Journal of Experimental & Theoretical Artificial Intelligence. 2019; 32(2):243-271

Population subset selection for the use of a validation dataset for overfitting control in genetic programming

Daniel Rivero, Enrique Fernandez-Blanco, Carlos Fernandez-Lozano, Alejandro Pazos

[a]*Dept. of Computer Science, University of A Coruna, Facultad de Informatica, CITIC, A Coruna, Spain;* [b]*Instituto de Investigacion Biomedica de A Corua (INIBIC), Complexo Hospitalario Universitario de A Corua (CHUAC), A Coruna, Spain*

Abstract

Genetic Programming (GP) is a technique which is able to solve different problems through the evolution of mathematical expressions. However, in order to be applied, its tendency to overfit the data is one of its main issues. The use of a validation dataset is a common alternative to prevent overfitting in many Machine Learning (ML) techniques, including GP. But, there is one key point which differentiates GP and other ML techniques: instead of training a single model, GP evolves a population of models. Therefore, the use of the validation dataset has several possibilities because any of those evolved models could be evaluated. This work explores the possibility of using the validation dataset not only on the training-best individual but also in a subset with the training-best individuals of the population. The study has been conducted with 5 well-known databases performing regression or classification tasks. In most of the cases, the results of the study point out to an improvement when the validation dataset is used on a subset of the population instead of only on the training-best individual, which also induces a reduction on the number of nodes and, consequently, a lower complexity on the expressions.

Keywords*:*

Genetic programming, overfitting, validation, evolutionary Computation

Introduction

For more than 20 years, Genetic Programming (GP) has been one of the most successful Evolutionary Computation (EC) techniques applied to solve a wide diversity of problems. The main reason for its success over time is its capability to generate mathematical expressions which are easily analysed and validated.

One of the main issues that most of the Machine Learning (ML) techniques has to tackle is setting constraints on the complexity of the system to be developed. For example, a polynomial regression model needs to set the polynomial order in advance, or an artificial neural network (ANN) needs to set the number of neurons within the network. Therefore, those constraints limit the complexity of the problems that the network or polynomial will be able to solve. Oppositely, GP has no beforehand constraints on problem complexity. Since the most common expression in GP has the shape of a tree and given the possibility that this tree may grow, the corresponding mathematical expression can have an arbitrary complexity, although some measures may be taken to avoid excessively large trees, such as limiting the maximum height of the tree.

However, this great advantage of GP is also one of its main drawbacks. Expressions allowed to be arbitrarily complex may become overfitted to the training dataset. In those cases, the noise within the training dataset is treated as a feature and the generated expressions try also to learn the noise information. Usually, this effect is evidenced by having a very low error on the training dataset, whereas the evaluation on the test dataset achieves a very high error.

Therefore, overfitting is a direct consequence of the possibility of generating complex expressions. The GP evolutionary process generates trees which progressively adapt to the problem being solved, which are typically more and more complex, resulting in trees with a larger number of nodes. This effect is related to the phenomenon known as bloat, which can be defined as the uncontrolled growth in a tree size with a small impact on fitness. Both effects, bloat and overfitting, are usually treated as related since the bloated models are usually more likely to overfit data than those with a smaller number of nodes.

A common approach to address this issue is to use a third dataset called validation dataset. This dataset allows to calculate the validation error on the expressions generated by GP. A rising error could indicate that the training dataset may be beginning to overfit, and the training should be stopped. Nonetheless, this is not an easy decision because the error in the validation dataset may decline in future generations.

The main aim of this work is to study the impact of using a validation dataset to select the individual to be returned by the GP process. This decision is usually made by using only the validation dataset on the individual with the best fitness during training, the training-best individual. Even there are works that use more than the training-best individual (Danandeh Mehr, Kahya, Uyumaz, & Erdem, 2014) (Danandeh Mehr & Nourani, 2017), this possibility has hardly been explored. The results of this paper point out to a better outcome by taking a percentage of the population to carry out the validation.

Related work

First and foremost, it should be highlighted that the use of a validation dataset is not a concept belonging to GP. Most of the ML techniques have been using this technique for a long time since it is one of the main resources to deal with overfitting. In the specific case of GP, other alternative techniques have been also developed.

For instance, in (Langdon, 2011) the authors propose to use a smaller set of patterns, and, as they state, this could lead to avoiding overfitting. Other approaches based on evaluating a random subset of the training set were also published. These approaches were compared to different strategies when using a validation set (Gonçalves & Silva, 2011). Finally, other works use model selection strategies based on cross-validation, although their results are not directly related to GP (Cawley & Talbot, 2010)

One of the first alternatives was a technique which consisted of controlling the bloat phenomenonpoli(Poli & McPhee, 2014; Silva & Costa, 2009; Vanneschi, Castelli, & Silva, 2010). The proposal defines a penalty term which multiplies the number of nodes in the tree-shape model by a weight number and, then, adds the result of that multiplication to the fitness value. This penalisation allows to control the bloat effect by reducing the size and, as a collateral effect, controlling overfitting (Ekárt & Nemeth, 2001; Gagné, Schoenauer, Parizeau, & Tomassini, 2006; Gustafson, Ekárt, Burke, & Kendall, 2004; Soule & Foster, 1998; Zhang & Mühlenbein, 1995). However, other studies have shown that even in an environment with bloat control, overfitting may occur (Vanneschi & Silva, 2009), and that low complexity expressions may have a greater generalisation error (Cavaretta & Chellapilla, 1999).

Other approaches are based on similarities between the trees. The method called Semantic Similarity-based Crossover calculates a measure of similarity between two trees or subtrees (Sampling Semantics Distance), based on the similarity of their semantics (Uy, Hien, Hoai, & ONeill, 2010). In that paper, it was stated that the crossover between two trees was more useful when they were not too similar or too different, resulting in lower overfitting. Alternatively, in (Vanneschi & Gustafson, 2009), the overfitted solutions were maintained in a list so that any new solution, before being included in the population, could be compared to solutions from that list, and be discarded if it was too similar to the others.

A similar technique is the so-called Random Sampling Technique (Gathercole & Ross, 1994), which is based on not using the whole training dataset in the fitness function. Instead of it, a random subset of the training dataset is generated in each generation to calculate the fitness of individuals. This technique was initially suggested as a way to speed up GP. Nevertheless, other studies showed that it could also be used to control the GP overfitting (Gonçalves & Silva, 2011; Gonçalves, Silva, Melo, & Carreiras, 2012; Liu & Khoshgoftaar, 2004)

On the same line, Interleaved Sampling (Azad, Medernach, & Ryan, 2014; Gonçalves & Silva, 2013) is found. This technique is based on the condition that the evaluation of the fitness function alternates between using the entire dataset and a single data point. Depending on whether the intention is to put more pressure on the set of patterns or on each isolated piece of data, more or fewer generations can be devoted to evaluating the entire dataset (interleaved all) or to alternatively evaluate a single data point (interleaved single). The results show that this technique allows the reduction of overfitting and the improvement of generalisation.

To prevent overfitting, the use of a validation dataset was also deeply studied (Gagné et al., 2006; Žegklitz & Pošík, 2015). For example, Canary Functions were introduced in a work in which a validation dataset is used to measure overfitting. These functions are different from the fitness function, even they pursue the same goal. When the values of those functions differ significantly from the fitness value, overfitting may be occurring (Foreman & Evett, 2005). Other studies using a validation dataset do not stop the evolutionary process, although the returned individual is the one which obtained the best fitness value in the validation dataset. This process is known as Backwarding (Robilliard & Fonlupt, 2001; Žegklitz & Pošík, 2015).

Other works have previously explored the possibility of selecting the best individual not only as a the training-best, but from a subset of the population, using different measures for this selection (Danandeh Mehr et al., 2014) (Danandeh Mehr & Nourani, 2017). However, a deep study on how many individuals to be evaluated with the validation dataset is still missing.

Finally, a different approach, which also uses a validation dataset, is Validation Start. In this case, the fitness is calculated by using the weighted sum of the error on the training dataset and the absolute difference of the error on the training dataset and the error on the validation dataset. The returned individual is the one with the lowest fitness value. The idea behind this modification of the fitness function is that a solution which produces a low error on the training dataset is expected, but the difference between this error and the one measured on another set of patterns (validation) should also be low (Gonçalves & Silva, 2011).

Method

Overfitting can be measured for a given expression as the difference between the error in the test dataset and the one in the training dataset (Gonçalves & Silva, 2013; Gonçalves et al., 2012). In order to control this value, as it was exemplified in Section 2, the validation dataset allows to detect overfitting. Since the validation dataset does not have an influence on the fitness value, it is a set of patterns that are not involved in the training and therefore will provide an approximation of the test error. Another definition of overfitting can be found in (Vanneschi et al., 2010), in which training and test errors are calculated throughout the training process.

An interesting work describing the use of different datasets for learning machine evolution is (Igel, 2013). However, that work has a big difference with this: in (Igel, 2013) the evolutionary algorithm evolved the hyperparameters of a learning machine. Therefore, the evaluation of each individual involves training this learning machine. This leads to having 3 different datasets: S_{train}, S_{evolve} and S_{final}. In the case of this work, each individual represents an expression that does not have to be trained. Thus, in this work, we dont need S_{train}. S_{evolve} corresponds to the training set. Regarding to S_{final}, as described in the reference, only promising candidates should be considered in this selection process and promising candidates could be the set of those individuals which were the best in some generation, which is similar to the validation set used here: evaluate only the best individuals of the population with this dataset. However, since S_{final} is used to make decisions, the results on this dataset are not truly independent from the whole learning process, i.e. they would be optimistic and thus another dataset is needed: a test dataset. In our work, the results are calculated in a dataset that does not have any influence in the whole evolutionary process.

This work applied the validation dataset in its most common form, which evaluates the output of the algorithm in each iteration with the validation dataset. This evaluation provides an estimation of how the model will behave on the test dataset. Therefore, if the validation error increases, this may be an indication that the system is starting to overfit. This could lead to the decision of prematurely stopping the training process. However, although the validation error increases in a given iteration, this error may decrease in successive generations. For this reason, prematurely stopping the training is usually not a good strategy. Due to this probable premature undesired stop and the unknown number of the optimum number of generations, this work has performed the training processes during a number of generations which is typically considered as high in the related literature (1000). When the generations had elapsed, instead of returning the model generated in the last generation, the returned model is the validation-best individual, which obtained the lowest error on the validation dataset, similarly to the approach proposed in other studies, called Backwarding (Žegklitz & Pošík, 2015). In this work, the error measurement was performed using the Root Mean Squared Error (RMSE) between the output and predicted values for regression problems, and the misclassification rate for classification problems, as described in Section 4.

However, if the population size is N individuals, in every generation GP will produce approximately N new individuals. Some individuals would be copies of the previous generation, but overall, it is expected that the new generation should be better than the previous one. Therefore, considering that N new models have been produced, it is possible not only to evaluate the training-best individual but all individuals, with the validation dataset.

