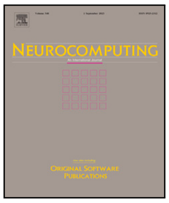
Contents lists available at ScienceDirect

Neurocomputing

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



HYB-PARSIMONY: A hybrid approach combining Particle Swarm Optimization and Genetic Algorithms to find parsimonious models in high-dimensional datasets

Jose Divasón, Alpha Pernia-Espinoza, Francisco Javier Martinez-de-Pison [*]

University of La Rioja, Spain

A R T I C L E I N F O

Keywords:
HYB-PARSIMONY
PSO-PARSIMONY
Parsimonious modeling
Auto machine learning
GA-PARSIMONY

A B S T R A C T

The PSO-PARSIMONY methodology (a heuristic for finding accurate and low-complexity models with particle swarm optimization (PSO)) allows obtaining machine learning models with a good balance between accuracy and complexity. However, when the datasets are of high dimensionality, the methodology does not sufficiently reduce the complexity of the models. This paper presents a new hybrid methodology, called HYB-PARSIMONY, that combines PSO with genetic algorithm (GA) based methods. In the early stages of the optimization process, GA methods have a preponderance to accelerate the search for parsimony. Later, PSO becomes more relevant to improve accuracy. This new methodology obtains significant improvements in the search for more accurate and low-complexity models in high-dimensional datasets.

1. Introduction

The success of machine learning techniques in practically all fields of science and industry has led to an increasing demand for optimization heuristics and tools to facilitate some typical tasks such as hyperparameter optimization (HO) and feature selection (FS). Companies try to reduce the risk of overfitting by selecting the least complex (most parsimonious) model among those with similar accuracy [1]. A less complex model will have more stable predictions and robustness to noise and disturbances, as well as being easier to maintain and analyze [2]. Likewise, companies are increasingly demanding models with a reduced number of input variables that facilitate the study and identification of biases that may affect decision making [3].

Another aspect to consider is energy consumption. Although intensive search heuristics may initially involve a higher energy cost, the use of simple models with few features versus ensemble models with a large number of inputs can help to compensate for the initial energy costs by using fewer resources (memory, cpu usage, etc.). Likewise, the reduction of features can lead to energy savings due to a lower use of sensors, lower information acquisition costs, lower maintenance, reduction of the need to retrain the models due to the reduction of features that can be perturbed over time (with noise, outliers, data drift), etc. Similarly, an automated systematic search can reduce time and energy costs compared to other classical methods based on multiple non-systematic tests [4].

An attempt to improve the results of other methodologies was to develop a new algorithm that combined the particle swarm optimization technique (PSO) and the parsimony criteria to obtain high-accuracy and low-complexity models.

The algorithm, named PSO-PARSIMONY [5], performs a finer adjustment and finds more accurate solutions than other methods, although the computational cost is higher and the obtained models from high-dimensional datasets are less parsimonious (higher complexity).

This work proposes a hybrid model between PSO [6] and genetic algorithms (GA) [7] in which, in the first stages and to accelerate the search for parsimony, the particles with worse fitness values are replaced in each iteration by new ones generated by typical GA operations: selection, crossover and mutation. After that, PSO optimization becomes more relevant to improve the accuracy search.

2. Related work

Nowadays, AutoML frameworks such as Autogluon [8], MLJar [9] and H2O AutoML [10], among others, address modeling problems with high dimensional tabular datasets by creating ensemble models formed by artificial neural networks and tree-based methods such as RandomForest, LightGBM [11], XGBoost [12] and CatBoost [13]. The use of robust training and validation methods makes it possible to

* Corresponding author.
E-mail addresses: jose.divason@unirioja.es (J. Divasón), alpha.pernia@unirioja.es (A. Pernia-Espinoza), fjmartin@unirioja.es (F.J. Martinez-de-Pison).

[1] https://github.com/eli5-org/eli5

https://doi.org/10.1016/j.neucom.2023.126840
Received 10 February 2023; Received in revised form 11 July 2023; Accepted 23 September 2023
Available online 27 September 2023

automatically obtain highly accurate models without having to perform a previous feature selection. However, this combination of highly complex models may contain biases that are difficult to detect. This is why companies increasingly prefer explainable models with a reduced number of input variables, even if their accuracy is lower than that obtained with ensemble models. For example, a linear model or a decision tree with few rules can be much more valuable in decision making. Even a black box model created with a reduced selection of the original features can be easier to analyze with current techniques such as ELI5.[1] and SHAP [14] Thus, the development of methods to perform simultaneous HO and FS remains a very active field of research.

However, the right choice of hyperparameters and a subset of features is a difficult combinational problem for which efficient heuristic methods are usually required. Currently approaches are usually inspired from nature, mainly from biological systems such as animal herding, bacterial growth and so on. They usually consist of a population of simple individuals interacting both locally and globally with each other following some simple rules. For example, Mirjalili et al. [15] proposed a new meta-heuristic called Grey Wolf Optimizer (GWO) inspired by grey wolves and was successfully applied to several classical engineering design problems. Mirjalili et al. also proposed the Salp Swarm Algorithm [16], being inspired by the swarming behavior of salps when navigating and foraging in oceans. Other techniques related to animals include bat [17], glowworm [18] and bee colonies [19].

One of the most commonly used optimization techniques is the Particle Swarm Optimization (PSO), originally proposed by Kennedy and Eberhart [20]. There has been much research on this technique and numerous improvements have been proposed [6,21], for instance, in terms of topology, parameter selection, and other technical modifications, including quantum-behaved and chaotic PSO, extensions to multiobjective optimization, cooperation and multi-swarm techniques. Indeed, the study of PSO modifications is itself an active area of research due to the success of the algorithm. Some hybridizations of PSO with other meta-heuristic methods have been also proposed. For instance, for feature selection Chuang et al. [22] proposed an improved binary particle swarm optimization using the catfish effect, that is, new particles are introduced into the search space if the best solution does not improve in some number of consecutive iterations. This is done by replacing the 10% of original particles with the worst fitness values by new ones at extreme positions. Some PSO hybridizations include operations from genetic algorithms, for example, there are several modifications that include the crossover [23] operator. In [24] the crossover is taken between each particle's individual best position. After the crossover, the fitness of the individual best position is compared with two offspring produced after crossing. Then, the best one is chosen as the new individual best position. In [25], the standard crossover and mutation operations from GA are applied to PSO.

3. Performance analysis of GA-PARSIMONY and PSO-PARSIMONY

Similar to these heuristics, GA-PARSIMONY [1,26,27] was proposed to search for parsimonious solutions with genetic algorithms by performing HO and FS. It has been successfully applied in many fields, such as steel industrial processes [28], hotel room-booking forecasting [29], mechanical design [30], solar radiation forecasting [31] and hospital energy demand [32]. Moreover, previous comparisons with other existing AutoML methodologies (such as Auto-sklearn, H2O and MLJAR) demonstrated its effectiveness [27].

