Neurocomputing 548 (2023) 126414

Contents lists available at ScienceDirect

Neurocomputing

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

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

Jose Divasón [d], Julio Fernandez Ceniceros [b], Andres Sanz-Garcia [c], Alpha Pernia-Espinoza [a,b], Francisco Javier Martinez-de-Pison [a,b,*]

[a] SCoTIC, Scientific Computation and Technological Innovation Center (SCoTIC), University of La Rioja, 26004 Logroño, La Rioja, Spain
[b] EDMANS Group, Department of Mechanical Engineering, University of La Rioja, Logroño, Spain
[c] Department of Mechanical Engineering, University of Salamanca, Béjar (Salamanca), Spain
[d] PSYCOTRIP Group, Department of Mathematics and Computation, University of La Rioja, Logroño, Spain

ARTICLE INFO

Article history:
Received 10 May 2022
Revised 8 February 2023
Accepted 29 May 2023
Available online 2 June 2023

Keywords:
PSO-PARSIMONY
t-stub connections
Parsimonious modeling
Auto machine learning
GA-PARSIMONY

ABSTRACT

We present PSO-PARSIMONY, a new methodology to search for parsimonious and highly accurate models by means of particle swarm optimization. PSO-PARSIMONY uses automatic hyperparameter optimization and feature selection to search for accurate models with low complexity. To evaluate the new proposal, a comparative study with multilayer perceptron algorithm was performed with public datasets and by applying it to predict two important parameters of the force–displacement curve in T-stub steel connections: initial stiffness and maximum strength. Models optimized with PSO-PARSIMONY showed an excellent trade-off between goodness-of-fit and parsimony. The new proposal was compared with GA-PARSIMONY, our previously published methodology that uses genetic algorithms in the optimization process. The new method needed more iterations and obtained slightly more complex individuals, but it performed better in the search for accurate models.

1. Introduction

Nowadays, there is a growing demand for auto machine learning (AutoML) tools to automatize tedious tasks such as hyperparameter optimization (HO), model selection (MS), feature selection (FS) and feature generation (FG).

One of the current topics in this field corresponds to the development of methods that help in the search for robust machine learning (ML) models that are able to predict with good accuracy under multiple and diverse input conditions. One of the most common strategies is the selection, among the most accurate models, of those with lower complexity. A model that has good accuracy and low complexity, i.e. a model with fewer input features, will be much more robust to perturbations, faster and easier to understand, and much cheaper to maintain and update.

In this article, we describe a new methodology, named PSO-PARSIMONY, which uses an adapted particle swarm optimization (PSO) to search for parsimonious and accurate models by means of hyperparameter optimization (HO), feature selection (FS), and the promotion of the best solutions according to two criteria: low complexity and high accuracy. This paper also includes a comparison in performance with GA-parsimony, our previously published methodology based on GA [20,27,19] that has been successfully applied in a variety of contexts such as steel industrial processes [23], hotel room-booking forecasting [28], mechanical design [11], hospital energy demand [1], and solar radiation forecasting [3].

Finally, a detailed analysis of the application of both methods for predicting force–displacement curves in steel T-joints is presented. The methods were tested for optimizing multilayer perceptron (MLP) models to predict two parameters defining the T-stub curve: the initial stiffness and the maximum force.

2. Related works

In recent years, there has been a trend toward providing methods to make ML more accessible for people without expertise in machine learning. The overarching aim is to reduce the human effort necessary in tedious and time-consuming tasks.

* Corresponding author at: SCoTIC, Scientific Computation and Technological Innovation Center (SCoTIC), University of La Rioja, 26004 Logroño, La Rioja, Spain.
E-mail address: fjmartin@unirioja.es (F.J. Martinez-de-Pison).

https://doi.org/10.1016/j.neucom.2023.126414
0925-2312/

Companies like *DataRobot*, *Strong Analytics*, *Mighty AI*, *Akkio*, *CloudZero* and *Unity Technologies*, among others, are currently providing services to automate a multitude of tasks in machine learning and artificial intelligence. In addition, new AutoML suites have emerged such as *Google Cloud AutoML*[1], *Microsoft Azure ML*[2], *Alteryx Intelligence Suite*[3], and *H2O AutoML*[4]. Free software is also available, such as `Auto-WEKA`, `Auto-Sklearn 2.0`[5], `Hyperopt`[6], `TPOT`[7], and, for Deep Learning, `Auto-Keras`[8] and `Auto-PyTorch`[9]. Many companies and tools use hybrid methods based on high computational resources fused with advanced optimization techniques.

2.1. The search of parsimony

The development of models with small datasets, i.e. a few hundred or a few thousand rows, can result in overfitted models [29]. These overfitted models learn "too much" well from the training dataset, but are unable to generalize adequately. That is, their predictions fail when new input conditions appear. This is very common when training models with algorithms that have many degrees of freedom and make them too "flexible", such as neural networks with many layers and neurons, large regression trees, etc.

In ML, the problem of overfitting is solved by incorporating regularization mechanisms within the algorithms, so that learning not only takes into account how the model learns from the data, but also the complexity of the model. Examples of regularization are widely used, such as controlling the size of weights in ridge and Lasso regressions, the use of dropout and weight decay in neural networks, or the depth of decision trees or the number of leaves in them [26].

If the dataset is small, there may be a bias in the error estimate if no adequate validation is performed [4,7]. Typically, the most common strategy is to use a robust cross-validation such as repeated cross-validation with a high number of repetitions [22].

An additional strategy to reduce the risk of over-fitting is to select the model with the lowest complexity (most parsimonious) among those models with similar accuracy [27]. The complexity of a model can be defined in multiple ways, such as the number of input features, the internal complexity of the model (by some defined metric), or a combination of the above [19]. The less complex model will be more robust to noise and disturbances, will have more stable predictions, and will be easier to maintain and analyze [14]. For example, imagine that several models have been trained with multiple combinations of features and hyperparameters and those with the highest accuracy have been selected through appropriate validation. Among them, the model with the lowest number of input features should be chosen because the disturbances or noise that may appear in the attributes that have not been selected will not affect it, and the problem of collinearity that may exist between the non-selected variables and the selected ones will be eliminated [21].

Some studies have focused on the context of automatically seeking parsimonious and accurate models. For example, Ma and Xia [15] used a tribe competition with GA to optimize FS in pattern classification. Wei et al. [33] applied binary particle swarm optimization (BPSO) in HO and FS to obtain an accurate SVM with a reduced number of features. Similarly, Vieira et al. [30] optimized a wrapper support vector machine (SVM) with BPSO to predict the survival or death of patients with septic shock. Wan et al. [31] combined GA with an ant colony optimization algorithm named MBACO to select an optimal subset of features with minimum redundancy and maximum discrimination ability. Ahila et al. [2] improved accuracy and generalization performance of Extreme Learning Machines (ELM) with PSO for power system disturbances. Wang et al. [32] reported a comparative of chaotic moth-flame algorithm against other methods in an HO and FS optimization strategy for medical diagnoses. This method showed a significant improvement in classification performance and obtained a smaller feature subset.

Similar to these methods, we proposed GA-PARSIMONY [23], a method to search for parsimonious solutions with GA by optimizing HO, FS, and parsimonious model selection.

GA-PARSIMONY can be considered as a special multi-objective optimization method focused on the search for accurate and parsimonious ML models, which takes into account two fundamental factors: the need to perform an optimal search due to the excessive computational cost of each solution, and that the accuracy of the model must prevail over the model complexity (parsimony). Previous work on this type of problem has shown that simultaneous optimization of both objectives (accuracy and complexity) produces sub-optimal solutions from the point of view of accuracy [23]. Therefore, GA-PARSIMONY is not a classical multi-objective system that tries to optimize two objective functions simultaneously, but has, as a primary objective, the improvement of accuracy and, as a secondary objective, the reduction of the complexity of the model. To this end, genetic algorithms (GA) are used to search for models with good accuracy but using a selection criterion that also seeks to improve parsimony.

GA-PARSIMONY has demonstrated in numerous case studies that it is capable of achieving accurate low complexity models. In addition, it has also been compared to other existing AutoML methodologies, showing a remarkable performance [19]. However, due to the excessive computational cost, the number of individuals that can be evaluated with GA in each generation is very small. This makes GA not as efficient as in other optimization problems.

The aim of this work has been to develop a new method adapted to this type of problems with the idea of improving the accuracy of the models, but considering parsimony as GA-PARSIMONY.

