Evaluation of a novel GA-based methodology for model structure selection: the *GA-PARSIMONY*

R. Urraca[a], E. Sodupe-Ortega[a], J. Antonanzas[a], F. Antonanzas-Torres[a], F.J. Martinez-de-Pison[a,*]

[a]*EDMANS Group, Department of Mechanical Engineering, University of La Rioja, Logroño, Spain.*

Abstract

Most proposed metaheuristics for feature selection and model parameter optimization are based on a two-termed $Loss + Penalty$ function. Their main drawback is the need of a manual set of the parameter that balances between the loss and the penalty term. In this paper, a novel methodology referred as the *GA-PARSIMONY* and specifically designed to overcome this issue is evaluated in detail in thirteen public databases with five regression techniques. It is a GA-based meta-heuristic that splits the classic two-termed minimization functions by making two consecutive ranks of individuals. The first rank is based solely on the generalization error, while the second (named *ReRank*) is based on the complexity of the models, giving a special weight to the complexity entailed by large number of inputs.

For each database, models with lowest testing RMSE and without statistical difference among them were referred as *winner* models. Within this group, the number of features selected was below 50 %, which proves an optimal balance between error minimization and parsimony. Particularly, the most complex algorithms (MLP and SVR) were mostly selected in the group of *winner* models, while using around 40-45 % of the available attributes. The most basic IBk, ridge regression (LIN) and M5P were only classified as *winner* models in the simpler databases, but using less number of features in those cases (up to a 20-25 % of the initial inputs).

Keywords: Genetic Algorithms, Parameter Tuning, Feature Selection, Parsimony Criterion, Model Comparative

*Corresponding author
Email address: `fjmartin@unirioja.es` (F.J. Martinez-de-Pison)

1. Introduction

The selection of a good overall model, with optimal generalization ability but with a reduced number of features, has multiple advantages for its implementation in real-world applications. The identification of the most relevant input variables facilitates the understanding of the problem being studied, and it generates more robust models against perturbations, noise and missing values. In this line, a reduction in the number of inputs has a positive impact on the human and economic efforts required for data acquisition and preprocessing. For instance, in environmental applications, it involves cutting down on costs in data acquisition systems as well as reducing the time to analyze and process the information. Finally, the development of less complex models significantly simplifies upcoming stages such as re-calibration and exploiting, and mitigates the well known overfitting issues.

One of the most frequent approaches to tackle overfitting is the use of regularization. This strategy has been included in the training stage of many machine learning algorithms, and it consists in minimizing a $Loss + Penalty$ function [1]:

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{minimize}}\{L(\mathbf{X},\mathbf{y},\beta) + \lambda P(\beta)\} \tag{1}$$

where $L(\mathbf{X},\mathbf{y},\beta)$ is the loss function that evaluates the performance of the model trained (β) given a set of input variables ($\mathbf{X}$) and an outcome ($\mathbf{y}$), and $P(\beta)$ is the penalty function that is related to the complexity of the model. Finally, λ is a non-negative parameter that balances cost and penalty terms in order to control the bias-variance trade-off. This type of regularization strategy is used by multiple methods such as ridge regression (L_2 penalty), LASSO (L_1 penalty), SVM (cost parameter) or ANNs (weight decay). In most of these methods, λ along with other secondary parameters are tuned with some classic optimization algorithms such as grid search (GS) or random search (RS). These optimization methods are combined with some resampling techniques such as k-fold Cross-Validation (CV) or Bootstrap to ensure a final model with adequate generalization ability. However, a second validation procedure is still required if other external parameters need to be optimized, such is the case of the number of features and coefficients involved in the data transformation process. This second validation procedure, performed among the best models from the first stage, must be again based on both criteria (generalization capability and complexity).

Soft computing (SC) appears as an effective alternative to reduce the computational and human cost of this task compared against the classic approaches [2, 3, 4, 5, 6, 7, 8, 9]. In the last years, several authors have reported the use of SC strategies for the model selection process, combining feature selection (FS) and parameter tuning (PT) to generate models with good generalization capabilities [10, 11, 12]. For instance, Huang and Chang [13] combined genetic algorithms (GAs) with k-fold cross-validation (CV) for FS and tuning of Support Vector Machines (SVM) in order to improve microarray classification. Vieira et al. [14] used binary particle swarm optimization (PSO) to tune a wrapper approach with SVM to predict whether a patient with septic shock survived or deceased. Ahila et al. [15] modified the PSO method to perform FS and tuning of Extreme Learning Machines (ELM) in a power system disturbances classification problem. Dhiman et al. [16] designed a hybrid approach with wavelet packet decomposition and a GA-SVM scheme for FS and MPO to obtain classification models capable of detecting epileptic seizures from background electroencephalogram signals. Castillo et al. [17, 18] used ant colony optimization (ACO) to adjust different membership functions of complex fuzzy controllers. Winkler et al. [19] used different evolutionary strategies to perform FS and to optimize linear models, k-nearest neighbors (k-NN), ANNs and SVM with the final purpose of identifying tumor markers. Sanz-García et al. [20] proposed a GA-based optimization method to create better overall parsimonious ANNs for predicting set points in an steel annealing furnace. Ding [21] used PSO for selecting spectral bands and optimizing SVM parameters in remote sensing.

The main objective of these works is to generate models with the lowest generalization error while maintaining the overall parsimony, which mainly concerns to the number of variables retained as inputs. However, most of these studies include an optimization via a classic two-termed $Loss + Penalty$ function that requires to set the penalty parameter (Λ). This Λ is similar to the aforementioned λ, but here is used to compare models instead of comparing variations of the same model. Hence, its value has to be manually set prior the execution of the optimization methodology. In this context, we introduced a new GA-based optimization methodology, named *GA-PARSIMONY* [22]. Our aim is to automate the optimization process when the complexity of the model is taken into account by getting rid of the penalty parameter Λ. To do so, we break the traditional $Loss + penalty$ optimization function by making two consecutive ranks of the individuals. First, individuals are ranked according to a loss term (k-fold CV error). Next, the position of individuals with no significant difference in their loss functions is modified based on the complexity of the models (process hereafter referred as *ReRank*). The complexity evaluation accounts for both, the inner complexity of the model and the number of features retained. Therefore, the methodology conducts the tuning of model parameters and feature selection at a time, while boosting the selection of parsimonious models. The methodology has been already successfully applied for predicting set points in industrial processes [23, 20, 24], for solar energy modeling [25, 26, 27] and for structure engineering [28] among other applications. When compared against other optimization methods, the obtained models proved to have similar generalization errors while using a lower number of inputs. The main goal of this work is to perform a more detailed analysis of the GA-PARSIMONY methodology by testing it into five well-known regression methods with different population sizes and public databases.

The remainder of this paper is organized as follows. GA-PARSIMONY methodology is presented in Section 2. The design of the experiments to evaluate the methodology is detailed in Section 3. The different regression techniques used are introduced, as well as the public databases and metrics used for evaluation. Numerical results obtained are presented and discussed in Section 4 and the conclusions drawn are shown in Section 5.

2. *GA-PARSIMONY* methodology

The objective of the methodology is to automate the model structure selection process. Specifically, feature selection and parameter tuning are simultaneously conducted in order to obtain accurate but parsimonious models. The methodology is referred by authors as GA-PARSIMONY [22], as it combines the traditional GA structure (see Figure 1) for FS and PT, with the selection of parsimonious models. Here, the main novelty compared to existing proposals is the elimination of the penalty parameter from the fitness function. The procedure begins with the definition of the initial population Λ_0.

$$\Lambda_0 : \{\lambda_0^1, \lambda_0^2, ..., \lambda_0^P\} \tag{2}$$

Hybrid chromosomes λ_g^i are used to select features and tune model parameters. The chromosomes are composed of two different entities: a binary coded vector, with the selected features as inputs to the predictive technique, and a real coded part, with the numerical values of the tuning parameters of the model. The first generation is created via Latin Hypercube Sampling (LHS) [29], a technique that generates a population with enough diversity in the search space and accelerates the convergence process.

The predictive technique is calibrated for each individual following the specifications (tuning parameters and input features) of its chromosome. Next, the generalization ability of each model is evaluated. This evaluation process is conducted by using m repeated k-fold cross validation to prevent overfitting [30]. Here, different metrics to evaluate the performance of models can be used, though MAE and RMSE are the most widespread ones.

$$J(\lambda_g^i) = \frac{\sum_{i=1}^{k \times m} error_i}{k \times m} \tag{3}$$

Once the fitness function J for all individual λ_g^i of the population Λ_g is computed, models obtained are sorted according to their fitness function:

$$\Lambda_g^J \leftarrow sort(J(\Lambda_g)) \tag{4}$$

Instead of including a complexity penalty term in the fitness function J, the first rank of the individuals is modified based on the complexity of the models. This process is referred as the *ReRank*.

$$\Lambda_g^s \leftarrow ReRank(\Lambda_g^J) \tag{5}$$

The *ReRank* algorithm works as follows (see Algorithm 1). Each model is compared against its predecessor starting from the top of the initial rank (based on Js). First, a statistical test is conducted to determine if a significant difference between their Js exists. Only in the case of being statistically equivalent, the complexity of both models is evaluated. If the first model is more complex than the latter, they swap their positions. The statistical test used is the Wilcoxon Signed-Ranked test [31]. The complexity of the models is evaluated with a expression that combines the number of features with the inner complexity of the predictive algorithm:

$$Complexity = 10^6 N_{FS} + C_{model} \tag{6}$$

where N_{FS} is the number of inputs and C_{model} is the internal model complexity, which depends on the regression algorithm. This expression is designed to give priority to the N_{FS} term. The complexity of the models is first evaluated in terms of the number of inputs. Only in the case of having two models with the same number of inputs, the inner complexity of the model is taken into account. Mathematically, this is accomplished by weighting the N_{FS} with a value high enough (10^6) and by setting an upper limit of 999999 for C_{model}. The indexes of the first chromosomes being compared are sequentially incremented up to the last element. The index of the second chromosome is incremented alone in the case of finding two individuals with no significant difference in their Js.

