

Genetic programming with semantic equivalence classes

Stefano Ruberto^b Leonardo Vanneschi^a Mauro Castelli^a

^a NOVA Information Management School (NOVA IMS), Universidade Nova de Lisboa,
Campus de Campolide, 1070-312, Lisboa, Portugal

^bGSSI, Gran Sasso Science Institute, INFN, 67100, L'Aquila, Italy

This is the accepted author *manuscript of the following article published by Elsevier*:

Ruberto, S., Vanneschi, L., & Castelli, M. (2019). Genetic programming with semantic equivalence classes. *Swarm and Evolutionary Computation*, 44(February), 453-469. DOI: 10.1016/j.swevo.2018.06.001



This work is licensed under a [Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License](https://creativecommons.org/licenses/by-nc-nd/4.0/).

Genetic Programming with Semantic Equivalence Classes

Stefano Ruberto^b, Leonardo Vanneschi^a, Mauro Castelli^{a,*}

^aNOVA Information Management School (NOVA IMS)
Universidade Nova de Lisboa, Campus de Campolide, 1070-312 Lisboa, Portugal
^bGSSI, Gran Sasso Science Institute, INFN, 67100 L'Aquila, Italy

Abstract

In this paper, we introduce the concept of semantics-based equivalence classes for symbolic regression problems in genetic programming. The idea is implemented by means of two different genetic programming systems, in which two different definitions of equivalence are used. In both systems, whenever a solution in an equivalence class is found, it is possible to analytically generate any other solution in that equivalence class. As such, these two systems allow us to shift the objective of genetic programming: instead of finding a globally optimal solution, the objective is now to find any solution that belongs to the same equivalence class as a global optimum. Further, we propose improvements to these genetic programming systems in which, once a solution that belongs to a particular equivalence class is generated, no other solution in that class is accepted anymore in the population during the evolution. We call these improved versions filtered systems. Experimental results obtained via seven complex real-life test problems show that using equivalence classes is a promising idea and that filters are generally helpful for improving the systems' performance. Furthermore, the proposed methods produce individuals with a much smaller size with respect to geometric semantic genetic programming. Finally, we show that filters are also useful to improve the performance of a state-of-the-art method, not explicitly based on semantic equivalence classes, like linear scaling.

Keywords: Genetic Programming; Semantics; Equivalence Classes

1. Introduction

The use of semantic methods is one of the hottest topics in Genetic Programming (GP) [1] and has recently attracted noteworthy attention from researchers, particularly in the applied domain of symbolic regression [2]. Let $\mathbf{X} = \{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_n\}$ be the set of input data, or fitness cases, of a symbolic regression problem, and $\vec{t} = [t_1, t_2, \dots, t_n]$ the vector of the respective expected output or target values (in other words, for each $i = 1, 2, \dots, n$, t_i is the expected output corresponding to input \vec{x}_i). A GP individual (or program) P can be seen as a function that for each input vector \vec{x}_i returns the scalar value $P(\vec{x}_i)$. Following [3], we call the *semantics* of P the vector $\vec{s}_P = [P(\vec{x}_1), P(\vec{x}_2), \dots, P(\vec{x}_n)]$. This vector can be represented as a point in an n -dimensional space, which we call a *semantic space*, that can be counterpoised to the *syntactic* or *genotypic space*, where individuals are represented by programs. While each program has one and only one semantics, the mapping between the semantic space and the genotypic space is generally not a bijection because several programs can have the same semantics. The target vector \vec{t} itself is a point in the semantic space and, in general, the objective of GP is to find at least one program in the genotypic space that maps into \vec{t} in the semantic space.

Several different methods to exploit semantic awareness in GP have so far been presented. For relatively complete surveys, the reader is referred to [4, 5, 6]. The work presented here proposes a new idea for exploiting semantic awareness in GP: *semantics-based equivalence classes*. Our concept of equivalence class is such that, once an individual in a class is found, it must be possible, and easy, to generate all the other individuals in that class. In this way, if we are

*Corresponding author Tel.: +351213828628 Fax: +351213872140

Email addresses: stefano.ruberto@gssi.infn.it (Stefano Ruberto), lvanneschi@novaims.unl.pt (Leonardo Vanneschi), mcastelli@novaims.unl.pt (Mauro Castelli)

able to find one solution that is in the same equivalence class as a globally optimal solution, then we are able to solve the problem by reconstructing the global optimum analytically. Two possible realizations of this idea are proposed in this paper. In the first, two individuals P_1 and P_2 are considered in the same equivalence class if $\vec{s}_{P_1} = k + \vec{s}_{P_2}$. In such a situation, we are identifying a collection of *affine subspaces* in the semantic space. In this collection, each subspace is a straight line (or hyperplane, if we think in n dimensions) and all the lines belonging to this collection are parallel to each other. In a system like this, it makes sense to use the variance of the difference between the coordinates of the semantic vector and the target as fitness. When this variance is equal to zero, the semantic vector of the individual is in the same equivalence class as the target, and the search can terminate, analytically reconstructing a globally optimal solution. In the second proposed system, two individuals P_1 and P_2 are in the same equivalence class if $\vec{s}_{P_1} = k \cdot \vec{s}_{P_2}$, where $k \in \mathbb{R}$ is a constant such that $k \neq 0$. In such a situation, we are defining a *projective space* (i.e. the space of all the straight lines – or hyperplanes, if we think in n dimensions – intersecting the origin) in the semantic space. The objective of GP then is to find any solution whose semantics is directly proportional to the target \vec{t} (i.e. that stands on the same straight line as the one intersecting the target and the origin). It therefore makes sense to define a new fitness function, corresponding to the variance of the ratios between the coordinates of the semantics of an individual and \vec{t} . When this variance equals zero, the individual is in the same equivalence class as the target, and the search can terminate, analytically reconstructing a globally optimal solution. The first of these two systems is called *GPPLUS* (given that addition is the operator that allows us to reconstruct the target), while the second one is called *GPMUL* (given that the target can be reconstructed using multiplication). Moreover, we introduce two new GP systems that are similar to *GPPLUS* and *GPMUL*, but in which a filter that allows us to reject individuals is applied. Specifically, a newly generated individual is rejected if another individual belonging to the same equivalence class already exists in the population. These “filtered” versions are called *FGPPLUS* and *FGPMUL*, respectively. The performance of *GPPLUS*, *GPMUL*, *FGPPLUS*, and *FGPMUL* is compared to one of the best known state-of-the-art semantics-based GP systems: Geometric Semantic GP (GSGP), introduced in [3] and described in detail in [7]. As test cases, we choose seven complex real-life symbolic regression problems, from as many different applied domains. Finally, in order to show the appropriateness of semantic filters also on systems that are not directly based on the idea of semantic equivalence classes, we apply filters to one of the most used GP techniques: linear scaling [8, 9]. For this reason, linear scaling without filters (called *LS*) and linear scaling with filters (called *FLS*) are also included in the comparative study.