3. PSO-PARSIMONY methodology

Algorithm 1: Pseudo-code of the PSO-PARSIMONY algorithm

1: Initialization of positions using a random and uniformly distributed Latin hypercube within the ranges of feasible values for each input parameter
2: Initialization of velocities
3: **for** $t = 1$ to T**do**
4: Train each particle $\mathbf{X}_i^t$ and validate with cross validation
5: Fitness evaluation and complexity evaluation 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 $\hat{\mathbf{L}}_i^p$
12: Update positions and velocities according the formulas
13: Mutation of % of features
14: Limitation of velocities and out-of-range positions
15: **end for**
16: **return** best individual $\hat{\hat{\mathbf{X}}}$

[1] Ref: https://cloud.google.com/automl
[2] Ref: https://azure.microsoft.com/es-es/products/machine-learning
[3] Ref: https://www.alteryx.com/products/intelligence-suite
[4] Ref: https://h2o.ai/platform/h2o-automl
[5] Ref: https://www.automl.org/automl/autoweka
[6] Ref: http://hyperopt.github.io/hyperopt
[7] Ref: http://epistasislab.github.io/tpot
[8] Ref: http://autokeras.com
[9] Ref: https://www.automl.org/automl/autopytorch

Tuning model setting hyperparameters and, at the same time, selecting a subset of the most relevant inputs require efficient heuristic methods to manage such a combinatorial problem. In this article, the search for the best model is based on an optimization technique developed by Kennedy and Eberhart: the particle swarm optimization (PSO) [13]. Its popularity has undoubtedly increased due to its straightforward implementation and demonstrated high convergence ratio. This section explains the PSO algorithm and how we have modified it to find parsimonious models.

3.1. The underlying idea behind the PSO algorithm

The PSO algorithm mimics the social behavior of birds flocking and fish schooling to guide the particles toward globally optimal solutions. The movement of a particle is influenced by its own experience (its best position achieved so far) and also by the experience of other particles (the best position within a neighborhood). The formulation of canonical PSO contains expressions for calculating velocities and positions of particles. Considering a D-dimensional search space, $\mathbf{X}_i^t$ and $\mathbf{V}_i^t$ describe the position and velocity of the i-th particle in the t-th iteration, respectively. If the iteration number t is clear from the context or unnecessary, we drop the letter t and simply denote them as $\mathbf{X}_i$ and $\mathbf{V}_i$, respectively. The personal best is represented by $\hat{\mathbf{X}}_i$ whereas the local best in each neighborhood is described by $\hat{\mathbf{L}}_i$. The minimum value among the personal bests represents the global best ($\hat{\hat{\mathbf{X}}}$) and, consequently, the optimal solution. In this context, the velocity and position of the next iteration are calculated as follows:

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

$$\mathbf{X}_i^{t+1} = \mathbf{X}_i^t + \mathbf{V}_i^{t+1} \tag{2}$$

where t represents the current iteration and ω is the inertia weight that depreciates the contribution of the current velocity. The purpose of this parameter is to prevent an uncontrolled increase in particle displacement. The coefficients φ_1 and φ_2 represent the cognitive and social learning rates. They control the trade-off between global exploration and local exploitation, i.e. how the local and personal bests affect the calculation of the next position. Finally, $\mathbf{r_{i,1}}$ and $\mathbf{r_{i,2}}$ are independent and uniformly distributed random vectors in the range $[0, 1]$ whose purpose is to maintain the diversity of the swarm.

3.2. A new PSO-based optimization technique with parsimony

In the framework of FS and model optimization, it is desirable to simultaneously minimize the fitness function (J) and the model complexity (M_c) to guarantee both accuracy and generalization capacity. Thus, the modified version of PSO proposed herein includes strategies for updating the personal best, local best and the global best considering not only the goodness-of-fit, but also the principle of parsimony (keeping in mind that goodness-of-fit should be prioritized). Algorithm 1 shows a summarized pseudo-code of our modified version of the PSO algorithm. The following subsections presents the proposed algorithm and each step in detail.

3.2.1. Steps 1 and 2: initialization

The process starts with a random initial swarm of models

$$\left\{\mathbf{X}_1^0, \mathbf{X}_2, \ldots, \mathbf{X}_s^0\right\} \tag{3}$$

generated by a random and uniformly distributed Latin hypercube within the ranges of feasible values for each input parameter. We denote this operation as $random_{LHS}(s,\ D)$, where s and D represent the number of particles in the swarm and the dimensions of the design space, respectively.

Each particle $\mathbf{X}_i^0$, which represents a model configuration, is characterized by a subset of input variables and the values of setting hyperparameters to be optimized. Similarly, initial velocities $\left\{\mathbf{V}_1^0, \mathbf{V}_2^0, \ldots, \mathbf{V}_s^0\right\}$ are randomly generated according to the following expression:

$$\frac{random_{LHS}(s,\ D) - \left\{\mathbf{X}_1^0, \mathbf{X}_2^0, \ldots, \mathbf{X}_s^0\right\}}{2} \tag{4}$$

3.2.2. Steps 3 to 5: training and evaluation with parsimony criterion

The main loop (line 3) is repeated, at most, a maximum number T of iterations. This loop consists of several substeps. First, each of the s particles of the swarm is trained and evaluated. The training and validation steps of the models are conducted by k-fold cross-validation (CV). This method is repeated n times to ensure the robustness of the model against different subsamples of training data.

Firstly, the goodness-of-fit of models is assessed by J. The root mean square error (RMSE) of the CV process is utilized herein as fitness function $J(\mathbf{X}_i^t) = RMSE_{CV}$, where $RMSE_{CV}$ denotes the mean of the RMSE of the n k-Fold CV runs. The RMSE penalizes large deviations from the target values. This feature is particularly advantageous in the context of steel connections because models should predict the connection response with a similar degree of accuracy throughout the entire design space. However, in other problems it could be more interesting to use other metrics, such as the mean absolute error (MAE). Thus, the user should choose the appropriate metric accordingly. In this way, the personal best ($\hat{\mathbf{X}}_i$) of each particle is computed in each iteration. Secondly, models are re-evaluated according to their complexity. The principle of parsimony states that in the case of two models with similar accuracy, the simplest is the best. Model complexity depends on the model structure and is related to its robustness to perturbations and noise. In this case, complexity has been defined by $M_c = 10^9 N_{FS} + Int_c$, where N_{FS} is the number of selected features and Int_c is an internal complexity measurement which depends on the ML algorithm. For example, the sum of the squared weights in MLP, $Int_c = \sum w_i^2$, the number of leaves in regression trees (RT), etc. Thus, in this formula, N_{FS} dominates the other measure and, then, Int_c only comes into play when the two compared solutions have the same N_{FS}.

From now on, $\hat{\mathbf{X}}_i^p$ denotes the personal best of the i-th particle considering the parsimony criterion. In accordance with this principle, we propose the following strategy for updating the $\hat{\mathbf{X}}_i^p$. For each particle of the swarm, the $\hat{\mathbf{X}}_i^p$ is updated to the new position if the fitness value of this new position $J(\mathbf{X}_i)$ is clearly lower than the current value, or if it is better and also has lower complexity. On the other hand, $\hat{\mathbf{X}}_i^p$ is not updated if $J(\mathbf{X}_i)$ is clearly higher than the current value. For intermediate cases, where $J(\mathbf{X}_i)$ is within a tolerance in regards to the $\hat{\mathbf{X}}$, the complexity criterion is applied. Then, the $\hat{\mathbf{X}}_i^p$ is updated to the new particle position only if its complexity $M_c(\mathbf{X}_i)$ is lower than the current value. This strategy, therefore, requires a user-defined tolerance (tol) to establish the limits wherein the complexity criterion is applicable. Note that, at this point, the tolerance is applied with respect to the fitness value of $\hat{\mathbf{X}}_i$ and not of the one of $\hat{\mathbf{X}}_i^p$. This prevents, in each step, an update of $\hat{\mathbf{X}}_i^p$ to a much worse solution than $\hat{\mathbf{X}}_i$.

The particle exhibiting the lowest fitness value in the process is chosen as global best, $\hat{\hat{\mathbf{X}}}$. A pseudo-code of the $\hat{\mathbf{X}}_i^p$ update follows:

Pseudo-code $\hat{\mathbf{X}}_i^p$ update

if $J(\mathbf{X}_i) < J\left(\hat{\mathbf{X}}_i^p\right) - tol$ **then**
$\hat{\mathbf{X}}_i^p = \mathbf{X}_i$
else if $J(\mathbf{X}_i) \leqslant J\left(\hat{\mathbf{X}}_i^p\right)$ **and** $M_c(X_i) \leqslant M_c\left(\hat{\mathbf{X}}_i^p\right)$ **then**
$\hat{\mathbf{X}}_i^p = \mathbf{X}_i$
else if $J(\mathbf{X}_i) \leqslant J\left(\hat{\mathbf{X}}_i\right) + tol$ **and** $M_c(\mathbf{X}_i) < M_c\left(\hat{\mathbf{X}}_i^p\right)$ **then**
$\hat{\mathbf{X}}_i^p = \mathbf{X}_i$
endif

3.2.3. Steps 7 to 9: early stopping

Although the optimization ends when the maximum number of iterations T is reached, the proposed PSO-PARSIMONY also includes an early stopping criterion that can be used if J does not improve within a predefined tolerance and a fixed number of iterations.

