], which are modifications of the favor and priority-favor relations, where a limit or ϵ-value is defined for each objective. However, if a different priority level can be defined for each objective, lexicographic optimization is preferred. This is the second hypothesis of this work: the co-evolution of the classifier hyperparameters, while the best subset of features is found, can be formulated as an LMOP.

Thus, the main contribution of this paper is approaching the feature selection problem as two co-evolving problems, the optimization of the classifier hyperparameters while the smallest subset of features is also being determined, and formulating this co-evolution as an LMOP, introducing a new lexicographic fitness function that minimizes the runtime of the wrapper procedure and avoids over-fitted solutions. Other contributions are:

- A study of the hyperparameters influence in the accuracy and number of features finally selected, and also in the wrapper procedure execution time, in order to reach an adequate balance between the quality of solutions and the wrapper procedure training time.
- A new lexicographic ranking methodology, based on pairwise comparisons of the p-values returned by the non-parametric Kruskal–Wallis statistical test, able to compare the average results of many feature selection methods on several datasets, taking into account multiple objectives.
- The application of the proposed wrapper procedure to some real high-dimensional datasets, obtaining high stable results.

The rest of the paper is organized as follows. Section 2 details the lexicographic relation for MOEAs, a relation that makes possible the full ranking of candidate solutions for a MOP where a different level of priority can be assigned to each objective. Later, Section 3 describes in detail the Lexicographic Optimization Cooperative Co-Evolutionary Algorithm (LeOCCEA), which simultaneously minimizes the number of features needed to describe a dataset while the hyperparameters of the classifier within a wrapper method are also being optimized. Then, Section 4 describes the different metrics considered in this paper to evaluate the solutions of the evolutionary algorithm and proposes a new lexicographic fitness function that aims to reduce both the computation time and the possible over-fitting of solutions, while optimizing the classification accuracy and reducing the number of selected features. After that, Section 5 studies the influence of the main hyperparameters of LeOCCEA, the percentage of training samples used for validation (p_{val}) and the vector of similarity thresholds applied in the lexicographic comparison of two solutions ($\boldsymbol{t}_l$), in the classification accuracy for test data, the number of features finally selected, and the computation time of the wrapper method. Afterwards, Section 6 compares the results obtained by LeOCCEA with those obtained by other wrapper methods using several datasets from the UCI machine learning repository [52], and Section 7 applies LeOCCEA to some real high-dimensional classification problems. Finally, Section 8 concludes this work.

2. A lexicographic relation for MOPs

As introduced above, lexicographic optimizers prioritize all the objectives and then try to satisfy them in order of priority. Thus, assuming a problem where n_o objectives have been defined, and also that these objectives can be sorted according to their priority, the fitness for any solution for the problem can be expressed as:

$$\boldsymbol{f} = \left[f^0, f^1, \ldots, f^{n_o-1}\right]^T \in \mathbb{R}^{n_o} \tag{1}$$

Depending on the kind of objectives being optimized, and also on the DM criteria, given two fitness evaluations $\boldsymbol{f_1}$ and $\boldsymbol{f_2}$, a difference between two fitness values in a given objective $o^i, d^i = |f_1^i - f_2^i|$, may be considered irrelevant or quite significant. Also, distinct precisions may be desired for the different objectives taken into account. Therefore a vector of n_o similarity thresholds $\boldsymbol{t}_l$ is introduced to let the DM setting the precision used to perform the comparison of each objective:

$$\boldsymbol{t}_l = \left[t_l^0, t_l^1, \ldots, t_l^{n_o-1}\right]^T \in \mathbb{R}^{n_o}_{\geqslant 0} \tag{2}$$

Two fitness values for an objective o^i will be considered similar if

$$|f_1^i - f_2^i| < t_l^i \tag{3}$$

If a traditional lexicographic comparison is desired, as introduced in [43], the DM only has to fix $t_l^i = 0, \forall i \in [0, n_o) \cap \mathbb{N}$.

Thus, the lexicographic relations between them, noted as $\prec_l$ and $\preceq_l$, are defined as [25]:

$$\boldsymbol{f_1} \prec_l \boldsymbol{f_2} \iff \exists k \in [0, n_o) \cap \mathbb{N} : f_1^k < f_2^k \wedge |f_1^k - f_2^k| \geqslant t_l^k \wedge |f_1^i - f_2^i| < t_l^i, \forall i < k \tag{4}$$

$$\boldsymbol{f_1} \approx_l \boldsymbol{f_2} \iff |f_1^i - f_2^i| < t_l^i, \forall i \in [0, n_o) \cap \mathbb{N} \tag{5}$$

$$\boldsymbol{f_1} \preceq_l \boldsymbol{f_2} \iff \boldsymbol{f_1} \prec_l \boldsymbol{f_2} \vee \boldsymbol{f_1} \approx_l \boldsymbol{f_2} \tag{6}$$

A fitness evaluation $\boldsymbol{f_1}$ will be better than another $\boldsymbol{f_2}$ (4) if and only if there exists an objective k such that $f_1^k < f_2^k$, that is, $\boldsymbol{f_1}$ improves $\boldsymbol{f_2}$ in objective k. The difference in such objective must also be higher than or equal to the similarity threshold t_l^k, that is $|f_1^k - f_2^k| \geqslant t_l^k$, while the difference in more important objectives (objectives lower to k) must be lower than the similarity threshold t_l^i ($|f_1^i - f_2^i| < t_l^i, \forall i < k$).

A fitness evaluation $\boldsymbol{f_1}$ will be similar to another $\boldsymbol{f_2}$ (5) if and only if the difference between each pair of objectives is lower than the similarity threshold t_l^i. That is, if $|f_1^i - f_2^i| < t_l^i$ for each objective i taken into account.

Finally, a fitness evaluation $\boldsymbol{f_1}$ will be better than or similar to another $\boldsymbol{f_2}$ (5) if and only if $\boldsymbol{f_1}$ is better than $\boldsymbol{f_2}$ (4) of if $\boldsymbol{f_1}$ is similar to $\boldsymbol{f_2}$ (5).

The behavior of the algorithm using this relation resembles the classical lexicographic optimization algorithms. It processes the objectives in order, but with an important difference. The search is not sequential, since an EA is being applied, what provides the search algorithm a mechanism to escape from local optima [53].

The use of this lexicographic relation within a MOP has many benefits. Since the population can be fully ranked, a simple EA, with smaller populations, can be applied. Another advantage of this approach is that the algorithm will provide only one optimal solution, composed of the combination of the best solution found in each subpopulation, instead of a large set of Pareto-optimal solutions, which greatly helps the DM. Besides, since priorities of objectives are defined according to the characteristics of the problem, the algorithm will search only towards solutions meeting this restriction, greatly reducing the search space.

3. The LeOCCEA wrapper method

This section describes extensively the LeOCCEA wrapper method, formerly introduced in [25]. This wrapper method is able to optimize the hyperparameters of the classifier while the set of features is also being minimized. Fig. 1 shows its flowchart, highlighting those steps that have been modified from the original CCEA to achieve LeOCCEA, along with the sections in this paper that detail these changes. As can be seen, subpopulation 0 optimizes the classifier hyperparameters whereas the rest of subpopulations are centered on solving the feature selection problem. Each subpopulation evolves individuals of a different species, being necessary an individual of each one of the species to form a complete solution for the two problems. Each time a complete solution is evaluated, the resulting fitness contributes to the fitness of all the individuals that have used to build that solution.

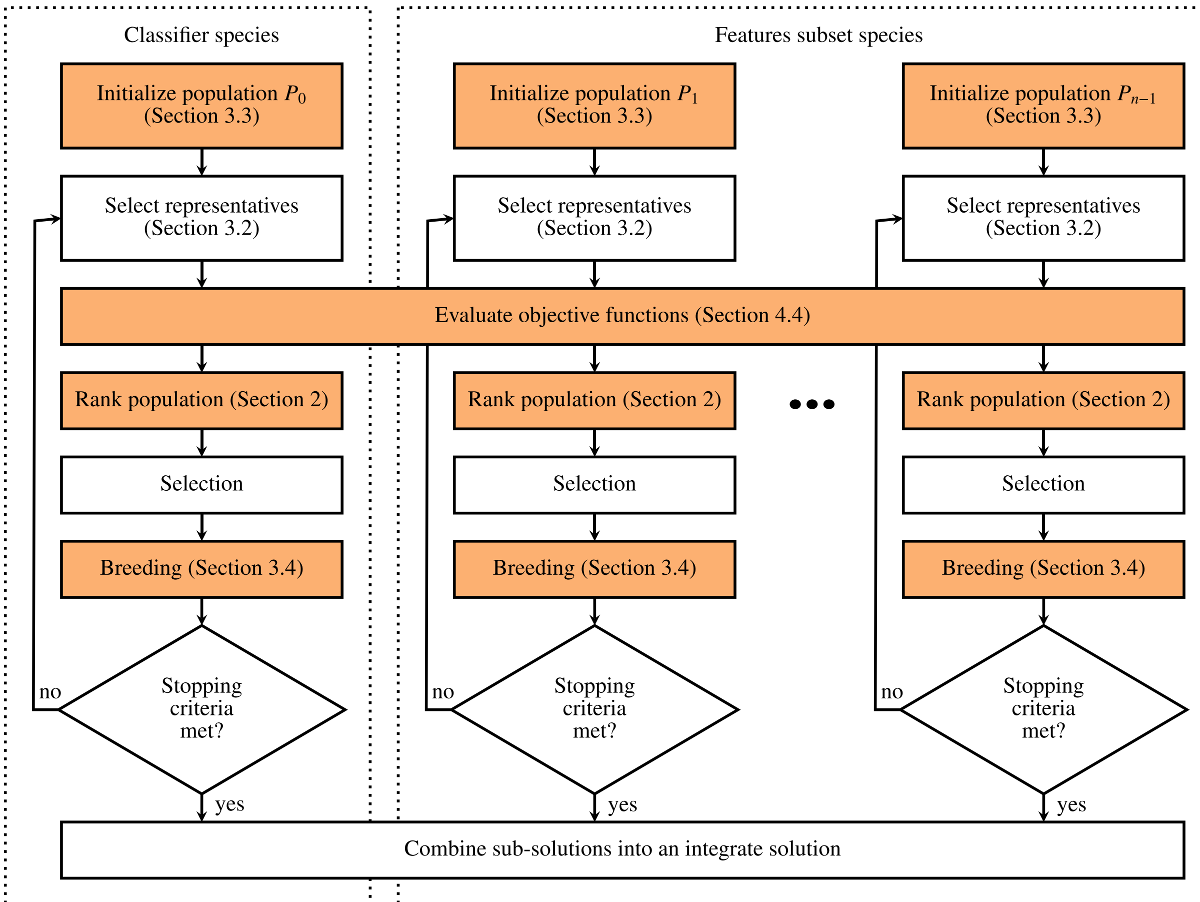


Fig. 1. Flowchart of the LeOCCEA wrapper method. The steps that are not highlighted are taken from the original CCEA.

3.1. Structure

Since LeOCCEA is based on a CCEA, potential solutions for the problem are co-evolved by different species, each one in a separate subpopulation. The most direct approach could be to use a two-level approach. However, a pure two-level approach would only use two species, one to optimize the hyperparameters of the classifier and another to minimize the features subset. This would not be an even division, since the number of hyperparameters needed to define a classifier is quite lower than the number of features in almost any feature selection problem, and thus, the search space would be much larger for the feature selection species, especially in high-dimensional problems. Moreover, such a kind of division would limit the exploitation of current parallel computing platforms drastically, since only two subpopulations would be used to implement the algorithm. In fact, recent CCEA-based wrapper methods are based on a single-level approach, mainly to leverage the high-performance computing platforms available nowadays. However, single-level approaches do not optimize the classifier. They only minimize the number of selected features. Therefore, LeOCCEA is based on a hybrid single-level and two-level approach. One species evolves the hyperparameters of the classifier while the input features are also split among several species (see Fig. 1). The number of species needed to process the features of the input dataset is not fixed a priori and should be adjusted for each problem according to the number of features in the dataset being processed and the number of computing nodes available, to balance the search spaces of all the subpopulations.