Algorithm 1 *ReRank*

```
1: input $G(J, Model - Complexity)$ : Individuals sorted by J
2: const $NUMINDIV = cte.; alpha = cte.$
3: var $PosFirst, PosSecond$ : 0..$NUMINDIV$
4: Begin
5:     $PosFirst \leftarrow 0$
6:     repeat
7:         $PosFirst \leftarrow PosFirst + 1$
8:         repeat
9:             $PosSecond \leftarrow PosFirst + 1$
10:            $p - value \leftarrow$ test $(G[PosFirst](J), G[PosSecond](J))$
11:            if $p - value > alpha$ AND $G[PosSecond](Size) < G[PosFirst](Size)$ then
12:                swap($G[PosFirst], G[PosSecond]$)
13:            end if
14:        until $p - value \leq alpha$ OR $PosSecond = NUMINDIV$
15:    until $PosFirst = NUMINDIV - 1$
16: End
```

Based on the modified rank obtained after applying the *ReRank* algorithm, best individuals are kept as parents for the next generation. The number of individuals selected as parents for the next generation is set by means of the elitism percentage x_e.

$$\Lambda_g[1 : P_e] \leftarrow select(\Lambda^e_{g-1}) \quad \text{with} \quad x_e \tag{7}$$

where Λ^e_{g-1} are the elitist individuals in $g - 1$ and P_e is the number of elitist individuals.

Then, couples of chromosomes for mating are selected following different approaches, from uniform selection to more advanced methods such as roulette or tournament. Each couple of parents produce two offsprings, and different mating methods can be chosen to generate the offsprings, such as crossover or blending [32] among others. Finally, the chromosomes of the new generation are randomly mutated to maintain the genetic diversity of population.

$$\Lambda_g[1:P] \leftarrow mutation(\Lambda_g[1:P]) \quad \text{with} \quad x_m \tag{8}$$

where x_m is the mutation rate, i.e., the percentage of total bits in the boolean part or digits in the numeric part mutated. The two best individuals are never muted. This procedure is repeated until the maximum number of generations G is reached.

3. Experimental

The GA-PARSIMONY methodology was evaluated in different scenarios, using five well-known regression algorithms and thirteen databases retrieved from public repositories.

3.1. Regression Schemes

Five of the currently most representative regression techniques were selected:

- MLP [33]: The Multi-Layer Perceptron (MLP) is the most common version of feed-forward artificial neural networks. The *Broyden-Fletcher-Goldfarb-Shanno* (*BFGS*) algorithm was selected. It is a more robust approach compared to the basic methods, and it is designed to avoid falling into local minima. Two parameters were tuned during the training process; the number of neurons in the hidden layer and the ridge parameter. The latter determines the penalty imposed due to the size of the weights in the training process.
- SVR [34]: Support Vector Regression (SVR) is the implementation of the well-known support vector machines (SVM) for regression tasks. It is actually one of the most used models since it is able to deal with non-linear situations thanks to the so-called 'kernel trick'. Besides, the technique is able to avoid local minimum values, providing high generalization capacity. The kernel function selected was the radial basis function (RBF). The setting

parameters were the penalty coefficient or cost C, which balances between error minimization and complexity, the γ of RBF kernel, a parameter which controls the width of the Gaussian function, and the insensitive loss parameter ϵ, which controls the number of support vectors.

- LIN [35]: Ridge regression is a classic variation of linear regression based on the Tikhonov regularization criterion. It introduces a L2 penalty to deal with ill-conditioned matrices, improving the robustness of the naive linear regression. The only parameter tuned was the *ridge* parameter, which controls the amount of regularization.

- IBk [36]: The IBk algorithm is an implementation of the k-nearest neighbors method (kNN) for regression. The outcome of the IBk model is the average or weighted average value of the closest neighbors. The tuning parameters for the IBk are the number of the nearest neighbors K and the type of weighting distance used.

- M5P [37]: The M5P algorithm is a conventional decision tree with linear regression models at the leaves. It is based on the M5 algorithm introduced by Quinlan [37] and later enhanced by Wang and Witten [38]. The tuning parameter was the minimum number of instances per leaf M.

3.2. GA-PARSIMONY *settings*

A real-coded chromosome was used with a total of $n + m$ values that include the n tuning parameters and a boolean array of m elements that correspond to the available inputs for the model. If the attribute is included in the model, the corresponding element of m is set to 1. The length of m depends on the number of features (dimension) of the database being used, while the number of tuning parameters n depends on the regression algorithm selected (see Table 1).

Data was normalized between 0 and 1 and then split into a training-validation set, to implement the methodology, and a testing set, to externally validate its accuracy. The normalized root mean squared validation error ($RMSE_{val}$) was the metric selected for the fitness function J. The validation procedure implemented was 5×2-fold CV (2 folds with 5 repetitions).

The Wilcoxon Signed Rank test was used for the statistical comparisons between Js in the

ReRank algorithm, with a significance level of $\alpha = 0.05$. The internal complexity of each predictive technique (C_{model}) was obtained with the analytic expressions shown in Table 1.

The selection strategy implemented was *random uniform* with an elitism percentage of 20%. The mating method used was *heuristic blending* [39], which is based on the following equation:

$$p_{new} = \beta\,(p_{mn} - p_{dn}) + p_{mn} \tag{9}$$

where p_{mn} and p_{dn} are the n_{th} variable in parent chromosomes, p_{new} is the new single offspring variable and β a random number in the range $[-0.1, 1.1]$. Finally, a mutation percentage of 10% was applied to all the experiments.

Experiments were carried with different population sizes (8, 16, 32, 48, 65), while the maximum number of generations was kept constant ($G = 40$).

3.3. Data

The described methodology was implemented in thirteen benchmark databases retrieved from public repositories: `UCI`[40] and `StatLib`[41]. Databases were selected to cover different regression scenarios, regarding the number of attributes and samples (see Table 2).

3.4. Evaluation

The performance of the different models trained (4 regression techniques, 13 datasets, and 6 population sizes) was evaluated based on the Root Mean Squared Error ($RMSE$):

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2} \tag{10}$$

Databases were normalized between 0 and 1 prior to training, so results could be evaluated in percentage terms. Subsequently, each database was split into a training set (80 % of samples) and a testing set (20 % of samples). The training set was used to calibrate the different models using the GA-PARSIMONY with 5×2-fold CV. A validation error ($RMSE_{val}$) was obtained from the calibration process, but still a external validation metric was computed with the testing set. This testing error was calculated for each model in each run of the 5×2-fold CV, so 5 testing values were available per model. Therefore, the testing error was reported in terms of its mean ($RMSE_{tst}$) and its standard deviation ($RMSE_{tst}^{sd}$).

For each database, models based on the different regression techniques and population sizes were ranked according to the average $RMSE_{tst}$. The model with the lowest error was initially considered the best or *winner* model. Then, this best model was statistically compared against the others using the Wilcoxon Signed-Ranked test with $\alpha = 0.05$. Models not showing significant difference with the best model were also included in the group of *winner* models. Another metric used to compare the model with lowest testing error and the rest was the $DRMSE_{tst,i}$, which is the difference between the $RMSE_{tst}$ of the model being studied (i) and the model with the lowest testing error:

$$DRMSE_{tst,i} = RMSE_{tst,i} - RMSE_{tst,best} \tag{11}$$

3.5. *Software*

All experiments were run in the free statistical software `R`[42]. The following packages were used to implement the different regression techniques: `e1071` [43] for the SVR and `RWeka`[44] for the remaining techniques in order to import `Weka` algorithms[35, 45] to `R`. All computations were run in a dual quad-core opteron server (Intel ®Xeon ®CPU E5410 @ 2.33 GHz).

4. Results and Discussion

Table 3 summarizes the results of the models with lowest $RMSE_{tst}$ for each database. Despite the fact that results are not comparable in terms of $RMSE$ due to the differences between databases, it is interesting to highlight that lowest errors were obtained with SVR and MLP in ten out of thirteen databases. In *triazines* and *wisconsin*, two databases with a high number of attributes, $M5P$ and LIN generated the lowest errors, while IBk was the best performing algorithm for *meta*.

The GA-PARSIMONY methodology succeeded in reducing the number of inputs, as eight out of thirteen models used less than the 50 % of available attributes. It has to be note that these high reduction ratios were obtained for the models with lowest testing error, which proves that the methodology is able to minimize the prediction error while still developing parsimonious models. The reduction in the number of features was more striking in databases with a high number of initial features (*tecator*, *puma*, *triazines*).