The paper is structured as follows: Section 2 discusses previous and related works. In Section 3, we present the general concept of semantics-based equivalence class and we define *GPPLUS*, *GPMUL*, *FGPPLUS*, and *FGPMUL*. Section 4 presents the experimental study. Specifically, the experimental settings and test problems are presented and, subsequently, the obtained results are reported and discussed. Finally, Section 5 concludes the paper, also by discussing ideas for future research.

2. Previous and Related Work

The definition of semantics-based GP systems is a hot topic in the field of evolutionary computation and several works that have appeared so far can be considered as relating to this paper or having inspired the work presented here. In [3], Moraglio and co-authors introduce genetic operators that have a direct effect on the semantics of the offspring. GP using these operators was called Geometric Semantic Genetic Programming (*GSGP*). These operators induce a unimodal fitness landscape for any problem consisting of finding a match between outputs and known targets. An implementation of this method, proposed by Vanneschi et al. in [10], counteracts the difficulties related to the rapid growth of the solutions in *GSGP* since it allows individuals to be encoded in effective ways, allowing us to manage them efficiently even when they have a very large size. This implementation permitted us to use *GSGP* for several real-life applications and allowed us to discover interesting generalization properties peculiar to these operators. The generalizability of *GSGP* was further discussed in [11], and its genetic operators were refined in [12]. The concept that binds our work to *GSGP* is the idea of direct semantic manipulation. We use this kind of manipulation to build and compare entire classes of individuals. Instead of the operators proposed in [3], we use traditional genetic operators. In doing so, we lose the interesting property of a unimodal fitness landscape as part of the trade for the possibility of obtaining human-readable solutions that are compact in size.

The idea of directly manipulating individuals to obtain a specific semantic result is also present in the work of Ruberto et al. [13], where two optimally aligned individuals in the error space are combined into a new one that

approximates the target, using a method called Error Space Alignment Genetic Programming (ESAGP). In ESAGP, two aligned individuals are considered in the same equivalence class. This concept is similar to what is introduced here, although the framework in this work is more general and can incorporate ESAGP as a particular instantiation. ESAGP, like the systems introduced here, does not have any guarantee of inducing a unimodal fitness landscape, but also in this case the resulting solutions are compact and readable. ESAGP has an option to filter those individuals considered redundant because of equivalence classes. This is an option that is also exploited in this work, but some contradictions of ESAGP are resolved here. ESAGP was maximizing semantic diversity and the exploration of the solution space, rejecting redundant individuals. However, at the same time, needing at least two aligned individuals, ESAGP was also looking for a certain amount of similarity. The algorithm presented here does not suffer from these contradictory objectives. A final difference consists in the use of a repository of class representatives for ESAGP. In the work presented here, the use of a repository is pointless because classes already explored were evaluated as a whole: having new classes better fitness, the evolutionary process never returns to the old ones. Another important similarity with ESAGP is that evolution proceeds without considering the error compared to the target values, indeed taking into account a more indirect measure, like the membership degree into a specific equivalence class (see the next section for more details).

Following the idea of ESAGP, the authors of [14] show how arbitrarily close alignments in the error space can be achieved by GP. One difference between this idea and ESAGP (and also the work presented here) is that the alignment is sought for using the genetic operators introduced in [3]. On the other hand, one similarity between [14] and this work is, once again, the fact that in [14] a first embryo of the idea of equivalence class can be found, identifying equivalent individuals as having the same semantics. A more general and sophisticated definition is given here.

In [15], the importance of metrics for *GSGP* was discussed. According to the authors, two main types of metrics can be identified: measures that structure the semantic space and govern the behavior of geometric search operators and metrics that determine how fitness is calculated. The first kind of metrics is clearly very important for the work presented here, while for the second kind of metric, this work differs from [15] given that no type of error between calculated values and targets is used here to assess the quality of the evolving individuals. An idea that can be considered as orthogonal to the one of the alignment in the error space is the one presented in [16], where a new type of crossover called subtree semantic geometric crossover (SSGC) was proposed. SSGC is an approximated semantic crossover and allows the main limitation of the *GSGP* crossover to be overcome, i.e. the exponential growth of the size of the solutions. Like alignment in the error space, SSGC is a type of crossover based on semantics and does not necessarily produce offspring larger than parents. Studies of semantics-based approaches of GP were recently extended to the domain of fuzzy rules in [17], where the objective was nonlinear modeling. Like this work, [17] is another example of how semantics-based methods, besides being effective, can be used for compact and readable models. Another interesting contribution was given in [18], where methods to reduce the dimensionality of the error space were developed. The large size of that space often adds to the complexity of semantics-based methods, and the ideas introduced in [18], although simple, can be very helpful in those cases. Another advantage of those ideas is that they are very versatile, and can naturally be integrated with many kinds of approaches, including the one of equivalence class discussed here. In the same year, the same authors introduced a dispersion operator for *GSGP* [19]. This operator was designed to counteract one of the issues of *GSGP*, i.e. the fact that crossover cannot generate solutions that are placed outside the convex hull formed by the individuals of the current population. Although the mutation operator alleviates this problem, we cannot guarantee it will find promising regions of the search space within a feasible computational time. In this direction, Oliveira and colleagues proposed a new geometric dispersion operator that uses multiplicative factors to move individuals to less dense areas of the search space around the target solution before applying semantic genetic operators. Clearly, using a multiplicative factor has analogies with the *GPMUL* approach presented here. On the other hand, the *GPMUL* approach also has important differences with the one used by Oliveira *et al.* For instance, *GPMUL* does not introduce any new operator, but only uses multiplicative factors to decide if two individuals belong to the same equivalence class or not.

