IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 2, 2019, accepted September 10, 2019, date of publication September 30, 2019,
date of current version October 23, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2944641

Wrapper for Building Classification Models Using Covering Arrays

HUGO DORADO[1,2], CARLOS COBOS[1], (Senior Member, IEEE), JOSE TORRES-JIMENEZ[3], (Senior Member, IEEE), DHARANI DHAR BURRA[4], MARTHA MENDOZA[1], AND DANIEL JIMÉNEZ[2]

[1]Information Technology Research Group (GTI), Universidad del Cauca, Popayán 190003, Colombia
[2]International Center for Tropical Agriculture (CIAT), Apartado Aéreo 6713, Cali 763537, Colombia
[3]Cinvestav Tamaulipas, Ciudad Victoria 87130, México
[4]International Center for Tropical Agriculture (CIAT), Asia Regional Office, Hanoi, Vietnam

Corresponding author: Hugo Dorado (h.a.dorado@cgiar.org)

This work was supported in part by the Research Program on Climate Change, Agriculture and Food Security (CCAFS) through the Project Towards a Digital Climate Smart Agriculture transformation in Latin America, in part by the Project CONACyT Métodos Exactos para Construir Covering Arrays Óptimos under Project 238469, in part by the CGIAR Platform for Big Data in Agriculture under the community of practice Data-Driven Agronomy, and in part by the ABACUS-CINVESTAV CONACYT under Grant EDOMEX-2011-COI-165873.

ABSTRACT Wrapper methods are a type of feature selection method that finds a subset of variables to improve the performance of a classifier by removing redundant and irrelevant variables. The use of a wrapper implies that each time a candidate solution is explored, the classifier is evaluated on the quality measures selected (e.g. accuracy or precision). Though robust, this iteration across several candidate solutions can become computationally intensive and time-consuming. In this paper we propose a wrapper, that is based on binary Covering Arrays (CAs), and binary Incremental Covering Arrays (ICAs), that have been widely used for experimental design and fault detection in software and hardware testing. The new wrapper was evaluated with six classifiers on seven data sets. The results show that the CAs and ICAs with strength 6 significantly improve the performance and reduces the number of variables required by the classifier. A comparative analysis of the proposed method against wrappers based on other search approaches such as genetic algorithms (GA) and particle swarm optimization (PSO), shows that the proposed method yields results similar to GA, but not to PSO, with differences to PSO, in accuracy, which in the majority of cases is below 0.04. This lack of accuracy, by which the new wrapper fails to match PSO, is offset by the fact that the user does not need to fine tune algorithm parameters, such as velocity ranges, timing, cognitive coefficient, and social coefficient, while it is also much easier to program in parallel.

INDEX TERMS Classification algorithms, covering arrays, random forest, support vector machines, genetic algorithms, particle swarm optimization.

I. INTRODUCTION

At a time when information technology revolution is making it possible to collect vast amounts of data through devices, sensors, images or sound at high spatial and temporal frequencies, there is a resulting increased focus on data-driven decision-making and innovation, specifically in the fields of science, business, and marketing [1], [2]. Machine learning models (algorithms) for classification (classifiers) that enable identification of complex relationships in this high dimensional data have therefore become popular, mainly due to the flexibility they offer in terms of less prior knowledge required on the variables and their interactions, non-linearity, and response times [3], [4]. A peculiar feature of high dimensional datasets, in comparison to traditional datasets (i.e. collected using manual data collection processes), is that they also capture several redundant and even irrelevant variables, thereby affecting the performance of the classifier in terms of prediction accuracy, and increased computational time and cost. As a consequence, a prior process of feature selection is always necessary in order to remove these redundant and irrelevant variables, which contribute little in terms of insight and information.

The associate editor coordinating the review of this manuscript and approving it for publication was Jafar A. Alzubi.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see http://creativecommons.org/licenses/by/4.0/

Feature selection enables determination of an ''optimal'' subset of variables, such that the variables responsible for noise, which reduce downstream classifier performance and render the training process computationally cumbersome, are discarded. This, therefore, becomes an indispensable step in the building of classifiers [5], [6]. Feature selection methods can be classified mainly into three groups:

- **Filter** methods carry out feature selection as a phase prior to training and classification. For this, a criterion is used to establish a ranking of features. The variables are then sorted starting with the most relevant and a subset below a certain threshold is discarded [5], [7], [8].
- **Wrappers** are methods that use the classifier as part of the feature selection process. In this case, the aim is to find the best subset of variables according to a search strategy. Each candidate solution is evaluated by running the classifier and obtaining a performance measure [8]–[10].
- **Embedded** methods carry out feature selection within the same learning process, i.e. as the classifier is being trained. This involves a function to evaluate the influence of the variables (features) as part of the algorithm. Unlike the wrapper, variables are selected with a single run or weighted using regularization methods [11], [12].

With respect to the advantages of each method, filters and embedded methods are faster at finding a solution without having to run the classification algorithm more than once. There is a risk, however, that the solution found is not always the most suitable in terms of classifier performance. Wrappers, on the other hand, are generally able to find better solutions than filters, but are disadvantaged as they are much more computationally expensive (computational cost grows exponentially with the number of input variables), since the classifier needs to iterate several times. This problem persists, despite the implementation of search strategies such as meta-heuristics. Such strategies in most cases require the optimization of several parameters and require a large number of experiments with some limitation of being parallelized. Covering arrays, given their characteristics, offer the potential for solving some of these problems, but unfortunately have not been unexplored for use in wrapper design.

Covering Arrays (CA) are expressed as integer matrices with N rows and k columns, and a parameter t that denotes the strength or degree of interaction.

Each row is a sample or test, each column represents a variable or parameter, and the parameter t indicates that, all the interactions between each t columns are covered at least once.

This way, CA does a sampling of all possible values of the k columns, with two important features: minimal cardinality (i.e. the number of rows is minimum), and maximal coverage (i.e. all the possible values for each t parameter is sampled in at least one row). They are often used in the design and testing of software solutions, hardware testing, etc. CAs are particularly useful in applications that require combinations of different parameters to be tested, in which case it is often not practical to perform exhaustive tests owing to high costs in time and effort. Each row of a CA represents a test that indicates a combination of k parameter values. Coverage or strength (represented by t) is a measure of the coverage of interactions between the different parameters. The greater the coverage, the higher the number of tests required, and when coverage equals the number of k parameters, the array is equivalent to an exhaustive test. CA use can effectively reduce the number of tests needed, while still maintaining the effectiveness (coverage) of the task being undertaken [13]. Incremental Covering Arrays (ICA) are a variation of CAs. They work in the following manner: Within a matrix of strength t are sub-matrices of CAs with strengths less than t, and if a new matrix is desired with strength $t + 1$ or greater, all that is required is to add a certain number of rows, without having to completely regenerate the array [14].

Although these unique characteristics of CAs have been exploited in several other fields, we have not found evidence of their use in designing feature selection approaches. The main contributions of this article are therefore summarized as: i) to present a wrapper that makes it possible to integrate the qualities of covering arrays in carrying out feature selection across various classifiers traditionally used in data mining processes; ii) to compare the results of the wrapper using covering arrays and incremental covering arrays with several classifiers such as K nearest neighbors (KNN), C4.5, Naive Bayes (NB), Multi-Layer Perceptron (MLP), Random Forest (RF) and Support Vector Machines (SVM) at different degrees of strength; and iii) to compare the results with Genetic Algorithm (GA) and Particle Swarm Optimization (PSO), which have been employed in the state of the art for wrappers.

The rest of the article is organized as follows: Section II shows previous work related to filters and wrappers, and introduces the concept of covering arrays; Section III presents the methodology used to construct a wrapper using CA with different classifiers; Section IV presents the results from the evaluation and testing of the wrapper, and the final section contains conclusions and opportunities for future work.

II. RELATED WORK

A. WRAPPERS FOR FEATURE SELECTION

The main characteristic of feature selection via wrappers is that it uses the classification algorithm within the evaluation process on each subset of variables – hence making the process computationally intensive. The search for the best subset of variables is an *NP* problem that becomes computationally prohibitive as the number of input variables grows. Defining a search strategy is a pre-requisite, which directs the algorithm to evaluate the most promising solutions [6]. An evaluation criterion is also necessary. This defines the performance of the classifier that is obtained from a subset of variables. The general diagram of a wrapper is presented in Fig. 1, where the feature selection of the classifier works like a ''black box'' that allows the best subset of variables to be identified.

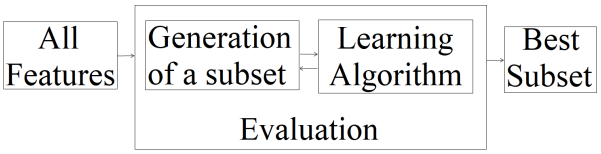

FIGURE 1. The wrapper approach to feature selection (Adapted from [15]).

Several variations have been adapted for the implementation of wrappers. The main feature that differentiates them is the search strategy used to define the subsets of variables that need to be evaluated [11]. *Methods of exponential complexity* involve computing a large number of tests. Among these is the exhaustive search. Although it guarantees a high probability of finding the best solution, it is impractical given the high computational costs, making it difficult to run even on small datasets [6], [9].

Another classic search strategy is the *sequential* approach, part of the family of greedy algorithms that use iterative functions. Among these algorithms, is the *forward* method, which starts from a null subset of variables, and in every subsequent step a new variable is added such that it enriches the performance of the classifier; this process is performed until no improvements are found, even after addition of another variable. Although *forward* returns reasonable results in a short time, its performance is poor in a number of cases due to its limitation of not being able to exclude variables that have been entered in previous iterations. The *backward* method, in contrast, starts with a solution that consists of the entire set of variables, and in an iterative way, variables are removed such that the performance of the classifier increases. The process is stopped when the exclusion of a variable no longer produces better results. The *backward* method becomes computationally expensive when processing classifiers with many variables [16].

Elsewhere, metaheuristics have also been widely used in the task of feature selection, beginning with *evolution-based* methods, the approach inspired by the process of evolution of species over time, with emphasis on the fact that recent species are better adapted to the changes generated in the environment compared to the older ones. Abd-Alsabour performed a review of evolutionary methods that have been implemented for feature selection [16]. This strategy starts with a set of solutions known as an initial population. These are optimized in subsequent iterations by means of operators that simulate processes of evolution or interactions observed in nature. The evaluation of each candidate solution is defined using a fitness function (accuracy, precision, recall, F measure, among others), and the search for the solution is stopped when a stop criterion is met. In general, population-based methods have several parameters that impact the quality of the results generated by the algorithm.