Models were ranked according to the $RMSE_{tst}$ in order to compare algorithms through different databases. Fig. 2 shows the different rankings of the models obtained for the thirteen databases. The plot on the left depicts that MLP and SVR generally coped the first positions in the rank when $popsize \geq 32$. For any $popsize$ setting, the medians of both techniques were below the first quartile of the remaining algorithm, which means that SVR and MLP obtained best ranking in more than 50% of databases. The MLP performed significantly well in the case of small population sizes ($popsize = 8$), being always ranked between the first and third position and showing a small interquartile range for the $DRMSE_{tst}$. This indicates the good generalization ability of this model for almost all databases. Surprisingly, MLP yielded good results even with very few individuals. On the other hand, SVR exhibited the best interquartile range when $popsize$ was between 32 and 48 individuals. It obtained first and second positions in more than 75% of databases with a low $DRMSE_{tst}$, similar or better than the one of MLP.

Table 4 focuses on the case of 64 individuals. Models with lowest testing error and no statistical difference among them are depicted in bold. MLP and SVR were chosen in the group of *winner* models in eleven out of thirteen databases. A similar trend is observed in Figure 3, where the percentage of *winner* models for each algorithm and $popsize$ is shown. It is observed that most *winner* models were obtained with MLP and SVR. In particular, SVR models were selected as *winners* in $70\% - 80\%$ of the cases for a $popsize \geq 16$.

Table 5 presents the average percentage of features used (N_{FS} in %) for each database. Results show that the number of *winner* models obtained with MLP and SVR was high when the number of attribute was less or equal to 32, while still showing similar N_{FS} values compared to the remaining algorithms. Differences between algorithms increased with databases of higher dimensionality (*triazines* or *tecator*). Nevertheless, results proved that basic algorithms such as LIN, IBk or M5P should be considered when selecting a predictive technique, as there are some simpler databases in which they are included in the group of *winner* models (*strike*, $bodyfat$, *pryim*, *wisconsin* or *meta*). What is more, in these simpler databases, these models exhibited higher ratios of input reduction compared to the the more complex SVR or MLP.

This idea was corroborated in Figure 4, where N_{FS} is plotted against the population size.

SVR and MLP were chosen in most of databases as *winner* models (approximately between 8 and 9 sets out of 13) but a lower N_{FS} (higher reduction) was obtained with simpler techniques when they were included in the group of *winner* models. N_{FS} was around 40 % for SVR and MLP, while it decreased close to a 20 % with linear regression and IBk. No relationship between N_{FS} and population size was observed.

Table 6 summarizes the total execution time required by each one of the configurations (database, regression technique and population size) implemented. The table was complemented with Figure 5, where the execution times are shown in relative terms for each database. Results show that execution times in the majority of databases were considerably low for an iterative optimization methodology, being close or under 10 minutes. This was a consequence of the low dimensionality of most databases, as nine out of thirteen databases presented less than 500 samples and 40 attributes. It has to be noted that MLP is the algorithm with higher execution times in all cases, due to the time consuming training algorithm of the MLP. In the case of *tecator*, the database with a higher number of attributes, this execution time raised over 1000 minutes, which proves the inadequacy of this algorithm for high dimensionality databases. The execution time of SVR and MLP also increased for databases with a higher number of samples (*ailerons*, *puma* and *space*), as the cost of these techniques raises exponentially with the number of samples. Lastly, a linear dependence was observed between the number of individuals used (population size) and the execution time required.

5. Conclusions

This study evaluates the *GA-PARSIMONY*, a new GA-based optimization methodology for model structure selection, with a wide variety of regression techniques and databases. It breaks the classic $Loss + Penalty$ optimization functions into a two-step process, in order to eliminate the necessity of setting the value of the penalty parameter a priori. A first rank of individuals is generated based on the prediction error, and this rank is subsequently modified based on the complexity of the model to spur the selection of parsimonious model. The complexity *ReRank* is made taking into account the inner complexity of the algorithm and the number of features used.

Results proved that this methodology was able to combine error minimization and parsimony effectively. Models with lowest testing RMSE and no statistical difference among them were identified and referred as *winner* models. Even in this group of *winner* models, which are the ones with lowest generalization error, the percentage of features selected was below 50 % for all predictive techniques implemented. The most complex algorithms, MLP and SVR, were selected more frequently in the group of *winner* models, while generally requiring around the 40 % of available attributes. This value decreased down to 20-25 % for the most simple IBk and ridge regression (LIN), despite of being included more occasionally in the group of *winner* models. Due to the relatively low dimension of the databases, no significant differences were observed among the different population sizes evaluated. Consequently, other experiments will be needed with higher-dimensional databases and with other techniques like ensemble methods (random forest for regression, boosting, bagging, etc.).

Acknowledgements

R. Urraca, J. Antonanzas and E. Sodupe would like to acknowledge the fellowship FPI-UR-2014 granted by University of La Rioja. F. Antonanzas-Torres would also like to express his gratitude for the FPI-UR-2012. We would like to convey our gratitude to the *Banco Santander* for the PROFAI13/06 fellowship and *Instituto de Estudios Riojanos* (IER) for funding parts of this research. Finally, we all are greatly indebted to the Agencia de Desarrollo Economico de La Rioja for the ADER-2012- I-IDD-00126 (CONOBUILD) fellowship for funding parts of this research

References

1. Hastie, T., Tibshirani, R., Friedman, J.. The elements of statistical learning: data mining, inference and prediction. 2 ed.; Springer; 2009.

2. Reif, M., Shafait, F., Dengel, A.. Meta-learning for evolutionary parameter optimization of classifiers. *Machine Learning* 2012;87(3):357–380.

3. Calvo-Rolle, J.L., Corchado, E.. A bio-inspired knowledge system for improving combined cycle plant control tuning. *Neurocomputing* 2014;126:95–105.

4. Guerrero, J.L., Berlanga, A., Molina, J.M.. A multi-objective approach for the segmentation issue. *Engineering Optimization* 2012;44(3):267–287.

5. Sedano, J., Curiel, L., Corchado, E., de la Cal, E., Villar, J.. A soft computing method for detecting lifetime building thermal insulationfailures. *Integrated Computer-Aided Engineering* 2010;17 (2):103–115.

6. Xue, B., Zhang, M., Browne, W.N.. Particle swarm optimisation for feature selection in classification: Novel initialisation and updating mechanisms. *Applied Soft Computing* 2014;18(0):261 – 276.

7. Oduguwa, V., Tiwari, A., Roy, R.. Evolutionary computing in manufacturing industry: an overview of recent applications. *Applied Soft Computing* 2005;5(3):281–299.

8. Caamano, P., Bellas, F., Becerra, J.A., Duro, R.J.. Evolutionary algorithm characterization in real parameter optimization problems. *Applied Soft Computing* 2013;13(4):1902–1921.

9. Valdez, F., Melin, P., Castillo, O.. A survey on nature-inspired optimization algorithms with fuzzy logic for dynamic parameter adaptation. *Expert System Applications* 2014;41(14):6459–6466.

10. Guo, T., Han, L., He, L., Yang, X.. A GA-based feature selection and parameter optimization for linear support higher-order tensor machine. *Neurocomputing* 2014;144:408 – 416.

11. Kabir, M.M., Shahjahan, M., Murase, K.. A new local search based hybrid genetic algorithm for feature selection. *Neurocomputing* 2011;74(17):2914 – 2928.

12. Kamyab, S., Eftekhari, M.. Feature selection using multimodal optimization techniques. *Neurocomputing* 2016;171:586 – 597.

13. Huang, H.L., Chang, F.L.. Esvm: Evolutionary support vector machine for automatic feature selection and classification of microarray data. *Biosystems* 2007;90(2):516–528.

14. Vieira, S.M., Mendonza, L.F., Farinha, G.J., Sousa, J.M.. Modified binary PSO for feature selection using SVM applied to mortality prediction of septic patients. *Applied Soft Computing* 2013;13(8):3494 – 3504.

15. Ahila, R., Sadasivam, V., Manimala, K.. An integrated PSO for parameter determination and feature selection of ELM and its application in classification of power system disturbances. *Applied Soft Computing* 2015;32(0):23 – 37.

16. Dhiman, R., Saini, J., Priyanka, . Genetic algorithms tuned expert model for detection of epileptic seizures from EEG signatures. *Applied Soft Computing* 2014;19(0):8 – 17.

17. Castillo, O., Lizarraga, E., Soria, J., Melin, P., Valdez, F.. New approach using ant colony optimization with ant set partition for fuzzy control design applied to the ball and beam system. *Information Sciences* 2015;294:203–215.

18. Castillo, O., Neyoy, H., Soria, J., Melin, P., Valdez, F.. A new approach for dynamic fuzzy logic parameter tuning in ant colony optimization and its application in fuzzy control of a mobile robot. *Applied Soft Computing* 2015;28:150–159.

19. Winkler, S.M., Affenzeller, M., Kronberger, G., Kommenda, M., Wagner, S., Jacak, W., Stekel, H.. Analysis of selected evolutionary algorithms in feature selection and parameter optimization for data based tumor marker modeling. In: Moreno-Diaz R.Z, R., Pichler, F., Quesada-Arencibia, A., eds. *EUROCAST (1)*; vol. 6927 of *Lecture Notes in Computer Science*. Springer; 2011:335–342.