The concept of equivalence class based on semantics, which is the focus of this work, also has interesting analogies with the concept of semantic niches, first introduced by Spector to describe the PUSH GP system [20]. Another study with some analogies to this paper is described in [21], where authors presented a GP system that maintains a distribution of different semantic behaviors and biases the search toward solutions that have similar semantics to the best solutions that have been found so far. In any case, the idea of semantic distribution used in [21] is very different with respect to that presented in this paper, while it presents some similarities with the idea exploited in [22]

to counteract bloat in GP (where a distribution of individuals' sizes is maintained). Our work has also important relationships with the well-known Linear Scaling technique (*LS*) introduced by Keijzer in [8, 9]. *LS*, if seen from our perspective in this work, can be reinterpreted as a technique that is using equivalence classes to evolve solutions. In the case of *LS*, in fact, two solutions gp_1 and gp_2 are equivalent if we can obtain gp_1 from gp_2 , and vice versa through a linear scaling. The evolutionary process in *LS* is looking for an individual that is equivalent to the target in a very similar way to what *GPMUL* and *GPPLUS* do. Being the most simple possible equivalence relationship, *GPPLUS* and *GPMUL* are also less powerful than *LS*. Indeed, *LS* incorporates both of them being able to recognize at once equivalences of both types. Nevertheless, this power comes at the price of more limited flexibility in use; in fact, *LS* optimizes the whole output at once, while more simple classes like *GPPLUS* and *GPMUL* can easily classify part of the input, recognizing what part of the problem is well explained by a proposed gp_i solution before optimizing the error. If we apply the equivalence definitions reported in Equation (4) and Equation (7) to the target (see Section 3), then we know in advance what part of this target is explained, or equivalent to the proposed gp_i , being k_i values homogeneous for these portions of the dataset. This kind of analysis is not possible with methods using a complex equivalence definition that requires knowing in advance what the target of the optimization is. Although not done here, it is easy to imagine an algorithm having repositories of partial target equivalences, which combines them to have a high-quality approximation. On the other hand, *LS* can benefit from this new point of view, integrating the concept of exploring the solution space by equivalence classes rather than one individual at a time. This fact inspired us with the possibility of using filtering techniques (like those explained below) to improve *LS*. Finally, another interesting approach is that proposed by Beadle et al. in [23], describing a Semantically Driven Crossover (SDC), that discards offspring with the identical semantics of parents. SDC works in the boolean domain and reduces all the candidate solutions to a canonical form to establish if two individuals have equivalent semantics. Our approach is more general and not limited to a particular domain. Convenient parametrization allows us to apply filtering to a certain degree, appropriate for a specific application. Further, we explore classes of individuals that are not just the ones with identical semantics, but also others that are equivalent from a wider point of view.

3. Methodology

Our objective is to use equivalence classes (EQCs) to create partitions of the solution space and to explore the space of EQCs instead of the space of single solutions. To achieve this objective, we start by defining equivalence relationships using criteria based on semantics. We introduce a relationship that we call *equivalence function* (*EF*). *EF* receives two semantics as arguments, and returns a vector of the same cardinality:

$$EF : \mathbb{R}^n \times \mathbb{R}^n \longrightarrow \mathbb{R}^n$$

We say that two individuals are equivalent (i.e. they belong to the same EQC) if *EF* calculated on the semantics of the two individuals returns a constant vector. In particular, we have that an individual whose semantics is \vec{gp} is equivalent to an individual whose semantics is the target \vec{t} if:

$$EF(\vec{t}, \vec{gp}) = \vec{k}, \text{ such that } \vec{k} = [k_1, k_2, \dots, k_n] \text{ and } \forall i = 1, 2, \dots, n-1 : k_i = k_{i+1} \quad (1)$$

Once we have a GP individual equivalent to \vec{t} , we can reconstruct an individual whose semantics is identical to \vec{t} (i.e. a globally optimal solution) analytically. The operation is trivial if the function *EF* is easy to invert (like for instance a linear function), as in Equation (2).

$$\vec{t} = EF^{-1}(\vec{k}, \vec{gp}) \quad (2)$$

Reconstructing an optimal tree from Equation (2) is straightforward. It is sufficient to combine the tree whose semantics is \vec{gp} (i.e. the genotype of the GP program) and constant k by means of operator EF^{-1} , that represents the root node of the optimal individual. With this in mind, now the objective of GP can become the one of finding a solution that belongs to the same EQC as the target. In order to obtain this, we need to define a new fitness function able to quantify the distance between the EQC of an individual and the EQC of the target. A simple possibility is to measure the *dispersion* of the k_i values in Equation (1). This dispersion must be as low as possible since we aim to obtain a

single constant value, which represents perfect equivalence. For this purpose, we can, for instance, use the *variance* of the components of \vec{k} , like in Equation (3).

$$Fitness = var(\vec{k}) \quad (3)$$

If we consider \vec{k} as the error vector of a solution (i.e. the vectorial difference between its semantics and the target), then the variance of \vec{k} can be interpreted as a measure of its “complexity”. Indeed, when the variance is equal to zero, the error is constant and thus it is trivial to eliminate it using Equation (2). In the continuation of this paper, these general concepts will be instantiated by defining two different concrete *EF* functions. Once that is achieved, the method for reconstructing the target, starting from an individual in the same EQC, should be clearer. An important observation is that, for continuous values, it is unlikely for two individuals to be in the same EQC. For this reason, a threshold is used to establish the EQCs. We point out that in none of the above phases we have used an error function (like for instance the Root Mean Square Error (RMSE)) to drive the evolutionary process: the evolved individuals can have a very poor value of the error function. Also, once we find an individual that belongs to the same EQC as another individual that is already in the population, it may be useful to reject it. After all, if we consider our system as exploring the space of EQCs (if once we generate an individual, we are immediately able to analytically obtain the best individual in terms of RMSE in the same EQC), then we do not need an EQC to be represented by more than one individual. In this paper, we propose a filtering process to implement this rejection: when a new individual is generated, it is checked against all other individuals in the population and rejected if it belongs to an EQC that is already represented by at least one other individual. A new individual is immediately generated in substitution and checked again.