Among these individuals, the training-best individual is the one that obtained the lowest error in the training dataset. If only the training-best individual of each generation of the population is considered for evaluating the validation dataset, other individuals which do not obtain such a low error in the training dataset may not be considered. These individuals, despite obtaining a higher error in the training dataset, may have a lower error in the validation dataset, and therefore, they may be less overfitted. The main difference between this method and Backwarding is that this method proposes the evaluation of more than just the training-best individual with the validation dataset. A formal description of this algorithm is shown on Algorithm 1. This pseudocode starts by defining a random population which is evaluated with the training dataset on steps 4–6. After that, steps 8–13 evaluate the N individuals with the best training fitness with the validation dataset and choose the validation-best individual as the one who obtained the best fitness in the validation dataset. Once the validation is finished, step 15 checks the termination criteria if this is not fulfilled, a new population will be built on the step 15 which will go through a new evaluation on the training and validation datasets. Otherwise, if the termination criteria is fulfilled, the validation-best individual will be returned after being evaluated with the test dataset. It should be highlighted that the validation dataset is not used to perform the selection genetic operator, but only to select the best-so-far individual between the N individuals with the best training fitness.

The extreme case of this method would take place by evaluating all new individuals in each generation with the validation dataset. This is not a good strategy either, because the more individuals are used for validation evaluation, the more chances will be to find one that overfits the validation dataset.

Therefore, a balance should be found between evaluating too few or too many individuals. In this work, experiments have been conducted with different percentages of the population, as shown in Section 5. In addition, since the population size is also another important parameter, experiments also took into account that information, in order to study its impact on the overfitting of expressions.

An important difference between this work and the classic GP approach is that, once the evolutionary process is finished, the individual to be returned may not have to be the training-best individual in any generation, but the validation-best individual.

Algorithm 1: Pseudocode of the used algorithm

```
1 best_individual = empty
2 population = new random population
3
4 foreach new_individual in the population do
5       new_individual.training_fitness = fitness(training_dataset)
6 end
7
8 foreach individual of the N best individuals in the population do
9 individual.validation_fitness = fitness(validation_dataset)
10 if individual.validation_fitness is better than
    best_individual.validation_fitness then
11 best_individual = individual
12 end
13 end
14
15 if termination criteria is NOT satisfied then
16 Build new population through genetic operators based on training_fitness
17 Go to step 4
18 end
19
20 best_individual.test_fitness = fitness(test_dataset)
21 return Best_individual
```

Datasets

The experiments in this study were conducted using five different databases. On one hand, two of them were chosen from the scope of symbolic regression problems because they have been previously used in other studies about overfitting in GP (Archetti, Lanzeni, Messina, & Vanneschi, 2007; Gonçalves & Silva, 2013).

The first of the databases, Toxicity, contains 234 data points. Each data point is a vector of 627 elements: 626 molecular descriptor values identifying a drug, and the known median lethal dose. The last value is the target to be predicted from the 626 remaining values. This database is available online (Archetti et al., 2007; Gonçalves & Silva, 2013).

The second database, Bioavailability, contains 359 instances. Every instance is a vector consisting of 242 values: 241 molecular descriptor values identifying a drug, and the percentage of the initial orally submitted drug dose that effectively reaches the systemic blood circulation after passing through the liver. As previously mentioned, the last value is the target to be predicted from the previous ones (Archetti et al., 2007; Gonçalves & Silva, 2013).

On the other hand, the remaining three datasets correspond to classification problems, and they are well-known datasets used as benchmark in many different works. These databases were taken from UCI (Dheeru & Karra Taniskidou, 2017), which is a public repository of databases used by the ML community.

One of the problems to be solved involves the classification of breast cancer into two possible types: benign and malignant samples. The database has 699 cases: 458 benign (65.5%) and 241 malignant (34.5%). Each data point is characterised by 9 attributes that are considered to be continuous, although they take discrete values ranging between 1 and 10.

Another problem consists of classifying radar measurements from the ionosphere. These data were taken by a system in Goose Bay, Labrador, Canada. This system consists of a set of 16 high-frequency antennas with a transmission power of about 6.4 kilowatts. The good measurements are those that demonstrate evidence of some type of structure in the ionosphere. The bad measurements are the ones that do not demonstrate this kind of evidence, i.e. the signals pass through the ionosphere. From a set of 34 attributes, the objective is to predict whether there are structures or not. In this case, there are 351 instances.

The objective of the last problem is to detect whether heart disease is present or not. These data correspond to 13 measurements taken from 303 patients at the V.A. Hospital of Cleveland.

Table 1 shows a summary of the most important features of the 5 problems to be solved. All of these datasets were divided into three disjoint subsets with 33% of the data each, for training, validation, and test datasets. Also, in the case of classification problems, this division was stratified, so the same proportion of data points of each class were used for training, validation and test. Table 1 also shows the size of the training/validation/test datasets.

Table 1. Summary of used datasets.

Problem	Num. of inputs	Num. of data points	Type of problem	Mean inputs	StdDev. inputs	Mean outputs	StdDev. outputs	Size of subsets
Toxicity	626	234	Regression	386.67	11,540.59	1672.73	11,540.59	78
Bioavailability	242	359	Regression	16.12	371.45	66.40	371.45	120
Wisconsin Breast Cancer	9	699	Classification	3.15	2.90	2.70	2.90	233
Ionosphere	34	351	Classification	0.25	0.58	0.64	0.58	117
Heart Disease	13	303	Classification	0.40	0.38	0.46	0.38	101

Summing up, 2 regression and 3 classification problems were chosen for the benchmark dataset. Since in this paper both types of problems, regression and classification, are used to evaluate the system, two different measures have been defined to evaluate the fitness of each individual. For the regression problems, RMSE has been used as fitness measurement. However, in classification problems, individuals with a higher RMSE can show higher accuracy while individuals with lower RMSE can have lower accuracy. For this reason, the misclassification pattern rate was used for the classification problems instead of RMSE. However, in any classification system, some correct classifications can be due to chance. To measure the level of agreement apart from chance, Heidke skill score, or kappa, can be used. This skill can be an important fitness measurement, that should be maximized. Misclassification rate and kappa are closely related, since the higher is the first, the lower is the second. Since tournament was used as selection algorithm, when two individuals are randomly taken from the population, their fitness is compared and the one with better fitness is chosen for reproduction. If another selection algorithm such as wheel selection, which gives a probability based on the fitness, was used, then the use of kappa or misclassification rate would have a higher impact on the behaviour of the algorithm. In this work, the misclassification rate is used because, apart from being the most common fitness measure in GP, it reflects better the performance of the classification system.

In the Wisconsin Breast Cancer classification problem, the targets take values of 2 and 4. These values were changed into 0 and 1 respectively. In the other 2 classification problems, the targets were already 0 and 1. In these 3 problems, a threshold of 0 was applied to the output of each individual, so a negative output was classified as 0, and a positive output was classified as 1. The fitness of this individual is calculated as the number of incorrectly classified patterns divided by the number of patterns.

Results and discussion

This section outlines the experiments carried out, as well as the results.

As it was aforementioned, this work is focused on evaluating the influence of using a certain number of individuals to calculate the validation value, instead of using only the best individual. Consequently, different experiments were conducted by using different percentages of the best individuals of the population used in the validation. The experimentation tried to cover a wide range of possibilities by repeating the test for 0%, 5%, 10%, 15%, 20%, 30%, 40%, 50%, 60%, 80% and 100%. It may be highlighted that, when a 0% is used, no individual was evaluated with the validation dataset and, therefore, after the evolutionary process, the individual to be returned is the one which obtained the best result in the training dataset. Oppositely, 100% means that, in each generation, all individuals are evaluated with the validation dataset. With an intermediate percentage, a given amount of individuals are evaluated. In addition to these percentages, in other experiments, only the individual with the lowest training error was used to evaluate the validation dataset.

Taking into account that the population size is a very important parameter in a normal GP run, in this situation this parameter is even more relevant. The same percentage of individuals in a larger population will lead to the use of a greater number of individuals for validation. To study its influence, two very different values in terms of population size were used: 100 and 1000 individuals.

As described in Section 3, the system was run until reaching 1000 generations, when, the individual with the lowest error in the validation dataset, was returned as the solution.

For each combination of percentage, population size and different problem, 50 independent runs were carried out. The results shown herein correspond to the median and average of 50 runs performed. In each of these 50 runs, the database was randomly split into training, validation and test datasets, having each of them 33% of the patterns. The reason for that unusually high percent for validation and test is that it ensures that validation and test datasets do not have non-representative of patterns because the objective is to measure the impact of the validation.

The function sets used by GP contain only arithmetic operators (+, -, *, %). The operation % is the protected division, which returns a result of 1 when the second argument is 0. The terminal only contains those variables corresponding to the independent terms of the database, and an ephemeral random constant between 0 and 1.

In addition to the above-mentioned parameters, for each run, the following values were used for the different GP parameters:

- Selection algorithm: two-individual tournament.
- Creation algorithm: ramped half-and-half.
- Crossover rate: 95%.
- Mutation probability: 4%
- Maximum tree depth: 9

The experiments described in this section were carried out using a proprietary GP library written in C++ and the equipment of the Galician Supercomputing Center (CESGA).

Figure 1 shows the results in the test, therefore, the lower the value, the better it is. Since average results are susceptible to outliers, solid lines show the median of the results in the test of the individuals which provided the lowest error in the validation dataset in each of the 50 different runs for each percentage. In addition, dashed lines show the average values of these independent runs. The results obtained for the population size of both 100 (left graphs) and 1000 (right graphs), as well as for the 5 problems are shown separately. Note that the case in which only the training-best individual of the population is selected for validation corresponds to a percent of 1% for a population size of 100, and 0.1% for a population size of 1000, and therefore their values are plotted very close to 0. Also, horizontal lines in this graph show the case when no validation dataset was used. In this case, the training-best individual of the last generation is returned as the algorithm output and used for evaluation of the test dataset.

As observed in this figure, and as expected, in all cases where validation is not performed, the error in the test dataset is much higher than when validation is done. A common approach when using a validation dataset is using only the best individual in each generation. As can be seen in Figure 1, better results can be found in all of the cases if more than 1 individual is validated. However, the best percentage of population size for validation is different for each problem and population size.

As it was previously defined in Section 2, overfitting can be defined numerically as the difference between the error in the test dataset and the error in the training dataset (Gonçalves & Silva, 2013; Gonçalves et al., 2012). Figure 2 shows, in terms of the 5 problems and 2 population sizes, the different overfitting values obtained for each of the percentage values. As before, the horizontal line indicates overfitting in the case where no validation dataset is used. Solid lines correspond to median results, and dashed lines correspond to average results.