3.2.4. Steps 10 to 12: updates with parsimony criterion

After training each particle and evaluating their performance, the neighborhoods have to be updated.

The topology of the swarm defines the subset of particles (neighborhood) with which each particle can exchange information. Many different topologies have been designed and studied for PSO [12]. We adopted the adaptive random topology originally proposed by Clerc [6]. In this topology, each particle in the swarm informs K particles randomly (the same particle can be chosen several times) and itself, where K is usually set to 3. As a result, each particle is informed by a number of particles (neighborhood) that can vary from 1 to s. If an iteration does not show an improvement in the fitness value of the global best $\hat{\hat{\mathbf{X}}}$, new neighborhoods are randomly generated following the same process. This topology provides more diversity in the swarm than using, for instance, a single global best and is less susceptible to being trapped in local minima. However, the convergence rate is generally slower than in a global best topology.

The next step in the PSO consists of updating the velocities and positions of the particles in order to evolve toward better solutions. This update is performed with a modified version of Eqs. 1 and 2 to promote simpler (and accurate) solutions: the personal bests computed with parsimony are used ($\hat{\mathbf{X}}_i^p$).

For this particular case where FS is included in the PSO, the binary status of features deserves special treatment. In our proposal, a continuous PSO is applied for both FS and model hyperparameters optimization. The real number x_d corresponding to the feature d is compared with a threshold value, α. Then, feature d is included in the input subset if $x_d \geqslant \alpha$. Otherwise, the feature is discarded. This approach avoids the premature convergence that characterizes the binary version of particle swarm optimization (BPSO) [30].

The inertia weight is defined as linearly decreasing with the iterations, according to Shi and Eberhart [24], $\omega = \omega_{max} - (\omega_{max} - \omega_{min})\frac{t}{T}$ where t represents the current iteration and T is a predefined maximum number of iterations. Thus, high values of ω in the first iterations have the ability to explore new areas. On the contrary, lower values of ω at the end of the process promote a refined search in local regions.

3.2.5. Steps 13 and 14: mutation and limitation

Finally, this modified version of the PSO also incorporates a mutation operator for the FS similar to that used in GA. The operator tries to prevent the premature convergence of the swarm. By default, the mutation rate m is set to $1/F$, where F represents the number of features of the problem. This value guarantees that at least one feature will change its status in each iteration (from 0 to 1 or vice versa). Concretely, for each particle i, the algorithm takes each x_d of the F components of $\mathbf{X}_i$ that belong to features (only those of the features, those of the hyper-parameters are not mutated). For each x_d of each particle, a random number between 0 and 1 is generated. If this random number is smaller than m, then x_d is mutated, i.e, if x_d is a discarded feature ($x_d < \alpha$) then x_d changes to a new random value between α and 1, in order for x_d to be selected (analogously if $x_d \geqslant \alpha$).

After updating velocities and positions and the mutation step, the particles are confined in order to avoid out-of-range positions. In this paper, the absorbing wall approach is utilized by default for this purpose [34]. Thus, when a particle is out of the feasible range, the position is set to the boundary and velocity is set to zero.

3.2.6. Steps 15 and 16: output

As explained before, the process ends if either the maximum number of iterations T is reached or the early stopping criterion is satisfied. The final result is the position of the best particle of the whole process, $\hat{\hat{\mathbf{X}}}$. The process explained in this section can be applied to each ML algorithm technique and dataset.

4. Performance analysis

4.1. Performance analysis of PSO-PARSIMONY vs GA-PARSIMONY with public datasets

Thirteen public datasets from the UCI repository [8], with different numbers of rows and features, were selected to perform the comparison between PSO-PARSIMONY and GA-PARSIMONY with two tolerances: $tol = 10^{-6}$ and $tol = 10^{-3}$. The effect of tol is highly dependent on the ability of machine learning models to explain the problem. Thus, a fine-tuning of tol should be performed on a case-by-case basis (as is explained in Section 4.2) to search for optimal trade-off between parsimony and J but, due to the high computational costs for the development of the experiment (where several months of computation were needed), it was decided to choose two representative tolerances: a very low one, 10^{-6} and an intermediate one that produced a good compromise between parsimony and J in higher dimensional data sets, 10^{-3}.

Table 1 shows a summary of the PSO-PARSIMONY settings for evaluating the performance with the public datasets. The GA-PARSIMONY settings were similar to previous experiments in [11], but with a population size of 24, a maximum number of generations of $G = 200$, and an early stopping of 35, the same as in the PSO-PARSIMONY experiments. Following empirical studies, ω_{max} and ω_{min} were set to 0.9 and 0.4, respectively [25,9]. In addition, the cognitive and social learning rate were set to $\frac{1}{2} + ln(2)$, as suggested by Clerc [5]. In this case, `MLPRegressor` algorithm from `sklearn` package was selected to train and validate the multilayer perceptron models (MLP) with a 5-repeated 5-fold cross validation.

The algorithm was defined by default with a single hidden layer of neurons with sigmoid activation functions. The hyperparameters to be modified, as well as the range of them, are defined in Table 1.

PSO-PARSIMONY was implemented by the authors in the Python language[10]. Also, a new GA-PARSIMONY Python version[11] of the `GAparsimony` R package [16] was developed. All experiments were implemented in nine separate 24 core servers from the Beronia

[10] Available at https://github.com/jodivaso/PSOparsimony.
[11] Available in Python Package Index (PyPI).

Table 1
Initial setting parameters for PSO-PARSIMONY and the MLP hyperparameter searching space.

Parameter	Description	Values
φ_1	Cognitive learning rate	$0.5 + ln(2)$
φ_2	Social learning rate	$0.5 + ln(2)$
numindiv	Number of swarm particles	24
iter	Number of iterations in optimization process	200
early	Num of iterations for early stopping	35
tol	Margin to consider *J* similar	10^{-6}
α	Lower feature's prob. of being considered input	0.5
$\omega_{max}, \omega_{min}$	Parameters to calculate inertia weights	0.9, 0.4
K	Number of particles to be informed for each one	3
m	Mutation rate	0.03
$k-fold$	Number of folds in Repeated-CV	5
runs	Number of run in Repeated-CV	5
MLP	Number of hidden neurons (search space)	[1, 2,. . ., 25]
	Weight decay (search space)	$10^{[-6.0,3.0]}$
	Solver	L-BFGS
	Activation function	sigmoid

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 2 shows the results of the best models obtained with PSO-PARSIMONY versus GA-PARSIMONY, with $tol = 10^{-6}$ and for the 13 public datasets. From left to right, columns are for each dataset: the name, the number of rows (*#rows*) and the total features (*#feats*); and for each algorithm (*PSO* and *GA*): the average of 5 runs of *J*, the number of features (N_{FS}), the elapsed time in minutes (*time*) and the number of iterations (*iters*). It can be observed that for each one of the 13 datasets, PSO found a solution with better accuracy. However, PSO was more costly (in terms of computational effort), since it required more iterations and more time. Roughly speaking, PSO was three times slower and needed doubling the iterations than GA.

The same experiments were repeated with $tol = 10^{-3}$ to observe the behavior of the methods in the search for parsimony. Table 3 shows that GA with $tol = 10^{-3}$ achieved a higher reduction in the number of N_{FS} in ten datasets with respect to $tol = 10^{-6}$, especially in datasets with high dimensionality (*bank*, *puma*, *ailerons*, *tecator*, *crime*). PSO also improved the reduction of N_{FS} in *pm*10, *concrete*, *puma*, *meta*, *tecator* and *crime*, although it was not as significant as with GA.

Finally, Table 4 shows a comparison of the average of features obtained with the two tolerances ($tol = 10^{-6}$ and $tol = 10^{-3}$, and for both methods. Columns $\overline{methodX}_{N_{FS}}$ corresponds with the mean of features obtained with *method* and $tol = 10^{-X}$. $\overline{dGA}_{N_{FS}}$ and $\overline{dGA}_{N_{FS}}$ indicate the difference between the number of features for both tolerances so that positive values show parsimony reductions and negative values the opposite. In small datasets, the observed parsimony reduction is very small or even negative, although the negative values are all lower (in absolute value) than -0.5. Actually, *tol* must be adjusted for each database to obtain the best trade-off between *J* and parsimony, as explained in the next section. In this case, the chosen value was not optimized for each dataset but it reduced complexity for higher dimensional datasets although no improvement in parsimony reduction was observed for other lower dimensional datasets. Thus, for most of these databases a finer tuning of "tol" will be necessary. However, a significant reduction is observed in high dimensional datasets such as *ailerons*, *tecator* and *crime*. GA-PARSIMONY obtained better parsimony, but it is important to emphasize that the main objective of the PSO-PARSIMONY was to improve the accuracy of the models. The results shown in Tables 2 and 3 demonstrate that, although PSO-PARSIMONY obtained less parsimonious models than GA-PARSIMONY, the new method obtained better accuracy models in all datasets. In short, PSO-PARSIMONY is a good alternative to find better solutions with a good balance between accuracy and parsimony.