This methodology (see pseudo-code 1) makes use of GA optimization where each individual i of each generation/iteration t is defined by $\mathbf{X}_i^t = [H_i^t, F_i^t]$ chromosome formed by the combination of two vectors: the training hyperparameters of the algorithm (H_i^t) and the input attributes selected for that individual (F_i^t). The particularity of this methodology is that it performs the selection of the best individuals of each generation in two steps. First, it orders the individuals according

Algorithm 1 Pseudo-code of the GA-PARSIMONY algorithm

```
 1: Initialization of individuals of initial population $\mathbf{X^0}$ using a random
    and uniformly distributed Latin hypercube within the ranges of
    feasible values for each input parameter
 2: for $t = 1$ to $T$ do
 3:     Train each individual $\mathbf{X}_i^t$ and validate with $CV$
 4:     Fitness evaluation $J$ and complexity evaluation $M_c$ of each
    individual
 5:     Sort individuals using $J$
 6:     Promote individuals with best $M_c$ between those with similar $J$
 7:     if early stopping is satisfied then
 8:         return best individual of the last generation, $\hat{\hat{\mathbf{X}}}$
 9:     end if
10:     Select elitist population $P_e$ for reproduction
11:     Cross over $P_e$ to create a new generation $\mathbf{X}^{t+1}$
12:     Mutation of a % of $H$ (hyperparameters)
13:     Mutation of a % of $F$ (features)
14: end for
15: return best individual $\hat{\hat{\mathbf{X}}}$
```

to their fitness value (J) and, subsequently, it reorders them so that, among the individuals with similar J (defined with a `tol` parameter), those with lower complexity (M_c) are promoted to higher positions. In this way, the methodology mainly aims at finding models with good fitness value, but guiding them towards the improvement of parsimony.

The methodology can be applied to both classification and regression problems. The algorithm by which the individuals are trained (Line 3 in pseudo-code 1) is decided by the user, for instance, MLP and KNN could be used for regression and classification problems, respectively. It is advisable to choose one depending on the dataset size, i.e., with a balance between predictive power and computational demands, since many individuals need to be trained repeatedly. Furthermore, the fitness evaluation J can also be customized by the user, but by default the *root-mean-square error* (RMSE) is used for regression and the *log loss* for classification. Similarly, the computation of the complexity (M_c) of each individual can be defined by the user; the following formula is used by default:

$$M_c = 10^6 N_{FS} + internal\_comp$$

where N_{FS} is the number of selected input features for that individual, $internal\_comp$ is an internal measure of model complexity, which depends on the algorithm used for training. For instance, the sum of the squared coefficients of a linear regression, the number of support vectors in a SVM and the mean of the number of leaves in a random forest. Six model complexity metrics were compared in a previous work [27], being this configuration the one that showed the best experimental results with different datasets.

In this type of problems where each solution has a high computational cost, it is not possible to evaluate a large number of individuals in each iteration. This makes GAs not as efficient as other optimization techniques where hundreds or thousands of individuals are evaluated. Thus, GA-PARSIMONY usually produces individuals that are very similar to each other, so that the GA optimization quickly converges to a local minimum rather than approaching a global minimum. This low diversity makes GA optimization unstable and requires several iterations of the method to find an optimal solution.

As a continuation of this methodology and in order to improve the obtaining of accurate and low-complexity models, the authors in [5] used particle swarm optimization combined with a parsimony criterion to find parsimonious and, at the same time, accurate machine learning models. The PSO algorithm is based on the use of a population (called a swarm) of possible solutions (called particles). Algorithm 2 presents a pseudo-code version of it.

Algorithm 2 Pseudo-code of the PSO-PARSIMONY algorithm

```
1: Initialization of positions $\mathbf{X^0}$ using a random and uniformly distributed Latin hypercube within the ranges of feasible values for each input parameter
2: Initialization of velocities according to $\mathbf{V^0} = \frac{random_{LHS}(s,\ D)-\mathbf{X^0}}{2}$
3: for $t = 1$ to $T$ do
4:     Train each particle $\mathbf{X}_i^t$ and validate with $CV$
5:     Fitness evaluation $J$ and complexity evaluation $M_c$ of each particle
6:     Update $\hat{\mathbf{X}}_i$, $\hat{\mathbf{X}}_i^p$ and the $\hat{\hat{\mathbf{X}}}$
7:     if early stopping is satisfied then
8:         return $\hat{\hat{\mathbf{X}}}$
9:     end if
10:    Generation of new neighborhoods if $\hat{\hat{\mathbf{X}}}$ did not improve
11:    Update each $\hat{\mathbf{L}}_i$
12:    Update positions and velocities according the formulae
13:    Mutation of % of $F$ (features)
14:    Limitation of velocities and out-of-range positions
15: end for
16: return best individual $\hat{\hat{\mathbf{X}}}$
```

These particles are moved around in the search-space of the combinational problem according to simple formulae:

$$\mathbf{V}_i^{t+1} = \omega\mathbf{V}_i^t + \varphi_1\mathbf{r_{i,2}} \times (\hat{\mathbf{X}}_i^p - \mathbf{X}_i^t) + \varphi_2\mathbf{r_{i,2}} \times \hat{\mathbf{L}}_i - \mathbf{X}_i^t \quad (1)$$

$$\mathbf{X}_i^{t+1} = \mathbf{X}_i^t + \mathbf{V}_i^{t+1} \quad (2)$$

where $\mathbf{V}_i^t$ and $\mathbf{X}_i^t$ denote the velocity and position of the ith particle in iteration t, respectively. Similar to GA-PARSIMONY, the position of the ith particle in iteration t is again a vector $\mathbf{X}_i^t = (H_i^t, F_i^t)$ where H_i^t corresponds to the values of model's hyperparameters and F_i^t is in this case a probability vector with values between 0 and 1 for selecting the input features if the probability is greater than a value α (by default, $\alpha = 0.50$). If the iteration is clear from the context, we just denote it as $\mathbf{X}_i$ (similarly for the velocities, i.e., $\mathbf{V}_i$).

Such formulae just state that the movement of a particle is influenced by three components: its previous velocity, its own experience (its best position achieved so far with a parsimony criterion, denoted as $\hat{\mathbf{X}}_i^p$) and also by the experience of other particles (the best position within a neighborhood, $\hat{\mathbf{L}}_i$). This permits particles to explore the search space based on their current momentum, each individual particle thinking (cognitive component) and the collaborative effect (cooperation component). More concretely, ω is the inertia weight used to control the displacement of the current velocity. φ_1 and φ_2 are positive constant parameters that balance the global exploration and local exploitation. $\mathbf{r_{i,1}}$ and $\mathbf{r_{i,2}}$ are uniformly distributed random variables used to maintain the diversity of the swarm.

The main novelty in the PSO-PARSIMONY methodology is that it includes a strategy where the best position of each particle (thus, also the best position of each neighborhood) is computed considering not only the goodness-of-fit, but also the principle of parsimony.

To do so, for each particle of the swarm, both $\hat{\mathbf{X}}_i$ (position of the best fitness value achieved by the ith particle, without parsimony) and $\hat{\mathbf{X}}_i^p$ (similarly, but with parsimony) are computed. $\hat{\mathbf{X}}_i^p$ is updated if the fitness value of this new position $J(\mathbf{X}_i^t)$ is clearly lower than the current value, or if it is better and also has lower complexity. If $J(\mathbf{X}_i^t)$ is within a tolerance in regards to the $\hat{\mathbf{X}}_i$, the complexity criterion is applied and $\hat{\mathbf{X}}_i^p$ is updated if the complexity $M_c(\mathbf{X}_i^t)$ is lower than its current value. As expected, $\hat{\mathbf{X}}_i^p$ is not updated if $J(\mathbf{X}_i^t)$ is clearly higher than the current value. In short, for each particle, its $\hat{\mathbf{X}}_i^p$ is updated if the best parsimonious solution improves or stays close (within the indicated tolerance) to the best solution without parsimony ($\hat{\mathbf{X}}_i$). This avoids inaccurate parsimonious solutions, i.e. the most important criterion is always the minimization of the fitness value J and parsimonious solutions should remain very close to the best global solution obtained without applying the parsimony criterion. At the end of the process, the position of the individual with the best score is returned, $\hat{\hat{\mathbf{X}}}$ (see [5] for further details).