3.2. Fitness evaluation

As introduced above, complete solutions for both co-evolving problems are formed by the collaboration of one representative individual from each one of the subpopulations. Once a complete solution is evaluated, the obtained fitness contributes to the final fitness of each one of the individuals used to build that solution, which is finally obtained as a statistic (the better, worse, or average value, for example) of all the fitness values obtained by the individual in the different collaborations it has been involved.

The collaboration strategy affects both the computation time and fitness estimation of the individuals. On the one hand, the more collaborations established to estimate the fitness of each individual, the better fitness estimation, but also the greater computation time. On the other hand, fewer collaborations minimize the computation time and worsen the fitness estimation of individuals, increasing the possibility of converging to sub-optimal solutions.

The cheaper individual-centric method, in terms of computation time, is the single-best collaboration method [27], where each individual collaborates with the best one of the remaining species to be evaluated. Assuming n_p subpopulations of size $m_i\,(i = 1, \ldots, n_p)$, the number of evaluations needed to assign a fitness value to all the individuals in a generation would be:

$$n_{e_{\text{best}}} = \sum_{i=0}^{n_p-1} m_i \tag{7}$$

Since this approximation is too greedy and might guide the algorithm towards local optima, more representatives could be chosen (randomly from each subpopulation, only from the Pareto front of each species, etc.). For a number n_r of representatives, the number of evaluations needed would increase linearly:

$$n_e(n_r) = n_r \sum_{i=0}^{n_p-1} m_i \tag{8}$$

In this case, each individual of each species is evaluated n_r times, with the best result being assigned as its fitness [54].

However, population-centric approaches allow the evaluation of all the individuals while minimizing the overall number of evaluations, avoiding also the greediness of individual-centric approaches [55]. Specifically, the shuffle-and-pair method shuffles the indices to access individuals in each subpopulation and then recombines all the individuals having the same index to form and evaluate a complete solution for the problem. Since only one evaluation per individual may poorly estimate its fitness, this process can be repeated n_r times to obtain a better evaluation for each individual. The only limitation for this method is that all the subpopulations must have the same size m. The total number of evaluations needed to complete a generation would be:

$$n_{e_{s\&p}}(n_r) = n_r \times m \tag{9}$$

which, on average, is n_p times lower than the number of evaluations needed for individual-centric methods. Considering that the hybrid single-level and two-level approach followed by LeOCCEA can make use of any number of subpopulations to balance the search space assigned to each species, the restriction of all subpopulations having the same size is completely irrelevant. Even more, as the number of features in the input dataset increases, more subpopulations will be needed, but since the number of subpopulations n_p no longer affects the overall number of evaluations needed, the shuffle-and-pair method is completely scalable for high-dimensional problems. Thus, this is the collaboration method chosen for LeOCCEA.

3.3. Species representation

Since LeOCCEA is based on a hybrid single-level and two-level approach, different representations for the species are needed. Specifically, one representation for the hyperparameters defining the classifier, and another one for the subsets of features co-evolved in the remaining species.

3.3.1. Classifier species

Some classifiers, such as KNN or SVM rely on configuration hyperparameters. Thus, these hyperparameters are encoded as a vector of floating-point numbers in the first subpopulation (P_0). In this case, the classifier applied within the wrapper procedure is an SVM based on a Radial Basis Function (RBF) kernel. Thus, S_0 is defined as follows:

$$S_0 = [C, \gamma]^T \in \mathbb{R}^2 \tag{10}$$

Each individual belonging S_0 encodes a possible value for the regularization hyperparameter of the SVM (C) and a possible width for its RBFs (γ).

3.3.2. Features subset species

The representation proposed in [11] is also used for the features subset species, but with a couple of differences: features are distributed over several subpopulations, and now there is not a maximum size for the set of features provided by the wrapper method. For a problem of n input features, indexed from 0 to $n-1$, and n_p subpopulations, assuming that subpopulation P_0 evolves the classifier hyperparameters, an individual belonging to subpopulation P_j is defined as:

$$I_j \subset S_j \tag{11}$$

with S_j being the whole subset of features evolved by the species in P_j:

$$S_j = \{x \in [a_j, b_j) \cap \mathbb{N}\}, 0 < j < n_p \tag{12}$$

That is, each species S_j is defined as the interval of natural numbers $[a_j, b_j)$ with a_j and b_j defined as:

$$a_j = (j-1)\lceil \frac{n}{n_p - 1} \rceil \tag{13}$$

$$b_j = min\left(j\lceil \frac{n}{n_p - 1} \rceil, n\right) \tag{14}$$

3.4. Breeding operators

Since each subpopulation co-evolves a different species, new breeding operators are needed for each one of them. The breeding operators for both, the classifier hyperparameters and the subsets of features, are described below.

3.4.1. Breeding operators for the classifier species

Given that species S_0 is represented by a vector of real numbers, Simulated Binary Crossover (SBX) [56] and polynomial mutation [57] are applied within the classifier hyperparameters species since both were specifically designed to deal with real numbers. These operators are based on a polynomial distribution depending on a user-defined index parameter v, which is usually fixed to 20 as standard default value.

3.4.2. Breeding operators for the features subset species

Concerning the features subset species, the breeding operators have also been adapted from those proposed in [11], in a way that generated offspring must belong to the same species as their parents. In what follows, the crossover and mutation operators are defined.

Crossover operator. Given a couple of individuals, I_{j_k} and I_{j_l}, belonging to subpopulation P_j, the two offspring, O_{j_k} and O_{j_l}, are obtained as follows.

Let $C_{j_{kl}}$ be the subset of features that have been selected by both I_{j_k} and I_{j_l}, that is, their common features:

$$C_{j_{kl}} = I_{j_k} \cap I_{j_l} \tag{15}$$

Let also $R_{j_{kl}}$ be the remaining features in I_{j_k} and I_{j_l}, once common features are removed:

$$R_{j_{kl}} = \left(I_{j_k} \cup I_{j_l}\right) \setminus C_{j_{kl}} \tag{16}$$

The offspring O_{j_k} and O_{j_l} are obtained as:

$$O_{j_k} = C_{j_{kl}} \cup R_{j_k}, \quad O_{j_l} = C_{j_{kl}} \cup R_{j_l} \tag{17}$$

provided that:

$$R_{j_l} \cup R_{j_k} = R_{j_{kl}} \text{and} R_{j_k} \cap R_{j_l} = \varnothing \tag{18}$$

$$|O_{j_k}| = |I_{j_k}| \text{and} |O_{j_l}| = |I_{j_l}| \tag{19}$$

All the selected features that are common in I_{j_k} and I_{j_l} are common in O_{j_k} and O_{j_l} too, since $O_{j_k} \cap O_{j_l} = C_{j_{kl}}$. The remaining subset of selected features $R_{j_{kl}}$, which are not common in I_{j_k} and I_{j_l}, are randomly distributed between R_{j_k} and R_{j_l} (18) assuring that the sizes of O_{j_k} and O_{j_l} match the sizes of I_{j_k} and I_{j_l} respectively (19). This crossover procedure always generates solutions conforming the constraints stated in Section 3.3.2 for the features species.

Mutation operator. This operator may alter each individual's gene (a selected feature) separately. Given an individual I_j belonging to a subpopulation P_j with $0 < j < n_p$, a gene mutation probability of p_m, and a random variable X following a standard uniform distribution ($X \sim \mathscr{U}(0,1)$), let M_j be defined as the random subset of features in I_j that will be mutated:

$$M_j = \{i \in I_j : X(i) \leqslant p_m\} \tag{20}$$

where $X(i)$ denotes the probability that feature i is mutated.

Once M_j is obtained, two possibilities exist to mutate all its elements. Each one of them could be modified or removed. Thus, M_j is randomly split into two new subsets, M_{j_s} and M_{j_r}, the elements of M_j that will be substituted and those that will be removed respectively:

$$M_{j_s} = \{m \in M_j : X(m) \leqslant 0.5\}, \quad M_{j_r} = M_j \setminus M_{j_s} \tag{21}$$

The mutated individual $I\prime_j$ is be obtained as:

$$I\prime_j = (I_j \setminus M_j) \cup N_{j_s} \cup N_{j_a} \tag{22}$$

where N_{j_s} is the subset of new features that substitutes those belonging M_{j_s}:

$$N_{j_s} \subset S_j \quad |N_{j_s}| = |M_{j_s}| \text{ and } N_{j_s} \cap I_j = \varnothing \tag{23}$$

and N_{j_a} is a subset of at most one new feature that will be added to $I\prime_j$, to make possible the increment of features in $I\prime_j$ respect to the original I_j:

$$N_{j_a} \subset \{x \in (S_j \setminus I_j) \setminus N_{j_s}\}, \quad |N_{j_a}| \in \{0,1\} \tag{24}$$

with S_j being defined as in (12).

4. Lexicographic fitness function proposal

As introduced above, several objectives should be taken into account to co-evolve the classifier hyperparameters and the best subset of input features simultaneously. Besides, since two problems are being jointly optimized, the objectives should cover both problems.

4.1. Metrics related to the feature selection problem

It seems clear that both, the size of the selected features subset and the classification accuracy obtained with it should be taken into account for a feature selection problem. The number of selected features can be measured easily. However, there are several metrics to estimate the accuracy of a classifier, such as the error rate [40] or the combination of sensitivity and specificity [39]. However, the Kappa index [58] has been finally chosen because it takes into account the accuracy of the classifier and also the per class error distribution. This index is defined as follows:

$$\kappa(\mathscr{C}, D_I) = \frac{p_o(\mathscr{C}, D_I) - p_e(\mathscr{C}, D_I)}{1 - p_e(\mathscr{C}, D_I)} \tag{25}$$

where $p_o(\mathscr{C}, D_I)$ is the relative observed agreement between the classifier $\mathscr{C}$ and the labeled data in the dataset D_I, (identical to accuracy), and $p_e(\mathscr{C}, D_I)$ is the hypothetical probability of chance agreement between the classifier $\mathscr{C}$ and the labeled data in D_I.

Special care has to be taken, especially when training with small datasets, to avoid over-fitting. Many works suggest the use of cross-validation when training classifiers [6,12]. Nevertheless, and although this approach has proven successful, it presents a critical inconvenience. It is quite computationally demanding, since all the potential solutions explored by the search algorithm must be evaluated several times. This drawback is even more serious in the case of CCEA approaches, since all the individuals in each species must be re-evaluated in every generation because their fitness depend on their collaboration with some individuals belonging to the remaining species too. Thus, less demanding alternatives are proposed below.

4.2. Metrics related to the optimization of SVM classifiers

The behavior of SVM classifiers depend on a regularization hyperparameter C that controls the trade-off between their training error and their generalization capability. Large values of C choose a smaller-margin separating hyperplane to minimize the training error, while small values try a larger-margin hyperplane, even if some training samples are misclassified. Thus, smaller values of C are preferred, since small margins may cause over-fitting, in particular for small datasets, and large margins generally lower the generalization error [59,60].

On the other side, and for most SVM implementations, training time may raise dramatically with large values of C. This is the case of the Sequential Minimal Optimization (SMO) algorithm, a widely used training algorithm for SVMs. In [61,62] it is shown how the increase in training time at large C values is sharp. This behavior can also be observed in the LibSVM library, since it is based on the WSS3 learning algorithm, a variant of the SMO algorithm [63].

Therefore, smaller values of C increase the generalization capability of the results provided by any wrapper method while minimizing its training time.

4.3. The VT fitness function

This is the lexicographic fitness function originally proposed in [25], where LeOCCEA was firstly introduced. It is based on the maximization of the Kappa index to estimate the accuracy of the classifier, the minimization of the number of features and the minimization of C, in this order. However, the Kappa index is not applied over the whole training set. As introduced above, the fitness function should prevent over-fitting. Thus, the VT fitness function takes the idea of distributed cross-validation proposed in [11], which saves a great amount of computation time respect to the original cross-validation method.