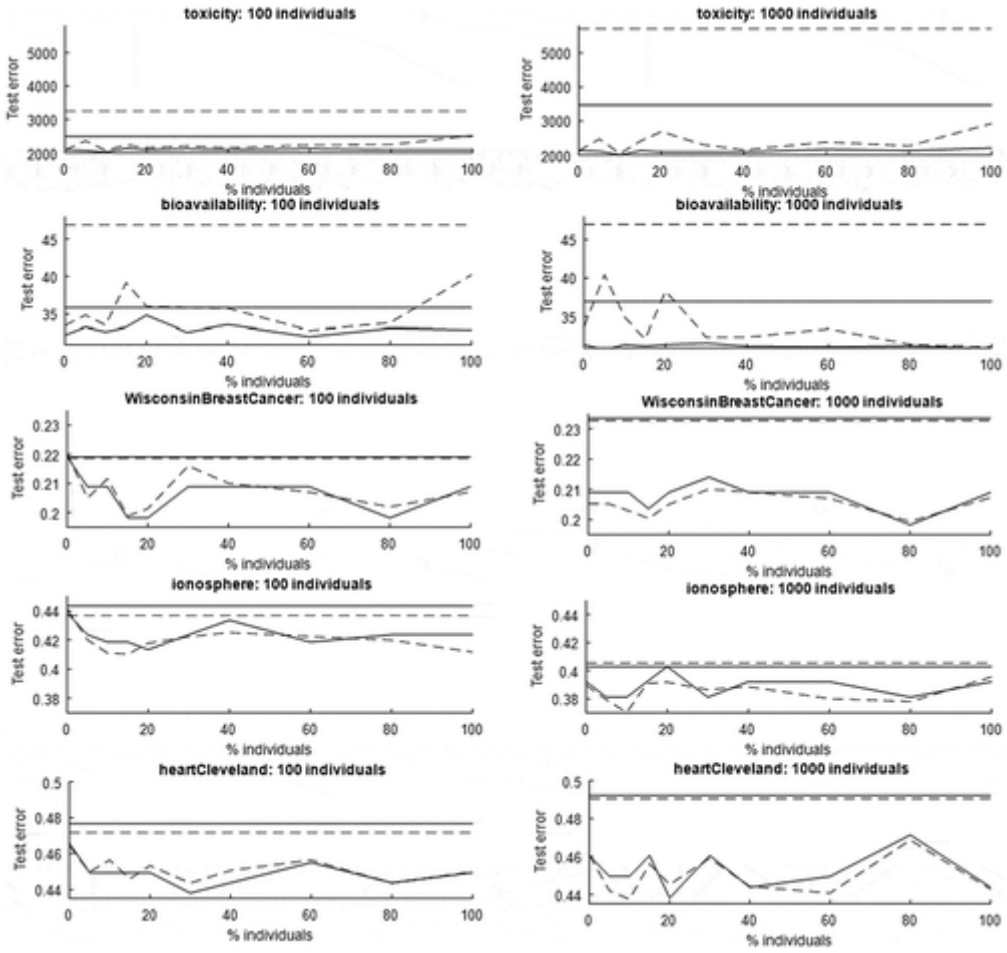

Figure 1. Test error for each problem and population size, considering different percentages of individuals to perform validation. Solid lines: median of the results in test for each percentage. Dashed lines: average results in test for each percentage. Horizontal lines: results without validation dataset (solid: median, dashed: average).

As was expected, overfitting gets the highest values when no validation dataset is used. However, its values are not always the lowest in those cases in which the best test results were returned (Figure 1). For example, in the case of the ionosphere problem, with a population size of 100 individuals, the best test results were obtained when evaluating 20% of the population. In the overfit graph, it gets its lowest value when all of the individuals are evaluated. This is happening because the evaluation of an individual with the validation dataset is independent of the training dataset, and it is possible to obtain an individual with a worse training result and a good validation. This possibility is higher as the percentage of individuals used for validation gets higher too.

In addition to Figure 2, a different measure of overfitting was calculated. This measure calculates the overfit for each generation (Vanneschi et al., 2010) and the results can be seen in Figure 3. This figure shows the overfit calculated for each generation when no validation was performed (dashed lines) and when validation was used with different percentages (solid lines). Although the solid lines in each graph are mixed and it is impossible to differentiate between percentages, the difference between using and not using validation is very clear. The values shown on this figure correspond to the median of the overfitting values of each of the different executions.

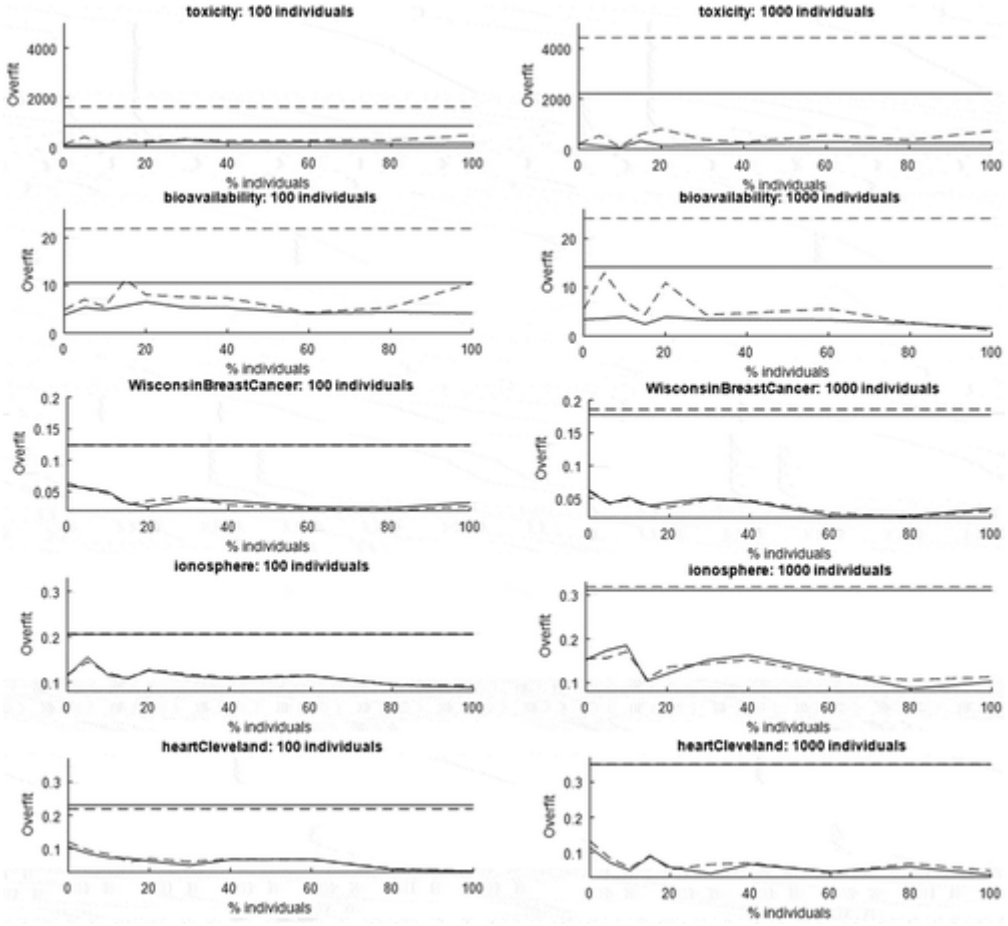

Figure 2. Overfitting for each problem and population size, considering different percentages of individuals to perform validation. Solid lines: median of the results in test for each percentage. Dashed lines: average results in test for each percentage. Horizontal lines: results without validation dataset (solid: median, dashed: average).

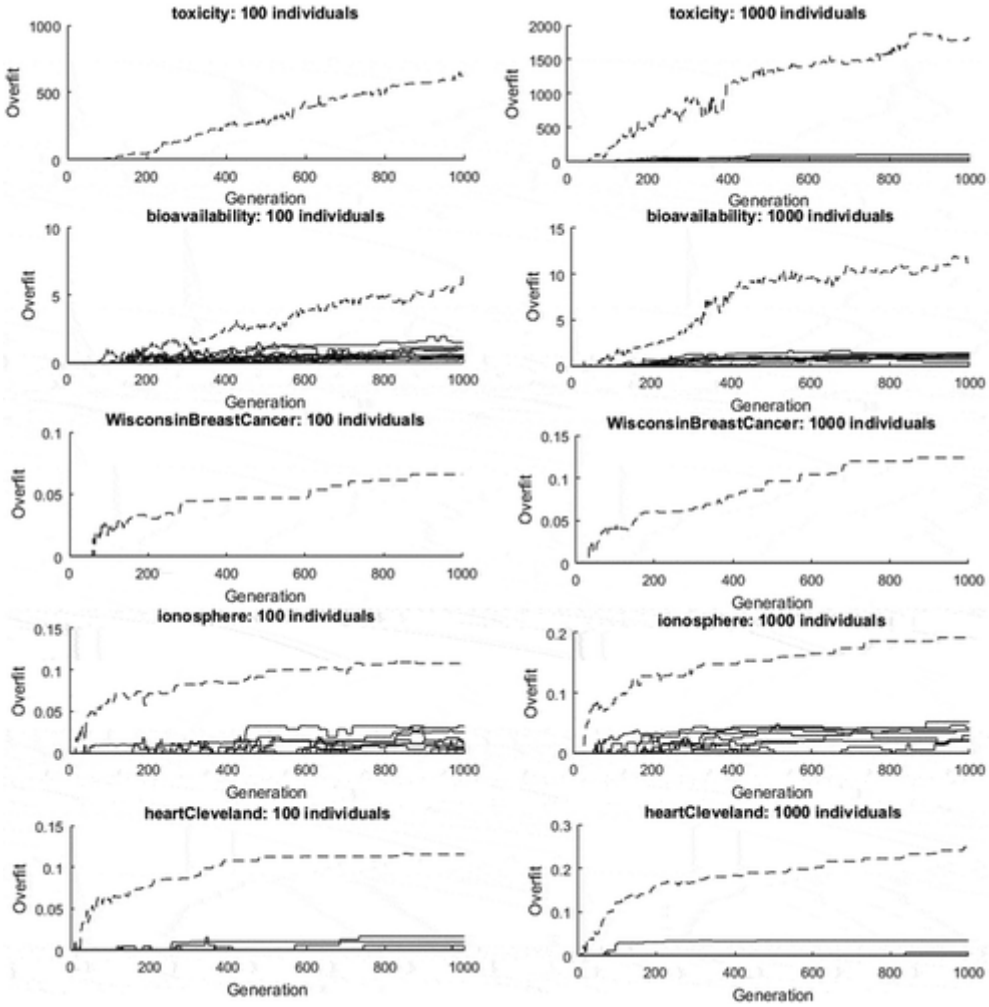

Figure 3. Overfitting on each generation. Solid lines: overfit calculated for each percentage. Dashed lines: overfit without validation dataset.

Figure 4 contains the representation for each problem and population size of the error obtained on the test dataset. The error is plotted for each generation with different cases: only the best individual is evaluated (dashed lines), no validation is performed (dotted lines), and with the best percentage found for each combination (solid lines). Instead of plotting the curves for all percent, which can be quite confusing, these 3 curves can clearly illustrate the process.

As it can be observed, when no validation is performed, the systems returns the worst test errors in most of the cases. In those cases in which it is not the worst, there is only a slight difference with the result using validation with only the best individual. Also, when the validation is carried out in the problems of toxicity, Wisconsin Breast Cancer and Heart Cleveland, the curve soon reaches the individual with the lowest error, with no decrease in test error from that point. Moreover, regarding the bioavailability problem, due to the aforementioned overfitting issue, the improvement of the test error seems to occur more slowly. This can be observed clearly when the population has fewer individuals (100) since it has to perform more generations to acquire an individual with a good validation score.