Among the most widely used population-based methods for feature selection with wrappers are: *genetic algorithms* (GA), for which solutions put forward are expressed in sequences of binary numbers indicating 0 for the absence of a variable and 1 for its presence. Each candidate solution represents a subset of variables and is called a chromosome; in an iteration of the evolution process, a set of operations (selection, crossover, mutation, and replacement) is applied to the population with the aim of improving the quality of the solutions in terms of their fitness function [17]–[20]. Analogous to GA, an approach based on Genetic Programming (GP) has been proposed as a wrapper. The operators for both algorithms are similar, however that used in GP has been customized in order to work with a tree-based structure [21]. For both GA and GP, although the search for better solutions ensures an improvement in quality and computational efficiency, there is a risk that the final solution will be biased towards a local optimum.

Another prominent metaheuristic approach is called *particle swarm optimization* (PSO). In this case the candidate solutions, known as particles, have the same characteristics as those of GA, with respect to the binary sequence representing the presence or absence of variables. A position and a velocity are then assigned to each solution, in order to define the magnitude of the change produced in the candidate solution in one iteration. Each solution moves through the search space, according to its current position and its velocity, but is also influenced by the information of the best position visited in previous iterations, the position of the best nearest neighbor or related particle, and the position of the best solution in the whole swarm [22], [23]. PSO-based algorithms have achieved satisfactory results in the realm of feature selection. They have therefore been the subject of much research, and some authors have proposed further variants that enable improved reproducibility of results relative to the original version [24]–[27].

Among other population-based methods used as search criteria in wrappers are *Ant Colony Optimization* (ACO) [28], [29], *Bee Colony Optimization* (BCO) [30], and *Grey Wolf Optimization* (GWO) [31], [32]. All of them perform the search for optimal solutions through operations that simulate collective integration and communication between individuals. However, as with GA, despite finding good solutions there is a risk of being trapped in local optimal [33]. Strategies based on metaheuristics usually involve a large number of parameters [34], such as the number of candidate solutions evaluated in each iteration, the probabilities of mutation or crossover in GA [35], and the maximum permitted velocity and acceleration constants in PSO. These parameters can influence the quality of the solution obtained [36], and for this reason implementations require parameter optimization, demanding computation time prior to performing definitive feature selection.

Hybrid approaches have been popular in the field of wrappers as well. For instance, a hybrid that uses the capability to exploit from PSO with the ability to explore from the GWO was proposed in [37]. Here, the authors modified the original version of the algorithm and used it in the context of a binary space, to make it suitable for feature selection. The respective tests were carried out using KNN. Another study proposed

in [38] for the diagnosis of diseases using SVM hybridized a dynamic ant colony system with wavelets transform and singular value decomposition in order to implement a feature selection approach and reduce the high-dimension of the data. The results of both hybrid-based wrappers were seen to be competitive, in terms of accuracy and number of features, with the state of the art of wrappers. However, for each research, wrapper performances were evaluated in each case with only one classifier.

In summary, wrappers are generally a good strategy for selecting an appropriate subset of variables in a dataset. However, the selection can be biased by the algorithm used in the modeling (SVM, KNN, Naive Bayes, C4.5, among others) and is computationally expensive. The search strategies used in wrapper design can be classified into three main groups: 1) exponential, whose use is feasible only with few variables, 2) sequential strategies that are mostly based on greedy methods (forward and backward) that deliver low-quality results and where the high dimensionality of the datasets affects the performance of the proposals (especially for Backward methods), and 3) metaheuristic approaches that include GA, PSO, ACO, BCO, GWO, Harmony Search [39], among others [40], which to date are the ones that have obtained the most promising results, although their use demands massive computation [11], [15].

Application in bioinformatics [41], content based image retrieval [42], and text mining [43] among other datasets that are high dimensional in nature pose challenges for metaheuristic-based wrappers. However, it is possible that clever hybrid proposals could be successful, due to the complex internal structure of these datasets [11], [15]. In addition, parallel metaheuristic proposals for feature selection should be considered, looking to use all computational resources, i.e. different kind of processors (CPU, GRP, and TPU), and all available memory, and to reduce the execution time to a minimum. Therefore, global (when only the fitness evaluation of the solution is parallelized), fine grained (when groups of solutions of the entire population evolve in parallel) and coarse grained (when the population is separated into islands and each of them runs independently) parallelization models should be analyzed in future proposals of metaheuristic wrappers [15].

Although CA has been extensively used in software testing, some authors have indeed used CAs as a sample strategy, to support the bagging process for embedded models, but not as a wrapper. Villegas *et al.* [44] employed CAs as a strategy for building bootstrap samples that are used in decision trees, for sentiment analysis. The tests with tweets showed the potential of the proposed method to reduce the indexes used in polarity detection. Vivas *et al.* [45] based on a strategy that uses CAs, modified the process of selecting features for training trees in Random Forest (RF). They proposed using the rows of CAs as a feature selector instead of the random selector used by the random forest algorithm by default. The combination of CA and RF produced promising results in terms of accuracy. Meanwhile, Dorado *et al.* [46] developed a variable importance measure that uses the subsets of features explored in any wrapper, for each variable. The importance measure was computed by obtaining the difference between the performance of the subsets that contain the variable versus the performance of the subsets that do not contain this variable. This difference is an estimation of the relative contribution to predicting the output in the classifier. However, in this work, CAs were used as an alternative approach without being formally presented. In addition, this work did not consider the use of ICAs or formal assessment for the feature selection algorithm, as the focus of the study was to develop a variable importance measure, and not the use of CAs/ICAs in wrappers.

$$CA(N = 5 : t = 2, k = 4, v = 2) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

FIGURE 2. Example of a binary CA of strength 2.

III. COVERING ARRAYS

CAs are mathematical objects in which several parameters or variables of interest are evaluated and each parameter contains a certain number of possibilities or values. Their applications have been limited to the fields of experimental design, biology, and engineering fault analysis, and the testing of software and hardware quality, all aimed at providing quality products without incurring exaggerated costs [47]. Compared to exhaustive tests, in which all possible combinations of parameters are evaluated, CAs make it possible to reduce the number of runs or tests based on a parameter called strength (t), which controls the minimum number of interactions between the parameters to be evaluated.

CAs may be expressed using the notation CA($N : t, k, v$), which represents a two-dimensional array (matrix) of size $N \times k$, where N refers to the number of tests, k the number of parameters, variables, or columns, v is the alphabet indicating the possible number of values that each parameter can take (for example, an array whose values are 0 and 1 is said to have an alphabet of 2, or is binary), and t is the strength or degree of interaction between parameters. The special feature of CAs is that any set t of columns that are extracted from the array contains all the possible combinations of v^t tuples in at least one of the rows [13]. Fig. 2 shows a CA of strength 2 (t=2) in which for any set of 2 columns the combination of values (0, 0), (0, 1), (1, 0) and (1, 1) always appears.

CAs can have different row sizes and the same strength. If one contains the least number of rows it is said to be an optimal CA and its notation is N = CAN(t, k, v). The construction of CAs is a complex problem, for which exact, algebraic, greedy and metaheuristics methods have been proposed [48]. Since CAs can normally be downloaded from open access repositories, the creation of these algorithms is not the aim of this work, but rather it is to use these CAs in building a new wrapper for feature selection purposes.

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \rightarrow Subset \rightarrow \begin{bmatrix} Null \\ x_2, x_3, x_4 \\ x_1, x_3, x_4 \\ x_1, x_2, x_4 \\ x_1, x_2, x_3 \end{bmatrix}$$

FIGURE 3. Extraction of a subset of variables using CA.

The underlying assumption with CAs and feature selection used in this study, is that 0 represents the absence of a variable and 1 represents its presence. Through such a formation, each row of a binary CA can be used as a reference to build a subset of candidate variables, which then can be used as input variables in the classifier. Later, all the rows can be evaluated, by implementing a classification model for each input variable subset and a fitness function (criterion), to select the best subset of variables. In this manner a new wrapper algorithm based on the CA approach can be developed and used as a search criterion.

The success of finding a good subset depends on the number of possible interactions that the array has considered, which can be controlled using the strength parameter (t). An example of a dataset with 4 variables is shown in Fig. 3, the subsets of variables formed by a CA of strength 2 ($t = 2$) and 4 parameters ($k = 4$), which correspond with the number of variables from the dataset.

IV. INCREMENTAL COVERING ARRAYS

An incremental covering array (ICA) [14] of strength t is denoted by: ICA($N_1, N_2, \ldots, N_t; t, k, v$) subject to $N_1 \leq N_2 \leq \cdots \leq N_t$, and satisfies that the first N_i rows is a CA($N_i; i, k, v$). As an example, an ICA of strength 4 ICA(2, 7, 13, 24; 4, 10, 2) can be seen in Fig. 4, where in the first two rows contain the CA(2; 1, 10, 2), additionally the first seven rows contain the CA(7; 2, 10, 2), furthermore the first thirteen rows contain the CA(13; 3, 10, 2), and lastly, all its twenty four rows contain the CA(24; 4, 10, 2). An ICA differs from a classical CA, in that it is built on the CAs of lowest strength, whereas the classical CA is constructed independently of the previous ones [14]. This characteristic of the ICA is useful when performing the experiments to analyze the influence of the strength on the results of feature selection, specifically in cases where it is desired to increase the strength, wherein to test at a greater strength (coverage), implies adding some rows (or tests) extra, ultimately resulting in a highly efficient process, unlike classic CAs, wherein increasing the strength, involves evaluating an array with many new rows and therefore a greater number of tests for the wrapper.

V. THE PROPOSED WRAPPER FOR FEATURE SELECTION BASED ON COVERING ARRAYS

In implementing the wrapper, the CAs used were of binary alphabet $v = 2$. This means that each component of the array has only values {0, 1}. In the construction of the wrapper, subsets of variables were constructed using the rows of the

$$\left.\begin{array}{l} t=3 \left\{\begin{array}{l} t=1 \left\{\begin{array}{cccccccccc} 0&1&0&1&0&1&0&1&0&1 \\ 1&0&1&0&1&0&1&0&1&0 \end{array}\right. \\ \left.\begin{array}{cccccccccc} 0&0&0&1&1&1&1&0&0&1 \\ 1&1&1&1&0&0&1&1&0&1 \\ 0&1&1&0&1&1&0&1&1&1 \\ 0&0&0&1&0&0&0&1&1&0 \end{array}\right\} t=2 \\ \begin{array}{cccccccccc} 1&1&0&0&0&1&0&0&0&0 \\ 1&1&0&1&1&0&0&0&1&1 \\ 1&0&0&1&0&0&1&0&0&0 \\ 1&1&1&0&0&0&0&1&1&0 \\ 1&0&1&1&1&1&0&1&0&0 \\ 0&1&1&1&1&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&1 \\ 0&0&1&1&1&0&1&1&1&1 \\ 0&1&0&0&0&0&1&0&1&1 \\ 0&0&0&0&1&1&0&0&1&0 \\ 1&1&1&0&1&1&1&0&0&1 \\ 0&1&0&0&1&0&1&1&0&0 \\ 1&0&1&1&0&1&0&0&1&1 \\ 1&0&0&0&0&1&1&1&1&1 \\ 1&0&0&0&1&0&0&1&0&1 \\ 0&1&1&1&0&1&1&0&1&0 \\ 1&1&0&1&1&1&1&1&1&0 \\ 0&0&1&0&0&1&1&1&0&0 \end{array} \end{array}\right\} t=4$$

FIGURE 4. ICA(2, 7, 13, 24; 4, 10, 2) with a maximum t of 4, $k = 10$ and $v = 2$.