3.1. GPPLUS: GP by Translation

The framework presented so far is very general and can be instantiated in many different ways, depending on how we express EQCs. Here, we propose the first case. The resulting technique, that we call *GPPLUS*, is based on the concept of error vector [13], i.e. the vector of the differences between the semantics of an individual and the target. Two solutions are considered in the same EQC if all the coordinates of the corresponding error vectors are identical to each other (i.e. if their semantics are equivalent by translation). We define the equivalence function *EF* like in Equation (4).

$$EF = \vec{t} - \vec{gp} = \vec{k} \quad (4)$$

Assuming that the final assessment of the quality of the solutions is made using the RMSE, *GPPLUS* optimizes Equation (5) for every point in the dataset.

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (t_i - gp_i - k)^2}{n}} \quad (5)$$

Assigning the first derivative to zero, $\frac{\partial}{\partial k} RMSE = 0$, we obtain that the optimal value of k is equal to the mean of \vec{k} . Then, a globally optimal solution (i.e. a solution whose semantics is identical to the target) can be built using Equation (6), where *GP* is the notation used to identify the program that has \vec{gp} as semantics:

$$T_{apx} = GP + k \quad (6)$$

As previously explained, it may be useful to reject individuals that are in the same EQC as another individual in the population. To check if two individuals, with respective semantics \vec{gp}_1 and \vec{gp}_2 , are in the same EQC, we calculate the variance of $(\vec{gp}_1 - \vec{gp}_2)$ and we compare it to a prefixed threshold. If it is below that threshold, then the individuals are considered in the same EQC and one of them can be rejected. A convenient threshold value was found experimentally, and the results of this experimental study are discussed later in this paper.

3.2. GPMUL: GP by proportions

In this section, we propose the second instantiation of the framework presented so far, called *GPMUL*. In *GPMUL*, two solutions are in the same EQC if the ratios of the coordinates of their semantics are identical to each other (i.e. if they are equivalent by scale). We define the new *EF* error function like in Equation (7).

$$EF = \frac{\vec{gp}}{\vec{t}} = \vec{k} \quad (7)$$

Analogously to *GPPLUS*, also for *GPMUL* fitness should be defined as a measure of dispersion of the components of \vec{k} . However, we experimentally observe a problem with *GPMUL* that did not exist with *GPPLUS*: in order to obtain a \vec{k} with all coordinates similar to each other, the evolution tends to create individuals with very large output values. In fact, the larger the coordinates of \vec{gp} in Equation (7), the more similar to each other are the coordinates of \vec{k} , even though these coordinates are also very large numbers. Since managing such large numbers may be impractical, a way of counteracting this issue is needed. For this reason, in *GPMUL* we consider a normalized version of \vec{k} whose coordinates are defined as in Equation (8). Using this normalization, there is no competitive advantage in evolving individuals with very large output values.

$$k_{norm_i} = \frac{k_i}{MEAN(\vec{k})} \quad (8)$$

Further, in *GPMUL* we have artificially imposed a severe penalty in fitness for all programs for which $MEAN(\vec{k})$ is equal to zero. In this way, those programs do not survive the selection phase, thus preventing the system from failures during the evaluation of the individuals. It is worth to point out that a solution for which the values of all the coordinates of \vec{k} are equal to zero is not the only case in which $MEAN(\vec{k})$ can be equal to zero. Thus, testing whether $MEAN(\vec{k})$ is different from zero (and eventually penalizing the solution with a poor fitness value in case the condition is not satisfied) is a crucial step for guaranteeing the correct functioning of the system. Given that an optimal solution is found when all coordinates of \vec{k} in Equation (7) are equal to 1, the fitness measure we employ for *GPMUL* is:

$$Fitness = MEAN \left[\left(\frac{k_i}{MEAN(\vec{k})} - 1 \right)^2 \right] \quad (9)$$

Analogously to *GPPLUS*, assuming that the RMSE is the measure that has to be minimized, we should find the scalar value k that optimizes Equation (10).

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (t_i - \frac{gp_i}{k})^2}{n}} \quad (10)$$

Assigning the first derivative to zero, $\frac{\partial}{\partial k} RMSE = 0$, we obtain the optimal value of k expressed by Equation (11).

$$k = \frac{\sum_{i=1}^n (gp_i)^2}{\sum_{i=1}^n (t_i \cdot gp_i)} \quad (11)$$

Thus, if GP finds a solution in the same EQC as the target, we use Equation (12) (where k is the one obtained in Equation (11)) to build a globally optimal solution T_{apx} (where *GP* represents a program whose semantic vector is \vec{gp}).

$$T_{apx} = \frac{GP}{k} \quad (12)$$

If, during the evolutionary process, a program is generated such that k , calculated in Equation (11), is equal to zero (for instance because all the gp_i values are equal to zero or because the t_i values approach infinity), then that program receives a severe fitness penalty, so that the selection process can rapidly remove it from the population. The filtering process is similar to the one introduced in Section 3.1: to check if two individuals, with respective semantics \vec{gp}_1 and \vec{gp}_2 , are in the same EQC, we calculate the variance of \vec{gp}_1 / \vec{gp}_2 and compare it to a prefixed threshold. If it is below

that threshold, the individuals are considered in the same EQC and one of them can be rejected. In this case, fitness cases for which the denominator of the equation is equal to 0 are ignored. As previously mentioned, an appropriate threshold value was determined empirically, and the results of this experimental study are reported and discussed later in this paper.

As a final remark, we point out that *GPMUL* might be difficult, or even impossible to apply for some classes of problems. An example of such problems is, for instance, a problem in which the target is a zero vector. In a case like that, Equations (7), (8), (9) and (11) cannot be calculated. Also for problems in which the target vector is “close” to a zero vector, the attempt to calculate those equations may generate problems, like the appearance of unbearably large numbers. However, in cases like these ones, one may implement a simple “workaround”, that would allow us to apply *GPMUL* effectively. The workaround may simply consist of adding a constant different from zero to all the target values, thus simply translating the target. In the generalization phase, one may simply apply the learned model and then subtract the chosen constant to the obtained result, to have the prediction. We also point out that having a target equal (or close) to the zero vector is extremely rare and, in a case like that, the prediction of the target can be done in a straightforward way, without the need of any machine learning system.