3.1. Performance of GA and PSO with high dimensional datasets

A comparison between both methods was performed [5] on 13 public data sets [33]. As training algorithm, multilayer perceptron model (`MLPRegressor` from `scikit-learn` package) was chosen, defined with a single hidden layer of neurons with *sigmoid* activation functions. As hyperparameters, the number of neurons ranged from 1 to 25, whereas the `alpha` value (the hyperparameter for regularization term) ranged within the interval $[10^{-6}, 10^3]$; see [5] for further details.

The results showed that PSO always improved accuracy over GA, but GA found solutions approximately 10% less complex on problems with a low number of features. However, datasets with a larger number of features, such as *ailerons*, *tecator* and *crime*, caused problems to PSO, which found better solutions than GA but with twice as many features and with higher computational cost. In summary, PSO obtained more accurate models but required more iterations and obtained much more complex models on high-dimensional datasets.

Experiments also provided insight on the evolution of the accuracy and parsimony in both GA and PSO. Fig. 1(a) shows an exponential-like decrease in the number of features selected by GA in the first iterations, whereas PSO performed a more linear decrease. On the other hand, Fig. 1(b) illustrates how good PSO was in terms of accuracy. This behavior can be explained by the low number of subjects that could be evaluated in each iteration, due to the high computational costs required to evaluate each individual. As it has been said previously, GA-PARSIMONY produces, in a few generations, populations of individuals very similar to each other, so this reduction in diversity causes GA optimization to quickly stall at local minima making the search for accurate models suboptimal. This is mainly due to three reasons. First, the high computational cost makes GA crossover mechanisms not as efficient as in other GA-based optimization problems where hundreds or thousands of individuals can be evaluated. Second, the parsimony criterion reduces the number of selected features very significantly; thus, individuals have fewer and fewer features and therefore the search space is smaller again, so that after a few iterations mutations fail to generate better individuals, the best ones being very similar to each other. Third, a well-known issue in classical genetic algorithms is the local optima problem: the population becomes genetically similar due to crossover and fitness selection as a local minim is approached [7]. On the other hand, the experiments show that PSO usually finds more accurate solutions because it performs a finer tuning, although it needs many more iterations and obtains more complex models.

With higher values of `tol`, this problem was accentuated as is demonstrated in Fig. 2a where diversity evolution with $tol = 10^{-3}$ is showed. For each generation/iteration, we defined diversity of T individuals as $F_{dist} = (\sum_i \sum_j d_{ij})/T^2$ where d_{ij} corresponds to the Euclidean distance of F_i and F_j, the binary vectors that define the characteristics that are selected for individuals i and j. It can be seen that with GA the diversity decreases rapidly in the first iterations, while with PSO the decrease is much slower. On the other hand, Fig. 2b shows the number of total feature state changes (change from selected to unselected, or vice versa) between two consecutive iterations. Thanks to GA's crossover and mutation mechanisms, GA performs more feature state changes than PSO in the first few generations, performing at the beginning a wider exploration of feature combinations than PSO, and finding solutions with smaller number of features on most of the datasets. However, this sharp reduction means that the obtained solutions cannot improve their accuracy in the next generations. In conclusion, PSO obtains better accurate solutions, but is much more inefficient when the number of features is high since the feature probability is slowly updated by the particle velocity. This is because a feature changes its state of being selected or not (or vice versa) only when the probability crosses the 0.50 threshold.

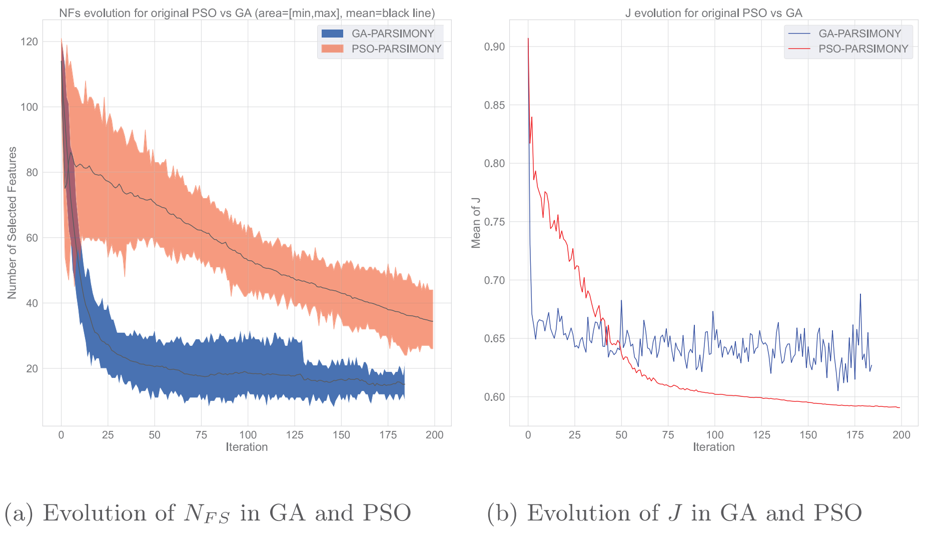


Fig. 1. PSO-PARSIMONY (red) vs. GA-PARSIMONY (blue) in terms of number of selected features and accuracy with the *crime* dataset with $tol = 10^{-6}$.

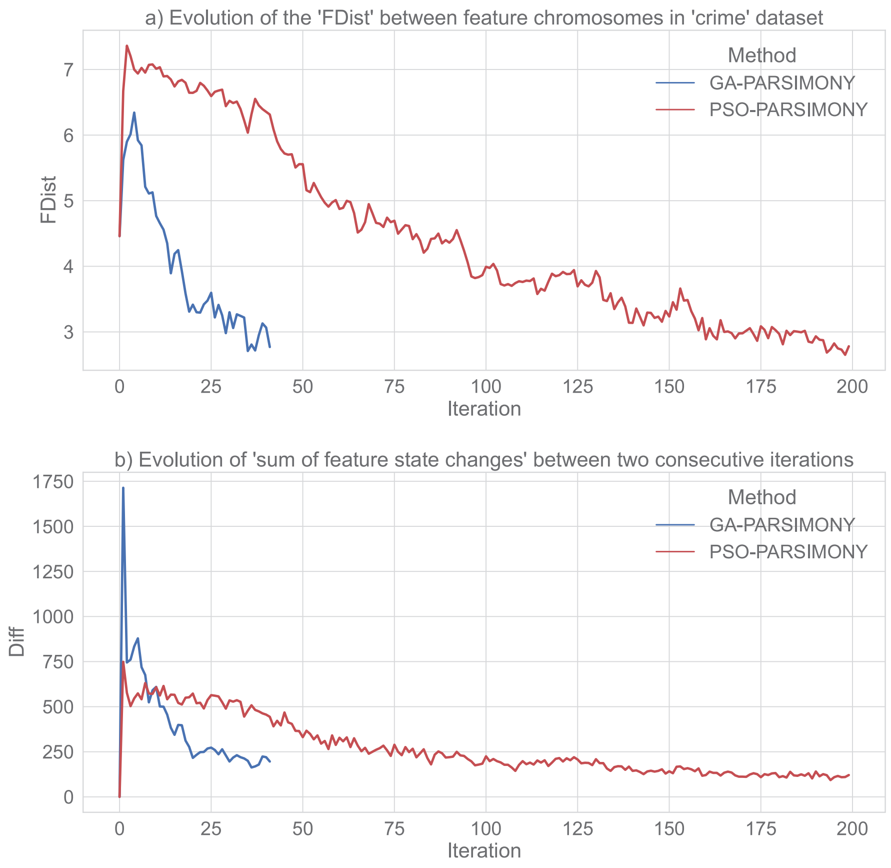


Fig. 2. Evolution of diversity (top) and number of features that change state (selected/unselected) between consecutive iterations (bottom) with GA-PARSIMONY and PSO-PARSIMONY and *crime* dataset.