CA, which defines the base of the search strategy. The number of parameters k, corresponds with the total number of input variables in the dataset, which is further explained at the end of Section III. A zero value (0) indicates the variable will not be considered, a one (1) means presence or inclusion (see Fig. 3).

Formalizing, where X of dimensions $n \times p$ is a set of data stored in an array, whose rows represent observations and columns represent input variables. Each row of the above array is related to a value of the output variable Y, which is a vector of dimensions $n \times l$ and contains the class or category to which each observation belongs. f represents a function of a classifier that predicts Y based on the values of X, such that the difference between actual and estimated values is minimal ($f(X) \approx Y$), and the quality of that estimate is evaluated from a quality indicator (e.g. classification accuracy), represented by α. Next, C_i, of dimensions $l \times p$, is the i-th row of a CA of strength t with parameters ($N : t = t, k = p, v = 2$) (N and n are different. N is the number of rows of the CA and n is the number of rows of X).

On expressing X as a set of vectors $[X_1\ X_2 \ldots X_j \ldots X_p]$, where X_j ($j = 1, 2, ..., p$) represents the j-th input variable and $C_{i.}$ a row of a CA that can be represented by a set of scalars $[C_{i1}, C_{i2}, \ldots, C_{ip}]$ where $C_{ij} \in \{0, 1\}$, both expressions are used as arguments to define a function proposed in this paper called *SelectInputSubset*, so that $Z_i = SelectInputSubset(X, C_{i.})$, in such a way that it receives the data matrix X, and the binary vector C_i to generate a new sub-matrix Z with a reduced number of columns of X, following the procedure described below:

For each column X_j that represents the j-th variable of the matrix, X and C_{ij} the j-th scalar of the vector $C_{i.}$ (both in the same column position), the Z matrix is constructed according

to the following rule: **If** $C_{ij} = 0$ **then** $X_j \notin Z_i$, **Otherwise** $X_j \in Z_i$.

Note that the number of columns of the new matrix Z_i corresponds to the number of components of the vector $C_{i.}$ whose value is 1. For example: *SelectInputSubset* ($X = [X_1\, X_2\, X_3\, X_4]$, $C_i = [1\, 1\, 0\, 1]$) $= [X_1\, X_2\, X_4] = Z_i$.

After using the *SelectInputSubset* function to obtain the sub-matrix Z_i, this is used to predict Y by means of the classifier f, i.e. $f(Z_i) \approx Y$. In this step, a classifier is trained using a cross-validation process, in addition to a parameter optimization process (the process is explained in Section VI.B). Finally, once the classification model is generated, an accuracy value α_i is calculated, which measures the performance of the subset of variables that make up the sub-matrix Z_i.

The wrapper consists of executing the previous process in which $f(SelectInputSubset\ (X, C_i)) = f(Z_i)$ $(i = 1, 2, \ldots, N)$ is evaluated using N experiments, i.e. for all the rows of the CA or ICA, a set of values of quality of classification $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_s, \ldots, \alpha_N\}$ is obtained in the end, which is the product of evaluating each of the classifiers. If α_s is the maximum value found in the vector α and it is located in position s, then the columns of the matrix Z_s correspond to the best subset of variables found by the wrapper. Algorithm 1 summarizes the proposed wrapper.

Algorithm 1 The Proposed Wrapper Based on Covering Arrays

Inputs : Data matrix X,
Variable of categorical response Y,
Covering Array or Tower of Covering Arrays C with N rows,
A classifier f, and
A quality indicator metric
Output: A matrix Z with equal or fewer columns than X which represent a subset of variables

```
begin
    Evaluate Y = f(X) optimizing the classifier
    parameters, calculate the metric and store result in α
    Best = X
    foreach row C_i in C do
        Z = SelectInputSubset(X, C_i) //define the
        subset of variables based on SelectInputSubset
        function
        Evaluate Y = f(Z) optimizing the classifier
        parameters, calculate the metric and stored in α_i
        for Y
        if α_i > α then Best = Z, α = α_i
        end if
    end foreach
    return Best
end
```

TABLE 1. General description of datasets.

Dataset	Variables or Features	Number of Class Values	Records or Instances
Glass	9	6	214
Wine	13	3	178
Zoo	16	7	101
Vehicle	18	4	846
WDBC	30	2	569
Ionosphere	34	2	351
Sonar	60	2	208

In short, the wrapper can be expressed according to (1).

$$\begin{aligned} WrapperCA(X, f, C) &= argmax_{Z_i}(\alpha_i) \\ Where\ \alpha_i &= metric(f(z_i))\ and \\ z_i &= selectInputSubset(X, C_i) \\ WrapperCA(X, f, C) &= argmax_{Z_i}(\alpha_i = metric(f(z_i))) \end{aligned} \quad (1)$$

VI. EXPERIMENTAL RESULTS

Implementation of the wrapper was performed in the statistical program R version 3.2.1 [49]. The main libraries used in the analysis were *snowfall* [50] to implement processes in parallel, *ggplot2* [51] as a graphical tool, and *caret* [52] for training and validation of the classifiers. The following section describes the test datasets, the classifiers implemented, and the results obtained by the wrapper.

A. DATASETS FOR VALIDATION

Evaluation of the wrapper with CAs and ICAs was performed using the data sets shown in Table. 1. These were selected because they are the most commonly used for evaluating feature selection procedures, as well as being different in the number of variables and number of classes in the response variable. Access to this data is free and is available in the repository of the University of California at Irvine (UCI) [53].

B. CLASSIFICATION MODELS AND PARAMETER TUNING

Table. 2 shows the six classifiers with which the proposed wrapper performance was evaluated. These were selected because they were widely used for classification tasks. Each model has its specific parameters, which were optimized using Cross Validation in a configuration of 10 folds with 5 replicates. The optimal parameters can change depending on the number of variables, and the information that these variables contain, so for each subset of candidate variables to be the solution of the wrapper, a parameter optimization was performed prior to obtaining a definitive classification model, with which the accuracy (the quality indicator selected for experiments) was evaluated. The optimization was carried out using the greedy algorithm presented in [54]. The algorithm divides the parameter grid into six regions of equal distance, and then evaluates the values corresponding to the limits of each region, to determine the most favorable configuration for the classifier. In case the classifier requires more than

TABLE 2. Classifiers and their corresponding parameters.

Acronym	Classifier	Parameter	Range
C4.5	J48 (Decision Tree C 4.5)	Confidence Factor	0.01-0.5
KNN	K nearest neighbors	Number of neighbors	5-17
NB	Naïve Bayes	Estimation density	Kernel, Normal
MLP	Multilayer perceptron	Number of units in the hidden layer.	1-13
RF	Random Forest	Number of variables randomly sampled as candidates at each split	1-p
SVM	Support Vector Machine	Cost of constraints violation and Gamma	0.25-16

one parameter, such as Multi-layer Perceptron (Number of units in the hidden layer and decay rate) or SVM (Cost and Gamma); those most important is selected for the optimization and, the remaining are maintained with default value configuration of the software. The column "Range" in table 2 shows the minimum and maximum values for the grid of each parameter. For NB, two possibilities in the density estimation were tested (Kernel or Normal). The Caret library in R facilitates this process [55].

C. CAS AND ICAS USED

The CAs and ICAs used for the experimentation and analysis were derived from the repository located in http://www.tamps.cinvestav.mx/∼oc. The CAs and ICAs were evaluated from strength 2 to strength 6, all with binary alphabet. Evaluation was carried out with different strengths to identify the one that achieves the best results in terms of performance for the wrapper. Each subset of data that was evaluated required a specific CA or ICA, since these are defined based on the number of variables in the data set (k value of the CA or ICA).

One special feature of ICA is that higher strength ICA's contains CAs of smaller strengths. This feature guarantees that as the strength of an ICA increases, the quality of the best solution found can only be maintained or improved. With classical CAs, the same does not happen, because a CA with higher strength does not necessarily have the same rows (test cases) as CAs of lower strengths, so performance does not necessarily increase as strength increases. To obtain an incremental accuracy using classical CAs, cumulative CAs (CCA)

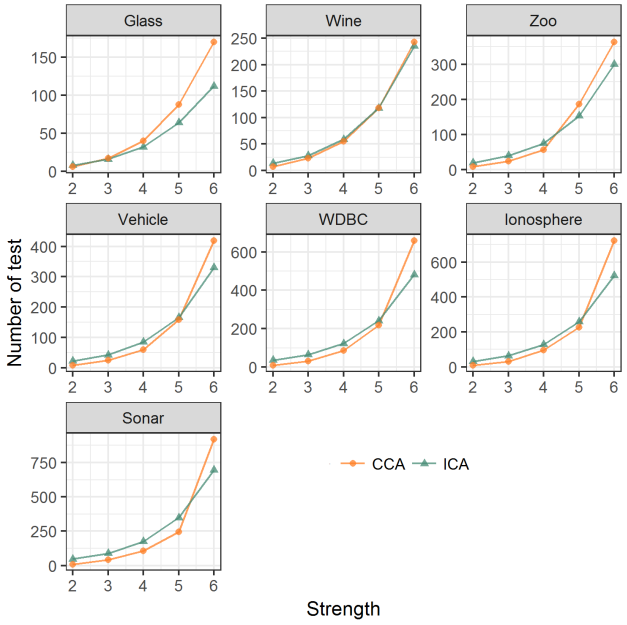

FIGURE 5. Number of tests required per strength level in each dataset.

were constructed and used, for which each time strength is increased, it is compared with the best solution found in the CA of lower strength, and if the performance found with the new CA does not improve, the best solution is still maintained. This implies that to evaluate a CCA of strength 4, CAs of strength 2 and 3 must be previously executed.

The number of tests required to evaluate the proposed wrapper depends on the number of variables in the dataset and on the strength of the CA or ICA to be used. Fig. 5 shows the strength of the CA or ICA on the horizontal axis (x-axis) and the number of CA or ICA test cases on the vertical axis. It can be seen that the number of test cases grows as the strength grows. In addition, for 6 of the 7 datasets it can be observed that at strengths of less than or equal to 5, the number of tests that are done with CCA is below the ICA. At strength 6, however, the reverse happens where the number of tests for CCA is higher than ICA.