20. Sanz-García, A., Fernández-Ceniceros, J., Fernández-Martínez, R., Martínez-De-Pisón, F.. Methodology based on genetic optimisation to develop overall parsimony models for predicting temperature settings on annealing furnace. *Ironmaking and Steelmaking* 2014;41(2):87–98.

21. Ding, S.. Spectral and wavelet-based feature selection with particle swarm optimization for hyperspectral classification. *Journal of Software* 2011;6(7):1248–1256.

22. Sanz-Garcia, A., Fernandez-Ceniceros, J., Antonanzas-Torres, F., Pernia-Espinoza, A., de Pison, F.M.. 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. *Applied Soft Computing* 2015;35:13 – 28.

23. Sanz-García, A., Fernández-Ceniceros, J., Antoñanzas-Torres, F., Martínez-de Pisón, F.J.. Parsimonious support vector machines modelling for set points in industrial processes based on genetic algorithm optimization. In: *International Joint Conference SOCO13-CISIS13-ICEUTE13*; vol. 239 of *Advances in Intelligent Systems and Computing*. Springer International Publishing; 2014:1–10.

24. Sanz-García, A., Antoñanzas-Torres, F., Fernández-Ceniceros, J., Martínez-de Pisón, F.J.. Overall models based on ensemble methods for predicting continuous annealing furnace temperature settings. *Ironmaking & Steelmaking* 2013;0(0):51–60.

25. Antonanzas-Torres, F., Urraca, R., Fernandez-Ceniceros, J., Martinez-de Pison, F.. Generation of daily global solar irradiation with support vector machines for regression. *Energy Conversion and Management* 2015;96:277–286.

26. Urraca, R., Antonanzas, J., Martinez-de Pison, F., Antonanzas-Torres, F.. Estimation of solar global irradiation in remote areas. *Journal of Renewable and Sustainable Energy* 2015;7(2):1–14.

27. Antonanzas, J., Urraca, R., Martinez-de Pison, F., Antonanzas-Torres, F.. Solar irradiation mapping with exogenous data from support vector regression machines estimations. *Energy Conversion and Management* 2015;100:380–390.

28. Fernandez-Ceniceros, J., Sanz-Garcia, A., Antonanzas-Torres, F., Martinez-de Pison, F.. A numerical-informational approach for characterising the ductile behaviour of the t-stub component. part 2: Parsimonious soft-computing-based metamodel. *Engineering Structures* 2015;82:249–260.

29. McKay, M., Beckman, R., Conover, W.. A comparison of three methods for selecting values on input variables in the analysis of output from a computer code. *Technometrics* 1979;21:239–245.

30. Kuhn, M., Johnson, K.. Applied Predictive Modeling. New York, NY: Springer; 2013.

31. Wilcoxon, F.. Individual comparisons by ranking methods. *Biometrics* 1945;1:80–83.

32. Michalewicz, Z., Janikow, C.. Handling constraints in genetic algorithms. In: *Proceedings of the Fourth International Conference on Genetic Algorithms*. Los Altos, CA: Morgan Kaufmann Publishers; 1991:151–157.

33. Menéndez de Llano, R., Bosque, J.L.. Study of neural net training methods in parallel and distributed architectures. *Future Generation Computer Systems* 2010;26(2):267–275.

34. Drucker, H., Chris, , Kaufman, B.L., Smola, A., Vapnik, V.. Support vector regression machines. In: *Advances in Neural Information Processing Systems 9*; vol. 9. 1997:155–161.

35. Witten, I.H., Frank, E., Hall, M.A.. Data Mining: Practical Machine Learning Tools and Techniques. 3 ed.; Amsterdam: Morgan Kaufmann; 2011. ISBN 978-0-12-374856-0.

36. Aha, D.W., Kibler, D., Albert, M.K.. Instance-based learning algorithms. *Machine Learning* 1991;6(1).

37. Quinlan, J.R.. Learning with continuous classes. In: *5th Australian Joint Conference on Artificial Intelligence*. 1992:343–348.

38. Wang, Y., Witten, I.. Induction of model trees for predicting continuous classes. In: *Proceedings of the 9th European Conference on Machine Learning Poster Papers*. Prague, Chez Republic; 1997:128–137.

39. Michalewicz, Z., Janikow, C.Z.. Handling constraints in genetic algorithms. In: *ICGA*. 1991:151–157.

40. Lichman, M.. UCI machine learning repository. 2013. URL: http://archive.ics.uci.edu/ml.

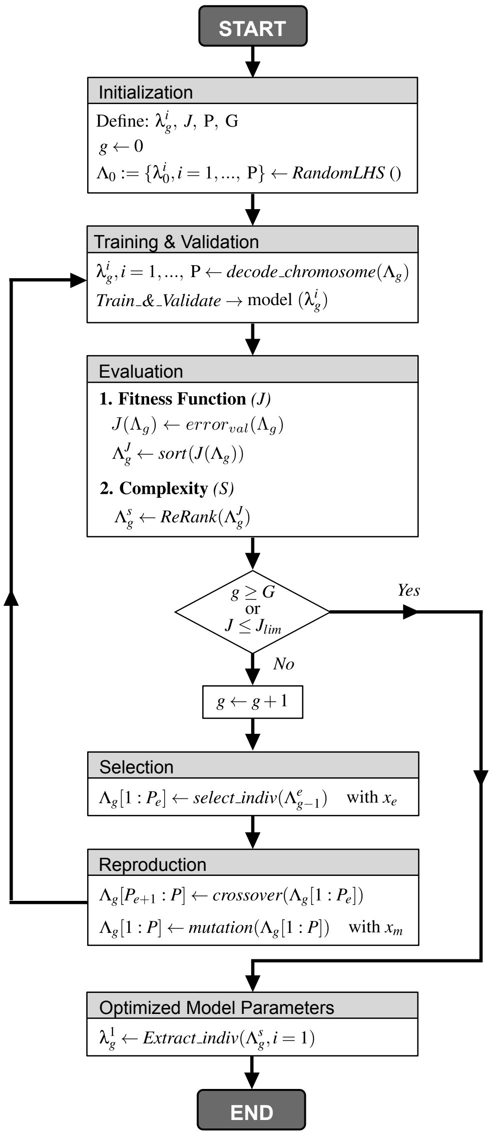
41. StatLib-Datasets Archive; 2015. URL: http://lib.stat.cmu.edu/datasets/.

42. R Core Team, . R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing; Vienna, Austria; 2013.

43. Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., Leisch, F.. e1071: Misc Functions of the Department of Statistics (e1071), TU Wien; 2014. URL: http://CRAN.R-project.org/package=e1071; R package version 1.6-4.

44. Hornik, K., Buchta, C., Zeileis, A.. Open-source machine learning: R meets Weka. *Computational Statistics* 2009;24(2):225–232.

45. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.. The weka data mining software: An update. *SIGKDD Explorations* 2009;11(1):10–18.



START
Initialization
Define: λ_g^i, J, P, G
$g \leftarrow 0$
$\Lambda_0 := \{\lambda_0^i, i = 1, ..., P\} \leftarrow RandomLHS\ ()$
Training & Validation
$\lambda_g^i, i = 1, ..., P \leftarrow decode\_chromosome(\Lambda_g)$
$Train\_\&\_Validate \rightarrow$ model (λ_g^i)
Evaluation
1. Fitness Function (J)
$J(\Lambda_g) \leftarrow error_{val}(\Lambda_g)$
$\Lambda_g^J \leftarrow sort(J(\Lambda_g))$
2. Complexity (S)
$\Lambda_g^s \leftarrow ReRank(\Lambda_g^J)$
$g \geq G$ or $J \leq J_{lim}$
Yes
No
$g \leftarrow g+1$
Selection
$\Lambda_g[1:P_e] \leftarrow select\_indiv(\Lambda_{g-1}^e)$ with x_e
Reproduction
$\Lambda_g[P_{e+1}:P] \leftarrow crossover(\Lambda_g[1:P_e])$
$\Lambda_g[1:P] \leftarrow mutation(\Lambda_g[1:P])$ with x_m
Optimized Model Parameters
$\lambda_g^1 \leftarrow Extract\_indiv(\Lambda_g^s, i = 1)$
END