Given a subset of input features I, a classifier $\mathscr{C}$ and an input dataset D, first the selected features coded in I are extracted from D, generating a reduced dataset D_I. Then, D_I is divided randomly into two new subsets, $D_{I_{tr}}$ and $D_{I_{val}}$ according to p_{val}, a hyperparameter indicating the rate of samples used to validate the classifier results with these selected features. This random division of D_I is also stratified, i.e., it ensures that a percentage p_{val} of samples of each class in D_I is always included in $D_{I_{val}}$. Then, classifier $\mathscr{C}$ is only trained with samples in $D_{I_{tr}}$, and later, two accuracies are evaluated: the Kappa indices obtained by the classifier using $D_{I_{tr}}$ and $D_{I_{val}}$ separately. Since all individuals are evaluated each generation because their fitness also depend on the collaborators used to form a complete solution for the problem, a sort of cross-validation, distributed over all the generations of the wrapper method, is finally carried out, with the benefit that each solution is evaluated only twice (for $D_{I_{tr}}$ and $D_{I_{val}}$) instead of the five or ten folds typically used with cross-validation. Another advantage of this method is that final solutions are not biased due to the way D is split, since different subsets $D_{I_{tr}}$ and $D_{I_{val}}$ are randomly generated each time an individual is evaluated.

In the case that two solutions achieve similar Kappa indices, the one with fewer features is preferred, and in the case that both solutions have similar Kappa indices and number of features, a smaller C is preferred. That is, the generalization performance of the final solution for both problems is optimized with two different objectives (one for each problem). The most important one is the validation Kappa index, but if two solutions have similar Kappa indices, a lower value of C improves the generalization capability of the classifier, since an SVM with wider margins is obtained. Thus, this lexicographic fitness function optimizes, in this order, the following objectives:

1. Maximize $\kappa(\mathscr{C}, D_{I_{val}})$.
2. Maximize $\kappa(\mathscr{C}, D_{I_{tr}})$.
3. Minimize the number of features.
4. Minimize the regularization hyperparameter C of the SVM.

The name of the fitness function (VT) comes from the use of both, the Validation and also the Training Kappa indices, to estimate the accuracy of solutions.

4.4. The VO fitness function

The VT fitness function was originally proposed to show how LeOCCEA was able to solve even Many-Objective Optimization Problems (MaOPs), but the fact is that it may cause over-fitting for really small datasets with a large number of features, since $\kappa(\mathscr{C}, D_{I_{tr}})$ is used as the second objective. Thus, this paper proposes the use of only $\kappa(\mathscr{C}, D_{I_{val}})$ to estimate the accuracy of solutions, that is the use of the Validation index Only (VO).

In consequence, the new lexicographic fitness function proposed in this paper only optimizes, in this order, these objectives:

1. Maximize $\kappa(\mathscr{C}, D_{I_{val}})$.
2. Minimize the number of features.
3. Minimize the regularization hyperparameter C of the SVM.

The reduction from four to three objectives makes the problem to become a MOP instead a MaOP, although this is not an issue for lexicographic methods. Moreover, it prevents over-fitting when selecting features from datasets with a large number of features and a low number of samples and also reduces the computation time of the wrapper method, since only one Kappa index is calculated for each solution instead of two. On the other side, the effect of distributed cross-validation remains because individuals are evaluated with a different $D_{I_{val}}$ each time, as described above.

5. LeOCCEA hyperparameter setting

The behavior of LeOCCEA depends on its two configuration hyperparameters: the vector of similarity thresholds $\boldsymbol{t}_l$, used by the lexicographic comparison of two individuals, and the percentage of samples p_{val} used to split the original training dataset D into $D_{I_{tr}}$ and $D_{I_{val}}$ each time an individual I is evaluated. Both the computation time and the balance between the minimization of the error rate or the number of features are affected by these hyperparameters, as shown below. This effect has been analyzed by means of an empirical study applying the wrapper method to six different datasets from the UCI machine learning repository [52]. Table 1 shows the details of these datasets. For all the experiments, each dataset has been randomly divided (stratified) into two separate sets as proposed in [12,22,70]: a test set containing 30% of all samples in each class, and a training set formed by the rest of samples.

Regarding p_{val}, in [11] it was fixed to 0.3, and although quite good results were obtained, the fact is that no other values were tried. Thus, in this work the value 0.5 is tested too. A reduction in the execution time is expected for this new value, since less training data will be used. However, the effect of this increment in the value of p_{val} in both the accuracy of the solutions and the number of features finally selected is unclear. On the other side, the vector of thresholds $\boldsymbol{t}_l$ regulate the maximum difference allowed between two values for each objective to be considered similar in their lexicographic comparison. Thus, the lower value of t_l^i, the higher probability of obtaining over-fitted solutions with respect to objective o^i, while high values of t_l^i will not guide the search properly towards o^i optima values because quite different values for o^i will be considered similar.

Although a different threshold value t_l^i can be fixed for each objective o^i, the use of the same value $t_l \in (0,1)$ for all thresholds in $\boldsymbol{t}_l$ was proposed in [25]. This decision is justified because:

- The number of selected features is always an integer value. Thus any value in $(1,0)$ is valid to distinguish two different integer values in (3).
- The Kappa index and the C regularization hyperparameter of SVMs are both real numbers, defined in $[-1,1]$ and in $(0,\infty)$ respectively. A priori it seems that these two objectives should have different similarity thresholds, since their range is also quite different. However, since the value of C is minimized to avoid over-fitting and reduce the training time, optima values for C will be in (0,1) most of the time. Thus, the same similarity threshold $t_l \in (0,1)$ can be fixed for both objectives.

Originally, t_l was fixed to 0.001 in [25], but other values may change the behavior of the wrapper method. So, different values for this hyperparameter have been tested. Concretely, values $0.001, 0.005, 0.01, 0.05, 0.1$ and 0.2 have been analyzed. The rest of hyperparameters of the wrapper method has been fixed as Table 2 shows.

The number of executions of the wrapper method (n_e) has been fixed to 40, as in [71]. On the other hand, the number of species (subpopulations) is different for each dataset, because it depends on the total number of features of each problem, as described above. Table 3 shows the number of subpopulations used for each one of the datasets of Table 1. The heuristic applied to fix these values has been that the number of features assigned to each subpopulation should be 4 or 5.

Regarding the wrapper method implementation, the base co-evolutionary algorithm, as well as the breeding operators applied within subpopulation P_0, which evolves the classifier hyperparameters, have been taken from ECJ [72], a research Evolutionary Computation (EC) framework written in *Java* and developed within the *Evolutionary Computation Laboratory* at the

Table 1
Datasets.

Dataset	# Features	# Classes	# Instances
German credit [64]	24	2	1 000
Johns Hopkins University Ionosphere [65]	34	2	351
Vehicle silhouettes [66]	18	4	846
Wisconsin Diagnostic Breast Cancer (WDBC) [67]	30	2	569
Wine recognition [68]	13	3	178
Zoo [69]	16	7	101

Table 2
Hyperparameters of the LeOCCEA wrapper method.

Parameter	Value
Subpopulations size (m)	150
Number of generations (n_g)	300
Feature selection species mutation probability $(p_{m_{fs}})$	0.01
SVM hyperparameters species Mutation probability $(p_{m_{svm}})$	0.05
Number of executions of the wrapper method (n_e)	40
Co-evolutionary evaluation number of shuffles (n_r)	2

Table 3
Number of subpopulations used for each dataset.

Wrapper	Number of subpopulations (n_p)
German credit	6
Ionosphere	8
Vehicle silhouettes	5
WDBC	7
Wine recognition	4
Zoo	5

George Mason University, VA, USA. Moreover, LibSVM has been used to implement the SVM classifiers [73]. The rest of the code has been written by the authors of this work. All the code is implemented within the *ristretto* library, publicly available in [74].

Fig. 2 shows the average values of the test Kappa index, the number of features finally selected and the execution time, over 40 executions of the wrapper method, for the six datasets listed in Table 1, and for all the possible combinations of $p_{val} \in \{0.3, 0.5\}$ and $t_l \in \{0.001, 0.005, 0.01, 0.05, 0.1, 0.2\}$. As expected, the execution time of LeOCCEA is reduced when p_{val} is increased. However, the effect of this hyperparameter on the Kappa index and the number of features changes depending on the datasets. For the Wine and Zoo datasets the Kappa index improves significantly with $p_{val} = 0.5$, while for the rest of datasets the accuracies obtained are similar. Regarding the number of features, it seems that a higher value of p_{val} makes the wrapper method to obtain smaller subsets of features for the Ionosphere, WBCD, Wine and Zoo datasets.

Concerning the similarity threshold t_l, the accuracy of the test Kappa index improves as t_l is increased until it stagnates for values higher than $t_l = 0.01$. This hyperparameter also influences on the number of features selected by the wrapper method the same way, although its effect is more or less clear depending on the dataset.

Thus, taking into account previous considerations, the hyperparameters of the LeOCCEA wrapper method have been fixed to $p_{val} = 0.5$ and $t_l = 0.01$ for the rest of executions performed in this paper, to achieve a compromise between the accuracy of results and the computation time of the algorithm.

6. Comparison with other wrapper methods

Once the hyperparameters of LeOCCEA have been fixed, this section compares its results with different wrapper methods. The same datasets used in the previous section are used for the comparison. The hyperparameters used by these methods are detailed in Table 4. As can be seen, there are quite different configurations. However, the direct comparison of these values with those in Table 2 is not fair, since LeOCCEA co-evolves two problems simultaneously (the optimization of the classifier hyperparameters and the selection of the smallest subset of features that better describe the dataset) while the remaining wrapper methods rely on a pre-parameterized classifier.

6.1. Brief description of the other wrapper alternatives

The results of LeOCCEA have been compared with those obtained by the following procedures. There is a wide variety of search algorithms and classifiers, including SVM, the classifier used by LeOCCEA in this paper.

6.1.1. Linear Forward Selection (LFS):

This wrapper procedure [75] is derived from the well known Sequential Forward Selection (SFS) [76], but with a fundamental difference. LFS limits the number of features considered in each step of the forward selection, which reduces the number of evaluations, optimizing the overall computation time. This method was applied to the datasets listed in Table 1 in [12], using KNN as the classifier with $k = 5$.

6.1.2. Greedy Stepwise Backward Selection (GSBS):

This method is based on the classical Sequential Backward Selection (SBS) algorithm [77]. It begins considering all the available features and takes off one feature per iteration until the removal of any of the remaining features worsens the accuracy of the classifier [78]. This method was also applied in [12] to the datasets listed in Table 1, using the same classifier, KNN with $k = 5$.

6.1.3. Commonly Used PSO Algorithm (ErFS):

This method uses the PSO metaheuristic [79] to minimize the error rate of the classifier. The implementation described in [12] fixes the inertia weight $w = 0.7298$ and the acceleration constants $c_1 = c_2 = 1.49618$, which are specific parameters of PSO. It also applies KNN as the classifier with $k = 5$.

6.1.4. PSO With a Two-Stage Fitness Function (2SFS):

This wrapper procedure, also based on PSO, splits the evolutionary process into two stages. The first one minimizes only the error rate of the classifier, whereas the second stage also considers the number of features in the fitness function [71]. Since this procedure is also proposed in [12], the parameters of both PSO and classifier are fixed in the same way as in ErFS.

6.1.5. Two-phase Mutation Grey Wolf Optimizer (TMGWO):

This wrapper procedure, described in [15], proposes a variant of the Grey Wolf Optimizer (GWO) [80] with a two-phase mutation operator to avoid local optima. This wrapper method applies KNN with $k = 5$ too.

6.1.6. FAM-BSO:

This wrapper method [70] uses the Brain Storm Optimization (BSO) algorithm [81], a swarm intelligence procedure inspired by the human brainstorming process, as search engine. Classification is performed applying the Fuzzy ARTMAP (FAM) model [82], a supervised neural network that combines fuzzy sets theory with Adaptive Resonance Theory (ART) [83].

6.1.7. Binary PSO (BPSO):

This wrapper procedure implements a version of PSO operating on discrete binary variables [84]. In [22] it was applied to some datasets of Table 1 using SVM as classifier.

6.1.8. BSEOA:

This wrapper procedure, proposed in [22], implements a binary version, inspired by BPSO, of the Social Emotional Optimization Algorithm (SEOA) [85], a swarm intelligent population-based optimization algorithm which simulates the decision-making of human beings in society based on human emotion. It also uses SVM as classifier.