D. RESULTS AND DISCUSSION

To analyze the results obtained by the wrapper with the two types of CAs, for each case (dataset-classifier), the accuracy obtained by the wrapper at each of the strengths (2-6) was compared against the accuracy obtained by the classifier (without the wrapper) with all variables (baseline, BL). Subsequently, the number of cases that exceeded that baseline at each of the strengths was calculated. The results obtained can be observed in Fig. 6, which show that for ICA there is a logarithmic growth whereas the CCAs show a more linear growth. Up to strength five, the results of the ICA are equal to or greater than the CCA, managing to exceed the BL 95% of the time. However, at strength 6, the CCAs outperform the ICAs with a difference of 2.3%, by managing always to exceed the baseline.

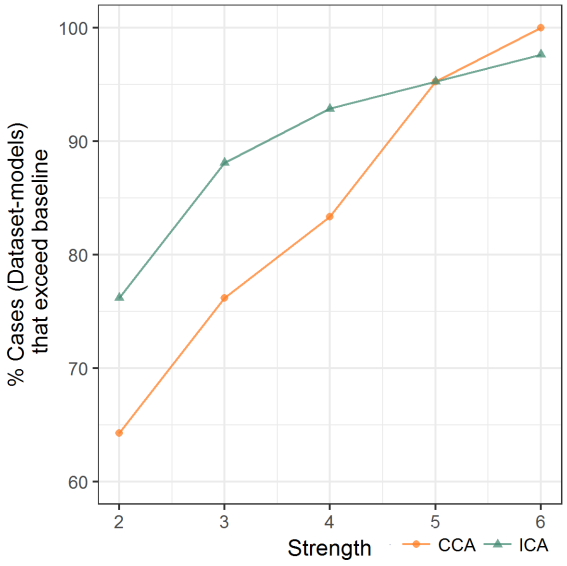

FIGURE 6. Percent of test datasets that exceed baseline according to the strength (CCA and ICA).

For both types of CA, meanwhile, the results of the wrapper suggest that the best strength is 6, wherein the highest number of cases that exceed the BL is achieved; however, even from strength two, both already surpass the BL 60% of the time. A similar finding was found in the field of software engineering: in [56], where it was demonstrated that software system failures can be detected with a high level of accuracy through the use of CAs even with small strengths, however, as this parameter increases, the results improve but require a greater number of tests. In [57], from experiments carried out with software for medical devices, servers, browsers and others, it was concluded that with strength 6, virtually 100% detection of faults are achieved without having to resort to exhaustive tests, which is consistent with the results found in Fig. 6.

Fig. 7 shows the increase in accuracy with respect to the BL in each case (Dataset-classifier) for the different wrappers evaluated, where it is observed that the accuracy after feature selection increased to 23.7% in the case of Wine with the KNN. However, for some data sets such as Vehicle, most classifiers reach a maximum of 1.44% increase, and therefore the wrapper does not produce a significant improvement. We observed that the data set and the classifier influence both the performance of the solution found, and the magnitude of increase observed, as CA strength increases. Although in some cases results lower than the BL are obtained, most of these occur at strength levels between 2 and 5, but this does not happen with strength 6.

Fig. 8 shows the increase in accuracy with respect to the BL across all datasets by a classifier. The atypical points can be seen in KNN that correspond to the improvement made on Wine, which exceeds 20%. It is also observed that the greatest benefit in feature selection (regardless of whether the wrapper uses accumulated CAs or ICAs) is KNN, followed by C4.5 and NB, which show an increase in accuracy greater than that obtained by the rest of classifiers. These results are expected, because both KNN and the first tree-based classifiers (C4.5 in this case) are quite sensitive to variables with noise [58]–[61]. Secondly, we found that for MLP and SVM classifiers, there is a slower growth in accuracy as strength increases, in relation to the above-mentioned classifiers. However, feature selection does benefit them, though to a lesser extent. This is due to the strong assumptions that these classifiers have about the independence of the variables and the presence of redundant variables [62]–[65]. Finally, the smallest increase comes from the RF classifier, which performs a feature selection process internally, controlled by the mtry parameter, which defines a number of variables that are randomly selected for the construction of each tree [66]–[68].

The results obtained by the proposed wrappers were also compared with two of the most successful algorithms used for optimizing search space in wrappers reported in the literature: Particle Swarm Optimization (PSO) and Genetic Algorithm (GA). Regarding the configuration of parameters for both, this was the same as that used in [2] (For GA, population size $= 20$, selection probability $= 0.8$, crossover probability $= 0.9$, and mutation probability $= 0.01$ was used. For PSO, swarm size $= 20$ and acceleration constants c_1 and $c_2 = 2$), and the number of evaluations of the objective function (objective fitness evaluations, OFEs) was adjusted to match the same number of cases evaluated by CCA in strength 6, thus ensuring a fair comparison, CCA being the one that requires a greater quantity of tests.

To carry out the execution, GA and PSO were programmed in R, and the *fitness function* was evaluated based on the *accuracy* obtained by the classifier and a subset of candidate variables chosen from a binary array. To ensure that the results are comparable, and to eliminate the effect produced by the partitions obtained in the cross-validation, the classifiers in all the wrappers were adjusted, so that the folds and training were with the same seed (initial random number), which enabled the same candidate solution to achieve the same result in PSO, GA, ICA or CCA.

The accuracy and number of variables achieved by the wrappers evaluated through the six classifiers and over the seven datasets can be observed in Table. 3. The comparison is performed similarly to that suggested in [69], [70], where the Friedman Aligned Ranks test is used to analyze the significance of difference between the results of the four wrappers plus the baseline, independently for each classifier. The significance level used was 0.05, and when significant differences were found in the results, a post hoc based on Friedman's Aligned Ranks test was performed, which indicates which of the wrappers are significantly different.

In Table. 3, capital letters are used for the results obtained for accuracy and lowercase for the comparison of the number of variables. Also, to facilitate the comparison of the performance of the approaches evaluated, an average of the rank of the Friedman test is used, where low values of accuracy

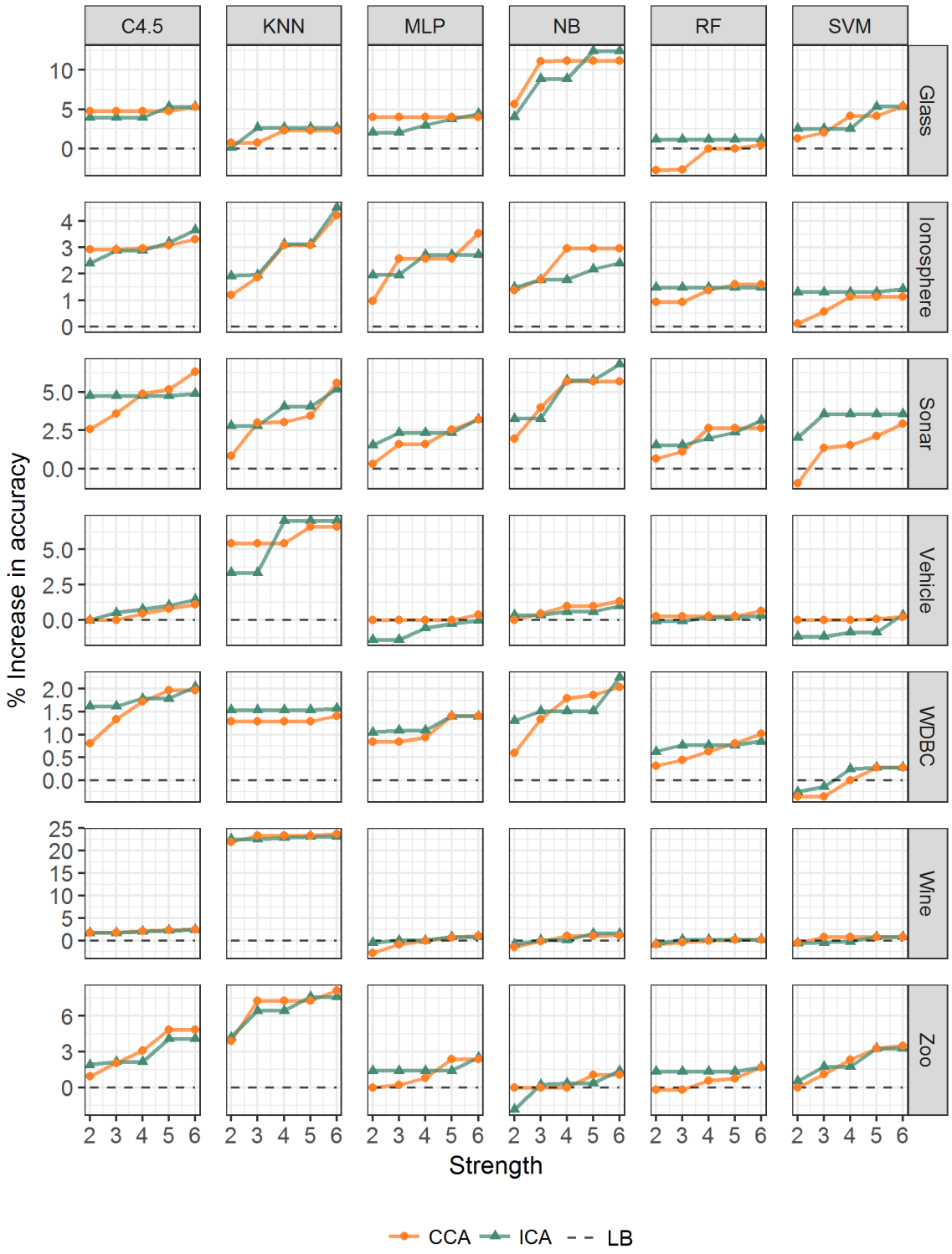

FIGURE 7. Increase in accuracy and strength for the wrappers proposed (CCA and ICA) by datasets and classifiers. Baseline (BL) accuracy obtained with the original dataset (all variables) is marked with a dashed black line.

close to one can be observed. These indicate the best performance, whereas, for counting of variables, values close to one indicate a large number of variables. On obtaining the results of the tests performed both for differences in accuracy and for variables, Friedman's Aligned Ranks showed statistical significance, and a comparison was therefore made using the post hoc tests in all the tests.