4. HYB-PARSIMONY: a hybrid method of PSO combined with GA crossover and mutation operators

As seen in the previous section, PSO-PARSIMONY obtained models with better accuracy than GA-PARSIMONY, although it needed many more iterations to find a suitable solution and the models obtained were of higher complexity, especially in high-dimensional datasets.

To improve the parsimony search, mainly in the first iterations, reduce the number of iterations and, therefore, the computational effort, it is proposed a hybrid combination to incorporate GA operations (selection, crossover and mutation) in the PSO algorithm.[2] For this purpose, Algorithm 2 was modified in several parts as is presented in Algorithm 3.

[2] They have been implemented in Python and are available at https://github.com/jodivaso/hyb-parsimony.

Algorithm 3 Pseudo-code of the HYB-PARSIMONY algorithm

```
1: Initialization of positions $\mathbf{X}^0$ using a random and uniformly distributed Latin hypercube within the ranges of feasible values for each input parameter
2: Initialization of velocities according to $\mathbf{V}^0 = \frac{random_{LHS}(s,\ D)-\mathbf{X}^0}{2}$
3: for $t = 1$ to $T$ do
4:     Train each particle $\mathbf{X}_i^t$ and validate with $CV$
5:     Fitness evaluation $J$ and complexity evaluation $M_c$ of each particle
6:     Update $\hat{\mathbf{X}}_i$, $\hat{\mathbf{X}}_i^p$ and the $\hat{\hat{\mathbf{X}}}$
7:     if early stopping is satisfied then
8:         return $\hat{\hat{\mathbf{X}}}$
9:     end if
10:    Generation of new neighborhoods if $\hat{\hat{\mathbf{X}}}$ did not improve
11:    Update each $\hat{\mathbf{L}}_i$
12:    Select elitist population $P_e$ from for reproduction
13:    Obtain a pcrossover % of worst individuals $P_w$ to be substituted with crossover
14:    Crossover $P_e$ to substitute $P_w$ with new individuals
15:    Update positions and velocities of $P_e$
16:    Mutation of % of $H$ (hyperparameters)
17:    Mutation of % of $F$ (features)
18:    Limitation of velocities and out-of-range positions
19: end for
20: return best individual $\hat{\hat{\mathbf{X}}}$
```

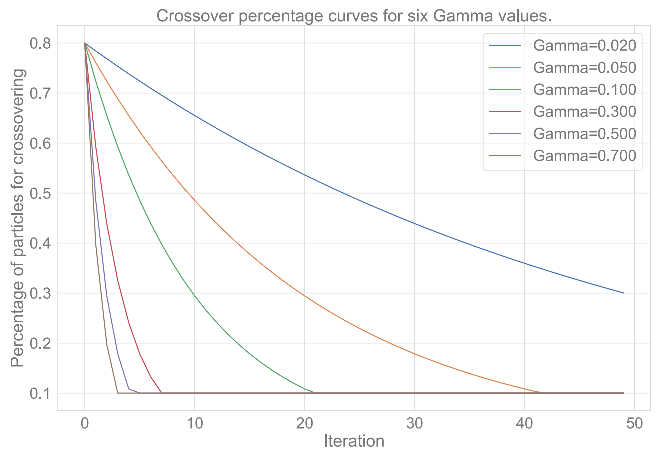


Fig. 3. Example of six curves created with different Γ values to establish the percentage of individuals to be replaced by crossover in each iteration.

4.1. Crossover strategy

A crossover phase is added just after calculating the local bests ($\hat{\mathbf{L}}$) of the neighborhoods. The purpose of the crossover is to distribute good parts of the genome among individuals. To perform this crossover, a selection phase is also added at that point, which is based on a nonlinear-rank selection following Michalewicz [34]. In this way, the selection of other individuals in addition to the best ones maintains the diversity of the population and prevents premature convergence. Furthermore, the best individuals are more likely to be selected for crossover. Thus, they are selected for breeding more times to foster good offspring. The crossover function was implemented by using heuristic blending [35] for hyperparameters and random swapping for features. It was also adapted to work properly with PSO: the positions are crossed with each other, as well as the velocities according to the crossover performed at the positions.

In addition, the way of replacing the particles differs from the typical GA crossover. In this case, the new particles created from the crossover replace the worst particles (those with the worst fitness value) that appeared in the population. For this purpose, a parameter `pcrossover` is incorporated, which fixes the percentage of worst individuals to be substituted from crossover. This parameter can be either a constant (such a percentage of particles is substituted in all iterations) or an vector which indicates a different percentage for each iteration. In this way, one can vary and encourage the crossover process in the first iterations by setting high values of `pcrossover` (to obtain a behavior similar to GA) and in further iterations decrease the percentage or even make it equal to 0 to obtain a pure PSO algorithm. Once the crossover step is done, PSO algorithm requires updating the positions and velocities according the formulae. In this case, this step is only applied to the particles that have not been substituted by the crossover.

For the new hybrid method, the following equation is proposed to calculate `pcrossover`, the percentage of particles to be substituted by crossover, in each iteration t:

$$pcrossover = max(0.80 \cdot e^{(-\Gamma \cdot t)}, 0.10) \tag{3}$$

Let us note that the formula presented above depends on a parameter Γ. Intuitively, this parameter regulates the number of particles to be substituted by crossover during the whole process: small values of Γ will cause many particles to be substituted by crossover over a large number of iterations, while a large value will do the opposite. Fig. 3 shows six curves obtained with different Γ values. In the first iterations the hybrid method performs the substitution by crossing a high percentage of particles. As the optimization process progresses, the number of substituted particles is reduced exponentially until it ends up fixed at a percentage of 10%. Thus, the hybrid method begins by facilitating the search for parsimonious models using GA-based mechanisms and ends up using more PSO optimization.

4.2. Mutation strategy

PSO-PARSIMONY already included a mutation operator, for which the mutation rate was set to $1/N$ by default, where N is the number of features of the problem. In contrast, GA-PARSIMONY was fixed to 10%, having a much more aggressive strategy in high dimensional datasets. This explains one reason why the PSO algorithm performed worse in terms of parsimony with high-dimensional datasets.

The new hybrid method also includes a similar aggressive uniform random mutation where three parameters are involved: `pmutation` represents the percentage of F (hyperparameters) to be muted, `feat_mut_thres` represents the probability of select a feature (include it in the selected features of the individual) when muting it, and `not_muted` is the number of top best individuals that will not be muted. Note that `not_muted` prevents losing the best individuals in the mutation step. The default values for `pmutation`, `feat_mut_thres`, and `not_muted` are set to 0.1, 0.1 and 3, respectively. These values are based on the default ones in GA-PARSIMONY, which have experimentally demonstrated good performance in previous works.

5. Experiments

5.1. Performance with public datasets

In order to test the capacity of the proposed methodologies to find accurate and parsimonious models, public datasets [33] with a high number of features were again selected. In particular, experiments compared HYB-PARSIMONY (HYB) with the original PSO method (Old-PSO), the PSO-PARSIMONY method but with the new mutation proposed in the hybrid method (New-PSO), and GA-PARSIMONY (GA).

To guarantee a fair comparison, all experiments were similar to previous works with a population size of $P = 40$, $tol = 0.001$, a maximum number of iterations of $T = 200$, and an early stopping of

Table 1
Hybrid with different Γ vs previous methods for *crime* dataset.