Figure 1: GA-PARSIMONY optimization methodology

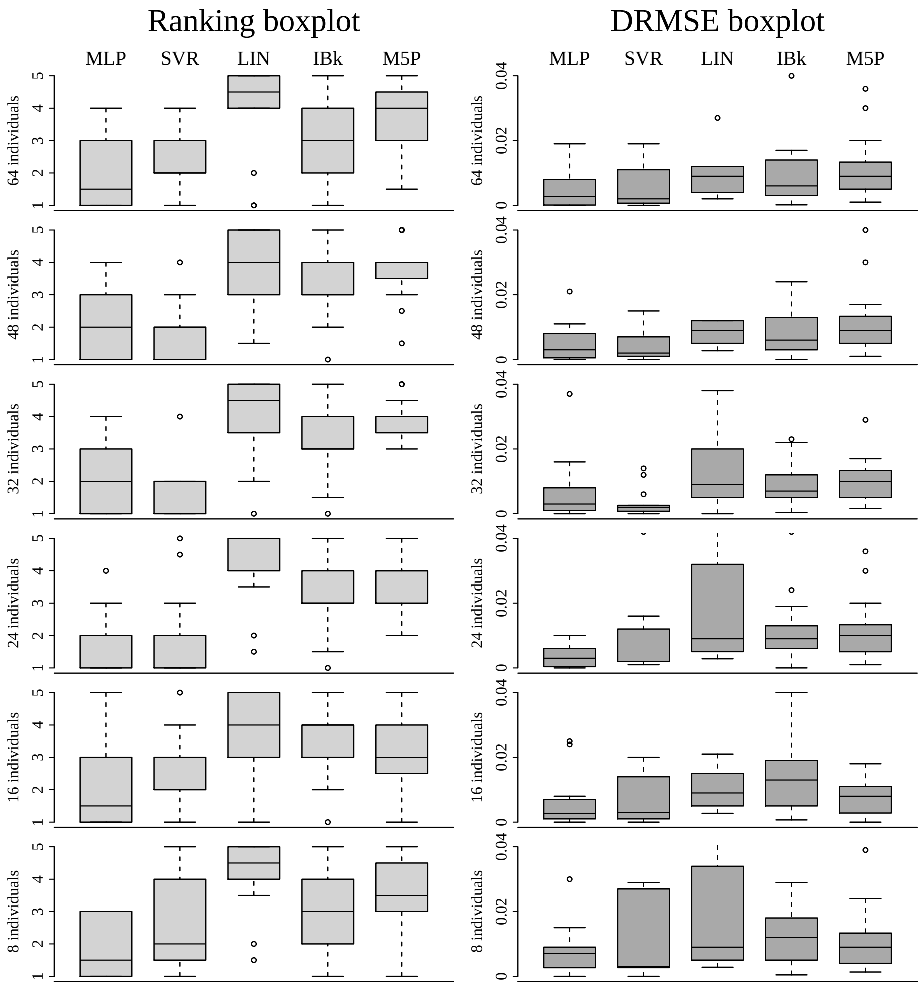


Figure 2: Boxplots of the results obtained with each regression technique and each population size for the different databases. The plot in the left shows the position of the algorithm in the rank based on the $RMSE_{tst}$ while the plot on the right depicts the $DRMSE_{tst}$.

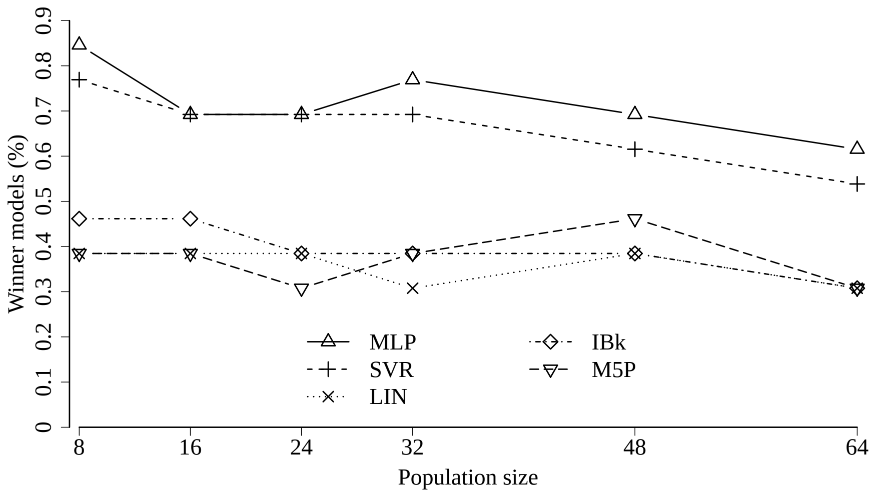


Figure 3: Percentage of *winner* models according to the population size (*popsize*).

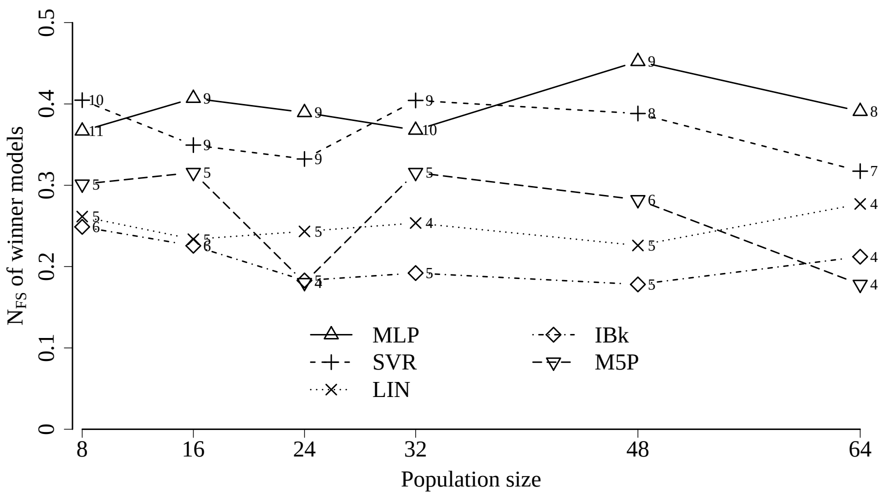


Figure 4: Number of features in parts per unit used by the *winner* models according the population size.

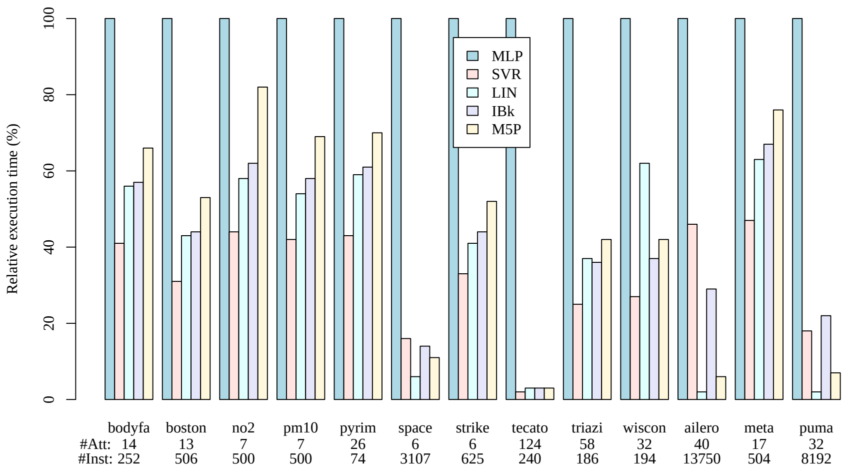


Figure 5: Relative execution time in % took by each regression technique for the implementation of the GA-PARSIMONY methodology.

Table 1: Regression techniques implemented, along with the functions used to compute the inner complexity of the algorithms (C_{model}) and the optimization range of the tuning parameters.

Algorithm	C_{model}	Tuning parameters
MLP	$\sum w_i{}^2$ (network weights)	number of hidden neurons $[1, 30]$ ridge $\left[10^{-6}, 0.\hat{9}\right]$
SVR	number of support vectors	$log_{10}(C)$ $\left[-3.\hat{9}, 1.4\hat{9}\right]$ γ $\left[10^{-6}, 0.\hat{9}\right]$ ϵ $\left[10^{-6}, 0.\hat{9}\right]$
LIN	$\sum \beta_i^2$ (regression weights)	ridge $\left[10^{-8}, 0.\hat{9}\right]$
IBk	$(10^6/K) - 1$	distance weighting $[1 = none, 2 = 1 - d, 3 = 1/d]$ K $[1, 30]$
M5P	number of leafs	M $[1, 30]$

Table 2: Data description

Database	# Attributes	# Instances
bodyfat	14	252
boston	13	506
no2	7	500
pm10	7	500
pyrim	26	74
space	6	3107
strike	6	625
tecator	124	240
triazines	58	186
wisconsin	32	194
ailerons	41	13750
meta	18	504
puma	33	8192

Table 3: Summary of the models with lowest $RMSE_{tst}$ for each database. N_{FS} in % stands for the relationship between the number of inputs used by the model (N_{FS}) and the number of available inputs of the database.

Database	Algorithm	$popsize$	$RMSE_{tst}$	$RMSE_{tst}^{sd}$	N_{FS}	$N_{FS}[\%]$
bodyfat	SVR	48	.030	.013	2	14.3
boston	SVR	64	.066	.008	8	61.5
no2	MLP	24	.094	.004	5	71.4
pm10	SVR	8	.157	.009	4	57.1
pyrim	SVR	32	.110	.066	5	19.2
space	MLP	8	.032	.002	6	100
strike	SVR	16	.052	.020	4	66.6
tecator	MLP	64	.009	.001	3	2.4
triazines	M5P	16	.162	.005	9	15.5
wisconsin	LIN	32	.261	.035	7	21.9
ailerons	MLP	24	.044	.001	8	20
meta	IBk	48	.067	.043	3	17.6
puma	SVM	32	.031	.001	4	12.5

Table 4: Mean and standard deviation (in parenthesis) of the $RMSE_{tst}$ obtained with each regression technique for the case of 64 individuals. The algorithm ranking according to the $RMSE_{tst}$ is shown in brackets. The group of *winner* algorithms for each database is depicted in bold.