If we analyze the overall performance of both methods we can conclude that PSO needs more iterations, but obtains more accurate models than GA, since the latter stops the optimization process too early. Due to the excessive computational cost, the number of individuals that can be evaluated with PSO or GA in each generation is very small. This makes GA crossover mechanisms not as efficient as in other GA-based optimization problems where hundreds or thousands of individuals can be evaluated. Thus, in a few generations, GA-PARSINOMY produces populations of individuals that are very similar to each other. This low diversity causes GA optimization to quickly be stuck at a local minimum, making the search for accurate models suboptimal. However, PSO can perform finer tuning and find more accurate solutions, although the computational cost is higher and the solutions are more complex.

The diversity in a population with *numindiv* individuals can be defined as:

$$Fdist = \frac{\sum_i \sum_j d_{ij}}{numindiv^2} \tag{5}$$

where d_{ij} corresponds to the Euclidean distance between the binary vectors defining the selected features of the *i* and *j* individuals. Thus, the larger this value is, the more diversity exists in that population. The left side of Fig. 1 shows the evolution of *Fdist* for GA-PARSIMONY and PSO-PARSIMONY with *crime* dataset and in the first 80 iterations. The right side of the figure shows the number of features that have changed their status (selected/unselected) between consecutive iterations. It can be observed how the diversity of GA drops sharply from the first iterations, while with PSO the diversity decreases slowly. However, GA performs many more feature state changes in the first iterations than PSO, which allows it to track a larger number of feature combinations, although it decreases drastically as iterations pass and individuals become more and more similar to each other.

In terms of the complexity of the solutions, GA outperformed PSO, finding more parsimonious solutions (with smaller number of features) on most of the datasets (11 out of 13). The crossover mechanisms of optimization with GA yield solutions with a reduced number of features within a few generations. However, this sharp reduction means that the obtained solutions cannot continue to improve in accuracy in the next generations. It can be observed that the difference between PSO and GA (with respect to parsimony) was smaller for data sets with a reduced number of features. In those cases, the solutions found by the PSO method were about 10% more complex than those found by GA. However, in datasets with a larger number of features, PSO found more accurate solutions, but using twice as many features as the solutions found by GA. While GA substantially reduces complexity in a few generations, thanks to the crossover mechanism between individuals, PSO is much more inefficient when the number of features is high, since the selection is performed on the basis of a probability that is slowly updated by the particle velocity. Thus, the change of state of a feature, between being selected or not (or vice versa), only occurs when the probability is higher than 0.50 (the α value).

4.2. Detailed performance analysis of PSO-PARSIMONY vs GA-PARSIMON in a case of study

The case study presented herein focuses on the bolted T-stub component (Fig. 2a), which corresponds to the tension zone in beam-to-column connections. The T-stub component comprises

Table 2
PSO-PARSIMONY vs. GA-PARSIMONY with 13 public datasets [8] (results are the average of 5 runs with 5-fold cross-validation with each methodology and $tol = 10^{-6}$).

dataset	*#rows*	*#feats*	$\overline{PSO}_J$	$\overline{GA}_J$	$\overline{PSO}_{N_{FS}}$	$\overline{GA}_{N_{FS}}$	$\overline{PSO}_{time}$	$\overline{GA}_{time}$	$\overline{PSO}_{iters}$	$\overline{GA}_{iters}$
strike	625	7	**.8259**	.8644	**1.6**	3.0	89.2	**20.9**	86.6	**57.8**
no2	500	8	**.6561**	.6598	6.0	6.0	60.5	**15.2**	98.2	**42.0**
pm10	500	8	**.8397**	.8417	5.8	**5.4**	35.5	**18.5**	76.8	**41.8**
concrete	1030	9	**.2810**	.2944	**7.6**	8.0	454.9	**194.9**	101.2	**46.4**
housing	506	14	**.3066**	.3221	10.6	**9.7**	244.1	**73.2**	145.9	**55.8**
bodyfat	252	15	**.1049**	.1110	**3.0**	4.8	100.2	**34.9**	135.8	**58.4**
meta	504	18	**.7102**	.7123	9.2	**6.0**	5.1	**3.8**	79.4	**45.2**
cpu_act	8192	22	**.1246**	.1254	14.4	**14.0**	2261.9	**936.9**	141.0	**62.6**
bank	8192	33	**.6339**	.6383	23.6	**23.4**	1308.9	**425.1**	162.8	**70.2**
puma	8192	33	**.1803**	.1810	6.1	**5.0**	3586.1	**1051.8**	115.0	**47.0**
ailerons	13750	41	**.3827**	.3828	17.7	**15.0**	3710.5	**1196.8**	115.9	**49.0**
tecator	240	125	**.0321**	.0328	74.0	**32.6**	114.5	**69.2**	115.2	**88.6**
crime	2215	128	**.5959**	.5960	56.4	**27.8**	843.2	**413.6**	194.2	**135.8**

Table 3
PSO-PARSIMONY vs. GA-PARSIMONY with 13 public datasets [8] (results are the average of 5 runs with each methodology and $tol = 10^{-3}$).

dataset	*#rows*	*#feats*	$\overline{PSO}_J$	$\overline{GA}_J$	$\overline{PSO}_{N_{FS}}$	$\overline{GA}_{N_{FS}}$	$\overline{PSO}_{time}$	$\overline{GA}_{time}$	$\overline{PSO}_{iters}$	$\overline{GA}_{iters}$
strike	625	7	**.8385**	.8648	**1.8**	3.0	105.1	**16.3**	88.0	**39.6**
no2	500	8	**.6560**	.6601	6.0	6.0	59.8	**17.5**	102.8	**46.4**
pm10	500	8	**.8399**	.8414	5.4	5.4	36.5	**17.5**	85.4	**36.8**
concrete	1030	9	**.2894**	.2953	**7.4**	7.8	468.0	**160.9**	107.2	**41.6**
housing	506	14	**.3126**	.3256	11.0	**10.0**	215.8	**74.6**	104.0	**61.0**
bodyfat	252	15	**.1071**	.1081	3.4	**2.0**	97.3	**40.2**	128.2	**69.2**
meta	504	18	**.7104**	.7112	6.8	**3.6**	**4.0**	4.9	67.8	**75.8**
cpu_act	8192	22	**.1241**	.1247	14.8	**13.4**	1241.7	**788.3**	84.6	**50.0**
bank	8192	33	**.6342**	.6379	23.6	**19.8**	1298.8	**629.1**	177.8	**101.6**
puma	8192	33	**.1805**	.1810	4.6	**4.2**	2188.6	**1028.4**	96.0	**51.2**
ailerons	13750	41	**.3823**	.3829	17.8	**10.4**	2663.2	**1144.7**	115.2	**65.6**
tecator	240	125	**.0323**	.0331	60.0	**25.8**	126.9	**44.5**	131.2	**69.4**
crime	2215	128	**.5934**	.5957	49.8	**19.8**	797.8	**410.5**	185.6	**143.6**

Table 4
Parsimony obtained with $tol = 10^{-3}$ and $tol = 10^{-6}$ for both methods.

dataset	*#feats*	$\overline{GA6}_{N_{FS}}$	$\overline{GA3}_{N_{FS}}$	$\overline{dGA}_{N_{FS}}$	$\overline{PSO6}_{N_{FS}}$	$\overline{PSO3}_{N_{FS}}$	$\overline{dPSO}_{N_{FS}}$
strike	7	3.0	3.0	0.0	**1.6**	1.8	−0.2
no2	8	**6.0**	**6.0**	0.0	**6.0**	**6.0**	0.0
pm10	8	**5.4**	**5.4**	0.0	5.8	**5.4**	0.4
concrete	9	8.0	7.8	0.2	7.6	**7.4**	0.2
housing	14	**9.7**	10.0	−0.3	10.6	11.0	−0.4
bodyfat	15	4.8	**2.0**	2.8	3.0	3.4	−0.4
meta	18	6.0	**3.6**	2.4	9.2	6.8	2.4
cpu	22	14.0	**13.4**	0.6	14.4	14.8	−0.4
bank	33	23.4	**19.8**	3.6	23.6	23.6	0.0
puma	33	5.0	**4.2**	0.8	6.1	4.6	1.5
ailerons	41	15.0	**10.4**	4.6	17.7	17.8	−0.1
tecator	125	32.6	**25.8**	6.8	74.0	60.0	14.0
crime	128	27.8	**19.8**	8.0	56.4	49.8	6.6

two t-shape profiles tied by their flanges with one or more rows of bolt (Fig. 2b). The tensile load applied to the web is transferred by the flange in bending and by the bolts in tension. During this process, the contact between the flanges produces a prying action that increases the forces developed in the bolts. The contact area, as well as the pressure magnitude, evolves during the loading process, complicating an adequate evaluation of the force–displacement response. Non-linear material laws, large deformations, and the existence of different failure patterns also present further challenges to calculating the T-stub component.