$Method$	Γ	J_{best}	N_{FSbest}	$\overline{J}$	$\overline{N_{FS}}$	$\overline{iters}$	$\overline{time}$
GA	0.00	.58070	19	.58503	20.4	146.8	86.2
OLD PSO	0.00	.58333	29	.58773	33.0	200.0	121.0
NEW PSO	0.00	.58155	26	.58419	25.2	362.8	211.9
HYB	0.02	.57981	22	.58650	24.4	229.6	132.0
HYB	0.04	.58142	**17**	.58571	22.6	200.8	118.8
HYB	0.06	.58397	23	.58747	26.8	206.6	119.1
HYB	0.10	**.57844**	24	.58402	26.6	200.4	115.6
HYB	0.12	.58106	24	.58576	25.8	184.4	97.6
HYB	0.14	.58143	24	.58856	28.8	190.0	109.9
HYB	0.16	.58151	19	.58304	23.4	228.8	121.5
HYB	0.18	.58603	26	.58773	27.4	148.6	85.9
HYB	0.20	.58251	23	.58517	25.2	185.6	111.0
HYB	0.22	.57964	23	.58434	23.4	241.4	138.9
HYB	0.24	.58229	25	.58554	26.4	167.4	98.0
HYB	0.26	.58368	23	.58656	25.4	176.0	101.2
HYB	0.28	.58054	24	.58340	23.6	235.6	135.0
HYB	0.30	.58343	29	.58540	25.6	**143.8**	**82.7**
HYB	0.32	.58050	22	**.58193**	**20.2**	242.2	139.1
HYB	0.34	.58247	**17**	.58421	23.6	233.2	123.5
HYB	0.36	.58001	23	.58378	21.6	221.6	127.4
HYB	0.38	.58119	20	.58544	23.2	197.2	117.9
HYB	0.40	.58117	27	.58493	24.6	209.8	120.6
HYB	0.45	.58304	22	.58494	24.8	176.8	101.9
HYB	0.50	.57938	24	.58319	23.6	213.4	123.0
HYB	0.55	.58314	24	.58555	25.4	193.8	111.5
HYB	0.60	.58158	22	.58378	24.2	218.2	115.8
HYB	0.70	.58080	24	.58582	26.6	195.8	116.5
HYB	0.80	.58065	19	.58232	23.2	215.0	123.7
HYB	0.90	.58101	25	.58437	24.8	187.4	108.5

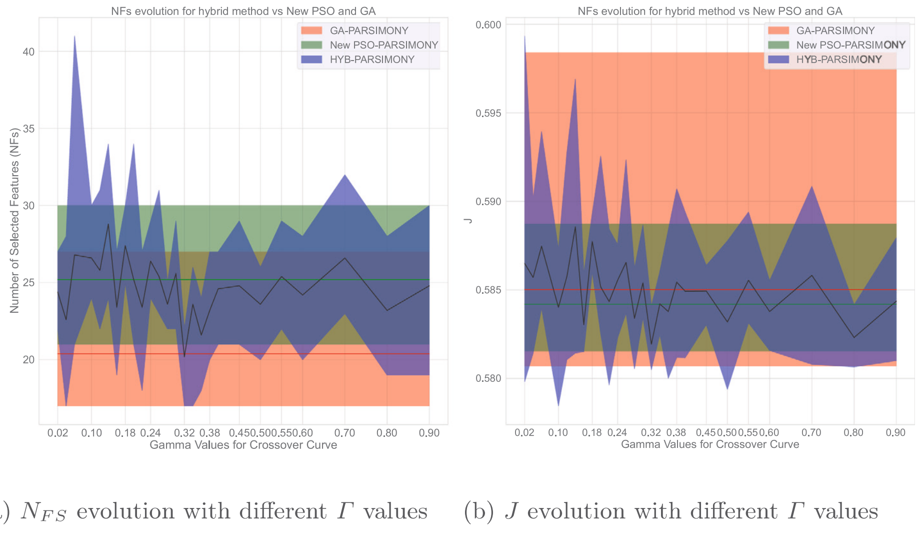


Fig. 4. Comparison between HYB-PARSIMONY (blue) vs. PSO-PARSIMONY (green) and GA-PARSIMONY (red) in terms of the number of selected features and accuracy with the *crime* dataset.

35. MLP was chosen again as the training algorithm, with the same hyperparameters to be optimized as the ones presented in Section 3.1. Experiments were implemented in 9 separately 24-core servers from the Beronia Cluster at the University of La Rioja. Each server was composed of two Intel Xeon E5-2670 (2.30 GHz) with 128 GB of RAM memory.

Table 1 presents the results with the *crime* dataset with 128 features. It shows results for the GA, the Old-PSO, the New-PSO and 26 Γ values of the hybrid method. The second and third columns indicate respectively the validation error (J) and the number of features (N_{FS}) of the best model obtained. The last four columns correspond to the mean of J, N_{FS}, *time* and the number of iterations (*iters*) of five runs for each algorithm. The hybrid method with $\Gamma = 0.10$ obtained the best model reducing J to 0.57844 versus the previous best model achieved with GA ($J = 0.58070$). However, the improvement in J involved the selection of 24 features (5 more) versus 19 in GA. On the other hand, the hybrid method with $\Gamma = 0.04$ obtained the most parsimonious model with only 17 features and an error of $J = 0.58142$, slightly higher than the J of GA. With respect to the mean values obtained from the five runs of each algorithm, it is observed that the hybrid method with $\Gamma = 0.32$ obtained the best mean values of J and N_{FS}.

Fig. 4 shows in blue the range (minimum and maximum) and in a solid black line the mean value of N_{FS} (left) and J (right) for five runs of the algorithm with different values of Γ for the dataset *crime*. It also includes the range and mean value for GA (red) and for New-PSO (green). Regarding N_{FS}, the hybrid method with Γ between 0.32 and 0.36 was more stable, since it obtained lower ranges than the one obtained with GA and with minimum values similar to the latter. On the other hand, it clearly outperformed the New-PSO method. Fig. 5 shows that the hybrid method reduced the number of features more drastically and converged earlier than the New-PSO method, similar

Table 2
NEW PSO-PARSIMONY vs HYB-PARSIMONY with a population size of $P = 40$ and $tol = 0.001$ (results are the average of the 5 runs).

$Dataset$	$\#rows$	$\#feats$	Γ	$\overline{PSO_J}$	$\overline{HYB_J}$	$\overline{PSO_{N_{FS}}}$	$\overline{HYB_{N_{FS}}}$	$\overline{PSO_{time}}$	$\overline{HYB_{time}}$
slice	5000	379	0.34	.0238	**.0231**	146.8	**132.2**	819.4	**609.0**
blog	4999	277	0.70	.4087	**.3983**	127.6	**113.8**	1117.5	**1051.6**
crime	2215	128	0.32	.5842	**.5819**	25.2	**20.2**	211.9	**139.1**
tecator	240	125	0.50	.0331	.0331	55.0	**48.6**	11.9	**8.7**
ailerons	5000	41	0.70	.3947	**.3934**	10.6	**10.2**	473.4	**466.1**
bank	8192	33	0.50	.6514	**.6511**	21.4	21.4	2146.4	**1536.6**
puma	8192	33	0.50	.1817	.1817	4.0	4.0	1063.8	**933.2**

Table 3
Best individual obtained with PSO-PARSIMONY vs HYB-PARSIMONY using a population size of $P = 40$ and $tol = 0.001$.