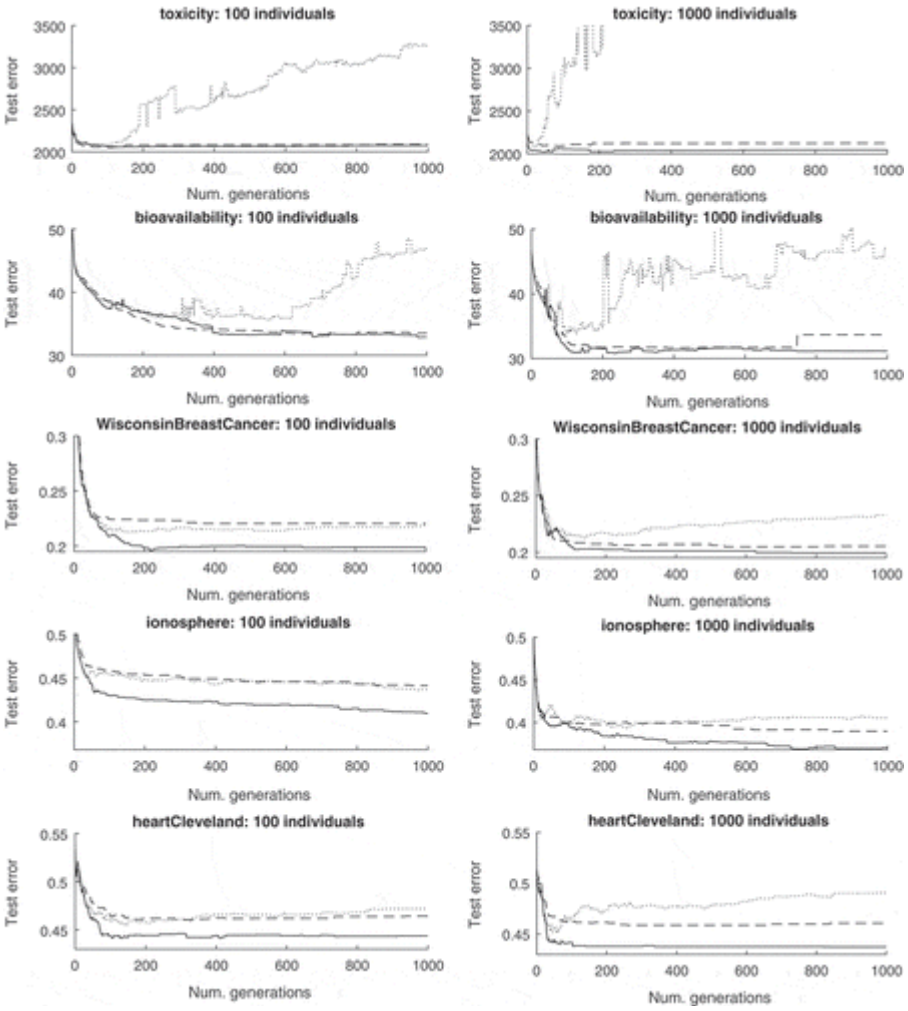

Figure 4. The evolution for the cases where no validation (0%) is performed, it is performed only with the best individual, and with the best percentage found. Dotted lines: results without validation is performed. Solid lines: results for the best percentage found for each combination. Dashed lined: results when only the best individual is evaluated.

To illustrate the development of the evolutionary process, Figure 5 shows for each problem, population size and percentage the average number of generations needed to find the individual with the lowest validation error. As in the previous graphs, solid lines represent median values while dashed lines represent average values, while horizontal lines represent the median (solid) and average (dashed) number of generations in the experiments with no validation. As can be seen, in toxicity and bioavailability problems the number of generations needed to obtain the best results without validation is the maximum number of generations (1000). This means that, for these two problems, not using a validation dataset makes the system to keep improving the training results until the last generation, while the test results were worsening. This can also be seen in Figure 4.

These graphs show that, except once, when the validation is performed, the individual to be returned was found after a longer period of time compared to when a single individual was used for validation. In addition, it seems that when using a higher percentage, the system takes more generations to find that individual. When using a higher percentage of individuals for evaluation on the validation dataset, more chances will be that the system finds individuals with lower validation error (i.e. higher accuracy) on a higher number of generations.

Moreover, in many of these graphs, the number of generations employed is substantially higher than in the case of using a single individual for validation. It seems that higher percentages have an impact on GP behaviour and its convergence is slower. However, many other factors may influence this effect.

Related to Figure5, the number of data points evaluated to find each solution can be calculated, which gives a measure of the computational effort needed by the process. When working with GP, computational efficiency plays a crucial role and, therefore, it needs to be evaluated. In order to measure the computational cost, Figure 6 shows the number of data points evaluated in the training and validation process in order to obtain the result. As in previous graphs, solid lines show median values, while dashed lines show average values. Obviously, the number of data points evaluated is significantly higher when the population size is 1000 than when it is 100, due to the number of patterns evaluated is around 10 times higher in the first case

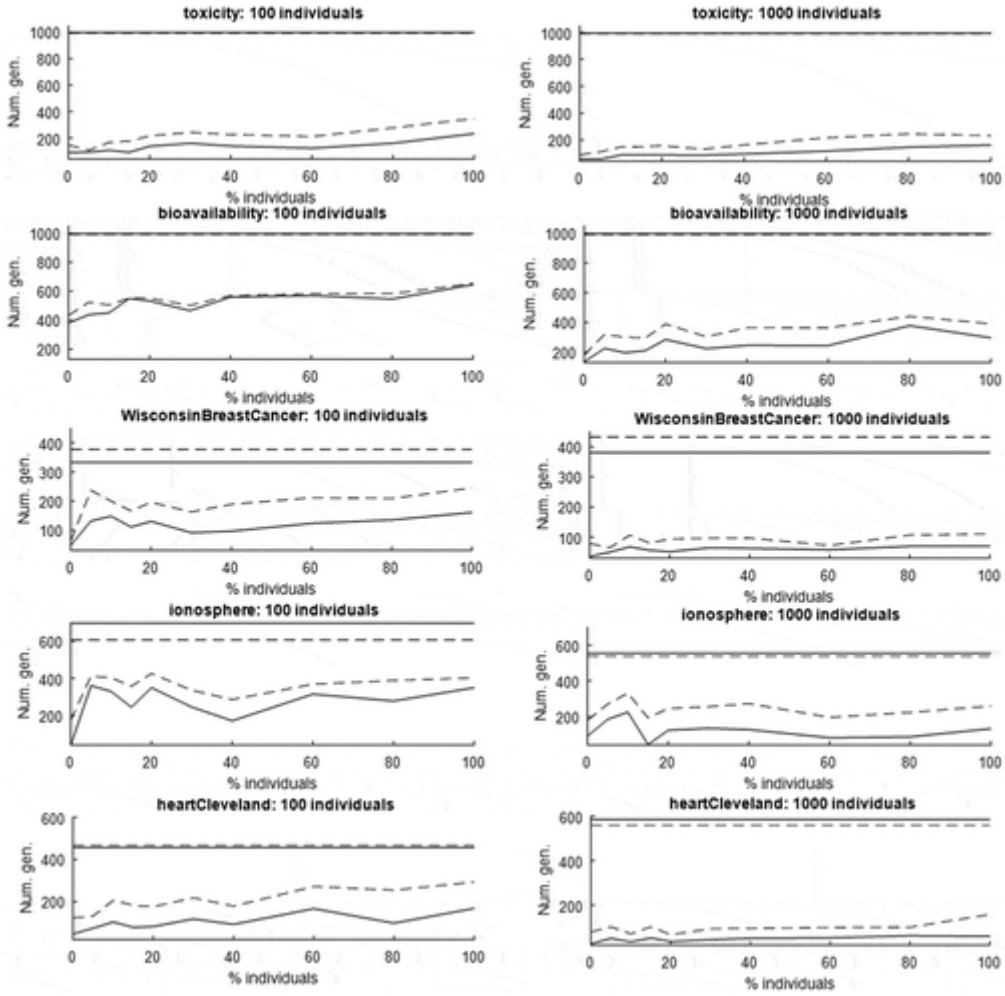

Figure 5. Average number of generations in which the individual to be returned for each percentage was found. Solid lines: median of the results for each percentage. Dashed lines: average results for each percentage. Horizontal lines: results without validation dataset (solid: median, dashed: average).

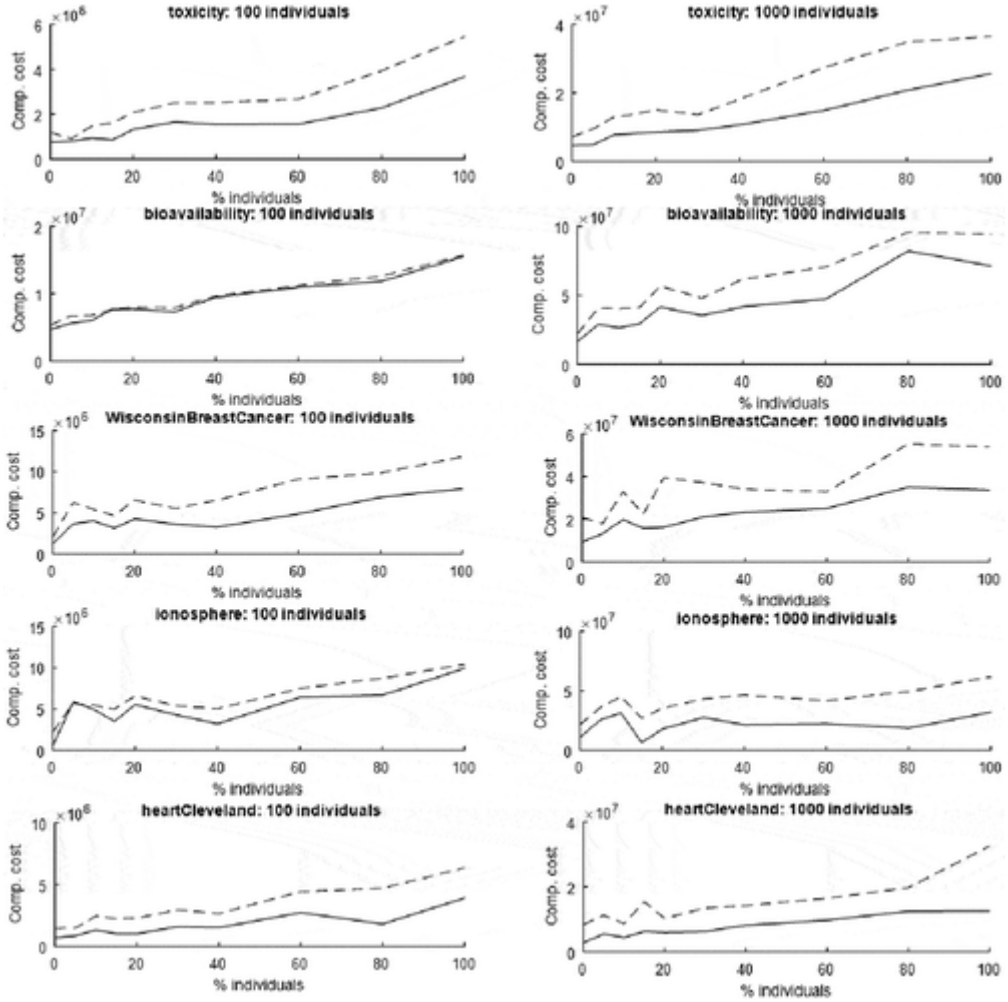

Figure 6. Computational cost for each percentage. Solid lines: median of the results for each percentage. Dashed lines: average results for each percentage.

This figure shows that the computational cost needed in these experiments for finding the validation-best individual was lower when the percentage of individuals used for validation was lower too. Also, these computational costs from the experiments seem to linearly increase as the percentage increases. However, choosing a low percentage due to its low computational cost is not a good idea. Figure 1 shows that, when the percentage is very low with a low computational cost, the returned results are not the best ones. Therefore, a trade-off between results and computational efficiency is needed to obtain the best results.