Alg	*bodyfat*	*boston*	*no2*	*pm10*	*pyrim*
MLP	**.034 (.018) [3]**	.073 (.011) [2]	**.095 (.004) [1.5]**	**.157 (.011) [1]**	**.129 (.035) [4]**
SVR	**.030 (.015) [1]**	**.066 (.006) [1]**	**.096 (.004) [3]**	**.158 (.011) [2]**	**.121 (.053) [2]**
LIN	**.035 (.020) [4.5]**	.113 (.008) [5]	.098 (.002) [4]	.169 (.013) [5]	**.120 (.023) [1]**
IBk	**.031 (.010) [2]**	.078 (.010) [3]	.099 (.007) [5]	**.160 (.012) [3]**	**.127 (.042) [3]**
M5P	**.035 (.021) [4.5]**	.086 (.012) [4]	**.095 (.006) [1.5]**	.168 (.014) [4]	.146 (.062) [5]
Alg	*space*	*strike*	*tecator*	*triazines*	*wisconin*
MLP	**.032 (.002) [1]**	**.060 (.018) [3]**	**.009 (.001) [1]**	.171 (.008) [1]	**.269 (.039) [2]**
SVR	**.033 (.003) [2]**	**.055 (.018) [1.5]**	.026 (.016) [4]	.181 (.015) [3]	**.275 (.042) [4]**
LIN	.041 (.003) [5]	**.061 (.018) [4.5]**	.013 (.002) [2]	.189 (.011) [4]	**.263 (.035) [1]**
IBk	.038 (.004) [3]	**.055 (.017) [1.5]**	.049 (.007) [5]	.176 (.012) [2]	**.278 (.031) [5]**
M5P	.040 (.003) [4]	**.061 (.018) [4.5]**	.016 (.002) [3]	.192 (.009) [5]	**.271 (.037) [3]**
Alg	*ailerons*	*meta*	*puma*		
MLP	**.044 (.001) [1.5]**	**.069 (.042) [3]**	**.031 (.001) [1]**		
SVR	**.044 (.001) [1.5]**	**.069 (.043) [3]**	**.032 (.001) [2]**		
LIN	.049 (.001) [5]	**.069 (.042) [3]**	.151 (.001) [5]		
IBk	.046 (.004) [4]	**.067 (.043) [1]**	.044 (.001) [3]		
M5P	**.045 (.001) [3]**	**.070 (.042) [5]**	.045 (.001) [4]		

Table 5: Average of the percentage of inputs retained by each model (N_{FS}) for the group of *winner* models in each database. The number of *winner* models is depicted in brackets

Database	#Att	MLP	SVR	LIN	IBk	M5P
space	6	1.00 (6)	0.96 (4)	- (0)	- (0)	- (0)
strike	6	0.56 (6)	0.44 (6)	0.67 (6)	0.33 (6)	0.33 (6)
no2	7	0.67 (6)	0.57 (6)	- (0)	- (0)	0.86 (4)
pm10	7	0.54 (5)	0.57 (6)	- (0)	0.57 (2)	- (0)
boston	13	0.62 (1)	0.64 (3)	- (0)	- (0)	- (0)
bodyfat	14	0.14 (6)	0.15 (6)	0.14 (6)	0.14 (5)	0.18 (6)
pyrim	26	0.22 (4)	0.21 (6)	0.14 (4)	0.15 (6)	- (0)
wisconsin	32	0.16 (6)	0.08 (6)	0.14 (6)	0.10 (6)	0.08 (6)
triazines	58	0.16 (1)	- (0)	- (0)	- (0)	0.16 (1)
tecator	124	0.25 (1)	- (0)	- (0)	- (0)	- (0)
ailerons	40	0.23 (6)	0.24 (1)	- (0)	- (0)	- (0)
meta	17	0.12 (6)	0.12 (6)	0.11 (6)	0.18 (6)	0.11 (6)
puma	32	0.12 (2)	0.12 (2)	- (0)	-(0)	- (0)

Table 6: Total execution time in minutes of the *GA-PARSIMONY* methodology for each regression algorithm, population size and database

		Database												
Popsize	Algorithm	bodyfat	boston	no2	pm10	pyrim	space	strike	tecator	triazines	wisconsin	ailerons	meta	puma
	MLP	10.8	24.7	15.4	12.6	11.4	134.7	18.4	225.5	52.6	15.0	422.7	9.5	98.4
	SVR	6.2	6.6	6.5	6.5	6.3	13.2	6.7	8.2	7.0	6.8	92.8	10.7	51.5
8	LIN	7.5	8.1	7.8	7.8	7.5	8.5	7.9	11.2	8.9	8.3	22.5	7.9	9.9
	IBk	7.6	7.9	7.9	7.9	7.8	14.8	8.0	10.4	8.8	8.4	154.1	8.0	72.3
	M5P	9.1	10.5	10.7	10.6	7.9	12.5	9.5	12.2	10.2	10.5	34.7	9.4	17.0
	MLP	16.4	22.7	15.7	16.9	15.4	177.1	22.2	485.2	30.7	28.0	1071.8	14.6	669.9
	SVR	6.7	7.1	6.9	7.2	6.7	27.7	7.3	8.9	7.6	7.5	497.3	6.8	122.5
16	LIN	9.1	9.8	9.2	9.1	9.1	10.5	9.2	14.1	11.3	17.3	23.0	9.2	13.3
	IBk	9.4	9.9	9.7	9.8	9.4	24.2	9.8	13.6	11.1	10.3	314.5	9.8	144.3
	M5P	10.9	12.0	12.9	11.7	10.8	18.8	11.6	15.6	12.8	11.8	63.9	11.2	48.6
	MLP	18.0	40.4	30.6	24.1	19.0	422.3	21.2	1219.8	37.7	22.6	1704.3	18.6	1243.6
	SVR	6.9	8.0	7.4	7.4	7.0	36.0	7.8	10.0	8.2	7.8	563.2	7.1	196.3
24	LIN	11.1	11.0	10.8	10.5	10.7	12.2	10.8	17.6	13.0	11.8	27.4	10.3	16.5
	IBk	11.1	11.9	12.0	11.8	11.5	33.3	11.9	16.9	13.4	12.6	458.4	11.7	217.6
	M5P	12.9	15.1	15.0	17.8	12.9	19.9	14.5	19.9	15.2	14.3	69.4	16.7	36.1
	MLP	22.9	44.5	26.1	34.8	20.6	459.9	37.8	1038.5	38.1	35.8	2816.1	23.1	1719.6
	SVR	7.4	8.6	8.1	8.3	7.4	45.5	8.5	11.8	9.1	8.3	692.1	7.4	668.1
32	LIN	12.1	13.2	13.1	12.1	12.8	14.3	12.1	21.4	15.5	13.9	34.0	11.8	24.0
	IBk	12.4	13.6	13.5	13.4	13.1	42.3	13.6	20.9	15.7	14.2	645.2	13.6	302.6
	M5P	14.7	16.9	17.1	16.4	14.2	27.3	16.1	22.7	18.4	15.9	86.4	28.6	43.2
	MLP	34.5	77.5	34.2	34.4	31.2	591.0	52.7	1401.3	93.0	64.1	2340.8	34.2	2339.1
	SVR	8.1	10.3	9.0	9.0	8.4	69.0	10.0	15.4	10.7	9.5	1873.5	29.2	597.3
48	LIN	14.8	15.8	15.5	14.7	15.9	18.1	14.7	28.5	20.9	17.3	47.2	19.7	26.8
	IBk	15.6	17.2	17.1	17.0	16.4	59.7	16.9	27.6	20.8	18.5	972.0	26.3	438.0
	M5P	18.7	20.8	21.0	19.9	18.7	38.0	19.7	32.1	23.1	20.3	145.2	18.2	61.7
	MLP	41.3	83.6	46.8	60.1	41.7	906.6	58.8	1756.6	212.5	80.4	4021.6	53.0	4075.4
	SVR	8.9	12.0	10.2	10.1	9.3	85.2	11.8	16.8	12.9	10.8	2405.7	17.7	1139.7
64	LIN	18.1	21.6	18.7	17.8	19.7	22.4	17.5	36.4	24.4	21.7	61.0	18.0	46.4
	IBk	18.6	20.9	20.5	20.5	20.0	78.6	20.3	33.6	25.5	21.8	1259.8	29.7	582.2
	M5P	21.5	25.2	34.9	23.4	22.0	47.1	23.7	41.8	28.6	23.4	181.3	32.5	85.4