$Dataset$	Γ	PSO_J	HYB_J	PSO_{Jtst}	HYB_{Jtst}	$PSO_{N_{FS}}$	$HYB_{N_{FS}}$	PSO_{time}	HYB_{time}
slice	0.70	.0228	**.0218**	**.0012**	.0017	124	**112**	1050.1	**627.5**
blog	0.38	.3948	**.3879**	.2523	**.2023**	**115**	129	1304.0	**1277.4**
crime	0.10	.5815	**.5784**	.5021	**.4780**	26	**24**	263.5	**138.5**
tecator	0.38	.0328	**.0327**	.0207	**.0206**	**48**	51	16.3	**10.7**
ailerons	0.15	.3935	**.3922**	**.3675**	.3698	13	**10**	**484.2**	494.3
bank	0.70	.6510	**.6507**	**.5839**	.5865	22	**21**	2428.5	**1675.0**
puma	0.38	.1817	.1817	.1776	.1776	4	4	1191.8	**712.9**

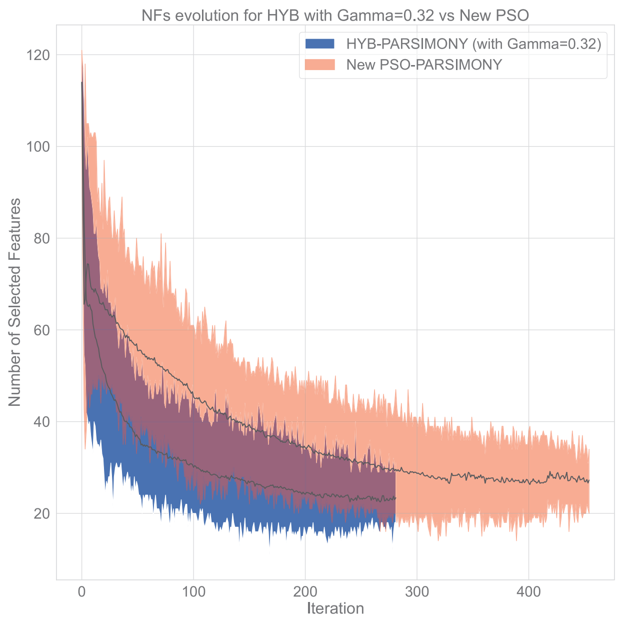


Fig. 5. Comparison of N_{FS} evolution between the Hybrid method (blue) (with Γ = 0.32) and the new PSO-PARSIMONY (red).

than GA method. With respect to J, Fig. 4b clearly shows that the new-PSO method was more robust than GA as it had a much smaller range of J in the five runs of the algorithm. However, the hybrid method obtained with $\Gamma = 0.32$ better J values than PSO with a significantly lower range.

Finally, Fig. 6 presents the mean values of N_{FS} and J for the four methods and with the *crime* dataset. In this case, the hybrid model with $\Gamma = 0.32$ (blue) improved the reduction of N_{FS} and J in a balanced way, reducing the convergence time and obtaining accurate but low-complexity solutions.

Similar results can be observed with other high-dimensional datasets. Tables 2 and 3 show respectively the average results and the best model obtained with the Hybrid Method and the New-PSO. In almost all datasets, the hybrid method obtained more accurate models with less complexity, although it was necessary to find a suitable Γ value.

In the previous tables, HYB-PARSIMONY has been compared only with the methods that preceded it: PSO-PARSIMONY and GA-PARSIMONY. As it was already said, previous publications have demonstrated the effectiveness of these methods against other AutoML tools. Now, it has been decided to perform new comparisons with baseline approaches.

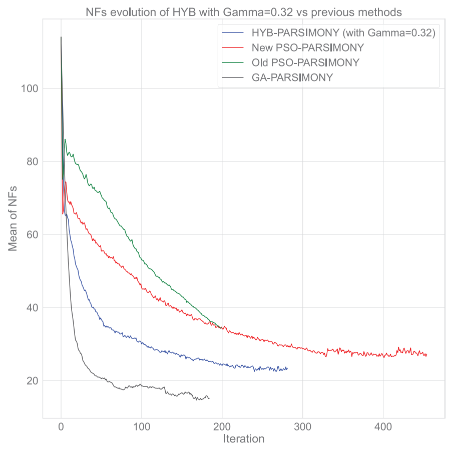
To allow a higher number of experiments, in this case the new experiments used sklearn's `KernelRidge` regression algorithm with `rbf` kernel. Although `KernelRidge` tends to have lower predictive power than MLP, it is much more efficient in terms of training time as well as being an explainable model. As hyperparameters, `alpha` ranged within the interval $[10^{-5}, 10^{5}]$ and `gamma` within the interval $[10^{-7}, 10^{0}]$.

In order to determine the generalization capability of the techniques, it was decided to choose a maximum of 2000 instances for training and validation; the remaining instances were used to determine the generalization error ($Jtst$). If the dataset had a size smaller than 2000 rows, it was divided in half. The procedure was performed 10 times for each dataset by randomly choosing each time different partitions for training/validation and testing.

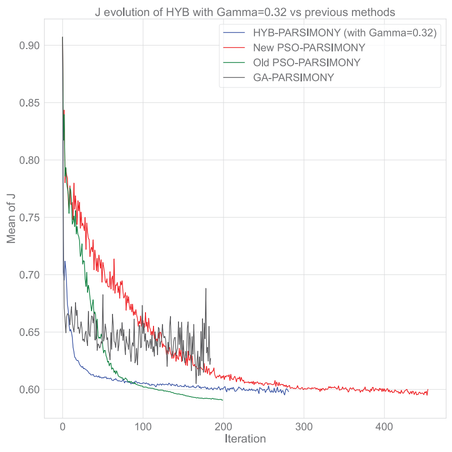
These three methods were compared:

- HO by Bayesian optimization (BO) with all features (BAY) and 200 iterations. The results of other baseline HO methods, such as grid search and random search, were worse or similar to those obtained with BAY, so they are not shown in the table. Among those baseline methods, BO was selected because it is one of the most used nowadays to perform HO.
- HYB-PARSIMONY (HYB) with $\Gamma = 0.50$, $tol = 0.001$, $T = 200$, $P = 15$. In this case, the number of individuals was reduced to only 15 to observe the ability of the methodology to find solutions with a small population.
- Using Python's `sklearn-genetics` package that allows the realization of HO and FS with genetic algorithms (SKG). However, this library does not allow to do simultaneous HO and FS so, first, HO with GA was performed with all the features, then, FS with GA was performed and, finally, HO with GA was performed again with the features selected in the previous step. This procedure is the most common approach when HO and FS have to be accomplished separately. In all three steps of the GA the default hyperparameters were selected, except $T = 200$ and $P = 15$.

In this sense, it is important to recall that most of the existing AutoML methods make use of highly complex ensemble models where FS is not performed, since each variable's importance is implicitly decided within the neural network or tree-based model. However, the goal of HYB-PARSIMONY, PSO-PARSIMONY and GA-PARSIMONY, is to obtain accurate simple models, but with a reduced number of input



(a) $\overline{N_{FS}}$ evolution



(b) $\overline{J}$ evolution

Fig. 6. Comparison with *crime* dataset of $\overline{N_{FS}}$ and $\overline{J}$ between the four methods: the HYB-PARSIMONY method (blue) (with $\Gamma = 0.32$), the new PSO-PARSIMONY (red), the old PSO-PARSIMONY (green) and GA-PARSIMONY (black).

Table 4
Comparison between three methods: HO with Bayesian optimization using all features (BAY), HYB-PARSIMONY with $\Gamma = 0.50$ (HYB) and `sklearn-genetics` (SKG) using three steps (HO with all features, FS and HO with the selected features); with `KernelRidge` algorithm. The results correspond to the mean of 10 runs of $Jtst$ and N_{FS}, and $p_{N_{Jtst}}$ and $p_{N_{FS}}$ to a p-value of a Wilcoxon–Mann–Whitney test between HYB and SKG.