4. Experimental study

4.1. Systems and test problems

The objective of the experimental study is to compare the methods *GPPLUS*, *GPMUL*, *LS* and their filtered variants *FGPPLUS*, *FGPMUL*, and *FLS*. To ensure a better understanding of the quality of the performance of these systems, we also consider the performance obtained by geometric semantic genetic programming (GSGP) [3]. We also decided to not report the results achieved by standard GP because a preliminary study indicated that standard GP was consistently outperformed by all the presented methods for all test problems. Moreover, while GSGP does not always perform better than standard GP [6], the reader can refer to existing studies [24, 25, 26] that directly compare GSGP and standard GP for several datasets. *GPPLUS*, *GPMUL* and *LS*, as well as their filtered variants, were implemented on top of the ECJ framework (<https://cs.gmu.edu/~eclab/projects/ecj/>). To compare the performance of the systems based on the concept of equivalence classes against that of GSGP, we considered seven different symbolic regression test problems that have already been used in previous GP studies. Table 1 reports the number of features and number of instances for each dataset. For a complete description of these datasets, the reader is referred to the references reported in the same table.

Table 1: Description of the test problems; the number of features (independent variables) and number of instances are reported for each dataset.

Dataset	# Features	# Instances
Airfoil Self-Noise [27]	5	1502
Concrete Compressive Strength [24]	8	1029
Parkinson Voice Recording (MOTOR) [25]	19	5875
Parkinson Voice Recording (TOTAL) [25]	19	5875
Protein Folding [26]	9	45000
Concrete Slump Test [28]	9	102
Yacht Hydrodynamics [29]	6	307

For all test problems, 100 runs were performed with each technique. In each run, a different partition between the training and test data was considered. In each run, 70% of the instances, selected at random with uniform probability, was used to train the model, while the remaining 30% was used to test its performance on unseen data. This means that in each run the training and test sets are drawn separately, and thus very probably each run uses a different training/test partition. We also considered different percentages for the training/test partitions, namely 60% – 40% and 80% – 20%. This analysis indicated that the results are comparable (i.e., no statistically significant difference) even though a percentage split of 80% training and 20% test results in a larger variation on the test performance among the considered partitions. This is probably due to the smaller amount of observations in the test set. All techniques share exactly the same configuration, as shown in Table 2, except for the filter value (in the various filtered

versions). To determine the best values of the parameters, a preliminary tuning phase was executed by performing a grid search. More in detail, we considered a range of standard values for each parameter and, inside this range, we selected some values to be tested (as shown in Table 2). The selected configuration is the one that, among the different configurations that returned comparable results (i.e., no statistical difference), minimizes the running time. As already demonstrated in previous studies [24, 25], a mutation rate that is too small does not correspond to the ideal choice for the problems taken into account. For the filter parameter, a preliminary extensive study was performed to determine the value that allows us to obtain the best results. To establish the ideal value of the filter parameter, an extensive

Table 2: GP parameters shared by all the methods considered in the experimental phase.

Parameter	Value
Function set	$\{+, -, *, /\}$, where $/$ is protected by returning 1 if the denominator is equal to zero [1]
Terminal set	set of n variables, where n = number of features in the dataset
Population size	200 [50;50;300]
Maximum number of generations	200 [50;50;300]
Initialization algorithm	Ramped Half-and-Half [1]
Maximum tree depth for initialization	6
Maximum tree depth during evolution	no limit
Elitism	the best individual in the population
Crossover rate	0.9 [0.8;0.02;1]
Mutation rate	0.1 [0;0.01;0.2]
Selection	Tournament
Tournament size	4 [2;1;5]

preliminary study was performed. The aim of this study was to compare different values of this parameter to achieve a better understanding of its impact on the system’s performance and on the size of the solutions. Table 3 reports the best value of the filter parameter found in this phase, for each benchmark problem. In particular, for each dataset and technique, the values reported in Table 3 are the ones that produced the best RMSE on the training set during the preliminary study that we performed and that we discuss in the first part of Section 4.2. The notation [min;step;max] that appears close to the parameters, refers to the values tested in the tuning phase. The first value (min) indicates the minimum value tested, the second value is the step, and the last value (max) is the maximum value tested. For all test

Table 3: Filter values; for each dataset and technique, we report the value of the filter used in the experiments in Figures 8 to 14. In particular, for each dataset and technique, the used filter value is the one that produced the best RMSE on the training set during the preliminary study presented in the first part of Section 4.2.

Dataset	FGPPLUS	FGPMUL	FLS
Airfoil Self-Noise	10^{-3}	10^{-5}	10^1
Concrete Compressive Strength	10^{-3}	10^{-3}	10^1
Parkinson Voice Recording (MOTOR)	10^{-6}	10^{-8}	10^{-1}
Parkinson Voice Recording (TOTAL)	10^{-3}	10^{-6}	10^{-1}
Protein Folding	10^{-5}	10^{-3}	10^{-7}
Concrete Slump Test	10^{-1}	10^{-3}	10^2
Yacht Hydrodynamics	10^{-1}	10^{-5}	10^{-2}

problems, the results are reported in terms of the RMSE between target and predicted values. It is worth reaffirming that only GSGP, among the techniques studied, uses the RMSE as a fitness function during the evolutionary process. In all plots reported in the rest of this section, medians of the RMSE of the best individual in the population in the training set, over 100 independent runs, are reported. The median was preferred over the mean because it is more robust to outliers. Median RMSE was studied against the computational effort instead of the number of generations

because generations do not have the same computational complexity for all studied methods. Following [30], as a measure of the computational effort we used the accumulated number of tree nodes evaluated until the current instant. Thus, the computational effort spent by a GP technique until generation g is:

$$CE(g) = NN_g + NN_{g-1} + \dots + NN_1$$

where, for each generation $h = g, g-1, \dots, 1$, NN_h is the sum of the number of tree nodes in all individuals that were evaluated in generation h . It is worth pointing out that while calculating $CE(g)$ we not only take into account the individuals that take part in the evolution, but also those that, eventually, are disregarded by an algorithm. For instance, all the filtered variants of the systems proposed reject several individuals during the evolution. Given that to decide whether an individual is rejected it is necessary to evaluate that individual, the nodes of the rejected individuals are also considered in the calculation of $CE(g)$. This allows us to make a fair comparison between the versions with filters and those without filters.