6.2. Proposed ranking methodology

Since this paper compares 9 different wrapper procedures, a new methodology is necessary to rank them lexicographically, taking into account both their error rates and number of features, in order to allow their comparison. This paper proposes the use of pairwise comparisons of all methods results using the non-parametric Kruskal–Wallis statistical test.

First of all, two *p*-values are obtained for each possible pair of wrapper methods W_a and W_b, $p_{e_{ab}}$ for their error rates comparison

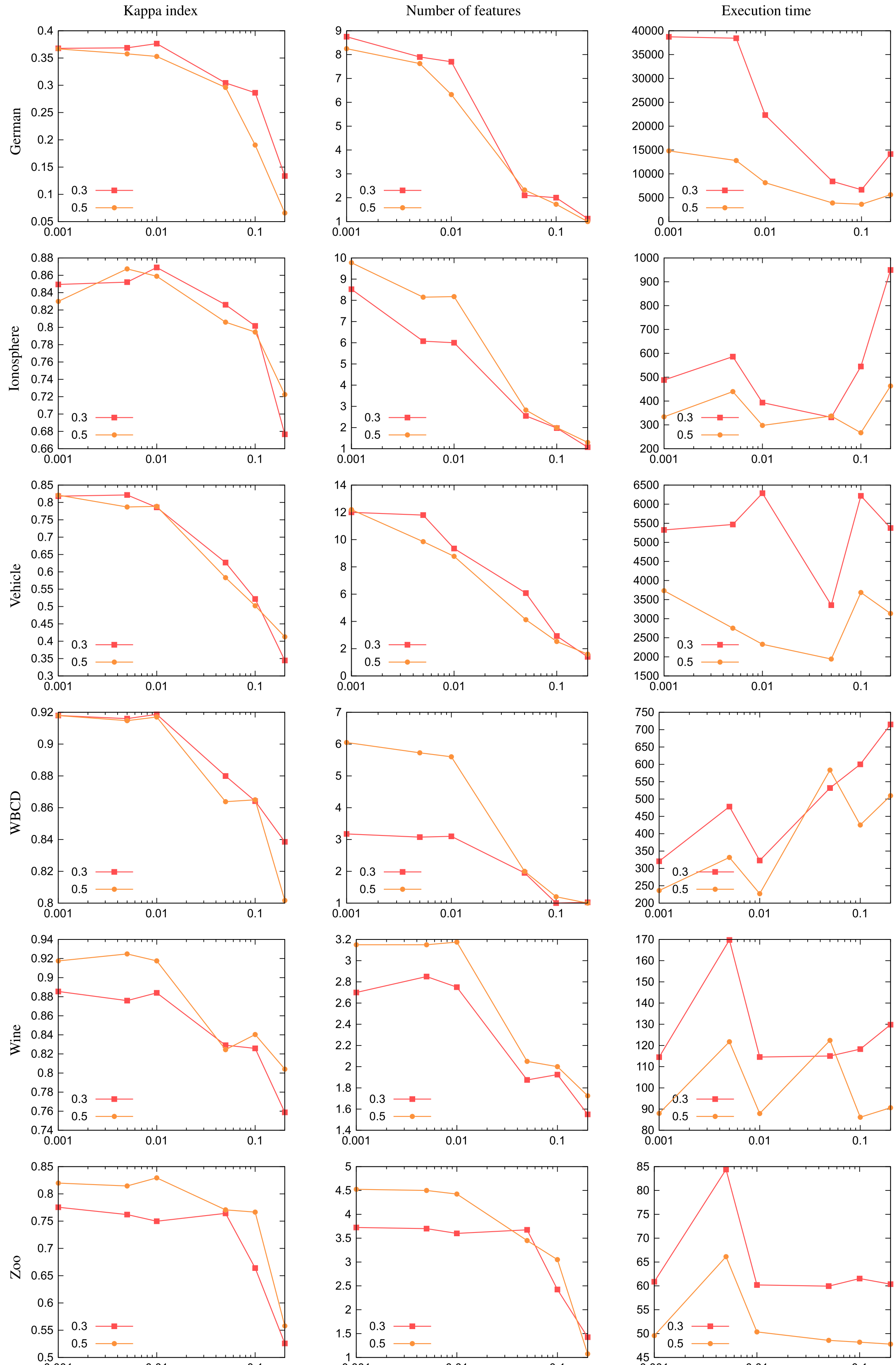


Fig. 2. Average values, over 40 executions of LeOCCEA, of the test Kappa index, number of features and execution time obtained for all the combinations of $p_{val} \in \{0.3, 0.5\}$ and $t_l \in \{0.001, 0.005, 0.01, 0.05, 0.1, 0.2\}$, and six different datasets.

Table 4
Hyperparameter setting of the wrapper methods compared with LeOCCEA. Depending on each method, m refers to the number of individuals, agents, or particles, while n_g is related to the number of generations or iterations, and n indicates the number of features in the dataset.

Wrapper	m	n_g
LFS [12]	*n.a.*	*n.a.*
GSBS [12]	*n.a.*	*n.a.*
ErFs [12]	30	100
2SFS [12]	30	100
TMGWO [15]	5	30
FAM-BSO [70]	100	2000
BSEOA [22]	$10 + 2\sqrt{n}$	50
BPSO [22]	$10 + 2\sqrt{n}$	50

(e_a and e_b), and $p_{n_{ab}}$ related to the difference between the number of features they finally selected (n_a and n_b). Then all the methods can be lexicographically ranked according to these expressions:

$$W_a \prec_l W_b \iff \left(p_{e_{ab}} \leqslant \alpha \ \wedge \ e_a < e_b\right) \ \vee \left(p_{e_{ab}} > \alpha \ \wedge \ p_{n_{ab}} \leqslant \alpha \ \wedge \ n_a < n_b\right) \quad (26)$$

$$W_a \approx_l W_b \iff p_{e_{ab}} > \alpha \ \wedge \ p_{n_{ab}} > \alpha \quad (27)$$

$$W_a \preceq_l W_a \iff W_a \prec_l W_b \ \vee \ W_a \approx_l W_b \quad (28)$$

where α is the significance level used for the comparison.

On the other side, it is quite possible that $W_a \prec_l W_b$ for some datasets while $W_b \prec_l W_a$ for others. So, the comparison should be converted to a real number to allow the computation of average values for all the datasets. This real number is $r_D(W)$, the rank of wrapper method W once the all the wrapper methods have been compared using dataset D:

$$r_D(W) = |B_D(W)| \quad (29)$$

where $B_D(W)$ is the set of wrapper methods that are better than W for D:

$$B_D(W) = \{W_i : W_i \prec_l W\} \quad (30)$$

Thus, an average value of the rank for each wrapper method, $\bar{r}(W)$, can be calculated for a set different datasets Δ:

$$\bar{r}(W) = \frac{1}{n_\Delta}\sum_{i=0}^{i<n_\Delta} r_{D_i}(W), \quad D_i \in \Delta \quad (31)$$

where n_Δ is the number of datasets in Δ.

6.3. Comparison

Tables 5 and 6 show the means of the error rates and number of features obtained by all the wrapper methods for datasets listed in Table 1, except for BPSO and BSEOA, whose results are not available in [22] for the German, Ionosphere and WBCD datasets.

A lexicographic comparison of all the wrapper methods has been made for all the datasets, according to the methodology proposed in Section 6.2. A significance level $\alpha = 0.05$ has been applied to detect statistically different values. Results are shown in Tables 7–12, where cells marked with a letter mean that the wrapper method in the column has a statistically significant better error (E) or a similar error but a statistically better number of features (N) than the method in the row, that is, that the method in the column is lexicographically better than the method in the row. On the other side, cells marked with a tick denote pairs of methods that obtain statistically similar results. As can be seen, these tables also allow to form clusters of similar methods for each dataset.

For the German, Ionosphere and WBCD datasets, Tables 7–9 also show the ranking of each wrapper method in the last row, while for Tables 10–12 two rankings are calculated, with and without BPSO and BSEOA, to allow the comparison of results with those of Tables 7–9, where results from BPSO and BSEOA are not available. Average rankings are calculated in Tables 13 and 14. The former calculates an average ranking considering all the datasets, while the latter considers all the wrapper methods. In both cases it can be appreciated that although LeOCCEA is not the best wrapper method for each one of the datasets, is achieves the best average ranking, with and without considering BPSO and BSEOA, what means that it performs better than the rest of methods on average. These results confirm the two starting hypotheses of this work: the simultaneous optimization of the classifier hyperparameters, while the best subset of representative features is being found, does improve the final results, and the co-evolution of these two interdependent problems can be successfully approached as a lexicographic problem. see Table 8.

6.4. Stability analysis

Stability is another desirable property of any wrapper method, since DMs usually prefer those methods which return consistent feature subsets from multiple runs. Table 15 shows the stability score achieved by LeOCCEA for all the datasets of Table 1. These scores have been calculated using the stability method proposed in [11], which is based on the average Spearman index of the full-ranked lists of features obtained in the different runs of the algorithm. Possible outcomes of this method lie in the range $[-1, 1]$, with 0 indicating no correlation at all, and 1 or -1 indicating a perfect positive or negative correlation, respectively. Thus, the higher (positive) value of the Spearman index, the more stability of the wrapper procedure. As can be seen, LeOCCEA is quite stable for all the datasets. see Table 11.

7. Application to real high-dimensional data

This last section applies LeOCCEA to two real high-dimensional multi-class classification problems. The former is related to the lung cancer diagnosis from microarray data, while the latter consists of the Motion Imagery (MI) classification of three datasets, corresponding to three different subjects.

7.1. Application to a lung cancer diagnosis

The first high-dimensional classification problem where LeOCCEA has been tested is the lung cancer diagnosis from microarray data. The data come from The Cancer Genome Atlas (TCGA) and consist of microarray data of 1 100 subjects with 410 features and three different states: 495 ACC Primary Tumor, 502 SCC Primary Tumor, and 103 Solid Tissue Normal samples, which have been split ($80\% - 20\%$, stratified) into training and test sets after the deletion of 10 outliers. This dataset was also used in [23] to select the most relevant features with several feature selectors, being the best one minimum Redundancy Maximum Relevance (mRMR) [86]. Three different classifiers were also tested on this dataset SVM, KNN, and Random Forest (RF).

Since the number of input features is much larger than those of datasets used in previous sections, LeOCCEA has been run 20 times, during 1 000 generations and using 16 subpopulations (the number of cores in the computing platform) of 1 000 individuals. The remaining hyperparameters have been kept as described above. Table 16 shows the error rate and the number of features finally selected by all the feature selection approaches. At first sight, it seems that LeOCCEA presents better results than all the feature selection approaches presented in [23]. Table 17 presents a pairwise lexicographic comparison of all of them, with a significance level $\alpha = 0.05$, as well as their ranking. LeOCCEA achieves the highest rank, confirming this fact. LeOCCEA also achieves high stable outcomes, reaching a stability score of 0.975. These results confirm

Table 5
Mean test error rate achieved by the different wrapper alternatives.

Method	German	Ionosphere	Vehicle	WBCD	Wine	Zoo
LFS	0.313	0.133	0.169	0.111	0.259	0.210
GSBS	0.357	0.219	0.242	0.164	0.148	0.200
ErFS	0.306	0.116	0.150	0.066	0.040	0.045
2SFS	0.308	0.119	0.151	0.065	0.040	0.045
TMGWO	0.244	0.069	0.262	0.052	0.053	0.040
FAM-BSO	0.171	0.081	0.182	0.035	0.028	0.043
BPSO	*n.a.*	*n.a.*	0.175 ± 0.068	*n.a.*	0.028 ± 0.011	0.013 ± 0.023
BSEOA	*n.a.*	*n.a.*	0.188 ± 0.062	*n.a.*	0.028 ± 0.013	0.008 ± 0.018
LeOCCEA	0.255 ± 0.021	0.067 ± 0.019	0.079 ± 0.007	0.039 ± 0.010	0.036 ± 0.014	0.036 ± 0.010

Table 6
Mean number of features achieved by the different wrapper alternatives.