Figure 1

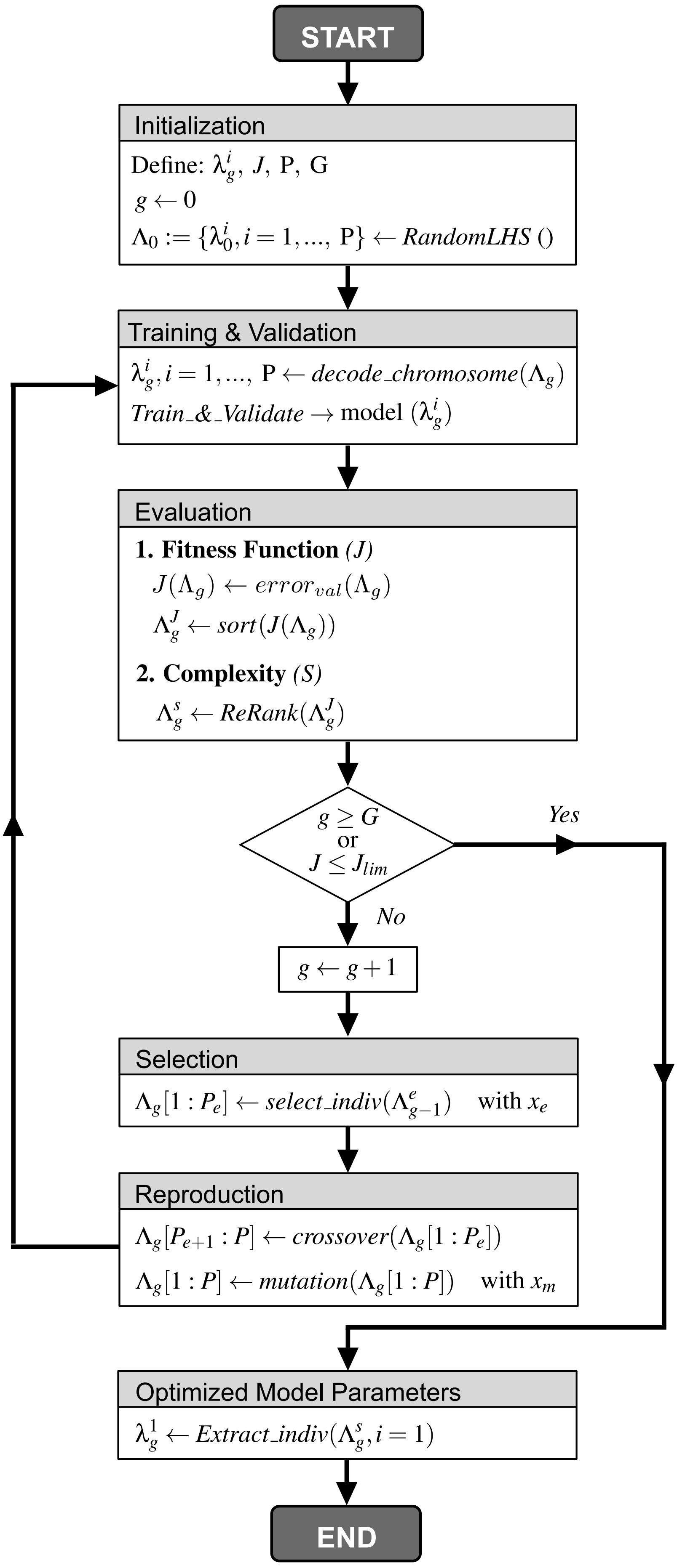
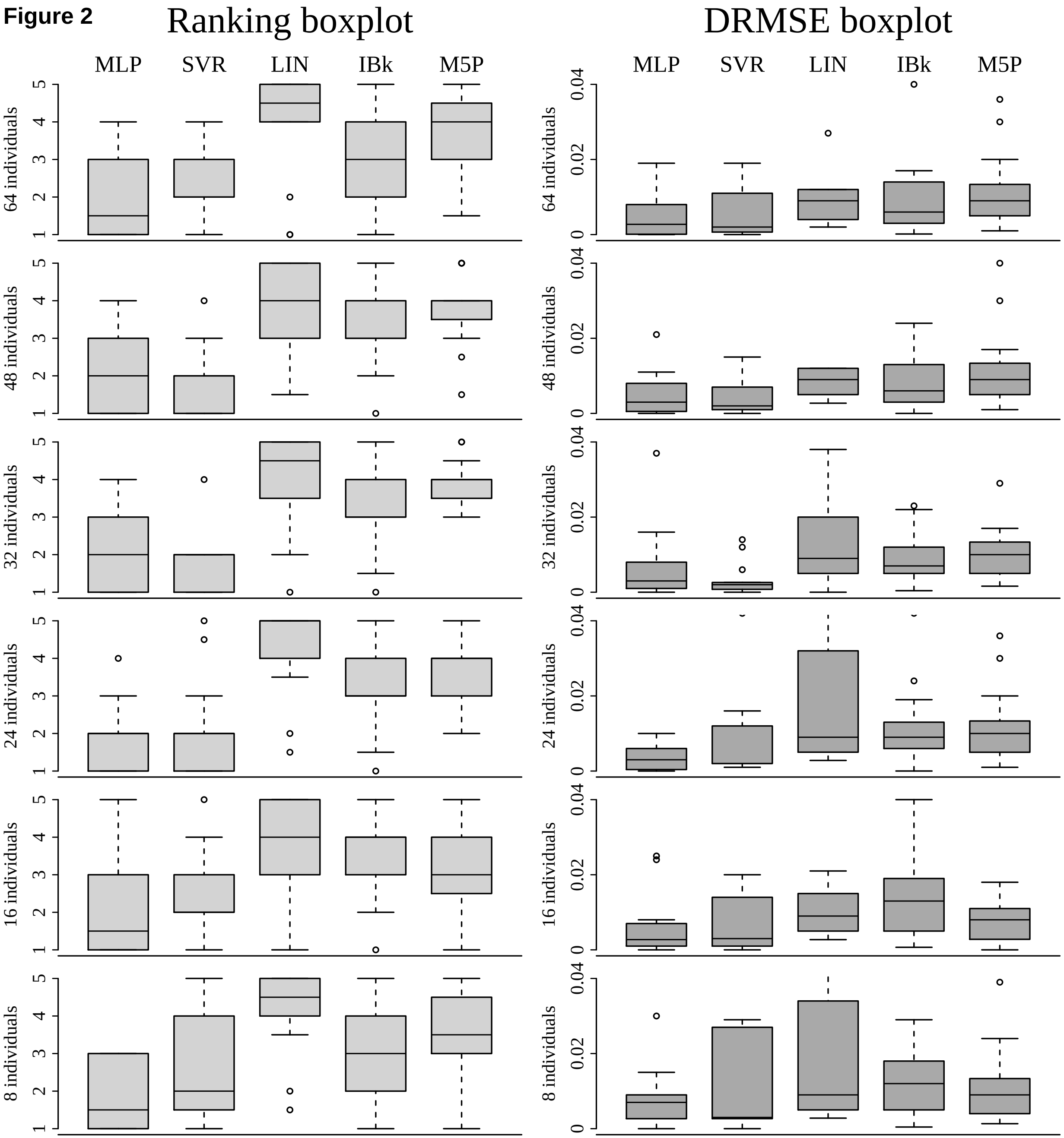
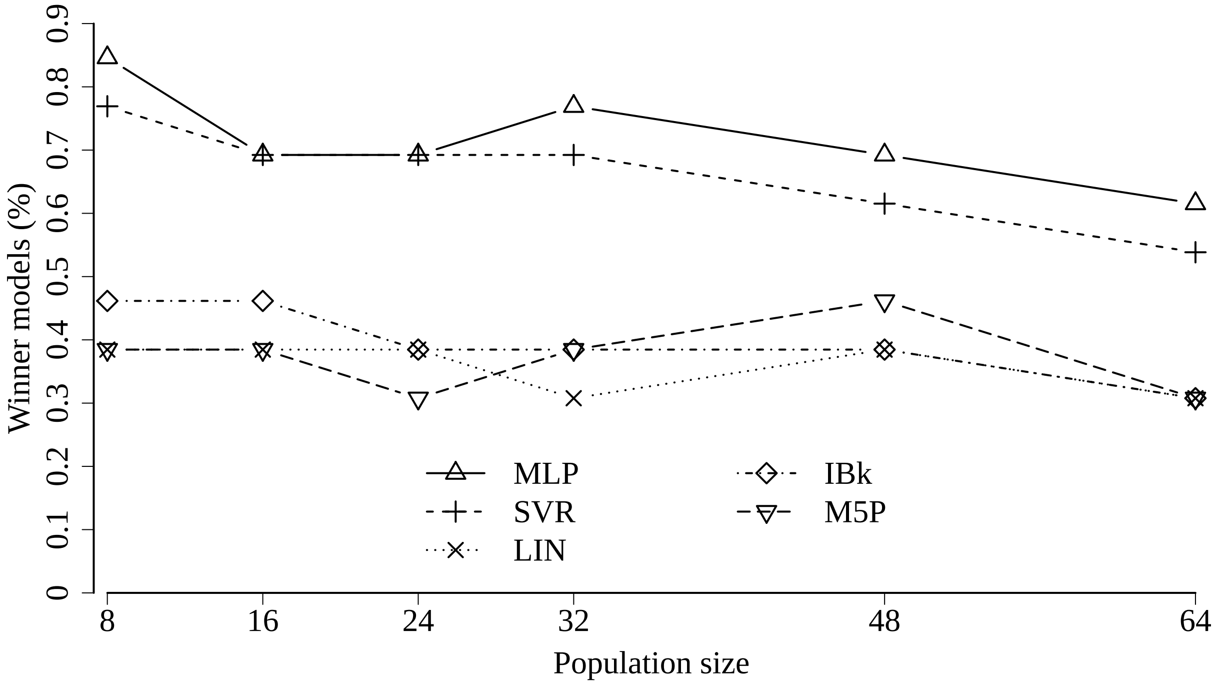