4.2. Experimental Results

We begin the discussion of the results with an analysis of the filter parameter (i.e., the parameter that tunes the effect of the filters, as presented in Section 3). The aim of this study is to compare different values of this parameter to achieve a better understanding of its impact on the system's performance and on the size of the solutions (expressed in terms of the average number of nodes of the individuals in the population). A wide range of values for the filter parameter was considered. The results are presented in Figures 1 to 7. In all of these figures, plots (a) and (b) report the results obtained by *GPPLUS*; plots (c) and (d) report the results obtained by *GPMUL*; and plots (e) and (f) report the results obtained by *LS*. Furthermore, plots (a), (c), and (e) report the RMSE on the test set for the different values of the filter parameter that were studied. The non-filtered version of each method is also reported: it is the leftmost box on every figure. Plots (b), (d), and (f) show the average number of nodes in individuals in the population (identified by the term individuals' size, or simply size, from now on), for the same values of the filter parameter. Concerning the different test problems, the results are organized as follows: Figure 1 reports the results for the airfoil self-noise problem (hereinafter *airfoil*); Figure 2 reports the results for the concrete compressive strength problem (hereinafter *concrete*); Figure 3 reports the results for the Parkinson's voice recording problem, using the MOTOR dataset (hereinafter *motor*); Figure 4 reports the results for the Parkinson's voice recording problem, using the TOTAL dataset (hereinafter *total*); Figure 5 reports the results for the protein folding problem (hereinafter *protein*); Figure 6 reports the results for the concrete slump test problem (hereinafter *slump*); finally, Figure 7 reports the results for the yacht hydrodynamics problem (hereinafter *yacht*).

Considering plots (a), (c), and (e) of these figures, we can see that *FGPMUL* and *FGPPLUS* generally achieve better results than *GPMUL* and *GPPLUS*, even for very small values of the filter parameter. Increasing the value of this parameter, the RMSE shows an oscillation until it reaches a minimum value that is, for the large majority of problems, just before a steep increase occurs. This pattern is visible in all the *FGPMUL* and *FGPPLUS* experiments, except for the cases of the slump and total datasets in which, although present, it is less evident. By increasing the value of the filter parameter above a certain threshold (that is different for every dataset), the error increases and all values of the filter parameter that are larger than that threshold will produce RMSE values that are comparable to (and even poorer than) the non-filtered counterpart of the studied method. Considering the *LS* technique, one can observe a similar trend, although less visible. The dataset for which the trend is less visible in the case of *LS* is slump. Still, also for that dataset, one can observe a worsening of the RMSE when the value of 10^3 is considered for the filter parameter. To summarize, filtering improved the performance of all techniques studied on almost all test problems studied. Finally, we note that the advantage of *GPPLUS* and *GPMUL* in using filters is larger than that of *LS*. Plots (b), (d), and (f) indicate that filters also have an important effect on the size of the individuals (i.e., the number of tree nodes of the solutions in the population). In particular, a generally observable trend is that higher values of the filter parameter result in smaller individuals. However, it is possible to observe that the size of the individuals does not monotonously decrease with an increase in the filter parameter.

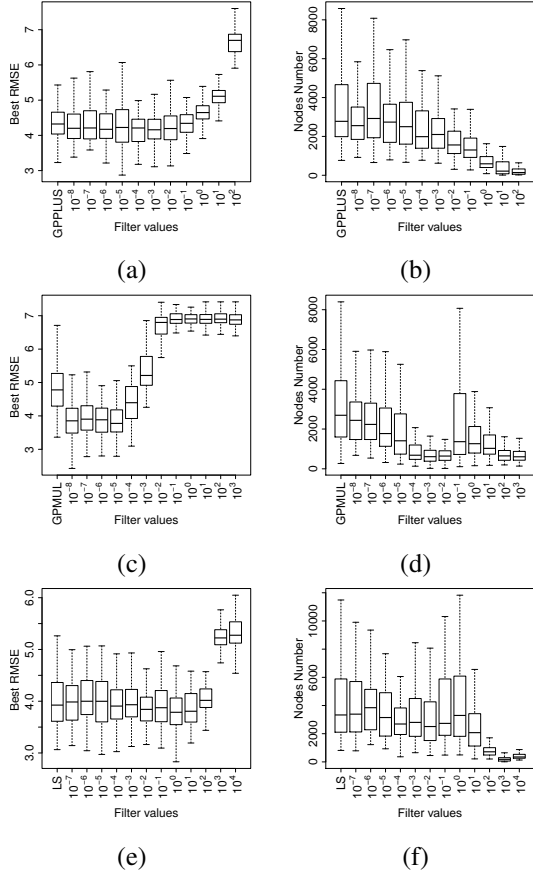


Figure 1: Dataset *airfoil*. Plots (a), (c), and (e) show the RMSE on the test set for different values of the filter parameter. The leftmost boxplot of each line shows the results without filtering: *GPPLUS* in plot (a), *GPMUL* in plot (c), and *LS* in plot (e). Plots (b), (d), and (f) show the average number of nodes in the population for the corresponding filter settings. Also in this case, the leftmost boxplot of each series shows the results without filtering: *GPPLUS* in plot (b), *GPMUL* in plot (d), and *LS* in plot (f).