Method	German	Ionosphere	Vehicle	WBCD	Wine	Zoo
LFS	3.000	4.000	9.000	10.000	7.000	8.000
GSBS	18.000	30.000	16.000	25.000	8.000	7.000
ErFS	13.480	12.580	9.520	13.420	8.000	9.180
2SFS	11.920	12.050	8.650	5.000	8.000	9.180
TMGWO	14.000	4.000	9.000	4.000	6.000	8.000
FAM-BSO	12.030	17.050	9.030	14.840	6.410	8.420
BPSO	*n.a.*	*n.a.*	10.500	*n.a.*	8.900	11.050
BSEOA	*n.a.*	*n.a.*	10.350	*n.a.*	8.150	10.200
LeOCCEA	6.325 ± 1.072	8.175 ± 0.501	8.775 ± 0.733	5.600 ± 0.744	3.175 ± 0.385	4.425 ± 0.594

Table 7
Lexicographic comparison of the wrapper methods for the German dataset ($\alpha = 0.05$).

Method	FAM-BSO	LeOCCEA	TMGWO	ErFS	2SFS	LFS	GSBS
FAM-BSO	✓						
LeOCCEA	E	✓					
TMGWO	N	N	✓				
ErFS	E	N	E	✓	N		
2SFS	E	E	E		✓	N	
LFS	E	E	E	E		✓	
GSBS	E	E	E	E	E	N	✓
Ranking	0	1	2	4	4	4	6

Table 8
Lexicographic comparison of the wrapper methods for the Ionosphere dataset ($\alpha = 0.05$).

Method	TMGWO	LeOCCEA	FAM-BSO	2SFS	LFS	ErFS	GSBS
TMGWO	✓						
LeOCCEA	N	✓					
FAM-BSO	N	N	✓	N			
2SFS	E	E		✓		✓	
LFS	E	E	E	E	✓		
ErFS	E	E	E	✓	N	✓	
GSBS	E	E	E	E	N	E	✓
Ranking	0	1	3	3	4	5	6

Table 9
Lexicographic comparison of the wrapper methods for the WBCD dataset ($\alpha = 0.05$).

Method	FAM-BSO	LeOCCEA	TMGWO	2SFS	ErFS	LFS	GSBS
FAM-BSO	✓	N					
LeOCCEA		✓	N				
TMGWO	E		✓	✓			
2SFS	E	E	✓	✓			
ErFS	E	E	E	N	✓	✓	
LFS	E	E	E	E	✓	✓	
GSBS	E	E	E	E	E	N	✓
Ranking	1	1	2	3	5	5	6

again the two starting hypotheses of this work, now on a high-dimensional dataset.

7.2. Application to a real motor imagery classification problem

Now LeOCCEA is applied to three different datasets concerning a real BCI problem. These MI datasets were recorded in the BCI laboratory at the University of Essex, UK. One different dataset was obtained from each one of the several subjects imagining the movement of their right hand, left hand, and feet (three classes). These BCI data were acquired applying the 10–20 international placement system [87], a standard method to apply the scalp electrodes in the context of EEG tests.

Data were obtained with 15 electrodes and from 12 healthy subjects (58% female, 50% naive to BCI, with ages ranging from 24 to 50), sampled at 256 Hz during four different sessions of 30 trials per class, producing a total of 120 trials per class for each subject. The training dataset was formed by samples from the first two sessions, leaving the rest of the data for the test dataset, obtaining two datasets of 180 samples per subject. After pre-processing and feature extraction, each sample is formed by 3 600 input features, each one representing a set of coefficients obtained from the original signal by means of multiresolution analysis (MRA) [88]. These datasets were also used to test the NSGA II-based wrapper method proposed in [11], where four different classification schemes were compared, KNN, NBC, and the application of LDA to reduce the input dimensionality before using the two former classifiers (LDA + KNN and LDA + NBC respectively).

The LeOCCEA hyperparameters have been fixed the same as in Section 7.1 for the three different datasets: subjects 104, 107, and 110. Tables 18 and 19 show the results obtained, along with those achieved by the four different classifier schemes used in [11]. It can be appreciated that the number of features finally selected by LeOCCEA is quite different for subjects 104, 107 and 110, although an a posteriori adjustment in the similarity threshold t_l could guide the search of the wrapper method towards solutions with a similar number of features, as it has been analyzed in Section 5. However, the value of the test Kappa index achieved by LeOCCEA is lower than those reported by the other wrapper alter-

Table 10
Lexicographic comparison of the wrapper methods for the Vehicle dataset ($\alpha = 0.05$). Ranking* has been calculated discarding BPSO and BSEOA, to allow the comparison of results with those of Tables 7–9.

Method	LeOCCEA	ErFS	2SFS	LFS	BPSO	BSEOA	FAM-BSO	TMGWO	GSBS
LeOCCEA	✓								
ErFS	N	✓	N						
2SFS	E		✓	✓					
LFS	E	E	✓	✓					
BPSO	E	E	N	N	✓	✓	✓		
BSEOA	E	E	N	N	✓	✓	✓		
FAM-BSO	E	E	E	N	✓	✓	✓		
TMGWO	E	E	E	E	E	E	E	✓	
GSBS	E	E	E	E	E	E	N	N	✓
Ranking	0	2	2	3	6	6	6	7	8
Ranking*	0	2	2	3	*n.a.*	*n.a.*	4	5	6

Table 11
Lexicographic comparison of the wrapper methods for the Wine dataset ($\alpha = 0.05$). Ranking* has been calculated discarding BPSO and BSEOA, to allow the comparison of results with those of Tables 7–9.

Method	LeOCCEA	FAM-BSO	BPSO	BSEOA	ErFS	2SFS	TMGWO	LFS	GSBS
LeOCCEA	✓								
FAM-BSO	N	✓							
BPSO	N	N	✓	✓					
BSEOA	N	N	✓	✓					
ErFS	N	E	E	E	✓	✓			
2SFS	N	E	E	E	✓	✓			
TMGWO	E	E	E	E	E	E	✓		
LFS	E	E	E	E	E	E	E	✓	
GSBS	E	E	E	E	E	E	N	N	✓
Ranking	0	1	3	3	5	5	6	7	8
Ranking*	0	1	*n.a.*	*n.a.*	3	3	4	5	6

Table 12
Lexicographic comparison of the wrapper methods for the Zoo dataset ($\alpha = 0.05$). Ranking* has been calculated discarding BPSO and BSEOA, to allow the comparison of results with those of Tables 7–9.

Method	LeOCCEA	BSEOA	TMGWO	BPSO	FAM-BSO	GSBS	ErFS	2SFS	LFS
LeOCCEA	✓	E							
BSEOA		✓	N	✓					
TMGWO	N		✓		✓				
BPSO	N	✓	N	✓					
FAM-BSO	N	E	✓	E	✓		✓	✓	
GSBS	E	E	E	E	E	✓			✓
ErFS	E	E	E	E	✓	N	✓	✓	
2SFS	E	E	E	E	✓	N	✓	✓	
LFS	E	E	E	E	E	✓	E	E	✓
Ranking	1	2	2	3	6	6	7	7	8
Ranking*	1	*n.a.*	2	*n.a.*	4	4	5	5	6

Table 13
Average ranking achieved for all datasets. BPSO and BSEOA are not included, since they have not been applied to all datasets.

Dataset	LeOCCEA	FAM-BSO	TMGWO	2SFS	ErFS	LFS	GSBS
German	1	0	2	4	4	4	6
Ionosphere	1	3	0	3	5	4	6
Vehicle	0	4	5	2	2	3	6
WBCD	1	1	2	3	5	5	6
Wine	0	1	4	3	3	5	6
Zoo	0	4	2	5	5	6	4
Average	0.50	2.17	2.50	3.33	4.00	4.50	5.67

natives, although results obtained by LeOCCEA are much more stable, especially for subjects 107 and 110, as shown in Table 20. This fact could mean that SVM is not the best classifier for the University of Essex BCI data. Indeed, this issue is discussed in [89,90], where some guidelines are provided to choose an adequate classifier taking into account the characteristics of a concrete BCI application.

On the other side, since results obtained by LeOCCEA are surprisingly stable and KNN and NBC performed better for the same data in [11], these classifiers have been tried with the Essex test

Table 14
Average ranking achieved considering all the wrapper methods.

Dataset	LeOCCEA	BSEOA	BPSO	FAM-BSO	2SFS	ErFS	TMGWO	LFS	GSBS
Vehicle	0	6	6	6	2	2	7	3	8
Wine	0	3	3	1	5	5	6	7	8
Zoo	1	2	3	6	7	7	2	8	6
Average	0.33	3.67	4.00	4.33	4.67	4.67	5.00	6.00	7.33

Table 15
Stability score achieved by the LeOCCEA wrapper method for all the datasets listed in Table 1.

Dataset	Score
German	0.863
Ionosphere	0.878
Vehicle	0.858
WBCD	0.743
Wine	0.867
Zoo	0.756
Average	0.828

Table 16
Average accuracy and number of features selected for the test patterns of the lung cancer dataset.

Method	Accuracy	# Features
mRMR + SVM	0.904	3
	0.947	6
	0.951	9
mRMR + KNN	0.904	3
	0.951	6
	0.951	9
mRMR + RF	0.914	3
	0.954	6
	0.944	9
LeOCCEA	0.962 ± 0.005	5.750 ± 1.552

dataset using the sets of features provided by LeOCCEA. KNN was parameterized with k equal to the odd number closest to the squared root of the number of samples in each dataset, as in [11]. Results are shown in Table 21. KNN seems to be quite sensitive to this change because their results have worsen. This is in line with the stability tests performed in [11]. On the contrary, results obtained with NBC are even better than those achieved by SVM, which was the classifier used by LeOCCEA to select the features in the training process. Thus, these results support the hypothesis that SVM may not be the best classifier for this dataset. Even more, perhaps LeOCCEA trained with NBC could even improve the results, although this experiment has been left for future work.

Table 18
Kappa values (avg $\pm$ std) for the test patterns of subjects 104, 107 and 110 of the University of Essex BCI data files.

Wrapper	104	107	110
KNN	0.704 ± 0.031	0.550 ± 0.033	0.590 ± 0.031
NBC	0.642 ± 0.029	0.521 ± 0.030	0.515 ± 0.038
LDA + KNN	0.647 ± 0.053	0.584 ± 0.035	0.580 ± 0.039
LDA + NBC	0.677 ± 0.047	0.550 ± 0.028	0.574 ± 0.055
LeOCCEA	0.578 ± 0.046	0.479 ± 0.049	0.493 ± 0.039

Table 19
Number of features (avg $\pm$ std) selected for subjects 104, 107 and 110 of the University of Essex BCI data files.

Wrapper	104	107	110
KNN	28.260 ± 1.209	28.840 ± 0.866	29.080 ± 0.829
NBC	27.220 ± 1.112	29.360 ± 0.921	29.380 ± 0.725
LDA + KNN	28.760 ± 1.117	29.850 ± 0.670	29.680 ± 0.695
LDA + NBC	28.480 ± 1.249	29.860 ± 0.756	29.720 ± 0.757
LeOCCEA	17.700 ± 1.342	30.900 ± 2.426	26.350 ± 2.540

Table 20
Stability scores achieved by the different wrapper procedures for subjects 104, 107 and 110 of the University of Essex BCI data files.

Wrapper	104	107	110	Average
KNN	0.948	0.959	0.963	0.957
NBC	0.679	0.928	0.793	0.800
LDA + KNN	0.694	0.834	0.920	0.816
LDA + NBC	0.721	0.859	0.879	0.820
LeOCCEA	0.991	0.988	0.989	0.989

Tables 22–24 show the pairwise comparison of the wrapper methods for the three subjects, with a significance level $\alpha = 0.05$, as well as their ranking. It can be appreciated that the subsets of features found by LeOCCEA and classified with NBC are among

Table 17
Lexicographic comparison of the feature selection methods applied to the lung cancer dataset. For the results obtained in [23], the different classifiers are indicated with the number of features obtained by mRMR in brackets ($\alpha = 0.05$).