Numerical approaches such as the Finite Element (FE) method constitute a reliable tool for assessing steel connections. Fig. 3 shows the results of one simulation with an advanced FE model of the T-stub component [10]. The FE model includes complete stress–strain nonlinear material relationships and a refined characterization of the bolt, including threaded length, nut, and washers. Additionally, the main novelty of the numerical model is the implementation of a continuum damage mechanics model to simulate the failure of the bolted connection. Thus, the force–displacement response of the T-stub can be fully characterized, from the initial stiffness up to the fracture point (Fig. 3b). Interested readers can refer to [10] for an in-depth description of the FE model.

Finally, in this study, the objective was to improve previous experiments [11,18] by obtaining more precise and parsimonious models to predict three key parameters of the T-stub force–displacement curve response: initial stiffness (k_i), maximum strength (F_u) and displacement at failure (d_f). However, previous work showed that the target d_f was not deterministic, so that the intrinsic error of the estimation cannot be further reduced even if models with lower validation error are achieved. Thus, modeling efforts were focused on the other two targets: k_i and F_u, that are fundamental to estimate the safety of the T-stub.

Fig. 1. Evolution of diversity (left) and number of features that change state (selected/unselected) between consecutive iterations (right) with GA-PARSIMONY and PSO-PARSIMONY and *crime* dataset.

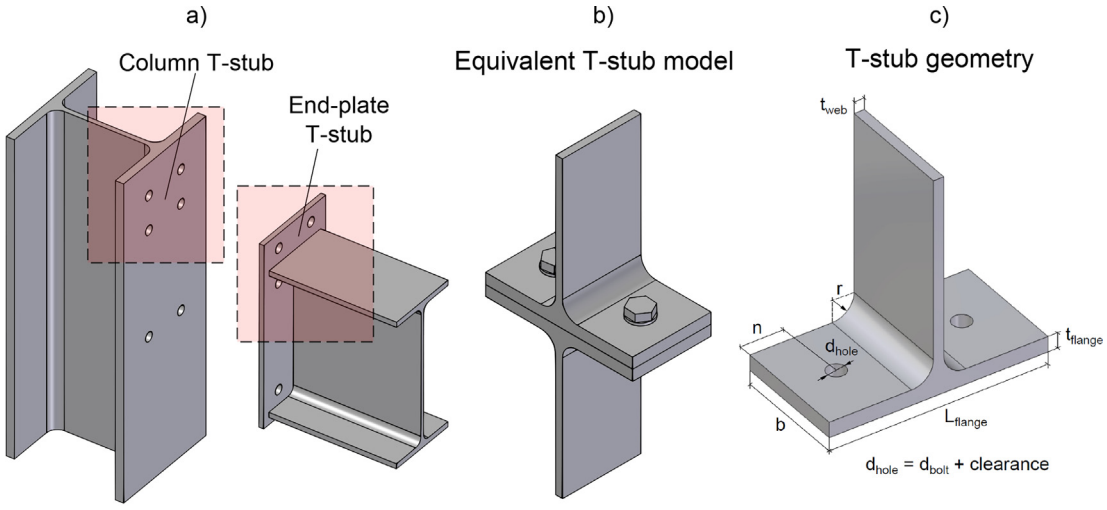


Fig. 2. Tension zone on steel connections. (a) End-plate beam-to-column connection, (b) equivalent T-stub model, and (c) T-stub geometry.

4.2.1. Dataset

The advanced FE model constituted an excellent tool for the generation of a training and testing dataset. Therefore, training data were created with 820 FE simulations from a Design of Computer Experiments (DoCE). DoCE accounts for the deterministic nature of computer experiments, assuming that numerical noise is negligible. For these cases, space-filling sampling techniques are appropriate because they uniformly distribute the points throughout the design space. One of the most widely used space-filling designs is the Latin hypercube sampling (LHS), introduced by McKay et al. in 1979 [17]. LHS divides each input into n equally probable intervals and selects a random value in each interval. The principal advantage of this method is that each input variable is represented in every division of its range. Additionally, a test dataset consisting of 76 samples was generated separately to check the accuracy and generalization capacity of models to predict unseen data.

LHS method was used to define the input values of the subsequent FE simulations with different conditions. Table 5 describes the feasible ranges of T-stub geometrical parameters (Fig. 2c) and the mechanical properties of the hot-rolled profiles and bolts used herein. For each combination of input values, a FE simulation was conducted to characterize the response of the T-stub component. Regarding the outputs, the performance of models was evaluated for their prediction of two key parameters of the force–displacement curve: initial stiffness (k_i) and maximum strength (F_u).

4.2.2. Experiment settings

The initial setting parameters for PSO-PARSIMONY in this experiment are rather similar to the ones presented in Table 1. The only differences are *tol* (the margin to consider J similar) and the size of the swarm. Concretely, six values of *tol* have been considered: $[10^{-6}, 0.001, 0.003, 0.005, 0.01, 0.025]$. The particle swarm was evaluated with eight different sizes in this experiment, from 5

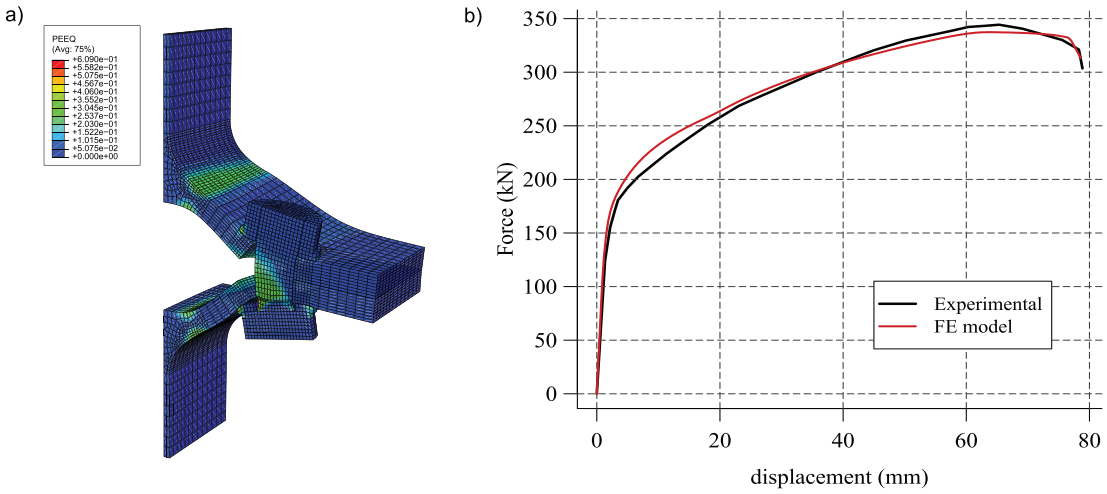


Fig. 3. Advanced FE model of the T-stub bolted component. a) FE simulation: equivalent plastic strain (PEEQ); b) Force–displacement response: FE model vs. Experimental test.

Table 5
Ranges of the input features included in the DoCE.

Variable	Description [units]	Range
dbolt	Nominal bolt diameter [–]	M12 - M27
clearance	Difference between bolt hole and bolt diameter [mm]	0.50–3.50
t_{flange}	Flange thickness of the T-shape profile [mm]	8.00–30.00
t_{web}	Web thickness of the T-shape profile [mm]	5.00–20.00
L_{flange}	Flange length of the T-shape profile [mm]	52.00–180.00
r	Flange-to-web connection radius [mm]	9.75–43.00
n	Dist. from center of the hole to edge of flange [mm]	15.75–106.00
b	Width of the T-shape profile [mm]	42.00–187.00
L_{thread}	Thread length of the bolt [mm]	2.50–60.25
σy	Yield strength of the struct. steel [MPa]	200–400
σ_u	Stress at the max. tensile load of the structural steel [MPa]	300–800
E_h	Strain-hardening coefficient of the structural steel [MPa]	1000–3000
σ_{yb}	Yield strength of the bolt steel [MPa]	640–1098
σ_{ub}	Stress at the maximum tensile load of the bolt steel [MPa]	800–1200
ϵ_{ub}	Strain at the maximum tensile load of the bolt steel [-]	0.07–0.14

to 40 particles. Also note that the number of input features is 15, which is a size that seems suitable for the PSO-PARSIMONY method according to the results presented in Section 4.1.

Again, experiments were implemented on nine separate 24-core servers from the Beronia cluster at the University of La Rioja.

4.2.3. Performance analysis

Table 6 shows results of the best models from 5 runs obtained with PSO-PARSIMONY versus GA-PARSIMONY and with different values of *tol*. In this experiments, PSO_J and GA_J corresponded to the $RMSE_{CV}$ error of the standardized logarithmic of the target. Logarithmic was used to transform the output to be close to a normal distribution. Therefore, results improved upon the previous experiments in [11].