START
Initialization
Define: λ_g^i, J, P, G
$g \leftarrow 0$
$\Lambda_0 := \{\lambda_0^i, i = 1, \dots, \text{P}\} \leftarrow \textit{RandomLHS}\ ()$
Training & Validation
$\lambda_g^i, i = 1, \dots, \text{P} \leftarrow \textit{decode\_chromosome}(\Lambda_g)$
$\textit{Train\_\&\_Validate} \rightarrow \text{model}\ (\lambda_g^i)$
Evaluation
1. Fitness Function *(J)*
$J(\Lambda_g) \leftarrow error_{val}(\Lambda_g)$
$\Lambda_g^J \leftarrow sort(J(\Lambda_g))$
2. Complexity *(S)*
$\Lambda_g^s \leftarrow ReRank(\Lambda_g^J)$
$g \geq G$ or $J \leq J_{lim}$
Yes
No
$g \leftarrow g + 1$
Selection
$\Lambda_g[1 : P_e] \leftarrow select\_indiv(\Lambda_{g-1}^e)$ with x_e
Reproduction
$\Lambda_g[P_{e+1} : P] \leftarrow crossover(\Lambda_g[1 : P_e])$
$\Lambda_g[1 : P] \leftarrow mutation(\Lambda_g[1 : P])$ with x_m
Optimized Model Parameters
$\lambda_g^1 \leftarrow Extract\_indiv(\Lambda_g^s, i = 1)$
END

Figure 2



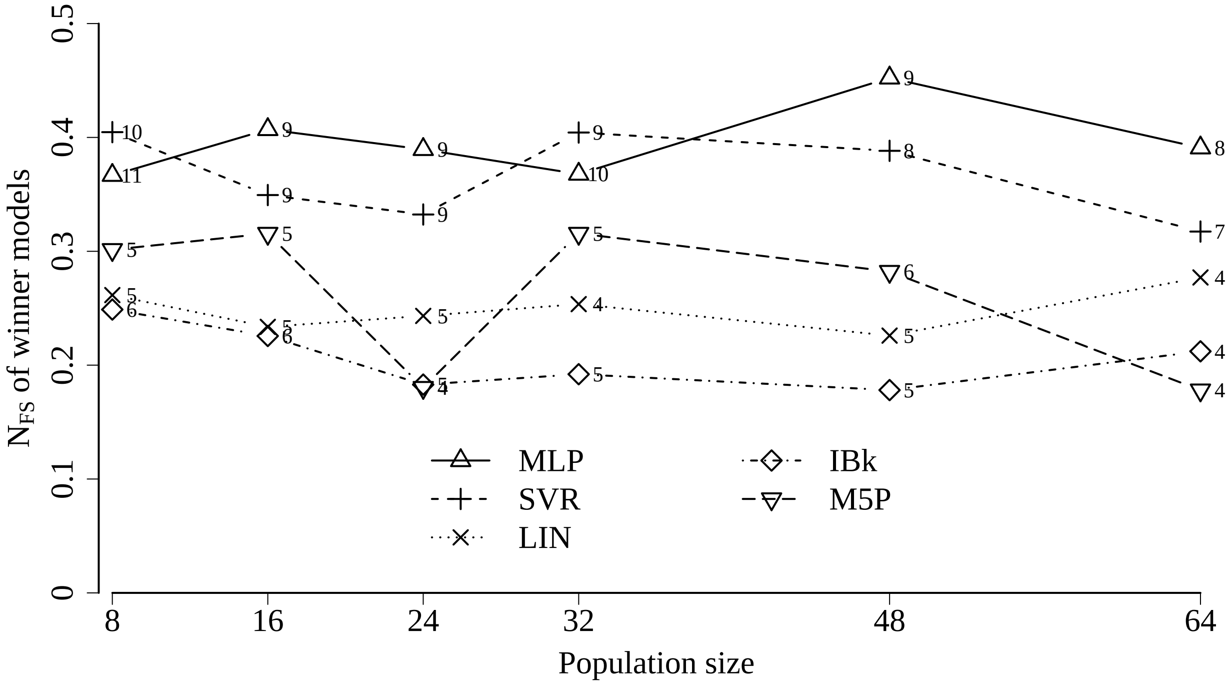
Ranking boxplot
DRMSE boxplot
MLP
SVR
LIN
IBk
M5P
64 individuals
48 individuals
32 individuals
24 individuals
16 individuals
8 individuals
1
2
3
4
5
0
0.02
0.04

Figure 3



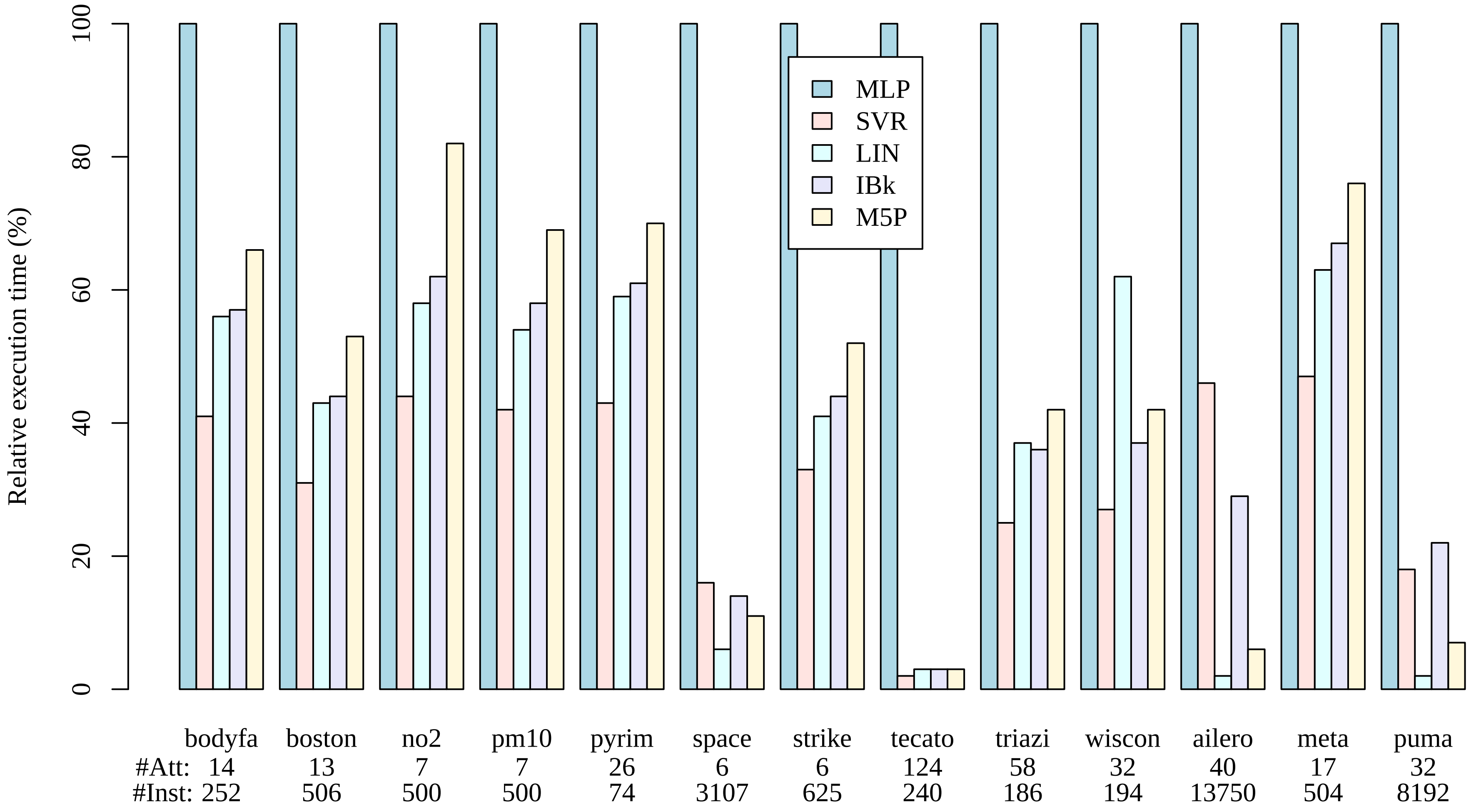
Winner models (%)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
8
16
24
32
48
64
Population size
MLP
SVR
LIN
IBk
M5P

Figure 4



0.5
0.4
0.3
0.2
0.1
0
N_{FS} of winner models
8
16
24
32
48
64
Population size
MLP
SVR
LIN
IBk
M5P
11
9
9
10
9
8
10
9
9
9
8
7
5
5
5
5
6
4
5
5
5
4
5
4
6
6
5
5
5
4

Figure 5



Relative execution time (%)
0
20
40
60
80
100
MLP
SVR
LIN
IBk
M5P
bodyfa
boston
no2
pm10
pyrim
space
strike
tecato
triazi
wiscon
ailero
meta
puma
#Att: 14
13
7
7
26
6
6
124
58
32
40
17
32
#Inst: 252
506
500
500
74
3107
625
240
186
194
13750
504
8192