In comparisons of accuracy, Table. 3 shows that the baseline differs significantly with respect to most wrappers except for the results found for the RF classifier, where there is

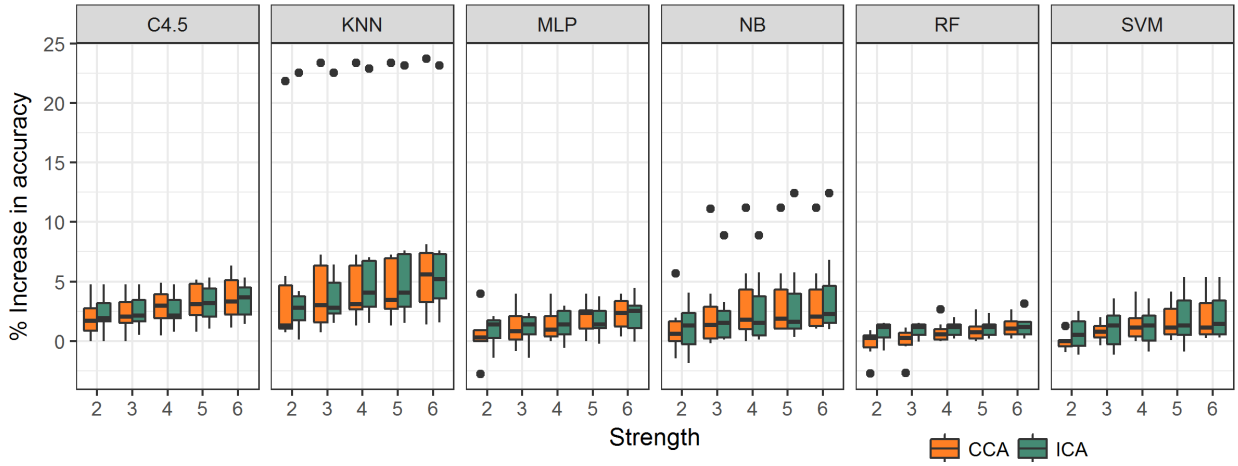

FIGURE 8. Accuracy comparison of the proposed method against baseline for the six classifiers. The x-axis shows the strength of the CCA and of the ICA.

no evidence of differences between the baseline with ICA and CCA. With regards to the mean rank value, PSO always obtained a higher accuracy than other wrappers, through all the models. In addition, in four of the six classifiers, this approach is the one that selects the least variables, making it the best-ranked wrapper.

In the NB, RF, MLP and SVM classifiers, the ICA and CCA wrapper have poorer performances and are statistically different from PSO. Based on the contrasts that involve GA versus CCA or ICA, only two significant differences were observed - one in MLP for ICA and another in RF for CCA. For the remaining models, the results of the comparison between GA, CCA and ICA did not reveal significant differences.

The advantage of PSO in regards to other wrappers evaluated is attributed to the fact that the swarm optimizes locally around each optimum found, and then if the optimum changes, the algorithm adapts and optimizes locally the new zone. In contrast, both CA-based wrappers do not implement local optimization, but only do a better distributed exploration using the interaction of the variables, which would explain the similarity of the results between them and the one based on GA.

Meanwhile, in the KNN classifier, the only difference presented was with respect to the baseline, whereas with the rest of the wrappers of the same classifier, there were none. In terms of the number of variables selected, it was observed that for all classifiers, the only differences observed were between the baseline and the wrappers. On contrasting the four wrappers, none was found to be different.

Another aspect analyzed in comparing the wrappers was the number of evaluations (OFEs) versus the accuracy achieved. These results can be observed in Fig. 9, where it can be clearly seen that CCA and ICA have a stepped growth, which in most cases does not match PSO or GA, which have a logarithmic gcrowth. This behavior is explained by the fact that CA-based wrappers do not use an exploitation approach, therefore upon finding a better solution, they continue to explore new solutions without prioritizing the findings, as opposed to population-based wrappers that focus on solutions with high potential. Also, we observed that the strength of CAs is dependent on the numbers of OFEs. Interestingly, the number of test cases performed at strength four on CCA is often located at the point where the GA curve stabilizes. In addition, the number of tests at strength six for CCA tends to agree with the number of tests where the PSO begins to stabilize. Hence it could be interpreted that the number of rows for CAs could serve as a guide to the number of OFEs needed to reach an acceptable solution for a GA or PSO, wrapper regardless of the classifier.

One of the limitations of the proposed algorithm is that the CCAs and ICAs should be generated apriori. This means for its implementation, a generator of CCAs or ICAs or the extraction of those arrays from external sources such as NIST [71] is required.

A property of CAs is that all rows tend to have the same number of ones, which in the wrapper means that they have the same number of features selected, which may be another limitation of the proposed method. This can be solved by making an iterative call off the proposed wrapper with the objective of looking for smaller subsets of features, experiments that the research group hopes to carry out in the near future.

Another limitation of the algorithm, as named above, is due to the lack of an exploitation approach. However, Fig. 10 shows the magnitude of the difference in accuracy between the wrappers proposed in this paper and the best algorithm (PSO). It can be observed, except for some maxima and atypical values, that the majority of these differences are below 0.02. Finally, although the results did not surpass this algorithm, the proposed wrappers have the advantage that they can be programmed in parallel more easily, and by fixing the strength of CCA and ICA, minimum additional

TABLE 3. Comparison of the accuracy (ACC) and number of variables (NF) obtained from the baseline, the proposed wrappers (CCA, ICA) and those based on PSO and GA.

Classifier	Dataset	Baseline		CCA		ICA		PSO		GA	
		Acc	NF	Acc	NF	Acc	NF	Acc	NF	Acc	NF
C4.5	Glass	0.6891	9	**0.7424**	6	**0.7424**	6	0.7406	6	0.74	**5**
	Ionosphere	0.8935	34	0.9265	14	0.9301	**12**	**0.9349**	**12**	0.9319	14
	Sonar	0.7508	60	0.814	27	0.7999	28	**0.8673**	**17**	0.8408	22
	Vehicle	0.7194	18	0.7305	14	0.7338	10	**0.7406**	**11**	0.7399	11
	WDBC	0.9406	30	0.9604	13	0.961	15	**0.9669**	**11**	0.963	12
	Wine	0.9348	13	0.9594	7	0.9584	8	**0.96**	**6**	0.9592	**6**
	Zoo	0.9298	16	**0.9783**	9	0.9707	**8**	0.9779	**8**	0.974	**8**
	Mean	0.84	25.71	0.87	12.86	0.87	12.43	**0.88**	**10.14**	**0.88**	11.19
	Std Desv	0.11	17.59	0.11	7.06	0.11	7.48	0.1	4.24	0.1	5.68
	Rank Mean	5	1	2.79	2.64	3.21	3.21	**1.43**	**4.21**	2.57	3.93
	Frid AR posH	A	b	BC	a	B	a	C	a	BC	a
KNN	Glass	0.6763	9	0.6991	**5**	**0.7027**	**5**	0.7006	6	0.6975	6
	Ionosphere	0.8434	34	0.8857	14	0.8885	12	**0.9183**	**8**	0.9098	9
	Sonar	0.8078	60	0.8635	32	0.8597	30	**0.8945**	**28**	0.8846	30
	Vehicle	0.6534	18	0.7195	**9**	0.7237	10	**0.7299**	11	0.7288	11
	WDBC	0.9354	30	0.9494	**14**	**0.9511**	15	0.9509	**14**	0.947	16
	Wine	0.7162	13	0.9533	**5**	0.9477	7	**0.9552**	8	0.9531	8
	Zoo	0.8498	16	0.9311	**10**	0.9258	12	**0.935**	**10**	0.9327	11
	Mean	0.78	25.71	0.86	12.71	0.86	13	**0.87**	**12.31**	0.86	12.91
	Std Desv	0.1	17.59	0.11	9.27	0.1	8.21	0.11	7.63	0.11	7.97
	Rank Mean	5	1	3.14	3.93	2.86	3.43	**1.29**	**3.79**	2.71	2.86
	Frid AR posH	B	b	A	a	A	a	A	a	A	a
NB	Glass	0.5677	9	0.6795	5	0.6918	6	**0.698**	5	0.6965	5
	Ionosphere	0.9105	34	0.9402	16	0.9345	18	**0.9473**	14	0.9447	15
	Sonar	0.7451	60	0.8019	32	0.8132	32	**0.8575**	26	0.8497	27
	Vehicle	0.6395	18	0.6529	11	0.6495	6	**0.6592**	9	0.6574	10
	WDBC	0.9409	30	0.9613	14	0.9634	9	**0.9746**	12	0.9732	12
	Wine	0.9763	13	0.9879	8	0.9922	8	**0.9938**	8	0.9935	8
	Zoo	0.8875	16	0.8979	10	0.9016	11	**0.9149**	9	0.9128	10
	Mean	0.81	25.71	0.85	13.71	0.85	12.86	**0.86**	**11.99**	**0.86**	12.4
	Std Desv	0.16	17.59	0.14	8.85	0.14	9.39	0.13	6.97	0.13	6.97
	Rank Mean	5	1	3.71	2.93	3.29	3.14	**1**	**4.29**	2	3.64
	Frid AR posH	A	b	B	a	B	a	C	a	BC	a
MLP	Glass	0.6555	9	0.6954	6	0.6999	**5**	**0.7114**	**5**	0.7017	**5**
	Ionosphere	0.8973	34	0.9328	17	0.9246	**12**	0.9392	15	**0.94**	15
	Sonar	0.8148	60	0.8469	24	0.8471	**22**	**0.8734**	29	0.873	29
	Vehicle	0.814	18	0.8177	**11**	0.8136	13	**0.8243**	14	0.8217	13
	WDBC	0.96	30	0.974	16	0.974	15	**0.9783**	**11**	0.9776	12
	Wine	0.9772	13	0.9874	**9**	0.9851	**9**	**0.988**	**9**	0.9875	**9**
	Zoo	0.9529	16	0.9765	9	0.9781	8	**0.9817**	**7**	0.9795	**7**
	Mean	0.87	25.71	0.89	13.14	0.89	**12**	**0.90**	12.91	**0.90**	13.03
	Std Desv	0.12	17.59	0.11	6.2	0.11	5.54	0.10	7.73	0.11	7.84
	Rank Mean	4.86	1	3.5	2.93	3.64	**3.86**	**1.14**	3.57	1.86	3.64
	Frid AR posH	A	b	BC	a	B	a	D	a	CD	a
RF	Glass	0.8062	9	0.8107	**6**	0.818	7	0.8156	7	**0.8157**	8
	Ionosphere	0.9321	34	0.9482	20	0.947	18	**0.9548**	**15**	0.9533	**15**
	Sonar	0.8437	60	0.8703	32	0.8752	**26**	**0.9197**	27	0.9107	29
	Vehicle	0.752	18	0.7586	**10**	0.7551	11	**0.7708**	**10**	0.7685	11
	WDBC	0.9627	30	0.9729	**13**	0.9712	15	**0.9777**	**13**	0.9765	**13**
	Wine	0.9864	13	0.9886	**8**	0.9886	12	**0.9906**	9	0.9901	9
	Zoo	0.9728	16	0.99	**7**	0.99	**7**	**0.9921**	8	0.9876	9
	Mean	0.89	25.71	0.91	13.71	0.91	13.71	**0.92**	**12.73**	0.91	13.36
	Std Desv	0.09	17.59	0.09	9.36	0.09	6.73	0.09	6.95	0.09	7.4
	Rank Mean	5	1	3.29	**3.86**	3.14	3.21	**1.29**	**3.86**	2.29	3.07
	Frid AR posH	A	b	A	a	AB	a	C	a	BC	a
SVM	Glass	0.7166	9	0.7702	**3**	0.7702	**3**	**0.7732**	**3**	0.7667	4
	Ionosphere	0.9515	34	0.9628	15	0.9658	18	**0.9752**	**17**	0.9721	18
	Sonar	0.8824	60	0.9115	30	0.9181	31	**0.9572**	**26**	0.947	30
	Vehicle	0.8308	18	0.8331	**13**	0.8343	**13**	**0.8478**	**13**	0.8452	**13**
	WDBC	0.9789	30	0.9817	**15**	0.9817	**15**	**0.984**	16	0.9832	18
	Wine	0.9853	13	0.9933	**7**	0.9932	8	**0.9947**	8	0.9943	9
	Zoo	0.9475	16	0.9821	11	0.9801	11	**0.9832**	**9**	0.9811	10
	Mean	0.9	25.71	0.92	13.43	0.92	14.14	**0.93**	**13.24**	**0.93**	14.38
	Std Desv	0.1	17.59	0.09	8.52	0.09	8.88	0.09	7.49	0.09	8.32
	Rank Mean	5	1	3.29	**4**	3.29	3.21	**1**	**4**	2.43	2.79
	Frid AR posH	A	b	B	a	B	a	C	a	BC	a