Related to the latter, Figure 7 shows the median and average error for each configuration when no validation dataset was used when the number of data points shown on 6 were evaluated. As in previous graphs, solid lines show median values and dashed lines show average values. As can be seen in comparison with Figure 1, the error obtained with the same computational cost is lower using a validation dataset.

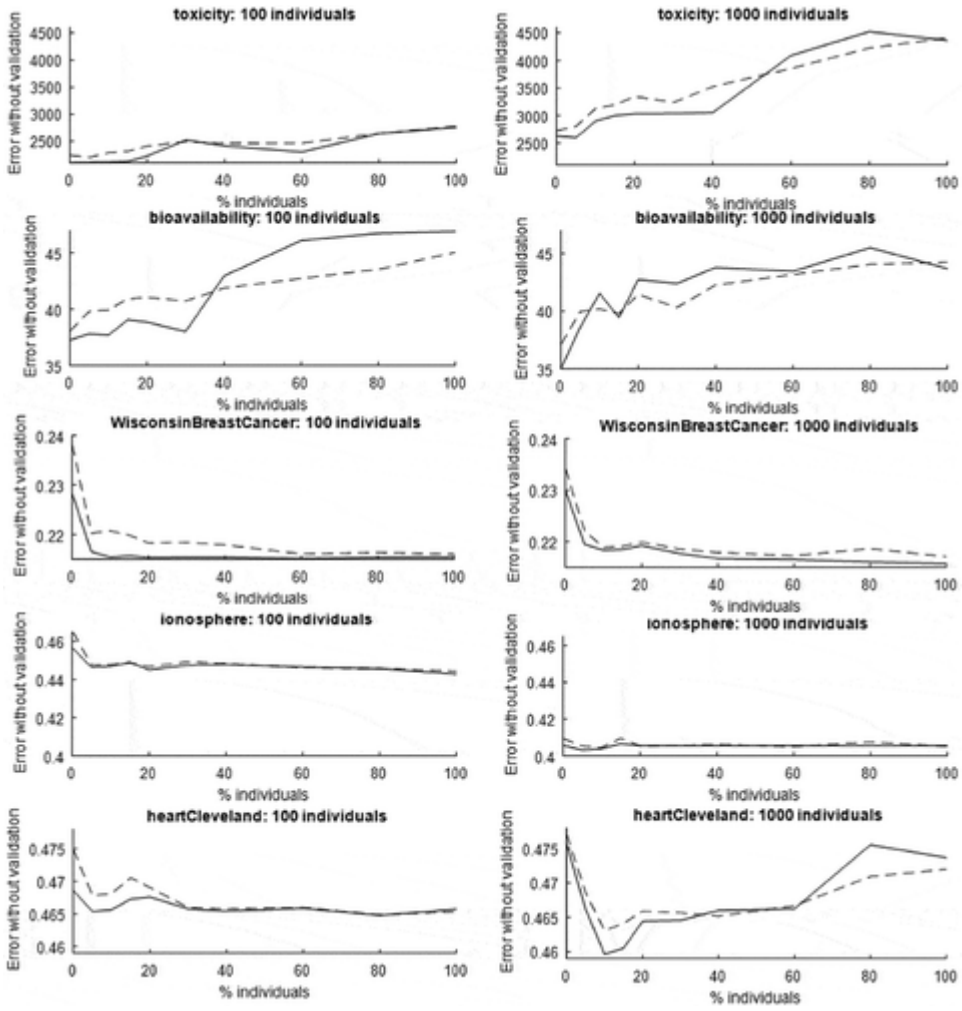

Figure 7. Error with no validation when the process is prematurely stopped. Solid lines: median of the results for each percentage. Dashed lines: average results for each percentage.

The next study carried out refers to the complexity of the resulting individuals. Figure 8 shows, for each problem and population size, the average number of nodes of the returned individual for each percentage. As before, in that figure, each horizontal line indicates the number of nodes in the case of not using a validation dataset; solid lines show median values and dashed lines show average values.

As expected, the number of nodes of the individuals in which no validation is performed, and therefore overfitted, is much higher than in the cases in which validation is performed. Comparing this graph to Figure 1, it can be observed that, for a given problem, obtaining a lower test error generally involves using fewer nodes, although this statement cannot be generalised, because, sometimes in order to improve an expression it is necessary to make it more complex.

Tables 2–11 summarise the most important results according to the information shown on Figures 1–8

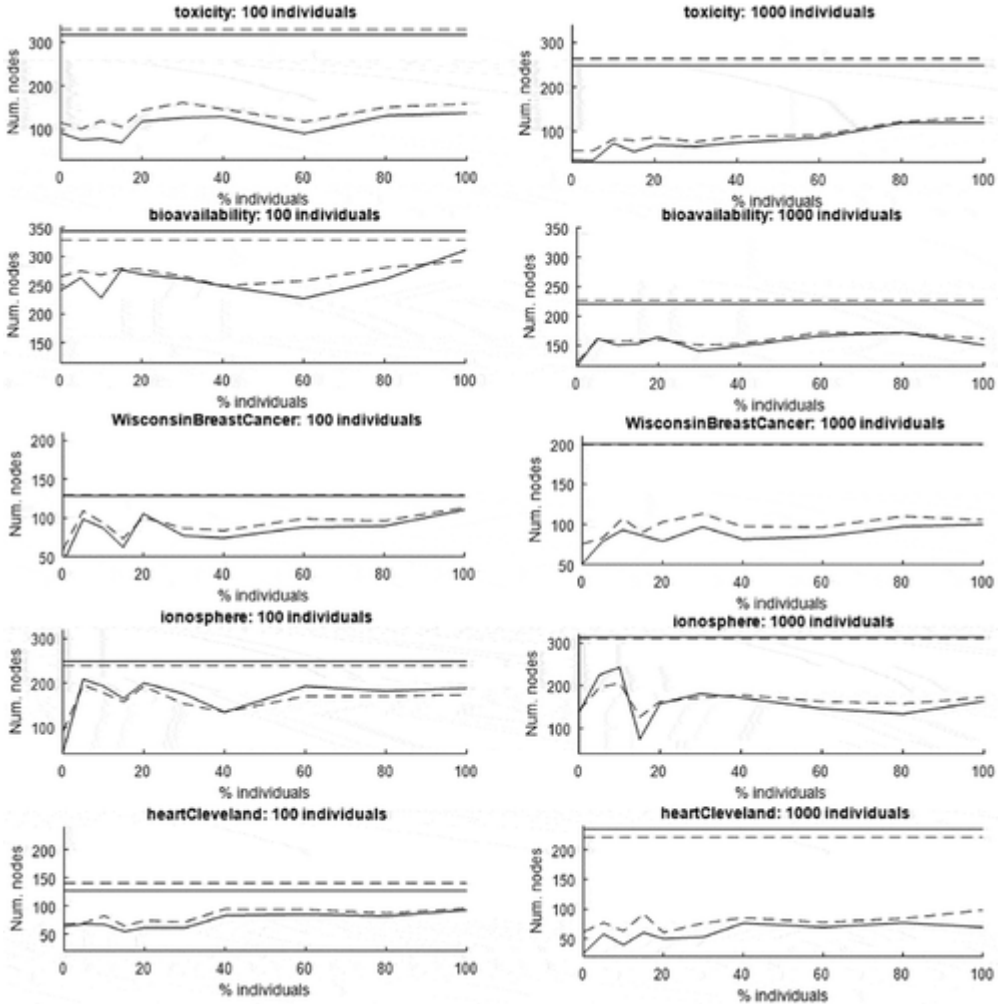

Figure 8. Average number of nodes of the individual to be returned for each percentage.Solid lines: median of the results for each percentage. Dashed lines: average results for each percentage. Horizontal lines: results without validation dataset (solid: median, dashed: average).

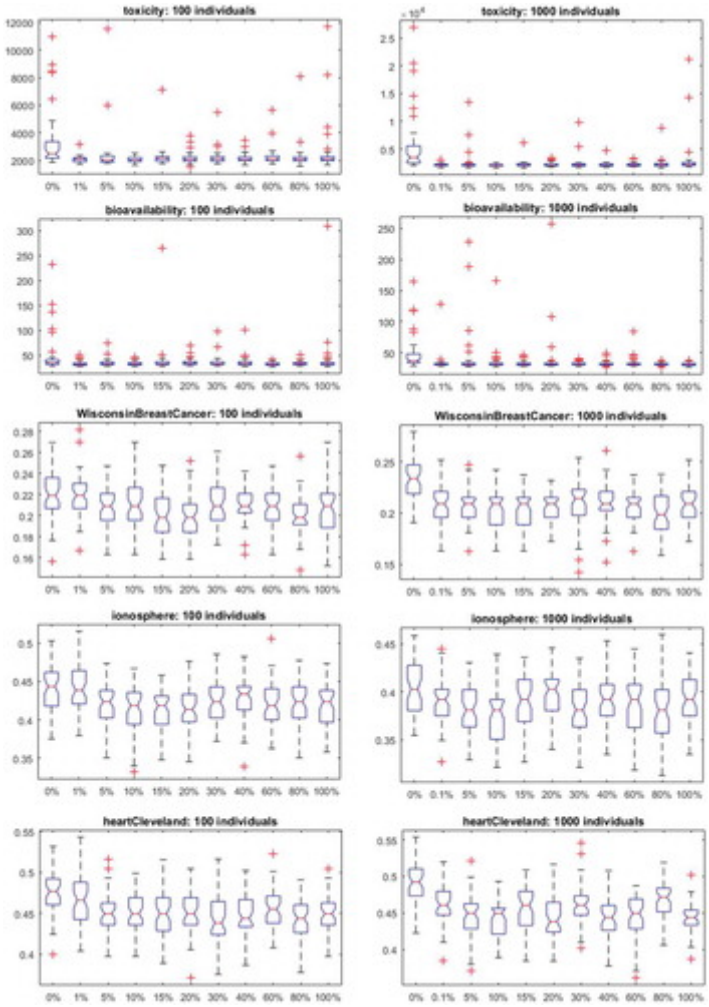

Figure 9. Boxplots of test results for the different problems and configurations.

Table 2. Summary of the most important results for the toxicity problem with 100 individuals according to the size of the partition used for validation.

Num. individuals to validate (Percentage of population)	0 (0%)	1 (1%)	5 (5%)	10 (10%)	15 (15%)	20 (20%)	30 (30%)	40 (40%)	60 (60%)	80 (80%)	100 (100%)
Test results, median	2494.76	2097.96	2074.11	2026.34	2138.02	2098.44	2121.08	2092.99	2123.47	2091.19	2094.18
Test results, average	3253.20	2087.38	2364.91	2074.78	2244.09	2155.12	2211.19	2154.13	2237.34	2247.78	2525.81
Test results, StdDev	1989.55	233.32	1447.76	196.88	739.43	391.60	548.83	307.19	604.59	897.12	1676.29
Overfitting, median	854.83	74.97	78.55	90.80	177.46	188.93	296.64	185.39	200.69	142.24	150.67
Overfitting, average	1640.97	116.09	417.36	107.82	286.27	245.50	314.11	267.14	275.36	277.39	494.79
Overfitting, StdDev	2028.11	380.07	1527.65	302.98	757.57	554.24	656.81	448.47	684.61	938.32	1734.03
Num. generations, median	998.00	91.50	94.50	106.50	93.00	138.00	160.00	139.50	123.50	160.50	233.50
Num. generations, average	995.47	147.02	108.38	167.24	177.26	217.31	242.31	226.80	211.58	277.69	347.63
Num. generations, StdDev	5.86	172.21	97.02	189.88	229.05	216.99	225.75	233.26	231.07	272.53	293.43
Comp. cost, median (x10000)	799.20	74.71	80.08	94.28	86.06	132.61	165.99	155.67	157.12	228.68	368.17
Comp. cost, average (x10000)	797.18	119.56	91.72	147.55	163.20	208.26	250.85	252.40	268.28	394.62	547.34
Comp. cost, StdDev (x10000)	4.69	139.09	81.35	166.52	209.69	207.01	232.75	258.46	291.61	385.91	460.68
Num. nodes, median	317.00	93.00	76.00	79.00	70.00	119.00	127.00	130.00	91.00	132.00	138.00
Num. nodes, average	330.43	116.10	101.92	120.08	106.04	144.12	161.98	146.96	118.00	151.38	159.48
Num. nodes, StdDev	127.05	110.16	88.00	108.85	106.96	113.13	120.78	107.85	95.80	118.69	114.15

Table 3. Summary of the most important results for the toxicity problem with 1000 individuals according to the size of the partition used for validation.

