New Hybrid Methodology Based on Particle Swarm Optimization with Genetic Algorithms to Improve the Search of Parsimonious Models in High-Dimensional Databases

Jose Divasón, Alpha Pernia-Espinoza, and Francisco Javier Martinez-de-Pison

University of La Rioja, Spain
{jose.divason,alpha.pernia,fjmartin}@unirioja.es

Abstract. Our previous PSO-PARSIMONY methodology (a heuristic to search for accurate and low-complexity models with particle swarm optimization) shows a good balance between accuracy and complexity with small databases, but gets stuck in local minima in high-dimensional databases. This work presents a new hybrid methodology to solve this problem. First, we incorporated to PSO-PARSIMONY an aggressive mutation strategy to encourage parsimony. Second, a hybrid method between PSO and genetic algorithms was also implemented. With these changes, particularly with the second one, improvements were observed in the search for more accurate and low-complexity models in high-dimensional databases.

Keywords: PSO-PARSIMONY · Hybrid method · parsimonious modeling · auto machine learning · GA-PARSIMONY

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). GA-PARSIMONY [13] is a well-established methodology 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 and hospital energy demand. Moreover, previous comparisons with other existing AutoML methodologies (such as Auto-sklearn, H2O and MLJAR) demonstrated its effectiveness [13]. However, certain limitations of GA-PARSIMONY are also well-known: it requires a high number of individuals and also several repetitions of the method to guarantee an optimal solution (because it is unstable). It often converges to local minima instead of global minima. In addition, it sometimes generates twin individuals that do not contribute new information to the search process.

A first attempt to improve the results of GA-PARSIMONY 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 was named PSO-PARSIMONY [1] and improved GA-PARSIMONY on small databases, but was stuck in local minima on large databases. To solve this problem, this paper describes new proposals to improve PSO-PARSIMONY. On the one hand, PSO is combined with an aggressive mutation strategy to foster parsimonious models. On the other hand, a hybrid model between PSO and genetic algorithms is proposed, in which the particles with worse fitness are replaced in each iteration by new ones generated from typical genetic algorithm operations: selection, crossover and mutation. The accuracy and complexity of the new proposals are tested with public databases of different sizes and compared with GA-PARSIMONY.

2 Related work

The importance of HO and FS when solving a machine learning problem is well-known: they permit improving the predictive accuracy of algorithms. 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. [11] 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 [10], being inspired by the swarming behavior of salps when navigating and foraging in oceans. Other techniques related to animals include bat [15], glowworm [7] and bee colonies [5]. One of the most commonly used optimization techniques is the Particle Swarm Optimization (PSO), originally proposed by Kennedy and Eberhart [6]. There has been much research on this technique and numerous improvements have been proposed [16,14], 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 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. [2] 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 [3] operator. In [4] 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 [12], the standard crossover and mutation operations from GA are applied to PSO.

3 Previous PSO

In a previous work by the authors [1], the particle swarm optimization (PSO) was combined with a parsimony criterion to find parsimonious and, at the same time, accurate machine learning models. The PSO algorithm works by having a population (called a swarm) of possible solutions (called particles). The position of a particle is simply a vector $X = (H, F)$ where H corresponds to the values of model's hyperparameters and F is a vector with values between 0 and 1 for selecting the input features. These particles are moved around in the search-space of the combinational problem according to simple formulas:

$$V_i^{t+1} = \omega V_i^t + r_1\varphi_1 \times \left(pbest_i^t - X_i^t\right) + r_2\varphi_2 \times \left(lbest_i^t - X_i^t\right) \tag{1}$$

$$X_i^{t+1} = X_i^t + V_i^{t+1} \tag{2}$$

where V_i^t and X_i^t denote the velocity and position of the i-th particle in iteration t, respectively. Such formulas 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, $pbest_i$) and also by the experience of other particles (the best position within a neighborhood, $lbest_i$). 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. r_1 and r_2 are uniformly distributed random variables used to maintain the diversity of the swarm.

Our modified PSO 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, see [1] for further details. Algorithm 1 presents a pseudo-code version of it.