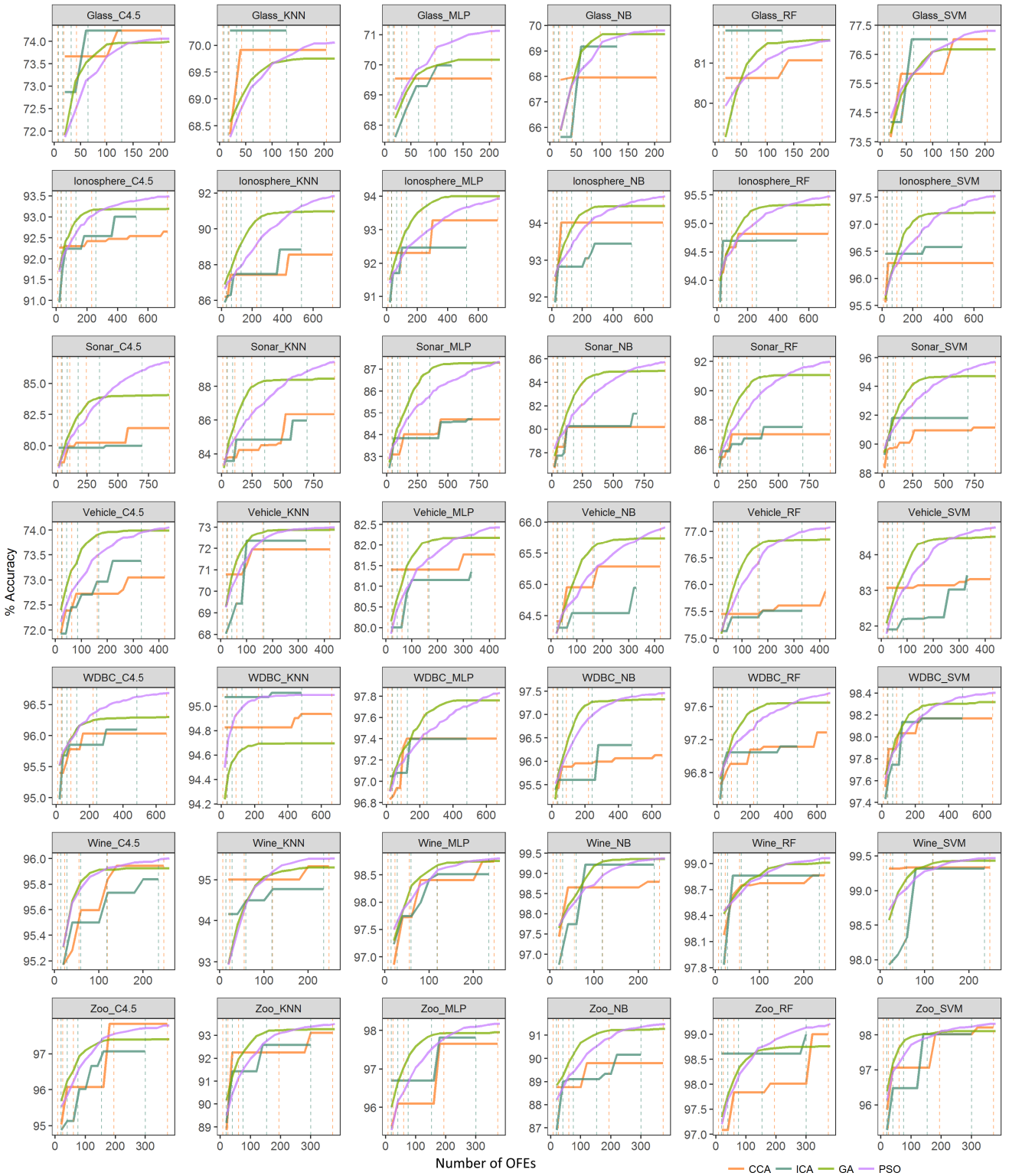

FIGURE 9. Comparison between the accuracy and number of evaluations for the wrappers (ICA, CCA, PSO and GA), by datasets and classifiers. The vertical lines represent the change in the strength of the covering array.

parametrization is required, in contrast to the classical version of GA and PSO where parameters such as size of the population or swarm, probabilities, velocities and various limiting values are involved. Therefore, according to the context in which the classifier is being evaluated, a researcher could take the option of minimally sacrificing accuracy for speed and simplicity, which is essential for large datasets that are becoming more common.

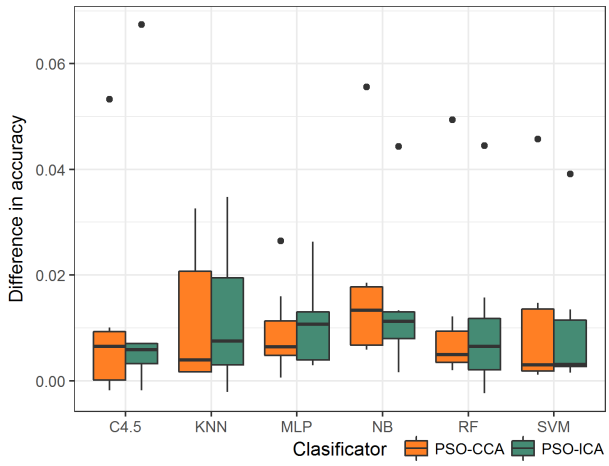

FIGURE 10. Difference between accuracy of PSO and the proposed algorithms (CCA and ICA).

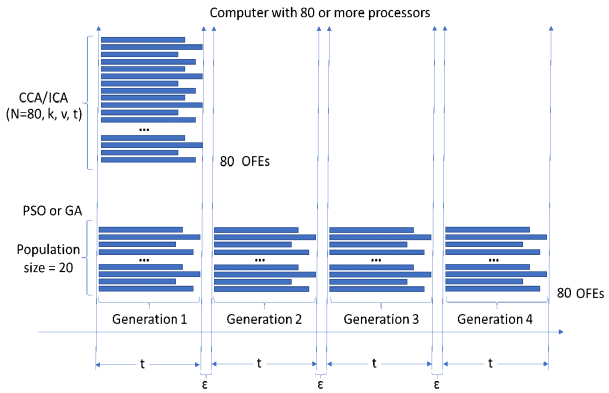

FIGURE 11. Comparison of parallel execution of the wrapper using CCAs/ICAs vs PSO/GA. t is the maximum time to evaluate an OFE, and ε is the internal time processing used by PSO/GA for updating particles or creating offspring in population/swarm.

E. PARALLEL EXECUTION

The parallel execution of the proposed wrapper using a CCA (or ICA) with 80 rows ($N = 80$) on a computer with 80 or more processors can be performed in time t. This time is the maximum evaluation time of an OFE, as shown in the upper left of Fig. 11. Moreover, when the wrapper is executed using PSO or GA using the global parallelization model, the execution time on the same computer is increased in proportion to the number of generations (or iterations) set for the algorithm. In the bottom part of Fig. 11, it can be seen that the execution of a PSO wrapper with a population size of 20, the particles are initialized randomly, and their evaluation (generation 1) in parallel takes time t,and time ε is expended, which is much smaller than t to update the velocities and positions of the particles (solutions) in the search space, to be able to perform the evaluation of generation 2 also with time t. The process with PSO, in this case, takes $4t + 3\varepsilon$ for the same 80 OFEs, which is greater than the time t used by the CCA. It is important to take into account that the number of generations in PSO and GA cannot be small, since with such iterations this metaheuristic approach will find better solutions for the optimization problem.

In general, the execution of the wrapper using PSO or a GA with a number g of generations and a population size ps, on a computer with a number C of processors, takes time equal to $gt + (g-1)\varepsilon$ when $C \geq ps$ and $(ps/C)gt + (g-1)\varepsilon$ when $C < ps$, leaving in the first case $C - ps$ idle processors. Taking into account that in practice it is not possible to use fractions of processors, the ceiling function is used to give more meaning to these expressions, and to unify them into the following $\lceil ps/C \rceil gt + (g-1)\varepsilon$. In the case of execution with a CCA of N rows, the execution time is t when $C \geq N$ and $(N/C)t$ when $C < N$, using in both cases more processors available in parallel since $N \geq ps$. These two expressions can be unified in $\lceil N/C \rceil t$.

F. APPLICATIONS

The wrapper proposed in this paper could readily be applied in many fields of science, including engineering, especially when the goal is to create predictive models and there are measurements of many variables. Some areas with the potential use of the proposed methods are text mining, to detect specific words that can be used to classify blogs, tweets, press releases, among others, having as potential applications: opinion evaluation, generation of summaries or spam detection. In this practice, the number of words could be elevated and discarding terms that do not contribute to the objective is a vital task before creating a definitive model. In addition, another area of application is industry, specifically in fault detection, where classification models are used in process monitoring and a number of variables are thus recorded in the operation. This number of variables can be more that the necessary and not all contribute for a correct diagnosis of the process. In the area of agriculture, models are developed to predict the yield based on variables related to climate, soil and agronomic management. Recording some variables from sensors, from soil analysis or direct monitoring in the crop could be expensive, and not all variables are relevant in terms of contributions for the predictive model, consequently, it will be useful to give priority only to the most important information. Finally, other fields of science, such as climate prediction, trends in the value of the country's currency, or bioinformatics, depend on multiple factors and often the relevance of these factors is unknown. Therefore, a wrapper based on CA offers a solution for detecting important variables that ought to be retained in the analysis.