Num. individuals to validate (Percentage of population)	0 (0%)	1 (0.1%)	50 (5%)	100 (10%)	150 (15%)	200 (20%)	300 (30%)	400 (40%)	600 (60%)	800 (80%)	1000 (100%)
Test results, median	3463.90	2130.54	2112.28	2008.37	2132.00	2095.28	2089.12	2091.10	2167.43	2130.15	2209.54
Test results, average	5717.03	2125.42	2472.91	2046.01	2415.81	2687.92	2308.90	2150.85	2382.32	2270.70	2926.34
Test results, StdDev	5745.12	241.48	1787.11	204.44	1033.36	3419.40	1196.12	451.70	990.74	1007.55	3225.67
Overfitting, median	2208.44	209.25	134.39	62.34	329.16	146.93	172.65	250.05	286.77	260.51	266.16
Overfitting, average	4427.59	210.37	535.77	103.15	587.96	802.62	396.52	279.37	553.32	369.27	722.48
Overfitting, StdDev	5777.87	403.10	1936.64	403.13	1127.77	3486.34	1271.02	617.72	1063.83	1100.12	2740.41
Num. generations, median	996.00	58.00	57.00	88.00	88.50	88.50	87.00	95.50	117.00	146.00	162.50
Num. generations, average	990.66	89.34	112.00	147.46	149.48	156.90	131.18	163.36	215.98	245.56	232.21
Num. generations, StdDev	17.26	129.49	208.91	204.40	196.43	184.38	132.85	188.09	235.40	245.54	206.42
Comp. cost, median (x10000)	7976.00	472.45	486.33	780.53	819.37	853.83	907.28	1069.22	1489.16	2081.52	2566.95
Comp. cost, average (x10000)	7933.26	723.42	947.51	1301.99	1377.64	1506.33	1362.78	1821.11	2738.28	3491.32	3661.37
Comp. cost, StdDev (x10000)	138.05	1036.90	1751.75	1792.59	1798.29	1759.02	1369.72	2084.09	2970.69	3476.89	3240.78
Num. nodes, median	248.00	35.00	34.00	73.00	54.00	69.00	66.00	74.50	85.00	119.00	119.00
Num. nodes, average	264.03	56.50	56.62	82.58	79.56	86.65	76.42	88.58	91.29	122.15	131.04
Num. nodes, StdDev	76.52	64.05	63.68	62.94	68.78	66.37	56.00	60.50	67.00	83.74	65.01

Table 4. Summary of the most important results for the bioavailability problem with 100 individuals according to the size of the partition used for validation.

Num. individuals to validate (Percentage of population)	0 (0%)	1 (1%)	5 (5%)	10 (10%)	15 (15%)	20 (20%)	30 (30%)	40 (40%)	60 (60%)	80 (80%)	100 (100%)
Test results, median	35.88	32.30	33.31	32.70	33.28	34.85	32.65	33.62	32.07	33.17	32.96
Test results, average	46.91	33.49	34.87	33.53	39.18	36.03	35.84	35.77	32.87	33.87	40.22
Test results, StdDev	36.56	4.68	7.13	3.59	33.13	6.90	10.64	10.35	3.77	4.04	39.96
Overfitting, median	10.56	3.80	5.35	4.95	5.70	6.55	5.32	5.29	4.18	4.38	4.19
Overfitting, average	21.95	4.91	7.06	5.49	11.26	8.06	7.54	7.37	4.30	5.34	10.52
Overfitting, StdDev	37.09	5.14	7.48	3.96	33.04	7.17	10.88	10.89	4.05	4.63	40.70
Num. generations, median	996.00	380.00	438.50	450.50	549.00	532.50	465.50	559.00	569.50	546.00	646.00
Num. generations, average	992.58	433.51	523.64	504.52	552.47	551.44	501.56	569.48	582.02	581.90	653.69
Num. generations, StdDev	9.92	244.58	286.19	288.36	295.28	282.83	273.84	266.87	271.82	247.56	261.60
Comp. cost, median (x10000)	1216.34	469.32	562.12	604.11	768.35	776.78	734.27	947.52	1099.92	1183.71	1552.80
Comp. cost, average (x10000)	1212.17	535.23	671.01	676.39	773.25	804.35	791.03	965.25	1124.06	1261.40	1571.27
Comp. cost, StdDev (x10000)	12.10	301.27	366.04	385.83	412.45	411.80	431.02	451.55	524.06	535.71	627.84
Num. nodes, median	344.00	241.00	263.00	228.00	277.00	269.00	261.50	249.00	227.00	260.00	311.00
Num. nodes, average	328.98	265.18	275.08	267.48	279.37	277.26	266.14	249.02	257.58	280.64	292.69
Num. nodes, StdDev	108.85	95.72	112.68	113.25	110.15	109.18	100.84	93.67	97.55	98.16	101.01

Table 5. Summary of the most important results for the bioavailability problem with 1000 individuals according to the size of the partition used for validation.

Num. individuals to validate (Percentage of population)	0 (0%)	1 (0.1%)	50 (5%)	100 (10%)	150 (15%)	200 (20%)	300 (30%)	400 (40%)	600 (60%)	800 (80%)	1000 (100%)
Test results, median	36.93	31.45	30.77	31.47	31.27	31.46	31.64	31.25	31.15	31.29	30.89
Test results, average	46.98	33.68	40.42	34.89	32.16	38.29	32.38	32.42	33.46	31.47	31.14
Test results, StdDev	26.55	14.02	36.14	19.36	3.76	33.76	3.14	4.71	8.43	2.11	2.53
Overfitting, median	14.13	3.46	3.64	3.93	2.47	3.91	3.43	3.41	3.32	2.66	1.60
Overfitting, average	24.15	5.57	12.95	7.17	4.43	10.96	4.38	4.74	5.61	2.72	1.27
Overfitting, StdDev	27.01	14.54	35.97	19.51	5.00	34.24	3.96	5.39	9.11	3.31	7.15
Num. generations, median	996.00	136.00	226.00	197.00	210.50	286.00	224.00	246.50	245.00	379.00	297.00
Num. generations, average	991.46	182.21	319.38	301.62	295.70	389.98	304.62	365.02	364.80	441.88	391.55
Num. generations, StdDev	10.50	139.60	248.04	226.59	240.50	304.53	254.46	274.14	258.49	298.15	266.81
Comp. cost, median (x10000)	12,163.40	1673.02	2903.33	2649.24	2954.66	4178.72	3541.50	4187.70	4742.88	8223.20	7152.00
Comp. cost, average (x10000)	12,107.97	2237.30	4097.66	4049.06	4144.90	5692.67	4810.46	6193.06	7052.55	9583.92	9421.22
Comp. cost, StdDev (x10000)	128.13	1704.72	3172.40	3031.79	3359.84	4433.96	4005.20	4638.44	4983.62	6452.00	6403.56
Num. nodes, median	221.00	115.00	162.00	151.00	153.00	165.00	141.00	149.00	167.00	172.00	151.00
Num. nodes, average	226.65	121.77	159.20	157.74	158.52	160.36	151.08	152.86	172.39	171.48	161.55
Num. nodes, StdDev	46.43	48.32	48.71	49.10	66.41	53.98	58.63	52.64	61.11	55.85	45.69

Table 6. Summary of the most important results for the Wisconsin Breast Cancer problem with 100 individuals according to the size of the partition used for validation.

Num. individuals to validate (Percentage of population)	0 (0%)	1 (1%)	5 (5%)	10 (10%)	15 (15%)	20 (20%)	30 (30%)	40 (40%)	60 (60%)	80 (80%)	100 (100%)
Test results, median	0.22	0.22	0.21	0.21	0.20	0.20	0.21	0.21	0.21	0.20	0.21
Test results, average	0.22	0.22	0.21	0.21	0.20	0.20	0.22	0.21	0.21	0.20	0.21
Test results, StdDev	0.05	0.03	0.03	0.04	0.04	0.03	0.04	0.03	0.03	0.03	0.04
Overfitting, median	0.12	0.06	0.06	0.05	0.03	0.03	0.04	0.04	0.03	0.02	0.03
Overfitting, average	0.12	0.06	0.05	0.05	0.03	0.04	0.04	0.03	0.02	0.02	0.03
Overfitting, StdDev	0.07	0.05	0.04	0.06	0.05	0.05	0.05	0.05	0.05	0.04	0.06
Num. generations, median	333.50	48.00	131.50	148.00	111.00	130.00	91.50	96.50	123.50	135.00	161.00
Num. generations, average	376.96	63.76	236.82	200.28	166.06	195.84	162.44	188.86	211.46	208.68	245.24
Num. generations, StdDev	280.97	70.44	216.54	203.82	158.76	200.69	180.60	227.93	230.45	204.95	209.94
Comp. cost, median (x10000)	796.11	122.56	366.84	403.40	313.55	426.30	362.61	325.44	487.70	689.16	794.96
Comp. cost, average (x10000)	899.54	207.05	625.47	542.25	467.65	653.32	553.51	653.47	909.04	985.12	1179.72
Comp. cost, StdDev (x10000)	668.71	230.38	555.73	525.42	427.99	661.30	549.29	749.17	939.54	869.33	969.30
Num. nodes, median	128.50	42.00	98.50	87.00	62.00	105.50	77.00	74.00	88.00	89.50	110.50
Num. nodes, average	129.70	58.72	109.22	93.54	73.32	100.58	86.82	83.76	99.16	96.40	113.16
Num. nodes, StdDev	53.39	46.10	66.22	47.82	46.06	56.07	51.35	46.89	57.40	47.23	44.71

Table 7. Summary of the most important results for the Wisconsin Breast Cancer problem with 1000 individuals according to the size of the partition used for validation.