Experiments were conducted with various databases. A subset of them is shown in Table 1. PSO always improved accuracy with respect to GA. In databases with a low number of features, the difference with respect to parsimony was usually small (GA found solutions about 10% simpler). However, datasets with a larger number of features caused trouble for PSO, which found better solutions than GA but with twice as many features. Moreover, GA required much less computational effort: approximately, GA was three times faster and needed halving the iterations. This showed that PSO-PARSIMONY is a good alternative to GA-PARSIMONY if the number of features is relatively small, because it finds better solutions with a good balance between accuracy and parsimony.

Algorithm 1 Pseudo-code of the modified PSO algorithm

Initialization of positions $\mathbf{X^0}$ using a random and uniformly distributed Latin hypercube within the ranges of feasible values for each input parameter
Initialization of velocities according to $\mathbf{V^0} = \frac{random_{LHS}(s,\ D) - \mathbf{X^0}}{2}$
for $t = 1$ to T **do**
    Train each particle $\mathbf{X}_i$ and validate with CV
    Fitness evaluation and complexity evaluation of each particle
    Update $pbest_i$, $pbest_{p,i}$ and the $gbest$
    if early stopping is satisfied **then**
        return $gbest$
    end if
    Generation of new neighborhoods if $gbest$ did not improve
    Update each $lbest_i$
    Update positions and velocities according the formulas
    Mutation of features
    Limitation of velocities and out-of-range positions
end for
return gbest

4 New proposals

The previous section shows that PSO-PARSIMONY works well, but has room for improvement in terms of parsimony (especially for datasets with a large number of features) and computational time. The experiments also provided some insight on the accuracy and the evolution of parsimony in both GA and PSO; see also Figure 1 for further results with the crime database. Specifically, Figure 1a shows an exponential-like decrease in the number of features selected by GA in the first iterations, whereas PSO performs a more linear decrease. Figure 1b shows how good PSO is in terms of precision.

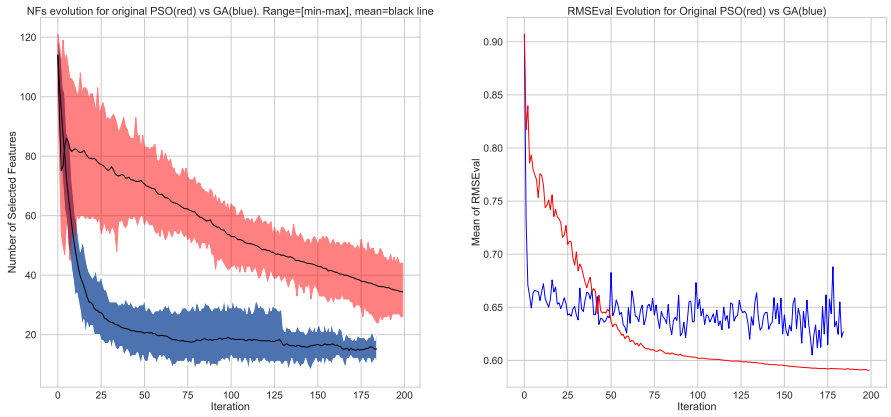
Table 1: PSO-PARSIMONY vs GA-PARSIMONY with 10 databases (results are the average of 5 runs with each methodology and $tol = 10^{-3}$).