Dataset	$\overline{BAY_{Jtst}}$	$\overline{HYB_{Jtst}}$	$\overline{SKG_{Jtst}}$	p_{Jtst}	$\overline{BAY_{N_{FS}}}$	$\overline{HYB_{N_{FS}}}$	$\overline{SKG_{N_{FS}}}$	$p_{N_{FS}}$
slice	.1411	**.1386**	.1420	.0539	378	**148.9**	202.4	**.0002**
blog	**.8231**	.8978	.8419	**.0002**	276	**85.1**	143.6	**.0002**
crime	.6382	.6428	**.6337**	**.0257**	127	**31.8**	67.7	**.0002**
tecator	.0558	**.0544**	.0578	.1859	124	**27.3**	63.0	**.0002**
ailerons	.3980	**.3978**	.3985	.3447	40	**9.1**	22.9	**.0002**
bank	**.6750**	.6784	.6761	.1212	32	**16.9**	19.8	**.0050**
puma	.8805	**.2079**	.2587	**.0004**	26	**3.6**	11.0	**.0001**
pol	.3192	**.2434**	.2767	**.0002**	21	**6.8**	15.7	**.0001**
cpu	**.1759**	.1948	.2075	.0757	18	**8.7**	11.7	**.0001**
elevators	.3456	**.3353**	.3419	**.0002**	17	**9.3**	14.2	**.0001**
meta	.9383	.9513	**.9373**	**.0002**	14	**1.9**	4.7	**.0003**
bodyfat	.2151	**.2089**	.2138	**.0002**	13	**1.2**	2.3	**.0005**
housing	.3700	**.3344**	.3350	.4727	10	**9.8**	10.0	.3681
concrete	**.3454**	.3687	.3698	.1212	6	5.1	**5.0**	.3681
pm10	**.8064**	.8371	.8377	**.0173**	7.0	5.0	5.0	1.000
no2	**.6973**	.7333	.7357	.2730	7.0	**4.9**	5.0	.3681
strike	.9606	**.9491**	.9645	**.0002**	6.0	**3.2**	4.0	**.0336**

features that facilitates bias detection, improves explainability, lowers maintenance and energy costs, enhances robustness, and so on.

Table 4 shows the mean value of $Jtst$ and N_{FS} for the 10 runs and for the three methods. The results of the 17 datasets are ordered from largest to smallest according to their dimensionality. $p_{N_{Jtst}}$ and $p_{N_{FS}}$ correspond to the p-value of a Wilcoxon–Mann–Whitney test between HYB and SKG regarding the accuracy and number of features, respectively.

The last four columns clearly show that HYB substantially reduced the number of features compared to the total dataset (used by BAY). Also, the improvement in N_{FS} compared to SKG was very relevant and statistically significant in all datasets with more than 10 dimensions.

Regarding $Jtst$, HYB obtained a better $Jtst$ in 9 datasets while SKG did it only in 2. However, BAY obtained the best model in `blog`, `bank` and `cpu` (besides the datasets with few features such as `concrete`, `pm10` and `no2` where the use of the HYB method does not make much sense). Probably, tuning the hyperparameters tol and Γ of the hybrid method could help to obtain more competitive models in these datasets.

6. Conclusions

This paper presents a new hybrid methodology that improves our previous PSO-PARSIMONY methodology in the simultaneous search for the best model hyperparameters and input features, with a balance between accuracy and complexity. Specifically, the hybrid method combines GA mechanisms such as selection, crossover and mutation within the PSO-based optimization algorithm.

The main novelty of the hybrid model is that the optimization is based on PSO but includes common genetic operations of selection, crossover and mutation to replace the worst particles. The percentage of variables to be replaced at each iteration is selected by a decreasing exponential function that is adjusted by Γ. Thus, in the first iterations parsimony is promoted by GA mechanisms, i.e., replacing by crossover a high percentage of particles at the beginning. Subsequently, optimization with PSO becomes more relevant for the improvement of model accuracy. This differs from other hybrid methods in which the crossover is applied between the best individual position of each particle or other approaches in which the worst particles are also replaced by new particles, but at extreme positions. Experiments show that, in general, and with a suitable Γ, the HYB-PARSIMONY methodology allows to obtain better, more parsimonious and more robust models compared to PSO-PARSIMONY. It also reduces the number of iterations and, consequently, the computational effort.

Although it is a promising method, further research is required to provide an explicit formula that fixes the Γ value for each dataset, for instance, depending on the number of instances and features or by means of adaptive strategies.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Jose Divason reports financial support was provided by European Regional Development Fund. F. J. Martinez-de-Pison reports financial support was provided by Banco Santander Central Hispano SA.

Data availability

The data that has been used is confidential.

Acknowledgments

The work is supported by grant PID2020-116641GB-I00 and the European Regional Development Fund under Project PID2021-123219OB-I00 funded by MCIN/AEI/10.13039/501100011033 FEDER, UE. We are also greatly indebted to Banco Santander for the REGI2020/41 and REGI2022/60 fellowships. This study used the Beronia cluster (Universidad de La Rioja), which is supported by FEDER-MINECO grant number UNLR-094E-2C-225.

References

[1] R. Urraca, E. Sodupe-Ortega, J. Antonanzas, F. Antonanzas-Torres, F.J. Martinez-de Pison, Evaluation of a novel GA-based methodology for model structure selection: The GA-PARSIMONY, Neurocomputing 271 (2018) 9–17, http://dx.doi.org/10.1016/j.neucom.2016.08.154.

[2] H. Li, D. Shu, Y. Zhang, G.Y. Yi, Simultaneous variable selection and estimation for multivariate multilevel longitudinal data with both continuous and binary responses, Comput. Statist. Data Anal. 118 (2018) 126–137, http://dx.doi.org/10.1016/j.csda.2017.09.004.

[3] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, A. Galstyan, A survey on bias and fairness in machine learning, ACM Comput. Surv. 54 (6) (2021) http://dx.doi.org/10.1145/3457607.

[4] H.B. Barua, K.C. Mondal, S. Khatua, Green computing for big data and machine learning, in: 5th Joint International Conference on Data Science & Management of Data, 9th ACM IKDD CODS and 27th COMAD, in: CODS-COMAD 2022, Association for Computing Machinery, New York, NY, USA, 2022, pp. 348–351, http://dx.doi.org/10.1145/3493700.3493772.

[5] J. Divasón, J.F. Ceniceros, A. Sanz-Garcia, A. Pernia-Espinoza, F.J.M. de Pison, PSO-PARSIMONY: A method for finding parsimonious and accurate machine learning models with particle swarm optimization. Application for predicting force–displacement curves in T-stub steel connections, Neurocomputing 548 (2023) 126414, http://dx.doi.org/10.1016/j.neucom.2023.126414.

[6] Y. Zhang, S. Wang, G. Ji, A comprehensive survey on particle swarm optimization algorithm and its applications, Math. Probl. Eng. (2015) 1–38, http://dx.doi.org/10.1155/2015/931256.

[7] S. Katoch, S.S. Chauhan, V. Kumar, A review on genetic algorithm: Past, present, and future, Multimedia Tools Appl. 80 (2021) 8091–8126.

[8] N. Erickson, J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li, A. Smola, AutoGluon-tabular: Robust and accurate AutoML for structured data, 2020, arXiv preprint arXiv:2003.06505.

[9] A. Płońska, P. Płoński, MLJAR: State-of-the-art Automated Machine Learning Framework for Tabular Data. Version 0.10.3, MLJAR Sp. z o.o., Łapy, Poland, 2021, URL https://github.com/mljar/mljar-supervised.

[10] E. LeDell, S. Poirier, H2O autoML: Scalable automatic machine learning, in: 7th ICML Workshop on Automated Machine Learning (AutoML), 2020, URL https://www.automl.org/wp-content/uploads/2020/07/AutoML_2020_paper_61.pdf.

[11] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, T.-Y. Liu, LightGBM: A highly efficient gradient boosting decision tree, in: Advances in Neural Information Processing Systems, Vol. 30, 2017, pp. 3146–3154.