In Table, $PSO_{N_{FS}}$ and $GA_{N_{FS}}$ are the number of features, $PSO_{\sum w^2}$ and $GA_{\sum w^2}$ the internal model complexity, and PSO_{time} and GA_{time} the elapsed time in minutes. The table show the best model obtained from 5 different runs of PSO-PARSIMONY and GA-PARSIMONY. Bold numbers indicate the best value for each *tol* value and target variable comparing between both methodologies. Red numbers indicate the best score for each target variable.

From the Table 6 it can be seen that PSO obtained better solutions than GA for low $tol \leqslant 0.001$. For these ranges, N_{FS} were identical although the internal complexity of the models was higher in PSO than in GA. In addition, PSO needed more time than GA to find the best solution. These results were similar to those obtained with the public datasets. In both target variables, the best solution is obtained by PSO with $tol = 0.001$. Note that PSO combined with a lower tolerance ($tol = 1e-6$) found a solution with the same score for k_i and the same number of features as the best one, but with higher internal complexity.

Additionally, Table 7, shows the averaging results of the 5 runs including the mean of the non-normalized testing error *Jtst* for PSO and GA methods: $\overline{PSO}_{Jtst}$ and $\overline{GA}_{Jtst}$; and the mean of the PSO and GA iterations: $\overline{PSO}_{iters}$ and $\overline{GA}_{iters}$. $\overline{PSO}_J$ and $\overline{GA}_J$ correspond to the normalized values to see the effect of *tol*, while $\overline{PSO}_{Jtst}$ and $\overline{GA}_{Jtst}$ correspond to the de-normalized values for comparison with previous works.

In this case, it can be observed that the best results were obtained with $tol = 10^{-6}$. For both targets, the best validation *J* errors were obtained with PSO. However, the testing errors (*Jtst*) were higher in PSO for F_u, probably because PSO obtained models with slightly higher internal complexity. In contrast, with k_i the average testing errors improved for the models obtained with PSO, although the average N_{FS} was lower in GA (10.8 vs. 11.0). Finally, iteration and times were clearly higher in PSO-PARSIMONY than in GA-PARSIMONY.

Figs. 4 and 5 show the average value of N_{FS} and the minimum, mean and maximum values of PSO_J and GA_J of 5 runs with $tol = 10^{-6}$ and different number of particles (*numindiv*). Finally, Fig. 6 represents the range of times employed by both methods and for each number of particles.

From these figures it can be clearly deduced that PSO-PARSIMONY has a much more robust and stable behavior, obtaining better solutions even with 10 or 5 particles. Although PSO-PARSIMONY needs more time to obtain the solutions and gets slightly less parsimonious solutions, the reduced minimum–maximum range observed in the PSO method ensures convergence to better solutions with a smaller number of particles and number of algorithm runs. For example, in our application case, 10 particles in PSO achieve better results than 40 individuals in GA. In terms of time, 10 particles in PSO take (roughly speaking, in mean) half as long as 40 in GA. In addition, GA needs more runs to guarantee that

Table 6
Comparative of the best MLP model obtained from 5 runs of PSO-PARSIMONY and GAPARSIMONY with a population size of $P = 35$, different *tol* values and for maximum strength (F_u) and initial stiffness (k_i).

Target	*tol*	PSO_J	GA_J	$PSO_{N_{FS}}$	$GA_{N_{FS}}$	$PSO_{\sum w^2}$	$GA_{\sum w^2}$	PSO_{time}	GA_{time}
F_u	1e-06	**.06109**	.06179	12	12	210.4	**161.4**	639.9	**266.1**
	0.001	**.06089**	.06131	12	12	220.5	**138.7**	389.0	**248.3**
	0.003	.06216	**.06185**	12	12	183.2	**132.3**	270.1	**228.0**
	0.005	**.06243**	.06397	12	**11**	182.8	**141.7**	293.1	**123.8**
	0.010	.06322	**.06265**	12	12	199.5	**134.2**	225.9	**94.9**
	0.025	.06268	**.06262**	12	12	**133.8**	153.1	230.6	**217.7**
k_i	1e-06	**.05021**	.05086	11	11	132.9	**120.7**	592.5	**154.9**
	0.001	**.05021**	.05067	11	11	**128.0**	145.0	331.3	**182.3**
	0.003	**.05031**	.05240	11	**10**	**139.5**	152.0	437.7	**334.8**
	0.005	.05105	**.05083**	11	11	174.7	**126.2**	**260.0**	359.0
	0.010	**.05119**	.05253	11	11	136.1	136.1	222.6	**173.9**
	0.025	.05306	**.05205**	11	**10**	**188.4**	141.2	**203.8**	217.5

Table 7
PSO-PARSIMONY vs GA-PARSIMONY with a population size of $P = 35$, different *tol* values and for maximum strength (F_u) and initial stiffness (k_i) (results are the average of the 5 runs). Note: $\overline{PSO}_J$ and $\overline{GA}_J$ correspond to the normalized values to see the effect of *tol*, while $\overline{PSO}_{Jtst}$ and $\overline{GA}_{Jtst}$ correspond to the de-normalized values for comparison with previous works.

tol	$\overline{PSO}_J$	$\overline{GA}_J$	$\overline{PSO}_{Jtst}$	$\overline{GA}_{Jtst}$	$\overline{PSO}_{N_{FS}}$	$\overline{GA}_{N_{FS}}$	$\overline{PSO}_{time}$	$\overline{GA}_{time}$	$\overline{PSO}_{iters}$	$\overline{GA}_{iters}$
					F_u					
1e-06	**.06154**	.06338	10.09	**9.80**	12.0	12.0	446.4	**199.4**	113.4	**52.8**
0.001	**.06166**	.06288	**9.77**	10.09	12.0	**11.2**	355.3	**195.7**	95.6	**56.8**
0.003	**.06263**	.06390	10.32	10.32	12.0	**10.8**	315.2	**155.8**	86.2	**49.4**
0.005	**.06309**	.06437	9.98	**9.47**	11.8	**10.8**	282.1	**146.3**	79.4	**42.2**
0.010	.06379	**.06374**	**10.43**	10.74	11.8	**10.8**	248.4	**151.0**	71.2	**54.4**
0.025	**.06391**	.06456	**10.09**	10.42	11.6	**10.8**	214.8	**189.3**	**62.8**	71.2
					k_i					
1e-06	**.05038**	.05205	**7.69**	7.81	11.0	**10.8**	469.5	**203.2**	124.5	**53.4**
0.001	**.05078**	.05133	**7.90**	8.15	10.8	10.8	343.0	**173.9**	92.4	**49.8**
0.003	**.05075**	.05344	**7.73**	7.92	11.0	**10.2**	450.6	**177.7**	122.6	**54.0**
0.005	**.05166**	.05226	**7.13**	8.07	11.0	**10.4**	253.5	**252.8**	**70.4**	79.4
0.010	**.05255**	.05366	**7.45**	8.17	10.4	**9.7**	247.8	**183.3**	**70.6**	76.7
0.025	.05353	**.05317**	**7.77**	7.86	10.8	**10.4**	206.2	**126.2**	62.4	**55.4**

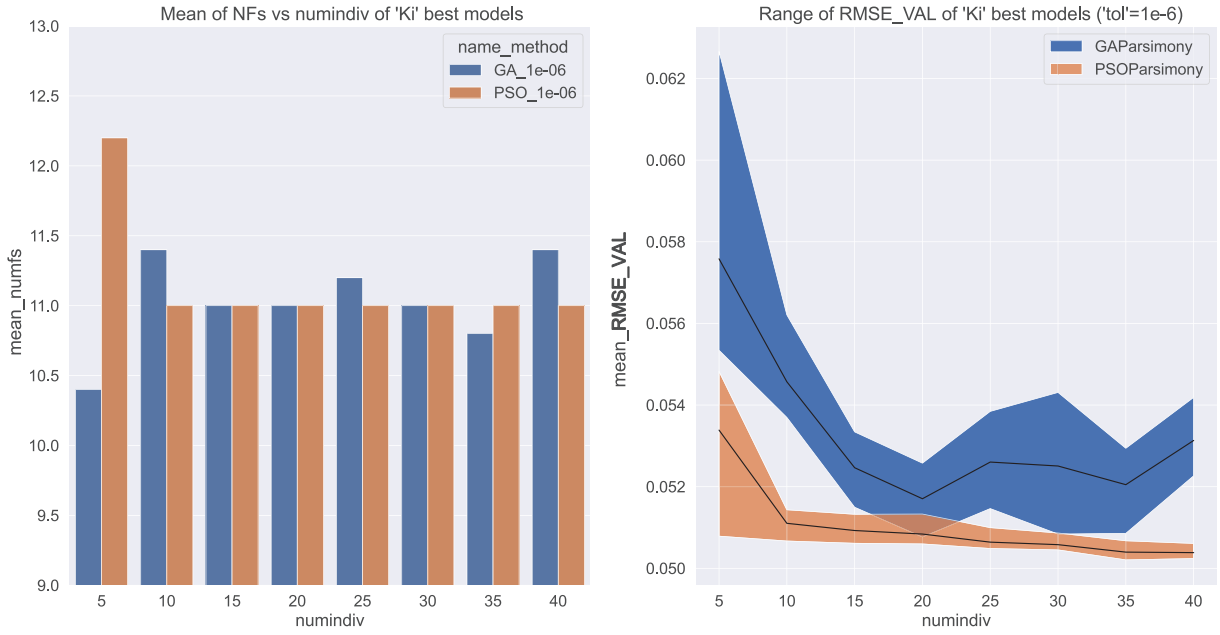


Fig. 4. Mean of N_{FS} (left) and Range: [min, mean, max] (right) of 5 runs of PSO_J and GA_J for F_u with different number of particles (*numindiv*) and $tol = 10^{-6}$.