Database	***#rows***	***#feats***	$\overline{\boldsymbol{PSO_J}}$	$\overline{\boldsymbol{GA_J}}$	$\overline{\boldsymbol{PSO}}_{\boldsymbol{N_{FS}}}$	$\overline{\boldsymbol{GA}}_{\boldsymbol{N_{FS}}}$	$\overline{\boldsymbol{PSO}}_{\boldsymbol{time}}$	$\overline{\boldsymbol{GA}}_{\boldsymbol{time}}$	$\overline{\boldsymbol{PSO}}_{\boldsymbol{iters}}$	$\overline{\boldsymbol{GA}}_{\boldsymbol{iters}}$
strike	625	7	**0.83856**	0.86479	**1.8**	3.0	105.1	**16.3**	88.0	**39.6**
no2	500	8	**0.65608**	0.66007	6.0	6.0	59.8	**17.5**	102.8	**46.4**
concrete	1030	9	**0.28943**	0.29526	**7.4**	7.8	468.0	**160.9**	107.2	**41.6**
housing	506	14	**0.31261**	0.32559	11.0	**10.0**	215.8	**74.6**	104.0	**61.0**
bodyfat	252	15	**0.10709**	0.10806	3.4	**2.0**	97.3	**40.2**	128.2	**69.2**
cpu_act	8192	22	**0.12405**	0.12473	14.8	**13.4**	1241.7	**788.3**	84.6	**50.0**
bank	8192	33	**0.63420**	0.63792	23.6	**19.8**	1298.8	**629.1**	177.8	**101.6**
puma	8192	33	**0.18049**	0.18097	4.6	**4.2**	2188.6	**1028.4**	96.0	**51.2**
ailerons	13750	41	**0.38228**	0.38258	17.8	**10.4**	2663.2	**1144.7**	115.2	**65.6**
crime	2215	128	**0.59336**	0.59565	49.8	**19.8**	797.8	**410.5**	185.6	**143.6**

To increase parsimony in PSO, two new variants of the algorithm are proposed in this section. The main goal is to promote the parsimonious behavior of GA in PSO. The first is a straightforward variant in which the mutation phase in PSO is performed exactly as in GA. The second is a hybrid approach between GA and PSO. They have been implemented in Python and are available at https://github.com/jodivaso/Hybrid-PSOGAParsimony.

4.1 PSO with a new mutation

PSO-PARSIMONY already includes a mutation operator, for which the mutation rate was set to $1/D$ by default, where D is the dimensionality of the problem. In contrast, GA has a much more aggressive strategy; it also performs uniform random mutation, but three parameters are involved: `pmutation` represents the percentage of parameters 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 best individuals that will not be muted. Note that `not_muted` prevents losing the best individuals in the mutation step. The default values are set to 0.1, 0.1 and 3, respectively. With these default values, GA approximately excludes 9% of the features in each iteration. In contrast, only $1/2D$ of the features are excluded in each step with PSO, which are very few if the dataset has many variables. This explains why the PSO algorithm performs worse in terms of parsimony with high-dimensional databases. To solve this problem, Algorithm 1 was modified in Line 13 to include the mutation step as is done in GA.



(a) Features evolution in GA and PSO (b) Evolution of RMSE in GA and PSO

Fig. 1: PSO-PARSIMONY (red) vs. GA-PARSIMONY (blue) in terms of number of selected features and accuracy with the crime dataset.

4.2 Hybrid method: PSO with crossover and mutation

To further encourage parsimony in PSO and make its behavior closer to GA, especially in the first iterations, a hybrid model was developed. For this purpose, Algorithm 1 was modified in several parts.

A crossover phase is added just after calculating the local bests of the neighborhoods (i.e., after Line 11). 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 [8]. 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 [9] 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 array to indicate 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 formulas. In this case, this step is only applied to the particles that have not been substituted by the crossover. The mutation phase is also modified to include the changes proposed in the previous subsection, i.e., a more aggressive mutation strategy to encourage parsimony. The rest of steps of the algorithm are preserved.

5 Experiments

In order to test the capacity of the proposed methodologies to find accurate and parsimonious models, databases with a high number of features were selected. In particular, experiments compared the PSO-PARSIMONY method with the new mutation (New-PSO) and the hybrid HYB-PARSIMONY (HYB) against GA-PARSIMONY (GA) and the previous PSO method (Old-PSO).

All experiments were similar to previous works with a population size of $P = 40$, $tol = 0.001$, a maximum number of generations of $G = 200$, and an early stopping of 35. Experiments were implemented in 9 separately 24-core servers from the Beronia Cluster at the University of La Rioja.