VII. CONCLUSION

In this paper, a new method for feature selection with a wrapper approach was proposed and evaluated. It contains a search strategy based on binary Covering Arrays. The experiment was performed using two types of Covering Arrays: Cumulative Covering Arrays and Incremental Covering Arrays. The wrapper proposed was found to exceed the accuracy compared to the baseline, with increments in some cases up to 25%, additionally, the highest accuracy is reached for

both ICA and CCA at strength 6, and both wrappers show similar results in terms of the number of variables selected and accuracy. However, ICA requires less tests, which can be a signficative difference in the presence of datasets that have a very high number of variables. Models which are sensible to irrelevant or redundant variables - such as K nearest neighbor or C4.5 - have substantial improvements on using the wrapper.

The wrappers proposed were also compared with two population-based algorithms (GA and PSO). A nonparametric statistical analysis was performed to compare the differences between them, in which it was concluded that all wrappers are comparable in terms of number of variables chosen. In accuracy however, the proposed wrappers do not match the results obtained by PSO, but are similar to GA. Finally, the magnitude of the difference between the PSO-based wrapper and the cumulative CA-based wrappers and ICA was also analyzed, where it was found that more than 90% of the cases have a difference in accuracy of less than 0.04, which could be compensated in terms of the time involved in optimizing parameters, and the possibility of computing a greater number of executions in parallel, while reducing the time to reach a better solution.

VIII. FUTURE WORK

As a follow up, we propose mixing the capacity of CAs with heuristics or metaheuristics such as hill climbing, simulated annealing, or Tabu Search, all of which exploit the best solutions found with CAs and could improve the proposed algorithm significantly. Also, we suggest evaluating the iterative execution of the proposed wrapper to reduce the subset of features selected from the dataset. Additionally, the inclusion of other wrappers in the comparison, such as random search, greedy algorithm, extensions of GA and PSO, and testing with large datasets such as ALL/AML, Leukemia, Colon, or Prostate. It should also now be possible, to construct families of cumulative CAs by ensuring that all the CAs of the same strength are isomorphic, and evaluate the impact of this on the feature selection process.

ACKNOWLEDGMENT

The authors acknowledge "Xiuhcoatl"-CGSTIC of CINVESTAV for providing access of high-performance computing. Both CCAFS and the Platform for Big Data in Agriculture are carried out with support from CGIAR Trust Fund Donors and through bilateral funding agreements. For details please visit https://www.cgiar.org/funders/.

REFERENCES

[1] C. L. P. Chen and C.-Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: A survey on big data," *Inf. Sci.*, vol. 275, pp. 314–347, Aug. 2014.
[2] H. Wang, Z. Xu, H. Fujita, and S. Liu, "Towards felicitous decision making: An overview on challenges and trends of big data," *Inf. Sci.*, vol. 367, pp. 747–765, Nov. 2016.
[3] J. Franklin, "The elements of statistical learning: Data mining, inference and prediction," *Math. Intell.*, vol. 27, no. 2, pp. 83–85, 2005.
[4] G. I. Webb, M. J. Pazzani, and D. Billsus, "Machine learning for user modeling," *User Model. User-Adapted Interact.*, vol. 11, nos. 1–2, pp. 19–29, 2001.
[5] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, Jan. 2003.
[6] S. Kotsiantis, "Feature selection for machine learning classification problems: A recent overview," *Artif. Intell. Rev.*, vol. 42, no. 1, pp. 157–176, 2011.
[7] J. R. Vergara and P. A. Estévez, "A review of feature selection methods based on mutual information," *Neural Comput. Appl.*, vol. 24, no. 1, pp. 175–186, 2014.
[8] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Comput. Elect. Eng.*, vol. 40, no. 1, pp. 16–28, Jan. 2014.
[9] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artif. Intell.*, vol. 97, nos. 1–2, pp. 273–324, 1997.
[10] O. Soufan, D. Kleftogiannis, P. Kalnis, and V. B. Bajic, "DWFS: A wrapper feature selection tool based on a parallel genetic algorithm," *PLoS ONE*, vol. 10, no. 2, 2015, Art. no. e0117988.
[11] A. Jović, K. Brkić, and N. Bogunović, "A review of feature selection methods with applications," in *Proc. 38th Int. Conv. Inf. Commun. Technol., Electron. Microelectron. (MIPRO)*, May 2015, pp. 1200–1205.
[12] Z. Y. Algamal and M. H. Lee, "Penalized logistic regression with the adaptive LASSO for gene selection in high-dimensional cancer classification," *Expert Syst. Appl.*, vol. 42, no. 23, pp. 9326–9332, 2015.
[13] G. Tzanakis, L. Moura, D. Panario, and B. Stevens, "Constructing new covering arrays from LFSR sequences over finite fields," *Discrete Math.*, vol. 339, no. 3, pp. 1158–1171, 2016.
[14] J. Torres-Jimenez, I. Izquierdo-Marquez, R. N. Kacker, and D. R. Kuhn, "Tower of covering arrays," *Discrete Appl. Math.*, vol. 190, pp. 141–146, Aug. 2015.
[15] N. El Aboudi and L. Benhlima, "Review on wrapper feature selection approaches," in *Proc. Int. Conf. Eng. MIS (ICEMIS)*, Sep. 2016, pp. 1–5.
[16] N. Abd-Alsabour, "A review on evolutionary feature selection," in *Proc. Eur. Modelling Symp.*, Oct. 2014, pp. 20–26.
[17] R. C. Anirudha, R. Kannan, and N. Patil, "Genetic algorithm based wrapper feature selection on hybrid prediction model for analysis of high dimensional data," in *Proc. 9th Int. Conf. Ind. Inf. Syst. (ICIIS)*, Dec. 2014, pp. 1–6.
[18] C.-F. Tsai, W. Eberle, and C.-Y. Chu, "Genetic algorithms in feature and instance selection," *Knowl.-Based Syst.*, vol. 39, pp. 240–247, Feb. 2013.
[19] S. Oreski and G. Oreski, "Genetic algorithm-based heuristic for feature selection in credit risk assessment," *Expert Syst. Appl.*, vol. 41, no. 4, pp. 2052–2064, 2014.
[20] A. K. Das, S. Das, and A. Ghosh, "Ensemble feature selection using bi-objective genetic algorithm," *Knowl.-Based Syst.*, vol. 123, pp. 116–127, May 2017.
[21] M. Alweshah, O. A. Alzubi, J. A. Alzubi, and S. Alaqeel, "Solving attribute reduction problem using wrapper genetic programming," *Int. J. Comput. Sci. Netw. Secur.*, vol. 16, no. 5, p. 77, 2016.
[22] X. Liu and L. Shang, "A fast wrapper feature subset selection method based on binary particle swarm optimization," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2013, pp. 3347–3353.
[23] B. Xue, M. Zhang, and W. N. Browne, "Particle swarm optimization for feature selection in classification: A multi-objective approach," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 1656–1671, Dec. 2013.
[24] Y. Lu, M. Liang, Z. Ye, and L. Cao, "Improved particle swarm optimization algorithm and its application in text feature selection," *Appl. Soft Comput.*, vol. 35, pp. 629–636, Oct. 2015.
[25] B. Xue, M. Zhang, and W. N. Browne, "Particle swarm optimisation for feature selection in classification: Novel initialisation and updating mechanisms," *Appl. Soft Comput.*, vol. 18, pp. 261–276, May 2014.
[26] P. Moradi and M. Gholampour, "A hybrid particle swarm optimization for feature subset selection by integrating a novel local search strategy," *Appl. Soft. Comput.*, vol. 43, pp. 117–130, Jun. 2016.
[27] T. Butler-Yeoman, B. Xue, and M. Zhang, "Particle swarm optimisation for feature selection: A hybrid filter-wrapper approach," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, May 2015, pp. 2428–2435.
[28] I. F. de Viana, P. J. Abad, J. L. Álvarez, and J. L. Arjona, "Applying ant colony hybrid metaheuristics to wrapper verification," *Expert Syst. Appl.*, vol. 57, pp. 62–75, Sep. 2016.
[29] B. Chen, L. Chen, and Y. Chen, "Efficient ant colony optimization for image feature selection," *Signal Process.*, vol. 93, no. 6, pp. 1566–1576, 2013.

[30] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga, "A comprehensive survey: Artificial bee colony (ABC) algorithm and applications," *Artif. Intell. Rev.*, vol. 42, no. 1, pp. 21–57, 2014.

[31] E. Emary, H. M. Zawbaa, and A. E. Hassanien, "Binary grey wolf optimization approaches for feature selection," *Neurocomputing*, vol. 172, pp. 371–381, Jan. 2016.

[32] Q. Al-Tashi, H. Rais, and S. Jadid, "Feature selection method based on grey wolf optimization for coronary artery disease classification," in *Proc. Int. Conf. Reliable Inf. Commun. Technol.* Cham, Switzerland: Springer, 2018, pp. 257–266.

[33] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Comput. Surv.*, vol. 35, no. 3, pp. 268–308, Sep. 2003.

[34] I. BoussaïD, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," *Inf. Sci.*, vol. 237, pp. 82–117, Jul. 2013.

[35] K. Sastry, D. Goldberg, and G. Kendall, "Genetic algorithms," in *Search Methodologies*. Boston, MA, USA: Springer, 2005, pp. 97–125.

[36] V. Kothari, J. Anuradha, S. Shah, and P. Mittal, "A survey on particle swarm optimization in feature selection," in *Proc. Int. Conf. Comput. Commun. Syst.* Berlin, Germany: Springer, 2011, pp. 192–201.

[37] Q. Al-Tashi, S. J. A. Kadir, H. M. Rais, S. Mirjalili, and H. Alhussian, "Binary optimization using hybrid grey wolf optimization for feature selection," *IEEE Access*, vol. 7, pp. 39496–39508, 2019.

[38] Q. Al-Tashi, H. Rais, and S. J. Abdulkadir, "Hybrid swarm intelligence algorithms with ensemble machine learning for medical diagnosis," in *Proc. 4th Int. Conf. Comput. Inf. Sci. (ICCOINS)*, Aug. 2018, pp. 1–6.

[39] N. Yusup, A. M. Zain, and A. A. Latib, "A review of Harmony Search algorithm-based feature selection method for classification," *J. Phys., Conf. Ser.*, vol. 1192, no. 1, 2019, Art. no. 012038.

[40] W. Liu and J. Wang, "A brief survey on nature-inspired metaheuristics for feature selection in classification in this decade," in *Proc. IEEE 16th Int. Conf. Netw., Sens. Control (ICNSC)*, May 2019, pp. 424–429.

[41] N. Almugren and H. Alshamlan, "A survey on hybrid feature selection methods in microarray gene expression data for cancer classification," *IEEE Access*, vol. 7, pp. 78533–78548, 2019.