Method	LeOCCEA	RF(6)	KNN(6)	SVM(6)	KNN(9)	SVM(9)	RF(3)	KNN(3)	SVM(3)	RF(9)
LeOCCEA	✓	✓								
RF(6)	✓	✓	✓							
KNN(6)	E	✓	✓	✓						
SVM(6)	E	E	✓	✓			N			
KNN(9)	E	N	N	N	✓	✓				
SVM(9)	E	N	N	N	✓	✓				
RF(3)	E	E	E		E	E	✓	✓	✓	
KNN(3)	E	E	E	E	E	E	✓	✓	✓	
SVM(3)	E	E	E	E	E	E	✓	✓	✓	
RF(9)	E	E	E	N	E	E	N	N	N	✓
Ranking	1	2	3	4	5	5	7	8	8	9

Table 21
Kappa values (avg ± std) applying different classifiers obtained for the test patterns of subjects 104, 107 and 110 of the University of Essex BCI data files. Features were selected with LeOCCEA (using SVM while training).

Classifier	104	107	110
SVM	0.578 ± 0.046	0.479 ± 0.049	0.493 ± 0.039
KNN	0.543 ± 0.053	0.361 ± 0.043	0.444 ± 0.059
NBC	0.639 ± 0.061	0.534 ± 0.037	0.542 ± 0.037

the best alternatives for subjects 104 and 107. Average ranking values are listed in Table 25, where LeOCCEA + NBC is in the second position just after KNN. However, it has to be remarked that the application of KNN to test data used the subsets of features found by the wrapper method proposed in [11] also using KNN, while for the LeOCCEA + NBC alternative NBC was applied to test data using the features provided by LeOCCEA, which used SVM while training.

8. Conclusions

This paper has described the LeOCCEA wrapper method in-depth, a wrapper procedure that hybridizes concepts of CCEAs and lexicographic optimization to make possible the simultaneous optimization of two interdependent problems: finding the best hyperparameter values for the classifier applied within the wrapper method while minimizing the number of features that better describe a dataset. The lexicographic approach allows the optimization of multiple objectives easily, even with a simple EA scheme for each species. Another benefit of LeOCCEA is that it finds only one solution per execution, instead of a set of Pareto optimal solutions, which makes easier the work of the DM. see Table 23.

Since the results and execution time of LeOCCEA depend on its two main configuration hyperparameters, p_{val} and t_l, an experimental study has been carried out in order to determine how these hyperparameters influence the wrapper method results and which

Table 22
Lexicographic comparison of the wrapper methods for subject 104 of the University of Essex BCI data files ($\alpha = 0.05$)

Method	KNN	LeOCCEA (NBC)	NBC	LDA + NBC	LDA + KNN	LeOCCEA (SVM)
KNN	✓					
LeOCCEA (NBC)	E	✓				
NBC	E	N	✓	E		
LDA + NBC	E	N		✓	✓	
LDA + KNN	E	N	N	✓	✓	
LeOCCEA (SVM)	E	E	E	E	E	✓
Ranking	0	1	3	3	4	5

Table 23
Lexicographic comparison of the wrapper methods for subject 107 of the University of Essex BCI data files ($\alpha = 0.05$).

Method	LDA + KNN	KNN	LeOCCEA (NBC)	LDA + NBC	LeOCCEA (SVM)	NBC
LDA + KNN	✓	E		✓		
KNN		✓	N			
LeOCCEA (NBC)	E		✓			
LDA + NBC	E	✓	✓			
LeOCCEA (SVM)	E	E	E	E	✓	
NBC	E	E	N	E	N	✓
Ranking	1	2	2	3	4	5

Table 24
Lexicographic comparison of the wrapper methods for subject 110 of the University of Essex BCI data files ($\alpha = 0.05$).

Method	KNN	LDA + KNN	LDA + NBC	LeOCCEA (NBC)	LeOCCEA (SVM)	NBC
KNN	✓		✓			
LDA + KNN	N	✓	✓			
LDA + NBC	✓	✓	✓	N		
LeOCCEA (NBC)	E	E		✓	✓	
LeOCCEA (SVM)	E	E	E	✓	✓	
NBC	E	E	E	N	N	✓
Ranking	1	2	3	3	4	5

Table 25
Average ranking achieved by the wrapper methods for the test patterns of subjects 104, 107 and 110 of the University of Essex BCI data files.

Subject	KNN	LeOCCEA (NBC)	LDA + KNN	LDA + NBC	LeOCCEA (SVM)	NBC
104	0	1	4	3	5	3
107	2	2	1	3	4	5
110	1	3	2	3	4	5
Average	1.00	2.00	2.33	3.00	4.33	4.33

values are likely to make the algorithm converge to satisfactory solutions in a reasonable computation time. Once these values have been obtained, LeOCCEA has been applied to several well-known datasets. A new lexicographic ranking methodology has been proposed to allow the comparison of its results with those provided by other state-of-the-art wrapper methods. LeOCCEA has achieved the best average ranking, which confirms the two starting hypotheses of this work: the simultaneous optimization of the classifier hyperparameters, while the feature selection problem is being solved, improves the final results, and the co-evolution of these two interrelated problems can be formulated as a lexicographic problem.

LeOCCEA has also been applied to several real high-dimensional datasets. For the lung cancer diagnosis, LeOCCEA also performs quite well, reducing the dimensionality of the dataset from 410 features to an average of 4.75, and achieving the better accuracy of all compared methods. However, for the MI application, the classification accuracy obtained is not as good as expected, although the wrapper method has presented a surprisingly high stability, which led us to think that perhaps SVM is not the best classifier for this BCI application. Thus, the subsets of features provided by LeOCCEA (using SVM while training) were used to classify the test datasets with KNN and NBC, achieving noticeably better accuracies with NBC, comparable with those obtained in [11]. These results open up future research where LeOCCEA should also take into account other classifiers, such as NBC, to improve its test accuracy in this application.

Finally, the stability scores achieved by LeOCCEA for all the high-dimensional datasets are quite higher than those obtained for the UCI datasets, even with the former being much more difficult problems. This effect may be related to the number of generations run in each case. Perhaps the 300 generations run for the UCI datasets do not suffice to achieve the highly stable results produced for the motor imagery data, obtained after 1000 generations.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work was supported by project PGC2018-098813-B-C31 (Spanish "Ministerio de Ciencia, Innovación y Universidades"), and by European Regional Development Funds (ERDF). Funding for open access charge: Universidad de Granada / CBUA. We would like to thank Dr. Daniel Castillo-Secilla and Dr. John Q. Gan for making available to us, respectively, the lung cancer dataset and the University of Essex BCI data files.

Appendix A. Summary of notations

The notations used along this paper are described below:

n Number of features in the original dataset.
k Number of neighbors chosen for the KNN classifier.
C Regularization hyperparameter for the SVM classifier.
γ Width of the RBF kernels for the SVM classifier.
n_o Number of objectives to be optimized.
o^i The i-th objective.
f^i Fitness value for the i-th objective.
$\boldsymbol{f}$ Vector of n_o components storing the fitness for all the objectives defined in the problem.
t_l^i Similarity threshold applied for the lexicographic comparison of values for i-th objective.
$\boldsymbol{t}_l$ Vector of n_o components storing the different similarity thresholds for all the objectives.
t_l Unique similarity threshold value. Used when the same similarity threshold value is applied for all the objectives ($t_l^i = t_l, \forall i \in [0, n_o) \cap \mathbb{N}$).
$\prec_l$ Better-than lexicographic relation (subindex l comes from lexicographic).
$=_l$ Equal-to lexicographic relation.
$\preceq_l$ Better-than or equal-to lexicographic relation.
n_p Number of subpopulations defined in the LeOCCEA wrapper method.
P_i i-th subpopulation defined in the LeOCCEA wrapper method.
S_i Species being evolved in subpopulation P_i. S_0 is used to represent the hyperparameters of the classifier whereas the rest of species evolve subsets of input features.
m_i Size of subpopulation P_i.
m Size of all the subpopulations being evolved (in case all the subpopulations have the same size).
I_j Individual belonging to species S_j. If $j = 0$ it encodes possible values for the hyperparameters of the classifier. For higher values of j it contains a subset of input features belonging to S_j.
n_r Number of representatives chosen from each subpopulation in order to assign a fitness value to each individual in the LeOCCEA wrapper method.
n_g Number of generations performed by the LeOCCEA wrapper method.
$p_{m_{fs}}$ Mutation probability for the feature selection species.
$p_{m_{svm}}$ Mutation probability for the classifier hyperparameter species.
n_e Number of executions of the wrapper algorithm.
D The whole training dataset.
p_{val} Percentage of data in D used for validation for the VO and VT lexicographic evaluation alternatives.
D_I Reduced dataset obtained keeping only the selected features coded in I from the original training dataset D.
$D_{I_{tr}}$ Subset of D_I used to train the classifiers for the VO and VT lexicographic evaluation alternatives.
$D_{I_{val}}$ Subset of D_I used for validation for the VO and VT lexicographic evaluation alternatives.
W Wrapper method.
$B_D(W)$ Set of wrapper methods lexicographically better than W for dataset D.
$r_D(W)$ Rank of wrapper method W for a dataset D.
$\bar{r}(W)$ Average rank of wrapper method W for several datasets.

References

[1] R.E. Bellman, Adaptive Control Processes. A Guided Tour, Princeton Legacy Library, Princeton University Press, Princeton, NJ, USA, 1961.
[2] I. Guyon, A. Elisseeff, An introduction to variable and feature selection, J. Mach. Learn. Res. 3 (2003) 1157–1182, URL:http://www.jmlr.org/papers/v3/guyon03a.html.
[3] P.M. Lewis, The characteristic selection problem in recognition systems, IRE Trans. Inform. Theory 8 (2) (1962) 171–178, https://doi.org/10.1109/TIT.1962.1057691.
[4] P. Min, On feature selection in multiclass pattern recognition, Ph.D. thesis, School of Electrical Engineering, Purdue University, Lafayette, IN, USA (June 1968).
[5] A.L. Blum, P. Langley, Selection of relevant features and examples in machine learning, Artif. Intell. 97 (1–2) (1997) 245–271, https://doi.org/10.1016/S0004-3702(97)00063-5.
[6] R. Kohavi, G.H. John, Wrappers for feature subset selection, Artif. Intell. 97 (1–2) (1997) 273–324, https://doi.org/10.1016/S0004-3702(97)00043-X.
[7] P. Pudil, P. Somol, Identifying the most informative variables for decision-making problems – a survey of recent approaches and accompanying

problems, Acta Oeconomica Pragensia 2008 (4) (2008) 37–55, https://doi.org/10.18267/j.aop.131.

[8] S. Khalid, T. Khalil, S. Nasreen, A survey of feature selection and feature extraction techniques in machine learning, in: K. Arai, A. Mellouk (Eds.), Proceedings of the 2014 Science and Information Conference, The Science and Information (SAI) Organization, London, UK, 2014, pp. 372–378, doi: 10.1109/SAI.2014.6918213.

[9] S. Basterrech, P. Bobrov, A. Frolov, D. Húsek, Nature-inspired algorithms for selecting eeg sources for motor imagery based bci, in, in: Proceedings of the 14th International Conference on Artificial Intelligence and Soft Computing (ICAISC 2015), Part II, Lecture Notes in Computer Science Springer, Zakopane, Poland, 2015, pp. 79–90, https://doi.org/10.1007/978-3-319-19369-4_8.

[10] R. Corralejo, R. Hornero, D. Álvarez, Feature selection using a genetic algorithm in a motor imagery-based brain computer interface, in: Proceedings of the 33rd Annual International Conference of the IEEE Engineering-in-Medicine-and-Biology-Society (EMBS), IEEE, Boston, MA, USA, 2011, pp. 7703–7706, https://doi.org/10.1109/IEMBS.2011.6091898.

[11] J. González, J. Ortega, M. Damas, P. Martín-Smith, J.Q. Gan, A new multi-objective wrapper method for feature selection – accuracy and stability analysis for bci, Neurocomputing 333 (14) (2019) 407–418, https://doi.org/10.1016/j.neucom.2019.01.017.

[12] B. Xue, M. Zhang, W.N. Browne, Particle swarm optimization for feature selection in classification: A multi-objective approach, IEEE Trans. Cybern. 43 (6) (2013) 1656–1671, https://doi.org/10.1109/TSMCB.2012.2227469.