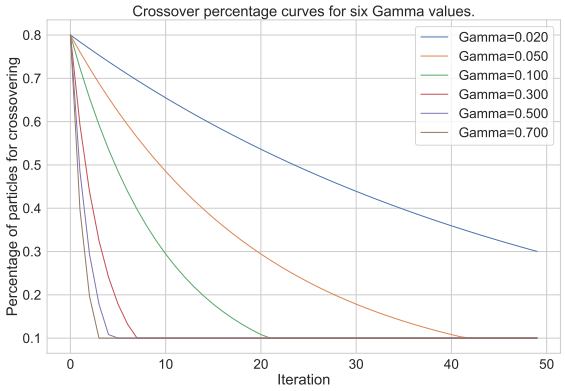


Fig. 2: Example of six curves created with different Γ values to establish the percentage of individuals to be replaced by crossover in each iteration.

For the hybrid method, the following equation was defined to calculate the percentage of particles to be substituted by crossover in each iteration $iter$:

$$\%particles = max(0.80 \cdot e^{(-\Gamma \cdot iter)}, 0.10) \tag{3}$$

Figure 2 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.

Table 2 presents the results with the *crime* database 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}.

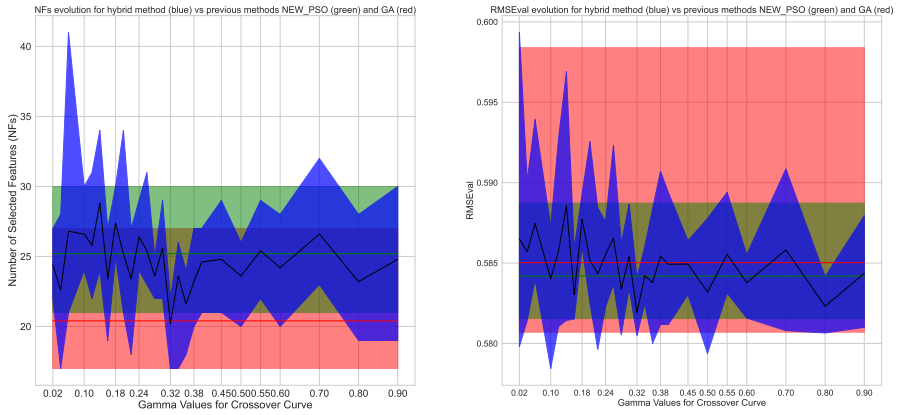
Figure 3 shows in blue the range (minimum and maximum) and in a solid black line the mean value of N_{FS} (left) and J (right) for five runs of the algorithm with different values of Γ for the database *crime*. It also includes the range and

Table 2: Hybrid with different Γ vs previous methods for *crime* database.

Method	Γ	J_{best}	N_{FSbest}	$\overline{J}$	$\overline{N_{FS}}$	$\overline{iters}$	$\overline{time}$
GA	0.00	0.58070	19	0.58503	20.4	146.8	86.2
OLD PSO	0.00	0.58333	29	0.58773	33.0	200.0	121.0
NEW PSO	0.00	0.58155	26	0.58419	25.2	362.8	211.9
HYB	0.02	0.57981	22	0.58650	24.4	229.6	132.0
HYB	0.04	0.58142	**17**	0.58571	22.6	200.8	118.8
HYB	0.06	0.58397	23	0.58747	26.8	206.6	119.1
HYB	0.10	**0.57844**	24	0.58402	26.6	200.4	115.6
HYB	0.12	0.58106	24	0.58576	25.8	184.4	97.6
HYB	0.14	0.58143	24	0.58856	28.8	190.0	109.9
HYB	0.16	0.58151	19	0.58304	23.4	228.8	121.5
HYB	0.18	0.58603	26	0.58773	27.4	148.6	85.9
HYB	0.20	0.58251	23	0.58517	25.2	185.6	111.0
HYB	0.22	0.57964	23	0.58434	23.4	241.4	138.9
HYB	0.24	0.58229	25	0.58554	26.4	167.4	98.0
HYB	0.26	0.58368	23	0.58656	25.4	176.0	101.2
HYB	0.28	0.58054	24	0.58340	23.6	235.6	135.0
HYB	0.30	0.58343	29	0.58540	25.6	**143.8**	**82.7**
HYB	0.32	0.58050	22	**0.58193**	**20.2**	242.2	139.1
HYB	0.34	0.58247	**17**	0.58421	23.6	233.2	123.5
HYB	0.36	0.58001	23	0.58378	21.6	221.6	127.4
HYB	0.38	0.58119	20	0.58544	23.2	197.2	117.9
HYB	0.40	0.58117	27	0.58493	24.6	209.8	120.6
HYB	0.45	0.58304	22	0.58494	24.8	176.8	101.9
HYB	0.50	0.57938	24	0.58319	23.6	213.4	123.0
HYB	0.55	0.58314	24	0.58555	25.4	193.8	111.5
HYB	0.60	0.58158	22	0.58378	24.2	218.2	115.8
HYB	0.70	0.58080	24	0.58582	26.6	195.8	116.5
HYB	0.80	0.58065	19	0.58232	23.2	215.0	123.7
HYB	0.90	0.58101	25	0.58437	24.8	187.4	108.5