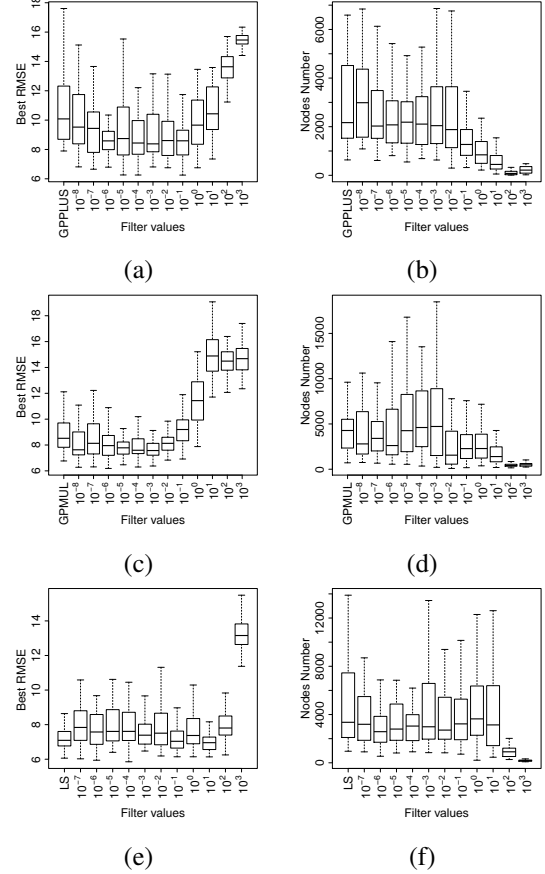


Figure 2: Dataset *concrete*. Plots (a), (c), and (e) show the RMSE on the test set for different values of the filter parameter. The leftmost boxplot of each line shows the results without filtering: *GPPLUS* in plot (a), *GPMUL* in plot (c), and *LS* in plot (e). Plots (b), (d), and (f) show the average number of nodes in the population for the corresponding filter settings. Also in this case, the leftmost boxplot of each series shows the results without filtering: *GPPLUS* in plot (b), *GPMUL* in plot (d), and *LS* in plot (f).

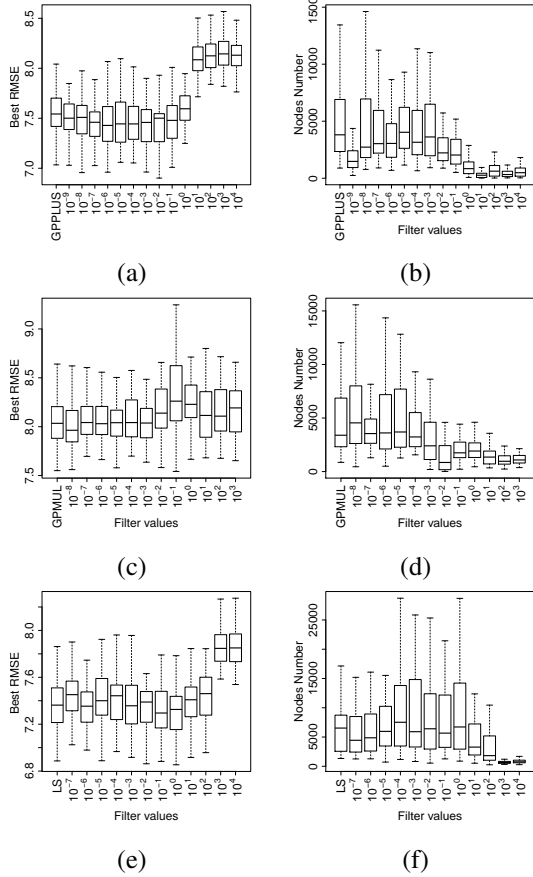


Figure 3: Dataset *motor*. Plots (a), (c), and (e) show the RMSE on the test set for different values of the filter parameter. The leftmost boxplot of each line shows the results without filtering: *GPPLUS* in plot (a), *GPMUL* in plot (c), and *LS* in plot (e). Plots (b), (d), and (f) show the average number of nodes in the population for the corresponding filter settings. Also in this case, the leftmost boxplot of each series shows the results without filtering: *GPPLUS* in plot (b), *GPMUL* in plot (d), and *LS* in plot (f).

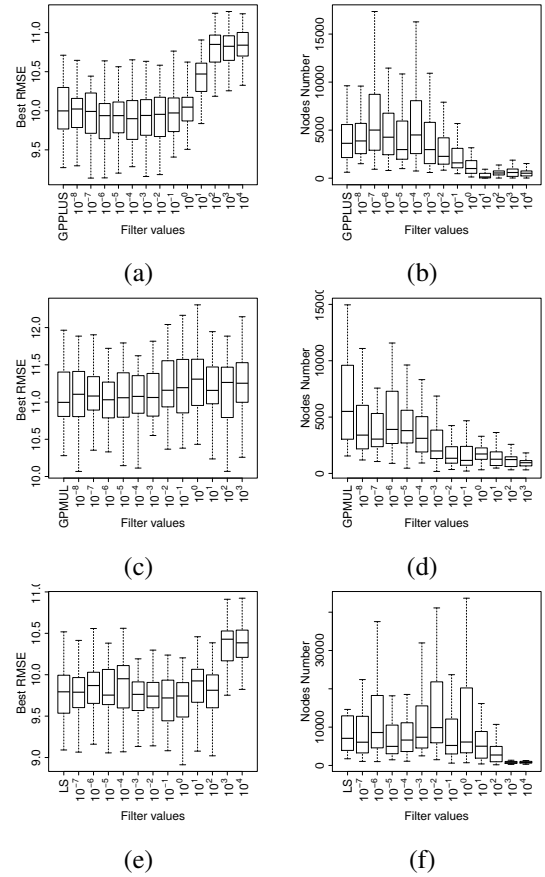


Figure 4: Dataset *total*. Plots (a), (c), and (e) show the RMSE on the test set for different values of the filter parameter. The leftmost boxplot of each line shows the results without filtering: *GPPLUS* in plot (a), *GPMUL* in plot (c), and *LS* in plot (e). Plots (b), (d), and (f) show the average number of nodes in the population for the corresponding filter settings. Also in this case, the leftmost boxplot of each series shows the results without filtering: *GPPLUS* in plot (b), *GPMUL* in plot (d), and *LS* in plot (f).