[13] A. Wang, N. An, G. Chen, L. Li, G. Alterovitz, Accelerating wrapper-based feature selection with k-nearest-neighbor, Knowl.-Based Syst. 83 (2015) 81–91, https://doi.org/10.1016/j.knosys.2015.03.009.

[14] C.H. Park, S.B. Kim, Sequential random k-nearest neighbor feature selection for high-dimensional data, Expert Syst. Appl. 42 (5) (2015) 2336–2342, https://doi.org/10.1016/j.eswa.2014.10.044.

[15] M. Abdel-Basset, D. El-Shahat, I. El-henawy, V.H.C. de Albuquerque, S. Mirjalili, A new fusion of grey wolf optimizer algorithm with a two-phase mutation for feature selection, Expert Syst. Appl. 139 (2020), https://doi.org/10.1016/j.eswa.2019.112824 112824.

[16] P. Tan, X. Wang, Y. Wang, Dimensionality reduction in evolutionary algorithms-based feature selection for motor imagery brain-computer interface, Swarm Evol. Comput. 52 (2020), https://doi.org/10.1016/j.swevo.2019.100597 100597.

[17] X.F. Song, Y. Zhang, Y.N. Guo, X.Y. Sun, Y.L. Wang, Variable-size cooperative coevolutionary particle swarm optimization for feature selection on high-dimensional data, IEEE Trans. Evol. Comput. 24 (5) (2020) 882–895, https://doi.org/10.1109/TEVC.2020.2968743.

[18] R.N. Khushaba, A. AlSukker, A. Al-Ani, A. Al-Jumaily, Intelligent artificial ants based feature extraction from wavelet packet coefficients for biomedical signal classification, in: Proceedings of the 3rd IEEE International Symposium on Control, Communications and Signal Processing (ISCCSP 2008), IEEE, St. Julians, Malta, 2008, pp. 1366–1371, doi: 10.1109/ISCCSP.2008.4537439.

[19] J. Ortega, J. Asensio-Cubero, J.Q. Gan, A. Ortiz, Classification of motor imagery tasks for bci with multiresolution analysis and multiobjective feature selection, BioMedical Eng. OnLine 15 (Suppl 1) (2016) 73, https://doi.org/10.1186/s12938-016-0178-x.

[20] M. Schroder, M. Bogdan, W. Rosenstiel, T. Hinterberger, N. Birbaumer, Automated eeg feature selection for brain computer interfaces, in: Proceedings of the 1st International IEEE/EMBS Conference on Neural Engineering, IEEE, Capri, Italy, 2003, pp. 626–629, https://doi.org/10.1109/CNE.2003.1196906.

[21] S. Madonado, R. Weber, A wrapper method for feature selection using support vector machines, Inf. Sci. 179 (13) (2009) 2208–2217, https://doi.org/10.1016/j.ins.2009.02.014.

[22] P. Rajasekharreddy, E.S. Gopi, Feature selection for vocal segmentation using social emotional optimization algorithm, in: A.J. Kulkarni, P.K. Singh, S.C. Satapathy, A.H. Kashan, K. Tai (Eds.), Socio-cultural Inspired Metaheuristics, Vol. 828 of Studies in Computational Intelligence, Springer Verlag, Singapore, 2019, pp. 69–91, doi: 10.1007/978-981-13-6569-0_4.

[23] D. Castillo-Secilla, J.M. Gálvez, F. Carrillo-Perez, M. Verona-Almeida, D. Redondo-Sánchez, F.M. Ortuno, L.J. Herrera, I. Rojas, Knowseq r-bioc package: The automatic smart gene expression tool for retrieving relevant biological knowledge, Comput. Biol. Med. 133 (2021), https://doi.org/10.1016/j.compbiomed.2021.104387 104387.

[24] C. Cortes, V. Vapnik, Support-vector networks, Machine Learning 20 (3) (1995) 273–297, https://doi.org/10.1007/BF00994018.

[25] J. González, J. Ortega, M. Damas, P. Martín-Smith, Many-objective cooperative co-evolutionary feature selection: A lexicographic approach, in: I. Rojas, G. Joya, A. Catalá (Eds.), Advances in Computational Intelligence, IWANN 2019, Vol. 11507 of Lecture Notes in Computer Science, Springer, Gran Canaria, Spain, 2019, pp. 463–474, doi: 10.1007/978-3-030-20518-8_39.

[26] M.A. Potter, K.A. De Jong, A cooperative coevolutionary approach to function optimization, in: Y. Davidor, H.-P. Schwefel, R. Männer (Eds.), Proceedings of the 3rd International Conference on Parallel Problem Solving from Nature, PPSN III, Vol. 866 of Lecture Notes in Computer Science, Springer, Jerusalem, Israel, 1994, pp. 249–257, doi: 10.1007/3-540-58484-6_269.

[27] M.A. Potter, K.A. De Jong, Cooperative coevolution: An architecture for evolving coadapted subcomponents, Evolutionary Computation 8 (1) (2000) 1–29, https://doi.org/10.1162/106365600568086.

[28] V.R. Khare, X. Yao, B. Sendhoff, Credit assignment among neurons in co-evolving populations, in: X. Yao, E.K. Burke, J.A. Lozano, J. Smith, J.J. Merelo-Guervós, J.A. Bullinaria, J.E. Rowe, P. Tiňo, A. Kabán, H.-P. Schwefel (Eds.), Proceedings of the 8th International Conference on Parallel Problem Solving from Nature, PPSN VIII, Vol. 3242 of Lecture Notes in Computer Science, Springer, Berlin, Germany, 2004, pp. 882–891, doi: 10.1007/978-3-540-30217-9_89.

[29] J. Tian, M. Li, F. Chen, Coevolutionary feature selection strategy for rbfnn classifier, in: M. Guo, L. Zhao, L. Wang (Eds.), Proceedings of the Fourth International Conference on Natural Computation, ICNC'2008, Vol. 7, IEEE, Jinan, China, 2008, pp. 131–135, doi: 10.1109/ICNC.2008.436.

[30] Y. Wen, H. Xu, A cooperative coevolution-based pittsburgh learning classifier system embedded with memetic feature selection, in: A.E. Smith, I. Parmee (Eds.), Proceedings of the 2011 IEEE Congress of Evolutionary Computation, CEC'2011, IEEE, New Orleans, LA, USA, 2011, pp. 2415–2422, doi: 10.1109/CEC.2011.5949916.

[31] B. Cao, J. Zhao, Z. Lv, X. Liu, A distributed parallel cooperative coevolutionary multiobjective evolutionary algorithm for large-scale optimization, IEEE Trans. Industr. Inf. 13 (4) (2017) 2030–2038, https://doi.org/10.1109/TII.2017.2676000.

[32] X. Ma, X. Li, Q. Zhang, K. Tang, Z. Liang, W. Xie, Z. Zhu, A survey on cooperative co-evolutionary algorithms, IEEE Transactions on Evolutionary Computation (early access), doi: 10.1109/TEVC.2018.2868770.

[33] A.E. Aguilar-Justo, E. Mezura-Montes, A local cooperative approach to solve large-scale constrained optimization problems, Swarm Evol. Comput. 51 (2019), https://doi.org/10.1016/j.swevo.2019.100577 100577.

[34] Z.H. Zhan, J. Li, J. Cao, J. Zhang, H.S.H. Chung, Y.H. Shi, Multiple populations for multiple objectives: A coevolutionary technique for solving multiobjective optimization problems, IEEE Trans. Cybern. 43 (2) (2013) 445–463, https://doi.org/10.1109/TSMCB.2012.2209115.

[35] X.F. Liu, Z.H. Zhan, Y. Gao, J. Zhang, S. Kwong, J. Zhang, Coevolutionary particle swarm optimization with bottleneck objective learning strategy for many-objective optimization, IEEE Trans. Evol. Comput. 23 (4) (2019) 587–602, https://doi.org/10.1109/TEVC.2018.2875430.

[36] Z.G. Chen, Z.H. Zhan, Y. Lin, Y.J. Gong, T.L. Gu, F. Zhao, H.Q. Yuan, X. Chen, Q. Li, J. Zhang, Multiobjective cloud workflow scheduling: A multiple populations ant colony system approach, IEEE Trans. Cybern. 49 (8) (2019) 2912–2926, https://doi.org/10.1109/TCYB.2018.2832640.

[37] S.-Z. Zhou, Z.-H. Zhan, Z.-G. Chen, S. Kwong, J. Zhang, A multi-objective ant colony system algorithm for airline crew rostering problem with fairness and satisfaction, IEEE Trans. Intell. Transp. Syst. (2020), https://doi.org/10.1109/tits.2020.2994779.

[38] C. Emmanouilidis, A. Hunter, J. MacIntyre, A multiobjective evolutionary setting for feature selection and a commonality-based crossover operator, in: A. Zalzala (Ed.), Proceedings of the 2000 Congress on Evolutionary Computation, CEC'2000, IEEE, La Jolla, CA, USA, 2000, pp. 309–316, doi: 10.1109/CEC.2000.870311.

[39] C. Emmanouilidis, A. Hunter, J. MacIntyre, C. Cox, A multi-objective genetic algorithm approach to feature selection in neural and fuzzy modeling, Evol. Optim. 3 (1) (2001) 1–26.

[40] J. Liu, H. Iba, Selecting informative genes using a multiobjective evolutionary algorithm, in: D.B. Fogel (Ed.), Proceedings of the 2002 Congress on Evolutionary Computation, IEEE, Honolulu, HI, USA, 2002, pp. 297–302, https://doi.org/10.1109/CEC.2002.1006250.

[41] L. Oliveira, R. Sabourin, F. Bortolozzi, C.Y. Suen, A methodology for feature selection using multiobjective genetic algorithms for handwritten digit string recognition, Int. J. Pattern Recognit Artif Intell. 17 (6) (2003) 903–929, https://doi.org/10.1142/S021800140300271X.

[42] F. Mendes, J. Duarte, A. Vieira, A. Gaspar-Cunha, Feature selection for bankruptcy prediction: A multi-objective optimization approach, in: X.-Z. Gao, A. Gaspar-Cunha, M. Köppen, G. Schaefer, J. Wang (Eds.), Soft Computing in Industrial Applications, Vol. 75 of Advances in Intelligent and Soft Computing, Springer, Berlin, Germany, 2010, pp. 109–115, doi: 10.1007/978-3-642-11282-9_12.

[43] V.V. Podinovskii, V.M. Gavrilov, Optimization with respect to successive criteria (Optimizatsiya po posledovatel'no primenyaemym kriteriyam), Sovetskoe Radio, Moscow, Russia, 1975.

[44] A. Ben-Tal, Characterization of pareto and lexicographic optimal solutions, in: G. Fandel, T. Gal (Eds.), Proceedings of the Third Conference on Multiple Criteria Decision Making Theory and Application, Vol. 177 of Lecture Notes in Economics and Mathematical System, Springer, Berlin, Germany, 1979, pp. 1–11, doi: 10.1007/978-3-642-48782-8_1.

[45] M.G. Klepikova, The stability of lexicographic optimization problem, USSR Comput. Math. Math. Phys. 25 (1) (1985) 21–29, https://doi.org/10.1016/0041-5553(85)90037-0.

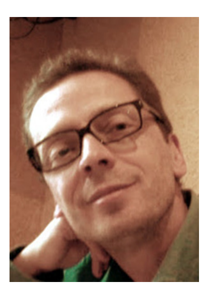
[46] S. Khosravani, M. Jalali, A. Khajepour, A. Kasaiezadeh, S.-K. Chen, B. Litkouhi, Application of lexicographic optimization method to integrated vehicle control systems, IEEE Trans. Industr. Electron. 65 (12) (2018) 9677–9686, https://doi.org/10.1109/TIE.2018.2821625.

[47] Y. Rasekhipour, I. Fadakar, A. Khajepour, Autonomous driving motion planning with obstacles prioritization using lexicographic optimization, Control Eng. Practice 77 (2018) 235–246, https://doi.org/10.1016/j.conengprac.2018.04.014.