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. Figure 4 shows that the hybrid method reduced the number of features more drastically and converged earlier than the New-PSO method, similar than GA method. With respect to J, Figure 3b 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, Figure 5 presents the mean values of N_{FS} and J for the four methods and with the *crime* database. 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.



(a) N_{FS} evolution with different Γ values (b) J evolution with different Γ values

Fig. 3: Comparison between HYBRID-PARSIMONY (blue) vs. PSO-PARSIMONY (green) and GA-PARSIMONY (red) in terms of the number of selected features and accuracy with the *crime* dataset.

Table 3: NEW PSO-PARSIMONY vs HYBRID-PARSIMONY with a population size of $P = 40$ and $tol = 0.001$ (results are the average of the 5 runs).

$\boldsymbol{Dataset}$	$\boldsymbol{\#rows}$	$\boldsymbol{\#feats}$	$\boldsymbol{\Gamma}$	$\overline{\boldsymbol{PSO}}_{\boldsymbol{J}}$	$\overline{\boldsymbol{HYB}}_{\boldsymbol{J}}$	$\overline{\boldsymbol{PSO}}_{\boldsymbol{N_{FS}}}$	$\overline{\boldsymbol{HYB}}_{\boldsymbol{N_{FS}}}$	$\overline{\boldsymbol{PSO}}_{\boldsymbol{time}}$	$\overline{\boldsymbol{HYB}}_{\boldsymbol{time}}$
slice	5000	379	0.34	0.0238	**0.0231**	146.8	**132.2**	819.4	**609.0**
blog	4999	277	0.70	0.4087	**0.3983**	127.6	**113.8**	1117.5	**1051.6**
crime	2215	128	0.32	0.5842	**0.5819**	25.2	**20.2**	211.9	**139.1**
tecator	240	125	0.50	0.0331	0.0331	55.0	**48.6**	11.9	**8.7**
ailerons	5000	41	0.70	0.3947	**0.3934**	10.6	**10.2**	473.4	**466.1**
bank	8192	33	0.50	0.6514	**0.6511**	21.4	21.4	2146.4	**1536.6**
puma	8192	33	0.50	0.1817	0.1817	4.0	4.0	1063.8	**933.2**

Similar results can be observed with other high-dimensional databases. Tables 3 and 4 show respectively the average results and the best model obtained with the Hybrid Method and the New-PSO. In almost all databases, the hybrid method obtained more accurate models with less complexity, although it was necessary to find a suitable Γ value.

6 Conclusions

This paper presents two new proposals to improve our previous PSO-PARSIMONY methodology for the simultaneous search of the best model hyperparameters and