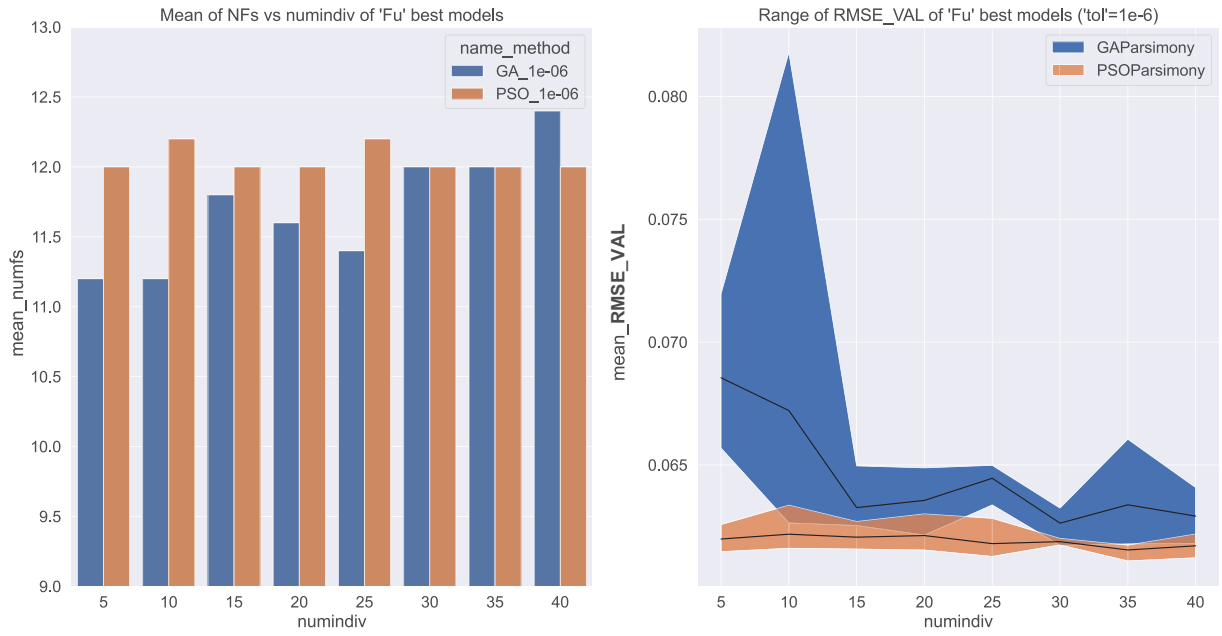


Fig. 5. Mean of N_{FS} (left) and Range: [min, mean, max] (right) of 5 runs of PSO_J and GA_J for k_i with different number of particles (*numindiv*) and *tol* = 10^{-6}.

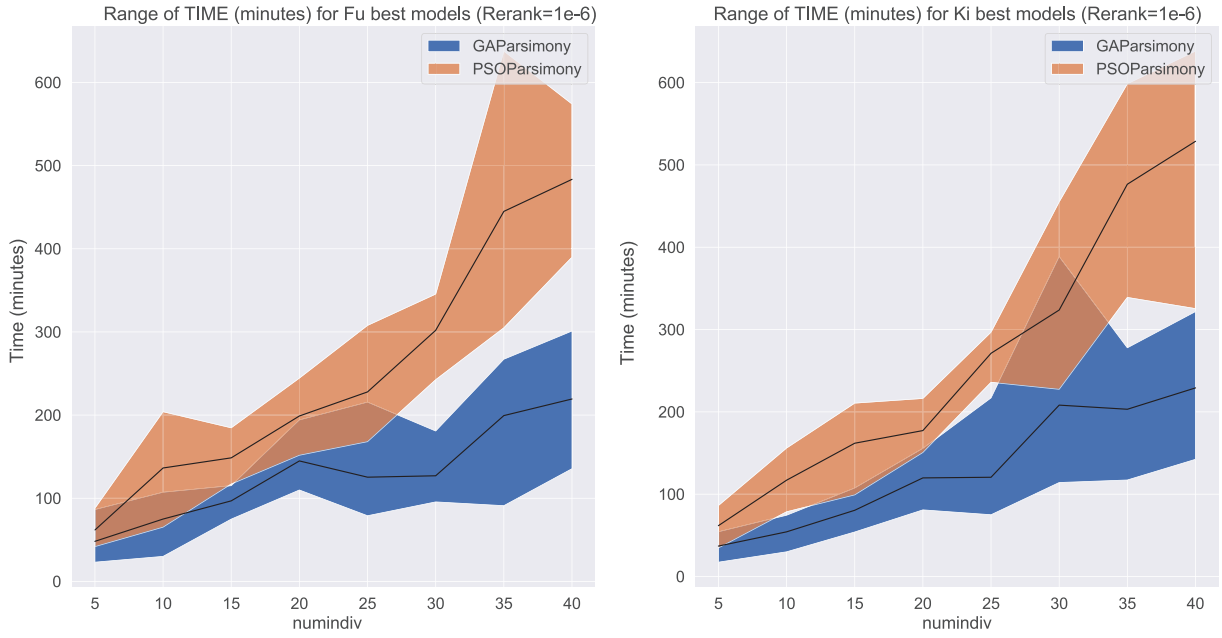


Fig. 6. Range: [min, mean, max] of 5 runs of PSO_{time} and GA_{time} for F_u (left) and k_i (right) targets with different number of particles (*numindiv*) and tol = 10^{-6}.

the solution obtained is accurate, due to the wider min–max range, while one run in PSO guarantees a good solution. Hence, if one performs 5 runs of GA and 1 in PSO, PSO would obtain a better solution than GA being 10 times faster.

As previously discussed, it is important to note that this is not a classical optimization problem where *J* and complexity are of equal relevance. As observed in these experiments, the search is focused on *J*, but it is highly probable to stop at a local minimum if the model complexity drops too fast.

Thus, the higher initial computational cost of PSO-PARSIMONY can be compensated by reducing the number of particles and the number of runs. This also makes it more suitable for searching models with larger datasets where the computational cost for each individual is higher.

5. Conclusions

The search for parsimonious models by means of feature selection and hyperparameter fitting can be a difficult and computationally intensive task. The development of strategies to reduce the number of models to be trained and evaluated is one of the fundamental objectives in this field. In this sense, this work shows a new methodology based on PSO for the simultaneous search of the best model parameters and the selection of the most influential input features, always aiming to balance robustness and parsimony. The main innovation of this method is the strategy to update the personal best using a complexity criterion $\hat{\mathbf{X}}_p$ (and thus, the local best and the global best). This allows us to

meet the goal of obtaining not only accurate but also parsimonious models.

Our experiments have shown that PSO-PARSIMONY obtains more accurate models than our previous GA-PARSIMONY methodology, although it needs a higher number of iterations to converge and obtains slightly less parsimonious models than the previous method. However, the new method has been shown to be much more efficient and robust in finding good solutions even with a reduced number of individuals and fewer runs. On the contrary, GA-PARSIMONY requires more individuals and more runs, since it has a wider minimum–maximum range of accuracy, being much less robust, and the individuals generated are similar to each other. Thus, by reducing the number of particles and algorithm runs, the PSO method can balance the computational effort required for each optimization.

Although this methodology is a promising method, it has been observed that in high-dimensional datasets the models are substantially more complex than those obtained with GA-PARSIMONY. While GA substantially improves parsimony in a few generations, thanks to the crossover mechanism between individuals, the PSO optimization process for feature selection, based on particle velocity changes, makes it much more inefficient when the number of features is high. Thus, although this methodology can be successfully used on datasets with a small number of features, further research is needed to improve the method with high-dimensional datasets in order to reduce the computational effort and find accurate and parsimonious solutions. One idea for future research is the realization of a hybrid methodology combining GA crossover mechanisms in the early stages of the search process to quickly improve parsimony and then using PSO to improve model accuracy.