[48] C.M. Fonseca, P.J. Fleming, Multiobjective optimization and multiple constraint handling with evolutionary algorithms – part i: A unified formulation, EEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans 28 (1) (1998) 26–37, https://doi.org/10.1109/3468.650319.

[49] N. Drechsler, R. Drechsler, B. Becker, Multi-objective optimisation based on relation favour, in: E. Zitzler, L. Thiele, K. Deb, C.A. Coello Coello, D. Corne

(Eds.), Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization, EMO'2001, Vol. 1993 of Lecture Notes in Computer Science, Springer, Berlin, Germany, 2001, pp. 154–166, doi: 10.1007/s11047-014-9422-0.
[50] F. Schmiedle, N. Drechsler, D. Große, R. Drechsler, Priorities in multi-objective optimization for genetic programming, in: L. Spector, E.D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt (Eds.), Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation, GECCO'2001, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001, pp. 129–136, URL:https://dl.acm.org/citation.cfm?id=2955256.
[51] N. Drechsler, A. Súlflow, R. Drechsler, Incorporating user preferences in many-objective optimization using relation ϵ)preferred, Nat. Comput. 14 (3) (2015) 469–483, https://doi.org/10.1007/s11047-014-9422-0.
[52] D. Dheeru, E. Karra Taniskidou, UCI Machine Learning Repository, School of Information and Computer Sciences, University of California, Irvine, CA, USA, 2017, URL:http://archive.ics.uci.edu/ml.
[53] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, 3rd Edition., Springer, Berlin, Germany, 1998.
[54] K.C. Tan, Y.J. Yang, C.K. Goh, A distributed cooperative coevolutionary algorithm for multiobjective optimization, IEEE Trans. Evol. Comput. 10 (5) (2006) 527–549, https://doi.org/10.1109/TEVC.2005.860762.
[55] E. Popovici, A. Bucci, R.P. Wiegand, E.D. De Jong, Coevolutionary principles, in: G. Rozenberg, T. Bäck, J.N. Kok (Eds.), Handbook of Natural Computing, Springer, Berlin, Germany, 2012, pp. 987–1033, doi: 10.1007/978-3-540-92910-9_31.
[56] K. Deb, R.B. Agrawal, Simulated binary crossover for continuous search space, Complex Systems 9(2) (1995) 115–148, URL:https://www.complex-systems.com/abstracts/v09_i02_a02/.
[57] K. Deb, S. Agrawal, A niched-penalty approach for constraint handling in genetic algorithms, in: A. Dobnikar, N.C. Steele, D.W. Pearson, R.F. Albrecht (Eds.), Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms, Springer, Portorož, Slovenia, 1999, pp. 235–243, doi: 10.1007/978-3-7091-6384-9_40.
[58] J.W. Cohen, A coefficient of agreement for nominal scales, Educ. Psychol. Measur. 20 (1) (1960) 37–46, https://doi.org/10.1037/h0026256.
[59] T. Eitrich, B. Lang, Efficient optimization of support vector machine learning parameters for unbalanced datasets, J. Comput. Appl. Math. 196 (2) (2006) 425–436, https://doi.org/10.1016/j.cam.2005.09.009.
[60] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, 2nd Edition, Springer series in statistics, Springer, New York, USA, 2009, https://doi.org/10.1007/978-0-387-84858-7.
[61] S.S. Keerthi, S.K. Shevade, Smo algorithm for least-squares svm formulations, Neural Comput. 15 (2) (2003) 487–507, https://doi.org/10.1162/089976603762553013.
[62] J. López, J.A.K. Suykens, First and second order smo algorithms for ls-svm classifiers, Neural Process. Lett. 33 (1) (2011) 31–44, https://doi.org/10.1007/s11063-010-9162-9.
[63] C. Sentelle, G.C. Anagnostopoulos, M. Georgiopoulos, A fast revised simplex method for svm training, in: Proceedings of the 19th International Conference on Pattern Recognition, IEEE, Tampa, FL, USA, 2008, https://doi.org/10.1109/ICPR.2008.4761810.
[64] H. Hofmann, German Credit Data, Universitat Hamburg, Hamburg, Germany, URL:https://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29.
[65] V. Sigillito, Johns Hopkins University Ionosphere Database, Johns Hopkins University, Baltimore, MD, USA, 1989, URL:https://archive.ics.uci.edu/ml/datasets/Ionosphere.
[66] P. Mowforth, B. Shepherd, Vehicle Silhouettes Dataset, Turing Institute, Glasgow, UK, URL:https://archive.ics.uci.edu/ml/datasets/Statlog+%28Vehicle+Silhouettes%29.
[67] W.H. Wolberg, W.N. Street, O.L. Mangasarian, Wisconsin Diagnostic Breast Cancer (WDBC), Clinical Sciences Center, MAdison, WI, USA, URL:https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29 (1995).
[68] M. Forina, R. Leardi, C. Armanino, S. Lanteri, PARVUS - An Extendible Package for Data Exploration, Classification and Correlation, Institute of Pharmaceutical and Food Analysis and Technologies, Genoa, Italy, URL:https://archive.ics.uci.edu/ml/datasets/Wine (1991).
[69] R.S. Forsyth, Zoo Dataset, Mapperley Park, Nottingham, UK, 1990, URL:https://archive.ics.uci.edu/ml/datasets/Zoo.
[70] F. Pourpanah, Y. Shi, C.P. Lim, Q. Hao, C.J. Tan, Feature selection based on brain storm optimization for data classification, Appl. Soft Computing J. 80 (2019) 761–775, https://doi.org/10.1016/j.asoc.2019.04.037.
[71] B. Xue, M. Zhang, W.N. Browne, New fitness functions in binary particle swarm optimisation for feature selection, in: H. Abbass, D. Essam, R. Sarker (Eds.), Proceedings of the 2012 IEEE Congress on Evolutionary Computation, CEC'2012, IEEE, Brisbane, QLD, Australia, 2012, doi: 10.1109/CEC.2012.6256617.
[72] S. Luke, et al., Ecj 26. a java-based evolutionary computation research system, URL:https://cs.gmu.edu/ eclab/projects/ecj/.
[73] C.-C. Chang, C.-J. Lin, LIBSVM: A library for support vector machines, ACM Transactions on Intelligent Systems and Technology 2 (3) (2011) 27, software available at URL:http://www.csie.ntu.edu.tw/ cjlin/libsvm.
[74] J. González, Ristretto, EFFICOMP team, University of Granada, URL:https://github.com/efficomp/ristretto.
[75] M. Gutlein, E. Frank, M. Hall, A. Karwath, Large-scale attribute selection using wrappers, in: K. Smith-Miles, E. Keogh, V.C. Lee (Eds.), Proceedings of the 2009 IEEE Symposium on Computational Intelligence and Data Mining, CIDM'2009, IEEE, Nashville, TN, USA, 2009, doi: 10.1109/CIDM.2009.4938668.
[76] A.W. Whitney, A direct method of nonparametric measurement selection, EEE Trans. Computers C-20 9 (1971) 1100–1103, https://doi.org/10.1109/T-C.1971.223410.
[77] T. Marill, D.M. Green, On the effectiveness of receptors in recognition systems, IEEE Trans. Inf. Theory 9 (1) (1963) 11–17, https://doi.org/10.1109/TIT.1963.1057810.
[78] R. Caruana, D. Freitag, Greedy attribute selection, in: W.W. Cohen, H. Hirsh (Eds.), Proceedings of the Eleventh International Conference on International Conference on Machine Learning, ICML'94, Morgan Kaufmann, New Brunswick, NJ, USA, 1994, pp. 28–36.
[79] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of the IEEE International Conference on Neural Networks, ICNN'95, Vol. 6, IEEE, Perth, WA, Australia, 1995, pp. 1942–1948, doi: 10.1109/ICNN.1995.488968.
[80] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, Adv. Eng. Softw. 69 (2014) 46–61, https://doi.org/10.1016/j.advengsoft.2013.12.007.
[81] Y. Shi, Brain storm optimization algorithm, in: Y. Tan, Y. Shi, Y. Chai, G. Wang (Eds.), Advances in Swarm Intelligence, ICSI 2011, Vol. 6728 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2011, pp. 303–309, doi: 10.1007/978-3-642-21515-5_36.
[82] G.A. Carpenter, S. Grossberg, N. Markuzon, J.H. Reynolds, D.B. Rosen, Fuzzy artmap: A neural network architecture for incremental supervised learning of analog multidimensional map, IEEE Trans. Neural Networks 3 (5) (1992) 698–713, https://doi.org/10.1109/72.159059.
[83] S. Grossberg, Adaptive pattern classification and universal recoding: Ii. feedback, expectation, olfaction, illusions, Biological Cybernetics 23(4) (1976) 187–202, https://doi.org/10.1007/bf00340335.
[84] J. Kennedy, R.C. Eberhart, A discrete binary version of the particle swarm algorithm, in: 1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation, Vol. 5, IEEE, Orlando, FL, USA, 1997, pp. 4104–4108, doi: 10.1109/ICSMC.1997.637339.
[85] Y. Xu, Z. Cui, J. Zeng, Social emotional optimization algorithm for nonlinear constrained optimization problems, in: B.K. Panigrahi, S. Das, P.N. Suganthan, S.S. Dash (Eds.), First International Conference on Swarm, Evolutionary, and Memetic Computing, SEMCCO 2010, Vol. 6466 of Lecture Notes in Computer Science, Springer, Chennai, India, 2010, pp. 583–590, doi: 10.1007/978-3-642-17563-3_68.
[86] C. Ding, H. Peng, Minimum redundancy feature selection from microarray gene expression data, J. Bioinformatics Comput. Biol. 3 (2) (2005) 185–205, https://doi.org/10.1142/S0219720005001004.
[87] J.S. Ebersole, A.M. Husain, D.R. Nordli, Current Practice of Clinical Electroencephalography, 4th Edition., Wolters Kluwer, Philadelphia, PA, USA, 2014.
[88] J. Asensio-Cubero, J.Q. Gan, R. Palaniappan, Multiresolution analysis over simple graphs for brain computer interfaces, J. Neural Eng. 10 (4) (2013), https://doi.org/10.1088/1741-2560/10/4/046014 046014.
[89] F. Lotte, M. Congedo, A. Lécuyer, F. Lamarche, B. Arnaldi, A review of classification algorithms for eeg-based brain-computer interfaces, J. Neural Eng. 4 (2) (2007) R1–R13, https://doi.org/10.1088/1741-2560/4/2/R01.
[90] F. Lotte, L. Bougrain, A. Cichocki, M. Clerc, M. Congedo, A. Rakotomamonjy, F. Yger, A review of classification algorithms for eeg-based brain-computer interfaces: a 10-year update, J. Neural Eng. 15 (3) (2018), https://doi.org/10.1088/1741-2552/aab2f2 031005.



Jesús González received the M.Sc. and Ph.D. degrees in Computer Engineering from the University of Granada, Spain, both with honors, in 1998 and 2001 respectively. He is a Full Professor in the Department of Computer Architecture and Technology at the University of Granada. His research interests include evolutionary computation, machine learning, and embedded systems.

Julio Ortega received the M.Sc. and Ph.D. degrees in Electronic Physics from the University of Granada, Spain, in 1986 and 1990 respectively. He is a Full Professor in the Department of Computer Architecture and Technology at the University of Granada and a senior member of the IEEE Computer Society. His research interests include the processing of parallel architectures, multi-objective optimization, neural networks, and evolutionary computation.

Juan José Escobar received the M.Sc. and Ph.D. degrees in Computer Engineering from the University of Granada, Spain, in 2014 and 2020 respectively. He is a Postdoctoral Researcher in the Department of Computer Architecture and Technology at the University of Granada. His research interests include code optimization, energy-efficient parallel computing, and workload balancing strategies on heterogeneous and distributed systems, especially in issues related to evolutionary algorithms and multi-objective feature selection problems.

Miguel Damas received the M.Sc. and Ph.D. degrees in Computer Engineering from the University of Granada, Spain, both with honors, in 1991 2000 respectively. Currently, he is a Full Professor in the Department of Computer Architecture and Computer Technology at the University of Granada. His research interests are related to the Industrial Internet of Things, human activity recognition systems, machine learning, and parallel programming for optimization problems.