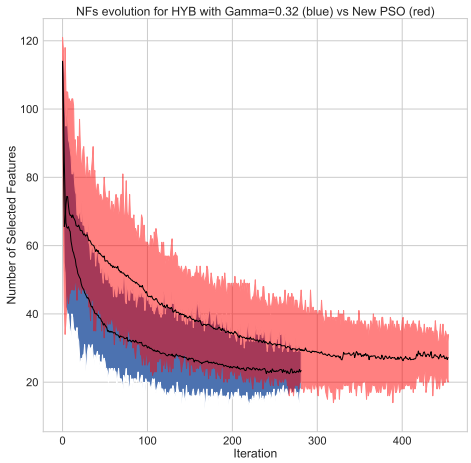
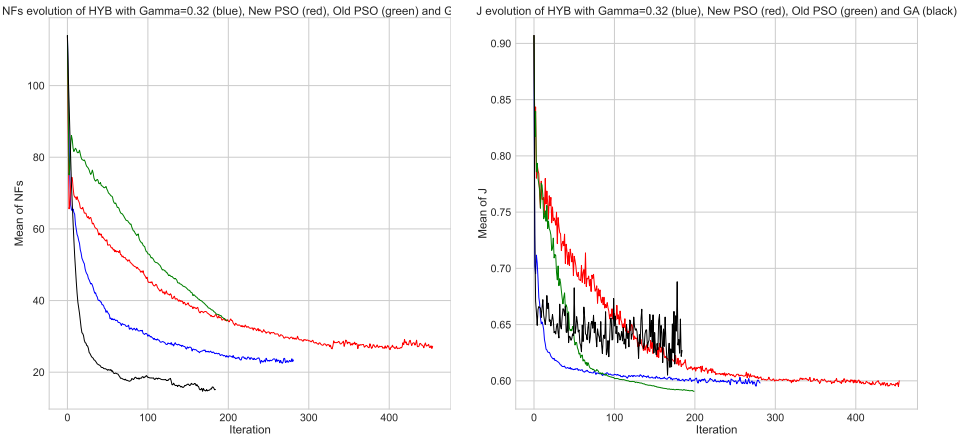


Fig. 4: Comparison of N_{FS} evolution between the Hybrid method (blue) (with $\Gamma = 0.32$) and the new PSO-PARSIMONY (red).



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

Fig. 5: Comparison with *crime* dataset of $\overline{N_{FS}}$ and $\overline{J}$ between the four methods: the Hybrid method (blue) (with $\Gamma = 0.32$), the new PSO-PARSIMONY (red), the old PSO-PARSIMONY (green) and GA-PARSIMONY (black).

Table 4: Best individual obtained with PSO-PARSIMONY vs HYBRID-PARSIMONY using a population size of $P = 40$ and $tol = 0.001$.

Dataset	Γ	PSO_J	HYB_J	$PSO_{J_{TST}}$	$HYB_{J_{TST}}$	$PSO_{N_{FS}}$	$HYB_{N_{FS}}$	PSO_{time}	HYB_{time}
slice	0.70	0.0228	**0.0218**	**0.0012**	0.0017	124	**112**	1050.1	**627.5**
blog	0.38	0.3948	**0.3879**	0.2523	**0.2023**	**115**	129	1304.0	**1277.4**
crime	0.10	0.5815	**0.5784**	0.5021	**0.4780**	26	**24**	263.5	**138.5**
tecator	0.38	0.0328	**0.0327**	0.0207	**0.0206**	**48**	51	16.3	**10.7**
ailerons	0.15	0.3935	**0.3922**	**0.3675**	0.3698	13	**10**	**484.2**	494.3
bank	0.70	0.6510	**0.6507**	**0.5839**	0.5865	22	**21**	2428.5	**1675.0**
puma	0.38	0.1817	0.1817	0.1776	0.1776	4	4	1191.8	**712.9**

input features, with a balance between accuracy and complexity. Concretely, a PSO with an aggressive mutation strategy as well as a hybrid method have been implemented. The main novelty relies on the hybrid model between GA and PSO, where the optimization is based on the PSO formulas, but the common genetic operations of selection, crossover and mutation are included to replace the worst particles. The percentage of variables to be substituted in each iteration can be customized. In this work, functions that depend on a Γ parameter have been used to promote parsimony in the first iterations (a high percentage of particles is substituted), but in further iterations the percentage is decreased. This differs from other hybrid methods where the crossover is applied between each particle's individual best position or other approaches where the worst particles are also substituted by new ones, but at extreme positions. Experiments show that, in general and once the appropriate gamma is fixed, this HYB-PARSIMONY methodology allows one to obtain better, more parsimonious and more robust models compared to our previous PSO-based methodology and the PSO with mutation. The computational effort is also reduced, since it requires less time.

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.