Num. individuals to validate (Percentage of population)	0 (0%)	1 (0.1%)	50 (5%)	100 (10%)	150 (15%)	200 (20%)	300 (30%)	400 (40%)	600 (60%)	800 (80%)	1000 (100%)
Test results, median	0.23	0.21	0.21	0.21	0.21	0.21	0.21	0.21	0.21	0.20	0.21
Test results, average	0.23	0.21	0.21	0.20	0.20	0.21	0.21	0.21	0.21	0.20	0.21
Test results, StdDev	0.04	0.03	0.03	0.03	0.03	0.03	0.04	0.03	0.03	0.03	0.03
Overfitting, median	0.18	0.06	0.04	0.05	0.04	0.04	0.05	0.04	0.02	0.02	0.03
Overfitting, average	0.19	0.06	0.04	0.05	0.04	0.04	0.05	0.05	0.03	0.02	0.03
Overfitting, StdDev	0.06	0.05	0.05	0.06	0.05	0.05	0.06	0.05	0.04	0.05	0.05
Num. generations, median	380.50	36.00	50.50	68.00	56.50	52.00	65.00	63.00	58.50	70.00	70.50
Num. generations, average	432.14	79.22	65.50	106.34	80.78	94.38	96.26	96.80	73.10	107.16	110.88
Num. generations, StdDev	280.24	147.19	41.65	129.07	75.65	121.21	115.65	122.16	44.34	131.26	126.96
Comp. cost, median (x10000)	9079.70	929.10	1322.62	1984.36	1567.74	1619.94	2120.37	2329.32	2504.39	3509.24	3376.80
Comp. cost, average (x10000)	10,308.73	2081.66	1791.27	3295.08	2241.73	3958.34	3703.58	3413.03	3292.99	5528.53	5391.62
Comp. cost, StdDev (x10000)	6669.67	3603.82	1151.16	3674.55	2054.75	5915.23	4371.80	4009.86	2619.01	6971.25	5895.16
Num. nodes, median	200.00	51.00	79.00	93.00	85.50	79.00	97.00	81.50	85.00	97.50	100.00
Num. nodes, average	199.56	75.74	82.76	107.26	89.14	103.14	113.20	97.46	96.74	110.16	105.74
Num. nodes, StdDev	67.09	75.22	43.17	59.40	49.16	67.86	69.59	48.19	48.16	51.00	53.0

Table 8. Summary of the most important results for the ionosphere problem with 100 individuals according to the size of the partition used for validation.

Num. individuals to validate (Percentage of population)	0 (0%)	1 (1%)	5 (5%)	10 (10%)	15 (15%)	20 (20%)	30 (30%)	40 (40%)	60 (60%)	80 (80%)	100 (100%)
Test results, median	0.44	0.44	0.42	0.42	0.42	0.41	0.42	0.43	0.42	0.42	0.42
Test results, average	0.44	0.44	0.42	0.41	0.41	0.42	0.42	0.43	0.42	0.42	0.41
Test results, StdDev	0.07	0.06	0.05	0.06	0.06	0.05	0.06	0.05	0.06	0.06	0.06
Overfitting, median	0.21	0.11	0.16	0.12	0.10	0.13	0.11	0.11	0.11	0.09	0.10
Overfitting, average	0.21	0.12	0.15	0.12	0.11	0.13	0.12	0.11	0.12	0.10	0.10
Overfitting, StdDev	0.07	0.09	0.07	0.08	0.08	0.06	0.08	0.06	0.07	0.09	0.07
Num. generations, median	697.00	41.00	361.00	329.50	200.00	350.00	245.00	171.00	315.00	278.00	349.50
Num. generations, average	607.54	175.54	412.66	403.32	355.38	426.10	336.06	285.96	368.56	388.78	396.62
Num. generations, StdDev	263.14	260.40	323.29	346.89	358.53	333.46	308.19	296.03	322.63	358.85	318.50
Comp. cost, median (x10000)	830.62	52.87	586.56	496.28	349.87	553.16	429.10	321.70	641.24	668.23	995.23
Comp. cost, average (x10000)	724.16	218.74	571.71	545.10	495.49	657.85	540.24	503.84	750.74	874.40	1030.01
Comp. cost, StdDev (x10000)	313.14	320.40	420.26	457.57	489.00	488.11	481.45	497.19	604.13	748.81	761.14
Num. nodes, median	249.00	43.50	208.50	193.00	148.00	200.00	175.50	134.00	192.50	182.00	188.50
Num. nodes, average	239.34	90.32	195.40	177.76	156.92	192.50	152.90	134.52	170.08	170.30	172.44
Num. nodes, StdDev	70.85	93.76	104.56	105.57	103.69	101.35	94.19	97.45	104.91	102.09	94.87

Table 9. Summary of the most important results for the ionosphere problem with 1000 individuals according to the size of the partition used for validation.

Num. individuals to validate (Percentage of population)	0 (0%)	1 (0.1%)	50 (5%)	100 (10%)	150 (15%)	200 (20%)	300 (30%)	400 (40%)	600 (60%)	800 (80%)	1000 (100%)
Test results, median	0.40	0.39	0.38	0.38	0.39	0.40	0.38	0.39	0.39	0.38	0.39
Test results, average	0.41	0.39	0.38	0.37	0.39	0.39	0.39	0.39	0.38	0.38	0.40
Test results, StdDev	0.06	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.06	0.06	0.05
Overfitting, median	0.31	0.15	0.17	0.19	0.10	0.12	0.15	0.16	0.13	0.09	0.10
Overfitting, average	0.32	0.16	0.16	0.17	0.11	0.14	0.14	0.15	0.12	0.11	0.11
Overfitting, StdDev	0.08	0.11	0.09	0.09	0.11	0.10	0.08	0.08	0.09	0.09	0.10
Num. generations, median	554.50	90.50	184.50	224.50	42.50	123.50	134.00	127.00	81.00	86.50	132.00
Num. generations, average	536.46	178.76	269.98	331.36	191.74	245.80	255.00	272.48	195.56	223.36	259.04
Num. generations, StdDev	279.36	224.05	272.71	289.21	256.01	288.31	278.25	306.01	224.19	262.36	299.40
Comp. cost, median (x10000)	6610.45	1089.91	2558.40	3134.40	627.44	1770.39	2753.02	2141.93	2225.48	1853.25	3196.00
Comp. cost, average (x10000)	6395.77	2157.91	3603.97	4507.27	2685.17	3527.13	4303.32	4620.28	4176.74	4939.18	6171.10
Comp. cost, StdDev (x10000)	3324.38	2658.05	3631.69	3794.62	3533.03	4140.25	4427.71	5135.45	4695.37	5691.46	7004.57
Num. nodes, median	313.50	137.50	225.00	243.00	74.50	158.00	181.00	171.50	146.00	133.50	163.50
Num. nodes, average	310.56	141.88	194.86	206.18	126.46	162.82	171.22	178.30	162.84	157.58	172.40
Num. nodes, StdDev	70.36	126.35	133.79	130.57	125.27	127.79	129.10	126.04	125.49	132.56	137.70

Table 10. Summary of the most important results for the heart disease problem with 100 individuals according to the size of the partition used for validation.

Num. individuals to validate (Percentage of population)	0 (0%)	1 (1%)	5 (5%)	10 (10%)	15 (15%)	20 (20%)	30 (30%)	40 (40%)	60 (60%)	80 (80%)	100 (100%)
Test results, median	0.48	0.47	0.45	0.45	0.45	0.45	0.44	0.44	0.46	0.44	0.45
Test results, average	0.47	0.46	0.45	0.46	0.45	0.45	0.44	0.45	0.46	0.44	0.45
Test results, StdDev	0.05	0.05	0.04	0.04	0.04	0.05	0.05	0.04	0.04	0.04	0.04
Overfitting, median	0.23	0.11	0.09	0.07	0.07	0.06	0.05	0.07	0.07	0.04	0.03
Overfitting, average	0.22	0.12	0.09	0.08	0.06	0.07	0.06	0.07	0.07	0.04	0.03
Overfitting, StdDev	0.07	0.09	0.06	0.09	0.08	0.08	0.08	0.07	0.07	0.06	0.06
Num. generations, median	457.00	47.50	74.50	104.00	78.50	82.50	118.50	92.50	167.00	98.50	168.50
Num. generations, average	466.78	123.54	131.82	206.18	183.22	176.52	218.52	179.28	272.38	254.64	293.58
Num. generations, StdDev	241.98	183.38	173.10	240.35	223.02	221.42	239.01	193.26	273.84	291.66	250.38
Comp. cost, median (x10000)	471.74	71.24	86.94	132.21	103.84	102.71	158.94	152.29	273.84	182.09	391.79
Comp. cost, average (x10000)	481.81	143.27	148.37	245.73	224.06	230.01	295.95	265.87	445.67	471.92	637.26
Comp. cost, StdDev (x10000)	249.24	192.68	185.86	274.66	271.13	286.03	326.37	283.21	446.36	531.93	541.35
Num. nodes, median	127.00	63.50	68.00	67.00	54.50	61.50	61.00	83.00	85.00	82.00	92.50
Num. nodes, average	140.72	68.02	69.82	82.68	64.28	74.16	71.40	94.96	93.82	87.16	95.76
Num. nodes, StdDev	65.45	53.47	39.38	58.78	47.22	56.77	49.59	66.96	60.44	49.45	57.56

Table 11. Summary of the most important results for the heart disease problem with 1000 individuals according to the size of the partition used for validation.

Num. individuals to validate (Percentage of population)	0 (0%)	1 (0.1%)	50 (5%)	100 (10%)	150 (15%)	200 (20%)	300 (30%)	400 (40%)	600 (60%)	800 (80%)	1000 (100%)
Test results, median	0.49	0.46	0.45	0.45	0.46	0.44	0.46	0.44	0.45	0.47	0.44
Test results, average	0.49	0.46	0.44	0.44	0.46	0.45	0.46	0.44	0.44	0.47	0.44
Test results, StdDev	0.05	0.04	0.05	0.04	0.05	0.05	0.05	0.04	0.05	0.04	0.04
Overfitting, median	0.35	0.11	0.07	0.05	0.09	0.06	0.04	0.07	0.05	0.06	0.04
Overfitting, average	0.35	0.13	0.08	0.06	0.09	0.05	0.07	0.07	0.04	0.07	0.05
Overfitting, StdDev	0.10	0.09	0.10	0.09	0.09	0.10	0.10	0.09	0.09	0.07	0.08
Num. generations, median	586.00	26.00	51.00	35.00	52.50	36.50	45.00	51.50	53.50	63.00	60.00
Num. generations, average	558.94	79.86	102.80	71.04	103.00	68.04	93.66	96.24	99.06	101.36	158.20
Num. generations, StdDev	251.62	152.16	157.66	124.09	135.35	116.69	161.07	144.65	175.41	137.50	225.69
Comp. cost, median (x10000)	6046.10	278.37	561.60	452.00	631.30	602.70	625.10	822.25	978.00	1253.55	1268.75
Comp. cost, average (x10000)	5767.38	846.04	1129.03	871.68	1544.38	1031.97	1354.21	1424.57	1654.12	1982.99	3261.40
Comp. cost, StdDev (x10000)	2591.69	1568.11	1698.85	1401.48	2295.80	1681.95	2156.58	2075.25	2852.20	2578.41	4594.62
Num. nodes, median	234.50	26.00	58.00	40.00	60.50	50.00	53.00	76.50	69.50	77.00	69.50
Num. nodes, average	221.60	61.94	77.20	63.14	91.94	60.66	76.52	85.32	77.94	84.22	98.28
Num. nodes, StdDev	76.61	66.73	67.57	52.76	84.29	57.20	70.82	64.94	53.98	62.11	69.99

Finally, a statistical study of these results has been carried out as recommended in previous works Derrac, García, Molina, and Herrera (2011); L.-M. Li, Lu, Zeng, Wu, and Chen (2016); X. Li, Zhang, and Yin (2014); Zeng, Xie, Chen, and Weng (2019) Figure 9 shows the boxplots for Kruskal-Wallis tests on the test data for each problem, population size and percentage. In each these boxplots, the first column is for the case when no validation is done (0%), the second for when the best individual is taken for validation (1% in 100 individuals, or 0.1% in 1000 individuals), and the remaining for the corresponding percent assigned on the x-axis.