[12] T. Chen, C. Guestrin, XGBoost: A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, ACM, New York, NY, USA, 2016, pp. 785–794, http://dx.doi.org/10.1145/2939672.2939785.

[13] A.V. Dorogush, V. Ershov, A. Gulin, CatBoost: Gradient boosting with categorical features support, 2018.

[14] S.M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS '17, Curran Associates Inc, Red Hook, NY, USA, 2017, pp. 4768–4777.

[15] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, Adv. Eng. Softw. 69 (2014) 46–61, http://dx.doi.org/10.1016/j.advengsoft.2013.12.007.

[16] S. Mirjalili, A.H. Gandomi, S.Z. Mirjalili, S. Saremi, H. Faris, S.M. Mirjalili, Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems, Adv. Eng. Softw. 114 (2017) 163–191, http://dx.doi.org/10.1016/j.advengsoft.2017.07.002.

[17] X.-S. Yang, A new metaheuristic bat-inspired algorithm, in: Nature Inspired Cooperative Strategies for Optimization, NICSO 2010, Springer, 2010, pp. 65–74.

[18] M. Marinaki, Y. Marinakis, A glowworm swarm optimization algorithm for the vehicle routing problem with stochastic demands, Expert Syst. Appl. 46 (2016) 145–163, http://dx.doi.org/10.1016/j.eswa.2015.10.012.

[19] D. Karaboga, B. Basturk, Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems, in: P. Melin, O. Castillo, L.T. Aguilar, J. Kacprzyk, W. Pedrycz (Eds.), Foundations of Fuzzy Logic and Soft Computing, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 789–798.

[20] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95 - International Conference on Neural Networks, Vol. 4, 1995, pp. 1942–1948, http://dx.doi.org/10.1109/ICNN.1995.488968.

[21] T.M. Shami, A.A. El-Saleh, M. Alswaitti, Q. Al-Tashi, M.A. Summakieh, S. Mirjalili, Particle swarm optimization: A comprehensive survey, IEEE Access 10 (2022) 10031–10061, http://dx.doi.org/10.1109/ACCESS.2022.3142859.

[22] L.-Y. Chuang, S.-W. Tsai, C.-H. Yang, Improved binary particle swarm optimization using catfish effect for feature selection, Expert Syst. Appl. 38 (10) (2011) 12699–12707, http://dx.doi.org/10.1016/j.eswa.2011.04.057.

[23] A.P. Engelbrecht, Particle swarm optimization with crossover: A review and empirical analysis, Artif. Intell. Rev. 45 (2) (2016) 131–165, http://dx.doi.org/10.1007/s10462-015-9445-7.

[24] Z.-F. Hao, Z.-G. Wang, H. Huang, A particle swarm optimization algorithm with crossover operator, in: 2007 International Conference on Machine Learning and Cybernetics. Vol. 2, 2007, pp. 1036–1040.

[25] M. Nazir, A. Majid-Mirza, S. Ali-Khan, PSO-GA based optimized feature selection using facial and clothing information for gender classification, J. Appl. Res. Technol. 12 (1) (2014) 145–152, http://dx.doi.org/10.1016/S1665-6423(14)71614-1.

[26] F.J. Martinez-de Pison, R. Gonzalez-Sendino, A. Aldama, J. Ferreiro-Cabello, E. Fraile-Garcia, Hybrid methodology based on Bayesian optimization and GA-PARSIMONY to search for parsimony models by combining hyperparameter optimization and feature selection, Neurocomputing 354 (2019) 20–26, http://dx.doi.org/10.1016/j.neucom.2018.05.136, Recent Advancements in Hybrid Artificial Intelligence Systems.

[27] F.J. Martinez-de Pison, J. Ferreiro, E. Fraile, A. Pernia-Espinoza, A comparative study of six model complexity metrics to search for parsimonious models with GAparsimony R Package, Neurocomputing 452 (2021) 317–332, http://dx.doi.org/10.1016/j.neucom.2020.02.135.

[28] A. Sanz-Garcia, J. Fernandez-Ceniceros, F. Antonanzas-Torres, A. Pernia-Espinoza, F.J. Martinez-de Pison, GA-PARSIMONY: A GA-SVR approach with feature selection and parameter optimization to obtain parsimonious solutions for predicting temperature settings in a continuous annealing furnace, Appl. Soft Comput. 35 (2015) 13–28, http://dx.doi.org/10.1016/j.asoc.2015.06.012.

[29] R. Urraca, A. Sanz-Garcia, J. Fernandez-Ceniceros, A. Pernia-Espinoza, F.J. Martinez-De-Pison, Improving hotel room demand forecasting with a hybrid GA-SVR methodology based on skewed data transformation, feature selection and parsimony tuning, Logic J. IGPL 25 (6) (2017) 877–889, http://dx.doi.org/10.1093/jigpal/jzx029.

[30] J. Fernandez-Ceniceros, A. Sanz-Garcia, F. Antoñanzas-Torres, F.J. Martinez-de Pison, A numerical-informational approach for characterising the ductile behaviour of the T-stub component. Part 2: Parsimonious soft-computing-based metamodel, Eng. Struct. 82 (2015) 249–260, http://dx.doi.org/10.1016/j.engstruct.2014.06.047.

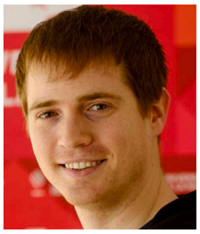
[31] F. Antonanzas-Torres, R. Urraca, J. Antonanzas, J. Fernandez-Ceniceros, F.J. Martinez-de Pison, Generation of daily global solar irradiation with support vector machines for regression, Energy Convers. Manage. 96 (2015) 277–286, http://dx.doi.org/10.1016/j.enconman.2015.02.086.

[32] E. Dulce-Chamorro, F.J.M. de Pison, An advanced methodology to enhance energy efficiency in a hospital cooling-water system, J. Build. Eng. 43 (102839) (2021) http://dx.doi.org/10.1016/j.jobe.2021.102839.

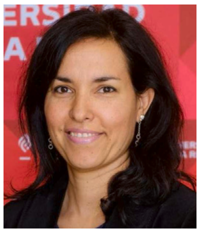
[33] D. Dua, C. Graff, UCI Machine Learning Repository, University of California, Irvine, School of Information and Computer Sciences, 2017, URL http://archive.ics.uci.edu/ml.

[34] Z. Michalewicz, Genetic Algorithms + Data Structures=Evolution Programs, Third Revised and Extended Edition, Springer, 1996.

[35] Z. Michalewicz, C.Z. Janikow, Handling constraints in genetic algorithms, in: Icga, 1991, pp. 151–157.



J. Divasón received the degrees in Computer Science and Mathematics from the University of La Rioja (Spain) in 2011. He worked for the ForMath FP7 European project in 2012 and finished his Ph.D. in computer science in 2016. His research interests include interactive theorem proving, computer algebra and machine learning.



Alpha Pernia-Espinoza is Associate Professor of Universidad de La Rioja (Spain) since 2012. She is Industrial Engineer and Ph.D. in Industrial Engineer by the Universidad de La Rioja, and Electrical Engineer and Master in Industrial Automation and Instrumentation by the Universidad de Los Andes (Venezuela). Her research interests involve industrial processes modeling and optimization through numerical simulation and advanced statistical tools. Another research interest is the application of additive manufacturing technologies in tissue engineering.

F.J. Martinez-de-Pison is the head of the EDMANS group and Professor at the University of La Rioja. He has a Ph.D. in Machine Learning applied to Industrial Processes from the University of La Rioja. His research activities focus on the use of soft computing, data mining and machine learning methods to solve real problems in various fields such as industry, energy, agriculture and business.