Acknowledgements We are greatly indebted to *Banco Santander* for the REGI2020/41 fellowship. This study used the Beronia cluster (Universidad de La Rioja), which is supported by FEDER-MINECO grant number UNLR-094E-2C-225. The work is also supported by grant PID2020-116641GB-I00 funded by MCIN/ AEI/ 10.13039/501100011033.

References

1. Ceniceros, J.F., Sanz-Garcia, A., Pernia-Espinoza, A., Martinez-de Pison, F.J.: PSO-PARSIMONY: A new methodology for searching for accurate and parsimonious models with particle swarm optimization. application for predicting the force-displacement curve in t-stub steel connections. In: Sanjurjo González, H.,

Pastor López, I., García Bringas, P., Quintián, H., Corchado, E. (eds.) Hybrid Artificial Intelligent Systems. pp. 15–26. Springer, Cham (2021)
2. Chuang, L.Y., Tsai, S.W., Yang, C.H.: Improved binary particle swarm optimization using catfish effect for feature selection. Expert Syst. Appl. **38**(10), 12699–12707 (2011). https://doi.org/10.1016/j.eswa.2011.04.057
3. Engelbrecht, A.P.: Particle swarm optimization with crossover: A review and empirical analysis. Artif. Intell. Rev. **45**(2), 131–165 (2016). https://doi.org/10.1007/s10462-015-9445-7
4. Hao, Z.F., Wang, Z.G., Huang, H.: A particle swarm optimization algorithm with crossover operator. In: 2007 International Conference on Machine Learning and Cybernetics. vol. 2, pp. 1036–1040 (2007)
5. Karaboga, D., Basturk, B.: Artificial bee colony (abc) optimization algorithm for solving constrained optimization problems. In: Melin, P., Castillo, O., Aguilar, L.T., Kacprzyk, J., Pedrycz, W. (eds.) Foundations of Fuzzy Logic and Soft Computing. pp. 789–798. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)
6. 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). https://doi.org/10.1109/ICNN.1995.488968
7. Marinaki, M., Marinakis, Y.: A glowworm swarm optimization algorithm for the vehicle routing problem with stochastic demands. Expert Systems with Applications **46**, 145–163 (2016). https://doi.org/10.1016/j.eswa.2015.10.012
8. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs, Third Revised and Extended Edition. Springer (1996)
9. Michalewicz, Z., Janikow, C.Z.: Handling constraints in genetic algorithms. In: Icga. pp. 151–157 (1991)
10. Mirjalili, S., Gandomi, A.H., Mirjalili, S.Z., Saremi, S., Faris, H., Mirjalili, S.M.: Salp swarm algorithm: A bio-inspired optimizer for engineering design problems. Advances in Engineering Software **114**, 163–191 (2017). https://doi.org/10.1016/j.advengsoft.2017.07.002
11. Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey wolf optimizer. Advances in Engineering Software **69**, 46–61 (2014). https://doi.org/10.1016/j.advengsoft.2013.12.007
12. Nazir, M., Majid-Mirza, A., Ali-Khan, S.: PSO-GA based optimized feature selection using facial and clothing information for gender classification. Journal of Applied Research and Technology **12**(1), 145–152 (2014). https://doi.org/10.1016/S1665-6423(14)71614-1
13. Martinez-de Pison, F.J., Ferreiro, J., Fraile, E., Pernia-Espinoza, A.: A comparative study of six model complexity metrics to search for parsimonious models with GAparsimony R Package. Neurocomputing **452**, 317–332 (2021). https://doi.org/10.1016/j.neucom.2020.02.135
14. Shami, T.M., El-Saleh, A.A., Alswaitti, M., Al-Tashi, Q., Summakieh, M.A., Mirjalili, S.: Particle swarm optimization: A comprehensive survey. IEEE Access **10**, 10031–10061 (2022). https://doi.org/10.1109/ACCESS.2022.3142859
15. Yang, X.S.: A new metaheuristic bat-inspired algorithm. In: Nature inspired cooperative strategies for optimization (NICSO 2010), pp. 65–74. Springer (2010)
16. Zhang, Y., Wang, S., Ji, G.: A comprehensive survey on particle swarm optimization algorithm and its applications. Mathematical Problems in Engineering pp. 1–38 (2015). https://doi.org/10.1155/2015/931256