On Table 12, the results of the Kruskal-Wallis one-way analysis of variance (ANOVA) are shown. The medians for each group can be seen on Tables 2–11, on the first row, and the standard deviations on the third row. The significance level was 1%. As can be seen from the chi-squared and p-values, ANOVA rejects the null hypothesis that for each problem, the median of the results is equal. In any case, the results obtained with a validation dataset were statistically better than the results without it, which highlights, once again, the importance of the validation dataset to deal with the overfitting. Two of the problems, toxicity and bioavailability, showed no statistical difference between the test results in the different percentages. However, in the remaining problems, the best results obtained when using a set for validation were statistically different from the results when only the best individual is evaluated in with the validation dataset. Therefore, these results seem to point out that the use of a validation dataset with more than just one individual, in the worst case, leads to have similar results while, in most of the cases, improve the results and performance of GP.

Table 12. Results from ANOVA analysis.

Problem	Number of Individuals	Sum of Squares	chi-square	p-value
Toxicity	100	1.1157e+06	47.9346	6.3797e-07
	1,000	1.0073e+06	69.0203	6.8513e-11
Bioavailability	100	5.5927e+05	23.0581	0.010534
	1,000	1.3739e+06	64.2684	5.5932e-10
Wisconsin Breast Cancer	100	1.2119 e + 06	48.0823	5.9955e-07
	1,000	1.7487e+06	69.5918	5.3157e-11
Ionosphere	100	1.0708e+06	42.4199	6.3125e-06
	1,000	1.0167e+06	40.2977	1.5016e-05
Heart Disease	100	1.2765e+06	50.5854	2.0826e-07
	1,000	1.9945e+06	118.6226	9.6281e-21

Conclusions and future work

In classic GP, the evolutionary algorithm output is the individual with the best fitness in the training dataset. If a validation dataset is added, this best individual is evaluated by using the validation dataset, and if the validation error is lower, it is stored to be returned only when the process is completed. However, the results found herein show that, in all cases, the individual with the lowest validation error was not one of the training-best individuals in each generation. Instead, the individual which provided the best validation results proved to be 'hidden' within the population.

In addition, the results seem to differ regarding what percentage of individuals to be evaluated with the validation dataset is the best one to find that individual. The complexity of these five problems is different, and for each of them, the best percentage is different. Further research has to be done in this sense to develop a method to set how many individuals should be evaluated with the validation dataset. This method could be even more complex, not setting the number of individuals to be evaluated with the validation dataset, but choosing among the population those individuals that are likely to return best results, and then evaluating them with the validation dataset.

The use of a validation dataset to reduce overfitting in GP has proved to be a very effective technique. Moreover, the possibility to evaluate more than one individual opens the door to the development of new techniques which allow to use the validation dataset in a more effective way to reduce overfitting.

From the results obtained in this work, new developments can be done. First, considering that the best individual was found in different percentages of the population, it would be of great interest to conduct a similar study where the number of individuals in the population is not fixed, but it depends on their fitness.

Related to this, a system could be developed, which, depending on the difference between the error values in the training and validation datasets, is able to modify the number of individuals to be evaluated with the validation dataset.

Acknowledgments

The experiments described in this work were performed on computers in the Supercomputing Center of Galicia (CESGA). Daniel Rivero and Enrique Fernndez-Blanco would also like to thank the support provided by the NVIDIA Research Grants Program.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work is supported by the Collaborative Project in Genomic Data Integration (CICLOGEN) PI17/01826 funded by the Carlos III Health Institute from the Spanish National plan for Scientific and Technical Research and Innovation 20132016 and the European Regional Development Funds (FEDER)A way to build Europe. This project was also supported by the General Directorate of Culture, Education and University Management of Xunta de Galicia (Ref. ED431G/01, ED431D 2017/16), he Galician Network for Colorectal Cancer Research (Ref. ED431D 2017/23), Competitive Reference Groups (Ref. ED431C 2018/49) and the European Regional Development Funds (FEDER) A way to build Europe Consellería de Cultura, Educación e Ordenación Universitaria, Xunta de Galicia[ED431C 2018/49,ED431D 2017/16,ED431D 2017/23,ED431G/01]; Instituto de Salud Carlos III [PI17/01826].

References

1. Archetti, F., Lanzeni, S., Messina, E., & Vanneschi, L. (2007). Genetic programming for computational pharmacokinetics in drug discovery and development. Genetic Programming and Evolvable Machines, 8(4), 413–432.
2. Azad, R., Medernach, D., & Ryan, C. (2014). Efficient interleaved sampling of training data in genetic programming. In Proceedings of the companion publication of the 2014 annual conference on genetic and evolutionary computation, Vancouver, Canada (pp. 127–128).
3. Cavaretta, M. J., & Chellapilla, K. (1999). Data mining using genetic programming: The implications of parsimony on generalization error. In Evolutionary computation, 1999. cec 99. proceedings of the 1999 congress on, Washington, DC, USA, USA (Vol.2, pp. 1330–1337).
4. Cawley, G. C., & Talbot, N. L. (2010). On over-fitting in model selection and subsequent selection bias in performance evaluation. Journal of Machine Learning Research, 11(Jul), 2079–2107.
5. Danandeh Mehr, A., Kahya, E., Uyumaz, A., & Erdem, H. (2014, 5). Rectangular side weirs discharge coefficient estimation in circular channels using linear genetic programming approach. Journal of Hydroinformatics, 16(6), 1318-1330. Doi:10.2166/hydro.2014.112
6. Danandeh Mehr, A., & Nourani, V. (2017, 3). A pareto-optimal moving average-multigene genetic programming model for rainfall-runoff modelling. Environmental Modelling and Software, 92, 239–251.
7. Derrac, J., García, S., Molina, D., & Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation, 1(1), 3–18.
8. Dheeru, D., & Karra Taniskidou, E. (2017). UCI machine learning repository. Retrieved from http://archive.ics.uci.edu/ml
9. Ekárt, A., & Nemeth, S. Z. (2001). Selection based on the pareto nondomination criterion for controlling code growth in genetic programming. Genetic Programming and Evolvable Machines, 2(1), 61–73.
10. Foreman, N., & Evett, M. (2005). Preventing overfitting in gp with canary functions. In Proceedings of the 7th annual conference on genetic and evolutionary computation, Washington, DC, USA (pp. 1779–1780).
11. Gagné, C., Schoenauer, M., Parizeau, M., & Tomassini, M. (2006). Genetic programming, validation sets, and parsimony pressure. In European conference on genetic programming, Budapest, Hungary (pp. 109–120).
12. Gathercole, C., & Ross, P. (1994). Dynamic training subset selection for supervised learning in genetic programming. In International conference on parallel problem solving from nature,Jerusalem, Israel. (pp. 312–321).
13. Gonçalves, I., & Silva, S. (2011). Experiments on controlling overfitting in genetic programming. In 15th portuguese conference on artificial intelligence (epia 2011), Lisbon, Portugal, (pp. 10–13).
14. Gonçalves, I., & Silva, S. (2013). Balancing learning and overfitting in genetic programming with interleaved sampling of training data. In European conference on genetic programming, Vienna, Austria (pp. 73–84).

15. Gonçalves, I., Silva, S., Melo, J. B., & Carreiras, J. M. (2012). Random sampling technique for overfitting control in genetic programming. In European conference on genetic programming, Málaga, Spain (pp. 218–229).
16. Gustafson, S., Ekárt, A., Burke, E., & Kendall, G. (2004). Problem difficulty and code growth in genetic programming. Genetic Programming and Evolvable Machines, 5(3), 271–290.
17. Igel, C. (2013, 06). A note on generalization loss when evolving adaptive pattern recognition systems. IEEE Transactions on Evolutionary Computation, 17, 345–352.
18. Langdon, W. (2011). Minimising testing in genetic programming. RN, 11(10), 1. [Google Scholar]
19. Li, L.-M., Lu, K.-D., Zeng, G.-Q., Wu, L., & Chen, M.-R. (2016). A novel real-coded population-based extremal optimization algorithm with polynomial mutation: A non-parametric statistical study on continuous optimization problems. Neurocomputing, 174, 577–587.
20. Li, X., Zhang, J., & Yin, M. (2014). Animal migration optimization: An optimization algorithm inspired by animal migration behavior. Neural Computing and Applications, 24(7–8), 1867–1877.
21. Liu, Y., & Khoshgoftaar, T. (2004). Reducing overfitting in genetic programming models for software quality classification. In null, Eighth IEEE International Symposium on High Assurance Systems Engineering, 2004. Proceedings.56–65. Doi:10.1109/HASE.2004.1281730
22. Poli, R., & McPhee, N. F. (2014). Parsimony Pressure Made Easy: Solving the Problem of Bloat in GP. In: Borenstein Y., Moraglio A. (eds) Theory and Principled Methods for the Design of Metaheuristics. Natural Computing Series. Springer, Berlin, Heidelberg. Doi:10.1007/978-3-642-33206-7_9
23. Robilliard, D., & Fonlupt, C. (2001). Backwarding: An overfitting control for genetic programming in a remote sensing application. In International conference on artificial evolution (evolution artificielle), Le Creusot, France, (pp. 245–254).
24. Silva, S., & Costa, E. (2009). Dynamic limits for bloat control in genetic programming and a review of past and current bloat theories. Genetic Programming and Evolvable Machines, 10(2), 141–179.
25. Soule, T., & Foster, J. A. (1998). Effects of code growth and parsimony pressure on populations in genetic programming. Evolutionary Computation, 6(4), 293–309.
26. Uy, N. Q., Hien, N. T., Hoai, N. X., & ONeill, M. (2010). Improving the generalisation ability of genetic programming with semantic similarity based crossover. In European conference on genetic programming, Istanbul, Turkey (pp. 184–195).
27. Vanneschi, L., Castelli, M., & Silva, S. (2010). Measuring bloat, overfitting and functional complexity in genetic programming. In Proceedings of the 12th annual conference on genetic and evolutionary computation, Portland, OR, USA (pp. 877–884).
28. Vanneschi, L., & Gustafson, S. (2009). Using crossover based similarity measure to improve genetic programming generalization ability. In Proceedings of the 11th annual conference on genetic and evolutionary computation, Montreal, QC, Canada (pp. 1139–1146).
29. Vanneschi, L., & Silva, S. (2009). Using operator equalisation for prediction of drug toxicity with genetic programming. In Portuguese conference on artificial intelligence, Aveiro, Portuga (pp. 65–76).
30. Žegklitz, J., & Pošík, P. (2015). Model selection and overfitting in genetic programming: Empirical study. In Proceedings of the companion publication of the 2015 annual conference on genetic and evolutionary computation, Madrid, Spain (pp. 1527–1528).
31. Zeng, G.-Q., Xie, X.-Q., Chen, M.-R., & Weng, J. (2019). Adaptive population extremal optimization-based pid neural network for multivariable nonlinear control systems. Swarm and Evolutionary Computation, 44, 320–334.
32. Zhang, B.-T., & Mühlenbein, H. (1995). Balancing accuracy and parsimony in genetic programming. Evolutionary Computation, 3(1), 17–38.