Although it is a promising method, it has been observed that it substantially worsens the search for parsimony with high-dimensionality datasets. Thus, although this methodology can be successfully used on datasets with a small number of features, further research is needed to improve the method for reducing the computational effort and finding accurate and parsimonious solutions with high-dimensionality datasets.

Data availability

The authors do not have permission to share data.

Declaration of Competing Interest

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

Acknowledgements

The work is supported by grant PID2020-116641 GB-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 No. UNLR-094E-2C-225.

References

[1] An advanced methodology to enhance energy efficiency in a hospital cooling-water system. Journal of Building Engineering 43, 102839 (2021). https://doi.org/10.1016/j.jobe.2021.102839

[2] R. Ahila, V. Sadasivam, K. Manimala, An integrated PSO for parameter determination and feature selection of ELM and its application in classification of power system disturbances, Appl. Soft Comput. 32 (2015) 23–37.

[3] 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, https://doi.org/10.1016/j.enconman.2015.02.086.

[4] Y. Bengio, Y. Grandvalet, No unbiased estimator of the variance of k-fold cross-validation, J. Mach. Learn. Res. 5 (2004) 1089–1105.

[5] Clerc, M.: Stagnation Analysis in Particle Swarm Optimisation or What Happens When Nothing Happens. Dept. Comput. Sci., Univ. Essex, Colchester, U.K., Tech. Rep. CSM-460 (2006).

[6] M. Clerc, Particle swarm optimization, vol. 93, John Wiley & Sons, 2010.

[7] E. Combrisson, K. Jerbi, Exceeding chance level by chance: The caveat of theoretical chance levels in brain signal classification and statistical assessment of decoding accuracy, J. Neurosci. Methods 250 (2015) 126–136, https://doi.org/10.1016/j.jneumeth.2015.01.010, cutting-edge EEG Methods.

[8] Dua, D., Graff, C.: UCI machine learning repository (2017),http://archive.ics.uci.edu/ml.

[9] Eberhart, R., Shi, Y.: Comparing inertia weights and constriction factors in particle swarm optimization. In: Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512). vol. 1, pp. 84–88 vol 1 (2000). DOI: 10.1109/CEC.2000.870279.

[10] J. Fernandez-Ceniceros, A. Sanz-Garcia, F. Antoñanzas-Torres, Martinez-de Pison, F.J.: A numerical-informational approach for characterising the ductile behaviour of the t-stub component. part 1: Refined finite element model and test validation, Eng. Struct. 82 (2015) 236–248, https://doi.org/10.1016/j.engstruct.2014.06.048.

[11] J. Fernandez-Ceniceros, A. Sanz-Garcia, F. Antoñanzas-Torres, Martinez-de Pison, F.J.: 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, https://doi.org/10.1016/j.engstruct.2014.06.047.

[12] E.M. Figueiredo, T.B. Ludermir, Investigating the use of alternative topologies on performance of the pso-elm, Neurocomputing 127 (2014) 4–12.

[13] Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN'95 - International Conference on Neural Networks. vol. 4, pp. 1942–1948 vol 4 (1995). DOI: 10.1109/ICNN.1995.488968.

[14] 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. Stat. Data Anal. 118 (2018) 126–137, https://doi.org/10.1016/j.csda.2017.09.004.

[15] B. Ma, Y. Xia, A tribe competition-based genetic algorithm for feature selection in pattern classification, Appl. Soft Comput. 58 (2017) 328–338.

[16] Martinez-de-Pison, F.J.: GAparsimony: Searching Parsimony Models with Genetic Algorithms (2019),https://CRAN.R-project.org/package=GAparsimony, R package version 0.9.4.

[17] M.D. McKay, R.J. Beckman, W.J. Conover, Comparison of three methods for selecting values of input variables in the analysis of output from a computer code, Technometrics 21 (2) (1979) 239–245, https://doi.org/10.1080/00401706.1979.10489755.

[18] A. Pernía-Espinoza, J. Fernandez-Ceniceros, J. Antonanzas, R. Urraca, F.J. Martinez-de Pison, Stacking ensemble with parsimonious base models to improve generalization capability in the characterization of steel bolted components, Appl. Soft Comput. 70 (2018) 737–750, https://doi.org/10.1016/j.asoc.2018.06.005.

[19] 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 (2020), https://doi.org/10.1016/j.neucom.2020.02.135.

[20] 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, https://doi.org/10.1016/j.neucom.2018.05.136, recent Advancements in Hybrid Artificial Intelligence Systems.

[21] F. Martinez-de Pison, E. Fraile-Garcia, J. Ferreiro-Cabello, R. Gonzalez, A. Pernia, Searching parsimonious solutions with GA-PARSIMONY and XGBoost in high-dimensional databases 527 (2017) 201–210, https://doi.org/10.1007/978-3-319-47364-2-20.

[22] J. Reunanen, Overfitting in making comparisons between variable selection methods, J. Mach. Learn. Res. 3 (1) (2003) 1371–1382.

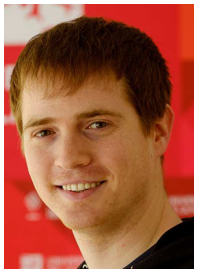
[23] A. Sanz-Garcia, J. Fernandez-Ceniceros, F. Antonanzas-Torres, A. Pernia-Espinoza, F.J. Martinez-de Pison, GA-PARSIMONY: A GA-SVR approach with feature selection and parameter optimization to obtain parsimonious solutions for predicting temperature settings in a continuous annealing furnace, Appl.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

[24] Shi, Y., Eberhart, R.: A modified particle swarm optimizer. In: 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360). pp. 69–73 (1998). DOI: 10.1109/ICEC.1998.699146.

[25] Shi, Y., Eberhart, R.: Empirical study of particle swarm optimization. In: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406). vol. 3, pp. 1945–1950 Vol. 3 (1999). DOI: 10.1109/CEC.1999.785511.

[26] Y. Tian, Y. Zhang, A comprehensive survey on regularization strategies in machine learning, Inf. Fusion 80 (C) (2022) 146–166, https://doi.org/10.1016/j.inffus.2021.11.005.

[27] 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, https://doi.org/10.1016/j.neucom.2016.08.154.
[28] 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, https://doi.org/10.1093/jigpal/jzx029.
[29] A. Vabalas, E. Gowen, E. Poliakoff, A. Casson, Machine learning algorithm validation with a limited sample size, PLoS One 14 (11) (2020), https://doi.org/10.1371/journal.pone.0224365.
[30] S.M. Vieira, L.F. Mendonza, G.J. Farinha, J.M. Sousa, Modified binary PSO for feature selection using SVM applied to mortality prediction of septic patients, Appl. Soft Comput. 13 (8) (2013) 3494–3504.
[31] Y. Wan, M. Wang, Z. Ye, X. Lai, A feature selection method based on modified binary coded ant colony optimization algorithm, Appl. Soft Comput. 49 (2016) 248–258.
[32] M. Wang, H. Chen, B. Yang, X. Zhao, L. Hu, Z. Cai, H. Huang, C. Tong, Toward an optimal kernel extreme learning machine using a chaotic moth-flame optimization strategy with applications in medical diagnoses, Neurocomputing 267 (2017) 69–84, https://doi.org/10.1016/j.neucom.2017.04.060.
[33] J. Wei, R. Zhang, Z. Yu, R. Hu, J. Tang, C. Gui, Y. Yuan, A BPSO-SVM algorithm based on memory renewal and enhanced mutation mechanisms for feature selection, Appl. Soft Comput. 58 (2017) 176–192.
[34] Zambrano-Bigiarini, M., Clerc, M., Rojas, R.: Standard particle swarm optimisation 2011 at cec-2013: A baseline for future PSO improvements. In: 2013 IEEE Congress on Evolutionary Computation. pp. 2337–2344 (2013). DOI: 10.1109/CEC.2013.6557848.



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.

Julio Fernández is PhD in Industrial Engineering since 2015. His research activities involve modelling, numerical simulations and soft computing methods applied to industrial processes, building and rubber products. He is currently working as CAE engineer at CMP Automotive group (Spain).

Andres Sanz-Garcia serves as Assistant Professor in the Mechanical Department at the University of Salamanca (USAL) since 2018. He is a senior member of the research group Laser Applications & Photonics (ALF) led by Prof. Luis Plaja and the Molecular Medicine group led by Prof. MD. Rogelio Gonzalez Sarmiento, both belonging to USAL. Andres is a visiting researcher at the Wyss Institute at Harvard University – Medical School (HMS) and the University of Helsinki (UH).

Alpha Pernia-Espinoza is Associate Professor of Universidad de La Rioja (Spain) since 2012. She is Industrial Engineer and PhD 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 modelling 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 PhD 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.