[42] M. S. Lotfabadi, Y. Zhan, and A. B. Tabrizi, "A review of wrapper feature selection in content based image retrieval systems," in *Proc. 10th Int. Conf. Mach. Learn. Comput.*, 2018, pp. 178–183.

[43] X. Deng, Y. Li, J. Weng, and J. Zhang, "Feature selection for text classification: A review," *Multimedia Tools Appl.*, vol. 78, no. 3, pp. 3797–3816, 2019.

[44] J. Villegas, C. Cobos, M. Mendoza, and E. Herrera-Viedma, "Feature selection using sampling with replacement, covering arrays and rule-induction techniques to aid polarity detection in twitter sentiment analysis," in *Proc. Ibero-Amer. Conf. Artif. Intell.* Cham, Switzerland: Springer, 2018, pp. 467–480.

[45] S. Vivas, C. Cobos, and M. Mendoza, "Covering arrays to support the process of feature selection in the random forest classifier," in *Proc. Int. Conf. Mach. Learn., Optim., Data Sci.* Cham, Switzerland: Springer, 2018, pp. 64–76.

[46] H. Dorado, C. Cobos, J. Torres-Jimenez, D. Jimenez, and M. Mendoza, "A proposal to estimate the variable importance measures in predictive models using results from a wrapper," in *Proc. Int. Conf. Mining Intell. Knowl. Explor.* Cham, Switzerland: Springer, 2018, pp. 369–383.

[47] B. S. Ahmed and K. Z. Zamli, "A review of covering arrays and their application to software testing," *J. Comput. Sci.*, vol. 7, no. 9, p. 1375, 2011.

[48] J. A. Timaná-Peña, C. A. Cobos-Lozada, and J. Torres-Jimenez, "Metaheuristic algorithms for building covering arrays: A review," *Revista Facultad Ingeniería*, vol. 25, no. 43, pp. 31–45, 2016.

[49] R Core Team. (2018). R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. [Online]. Available: https://www.R-project.org/

[50] J. Knaus, C. Porzelius, H. Binder, and G. Schwarzer, "Easier parallel computing in R with snowfall and sfcluster," *R J.*, vol. 1, no. 1, pp. 54–59, 2009.

[51] H. Wickham, *Ggplot2: Elegant Graphics for Data Analysis*, 2nd ed. New York, NY, USA: Springer-Verlag, 2016. [Online]. Available: https://ggplot2.tidyverse.org. doi: 10.1007/978-3-319-24277-4.

[52] M. Kuhn, "Building predictive models in R using the caret package," *J. Stat. Softw.*, vol. 28, no. 5, pp. 1–26, 2008. [Online]. Available: https://www.jstatsoft.org/v028/i05. doi: 10.18637/jss.v028.i05.

[53] D. Dua and C. Graff. (2017). *UCI Machine Learning Repository*. [Online]. Available: http://archive.ics.uci.edu/ml

[54] S.-W. Lin, Z.-J. Lee, S.-C. Chen, and T.-Y. Tseng, "Parameter determination of support vector machine and feature selection using simulated annealing approach," *Appl. Soft Comput.*, vol. 8, no. 4, pp. 1505–1512, 2008.

[55] M. Kuhn, "Building predictive models in R using the caret package," *J. Statist. Softw.*, vol. 28, no. 5, pp. 1–26, 2008.

[56] C. Yilmaz, M. B. Cohen, and A. A. Porter, "Covering arrays for efficient fault characterization in complex configuration spaces," *ACM SIGSOFT Softw. Eng. Notes*, vol. 32, no. 4, pp. 45–54, 2006.

[57] D. R. Kuhn, D. R. Wallace, and A. M. Gallo, "Software fault interactions and implications for software testing," *IEEE Trans. Softw. Eng.*, vol. 30, no. 6, pp. 418–421, Jun. 2004.

[58] J. A. SáEz, J. Luengo, and F. Herrera, "Predicting noise filtering efficacy with data complexity measures for nearest neighbor classification," *Pattern Recognit.*, vol. 46, no. 1, pp. 355–364, 2013.

[59] C. J. Mantas and J. Abellán, "Credal decision trees in noisy domains," in *Proc. ESANN*, 2014, pp. 1–6.

[60] E. Tuv, A. Borisov, G. Runger, and K. Torkkola, "Feature selection with ensembles, artificial variables, and redundancy elimination," *J. Mach. Learn. Res.*, vol. 10, pp. 1341–1366, Jul. 2009.

[61] D. D. Lewis, "Naive (Bayes) at forty: The independence assumption in information retrieval," in *Proc. Eur. Conf. Mach. Learn.* Berlin, Germany: Springer, 1998, pp. 4–15.

[62] B. B. Averbeck, P. E. Latham, and A. Pouget, "Neural correlations, population coding and computation," *Nature Rev. Neurosci.*, vol. 7, no. 5, p. 358, 2006.

[63] S. Halkjær and O. Winther, "The effect of correlated input data on the dynamics of learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 1997, pp. 169–175.

[64] S. Maldonado and R. Weber, "A wrapper method for feature selection using support vector machines," *Inf. Sci.*, vol. 179, no. 13, pp. 2208–2217, 2009.

[65] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A practical guide to support vector classification," Dept. Comput. Sci., Nat. Taiwan Univ., Taipei, Taiwan, Tech. Rep., 2003. [Online]. Available: https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

[66] P. Wei, Z. Lu, and J. Song, "Variable importance analysis: A comprehensive review," *Rel. Eng. Syst. Saf.*, vol. 142, pp. 399–432, Oct. 2015.

[67] K. Archer and R. V. Kimes, "Empirical characterization of random forest variable importance measures," *Comput. Statist. Data Anal.*, vol. 52, no. 4, pp. 2249–2260, Jan. 2008.

[68] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[69] S. García, A. Fernández, J. Luengo, and F. Herrera, "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power," *Inf. Sci.*, vol. 180, no. 10, pp. 2044–2064, 2010.

[70] A. Zarshenas and K. Suzuki, "Binary coordinate ascent: An efficient optimization technique for feature subset selection for machine learning," *Knowl.-Based Syst.*, vol. 110, pp. 191–201, Oct. 2016.

[71] A. Hartman, "Software and hardware testing using combinatorial covering suites," in *Graph Theory, Combinatorics and Algorithms*. Boston, MA, USA: Springer, 2005, pp. 237–266.

HUGO DORADO received the B.Sc. degree in statistics from the Universidad del Valle, Cali, Colombia, in 2013, and the M.Sc. degree in computing from the Universidad del Cauca, Popayán, Colombia, in 2019.

He is currently a Research Assistant with the Research Group of Decision and Policy Analysis (DAPA), International Center for Tropical Agriculture (CIAT). His role has been focused in analyzing observational information in agriculture by using analytical methodologies for detecting patterns associated with either high or low yields. His research interests include data mining, computational intelligence, and statistical science.

CARLOS COBOS (SM'13) was born in Bucaramanga, Colombia, in 1971. He received the B.S. degree in systems engineering and the M.S. degree in informatics from the Universidad Industrial de Santander, Bucaramanga, in 1995 and 2004, respectively, and the Ph.D. degree *(summa cum laude)* in systems engineering and computation from the Universidad Nacional de Colombia, Bogota, in 2014. From 1995 to 1999, he was a Lecturer with the Universidad Industrial de Santander. Since 1999, he has been a Titular Professor and Researcher with the Computer Science Department, Universidad del Cauca, Colombia. He has coauthored three books, 14 JCR articles, and more than 40 SJR and Scielo articles. He has tutored two Doctorates and 13 master graduates. His research interests include data mining, computational intelligence, deep learning, machine learning, natural language processing, information retrieval, and text mining. His main international collaborations are with Spain, Mexico, Australia, and USA. He is classified as a Senior Researcher in the National System of Researchers, Colombia.

JOSE TORRES-JIMENEZ (SM'99) received the Ph.D. degree from Monterrey Institute of Technology and Higher Education (ITESM), Cuernavaca, Mexico. He is currently a Lecturer and a Researcher with Cinvestav Tamaulipas, Mexico. He is also an Expert in combinatorial optimization. He has graduated more than ten Ph.D. professionals and more than 50 M.Sc. professionals. He has dedicated more than a decade to build many of the best-known covering arrays (mathematical objects that are used to do software and hardware testing). He has many international collaborations with USA, Spain, Colombia, France, and Austria. He is a Level III Member of the National System of Researchers, Mexico.

DHARANI DHAR BURRA received the M.S. degree in biological sciences from the King Abdullah University of Science and Technology, Saudi Arabia, in 2011, and the Ph.D. degree in agricultural sciences and statistics from the Swedish University of Agricultural Sciences, Sweden, in 2016. He is currently a Data Scientist with the CGIAR Platform for Big Data in Agriculture, he develops and tests new methodologies, that use big data sources (structured and unstructured) in combination with ground truth data and analytics in pursuit of developing near real time agri-food monitoring systems for national governments, farmer cooperatives, and donor agencies. He is currently involved in developing intelligent and integrated decision support systems for smallholder farmers in the tropics and national level food security monitors for a host of countries in South East Asia, under the ambit of monitoring of sustainable developmental goals.

MARTHA MENDOZA received the B.S. degree in systems engineering and the M.S. degree in informatics from the Universidad Industrial de Santander (UIS), Bucaramanga, in 1995 and 2004, respectively, and the Ph.D. degree *(summa cum laude)* in systems engineering and computation from the Universidad Nacional de Colombia, Bogota, in 2016. From 1995 to 1999, she was a Lecturer with the Universidad Industrial de Santander. Since 1999, she has been a Titular Professor and a Researcher with the Computer Science Department, Universidad del Cauca, Colombia. She has coauthored nine JCR articles and more than 40 SJR and Scielo articles. She has tutored four M.Sc. and 20 B.Sc. graduates. Her research interests include data warehouse, computational intelligence, and machine learning.

DANIEL JIMÉNEZ received the B.Sc. degree in agronomy from the Universidad de Caldas, Colombia, in 2002, and the Ph.D. degree in applied agriculture from Ghent University, Belgium, in 2013. He was with Bioversity International and the University of Applied Sciences Western Switzerland (HEIG-VD). He was also a Consultant with the French Agricultural Research Centre for International Development (CIRAD). He is currently a Scientist with the International Center for Tropical Agriculture (CIAT). He has been a Pioneer of using artificial intelligence techniques for agricultural research in developing countries, and he coordinates the Data-Driven Agronomy Community of Practice of the CGIAR Platform for Big Data in Agriculture. His work has received the recognition from the World Bank Group, in 2015, and the United Nations, in 2014 and 2017, and the team he leads at CIAT took the first prize at the Syngenta 2018 Crop Challenge in Analytics.