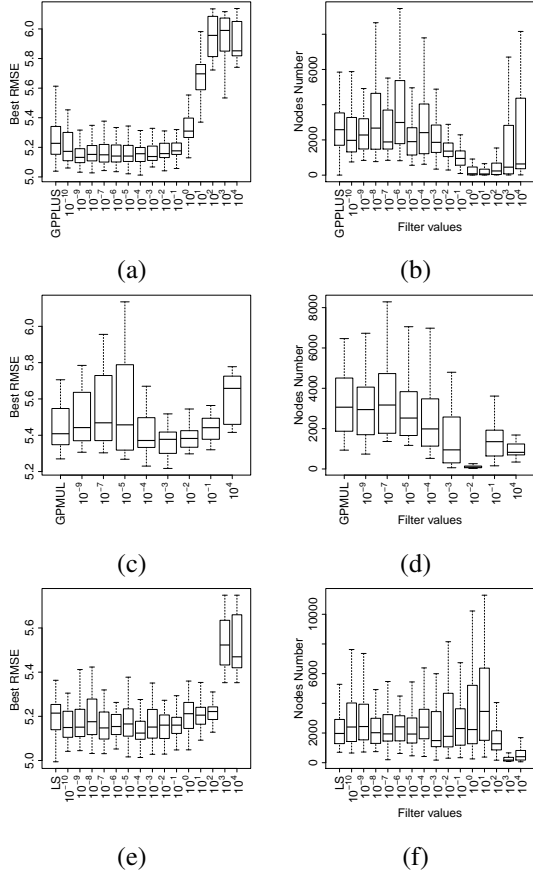


Figure 5: Dataset *protein*. Plots (a), (c), and (e) show the RMSE on the test set for different values of the filter parameter. The leftmost boxplot of each line shows the results without filtering: *GPPLUS* in plot (a), *GPMUL* in plot (c), and *LS* in plot (e). Plots (b), (d), and (f) show the average number of nodes in the population for the corresponding filter settings. Also in this case, the leftmost boxplot of each series shows the results without filtering: *GPPLUS* in plot (b), *GPMUL* in plot (d), and *LS* in plot (f).

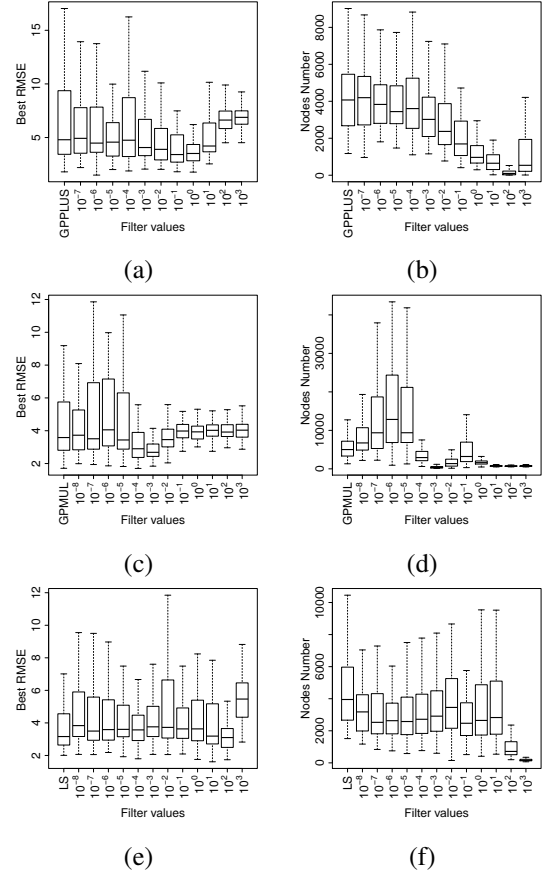


Figure 6: Dataset *slump*. Plots (a), (c), and (e) show the RMSE on the test set for different values of the filter parameter. The leftmost boxplot of each line shows the results without filtering: *GPPLUS* in plot (a), *GPMUL* in plot (c), and *LS* in plot (e). Plots (b), (d), and (f) show the average number of nodes in the population for the corresponding filter settings. Also in this case, the leftmost boxplot of each series shows the results without filtering: *GPPLUS* in plot (b), *GPMUL* in plot (d), and *LS* in plot (f).

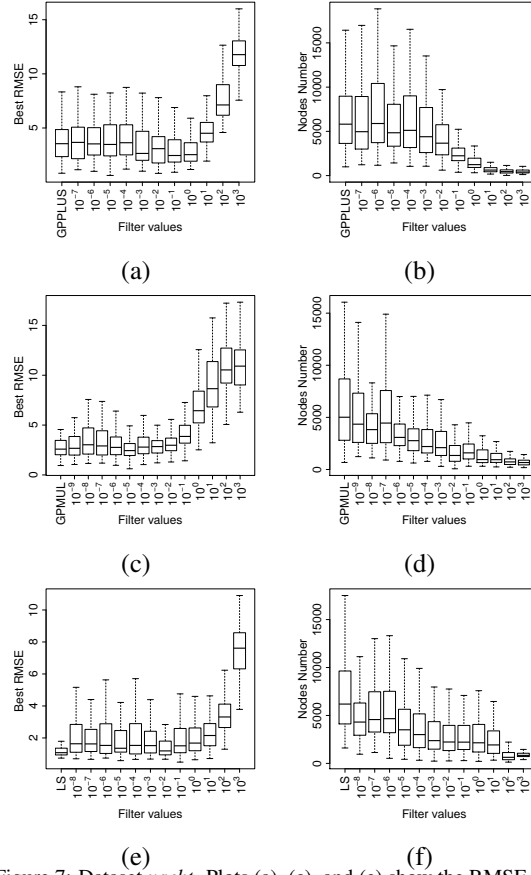


Figure 7: Dataset *yacht*. Plots (a), (c), and (e) show the RMSE on the test set for different values of the filter parameter. The leftmost boxplot of each line shows the results without filtering: *GPPLUS* in plot (a), *GPMUL* in plot (c), and *LS* in plot (e). Plots (b), (d), and (f) show the average number of nodes in the population for the corresponding filter settings. Also in this case, the leftmost boxplot of each series shows the results without filtering: *GPPLUS* in plot (b), *GPMUL* in plot (d), and *LS* in plot (f).

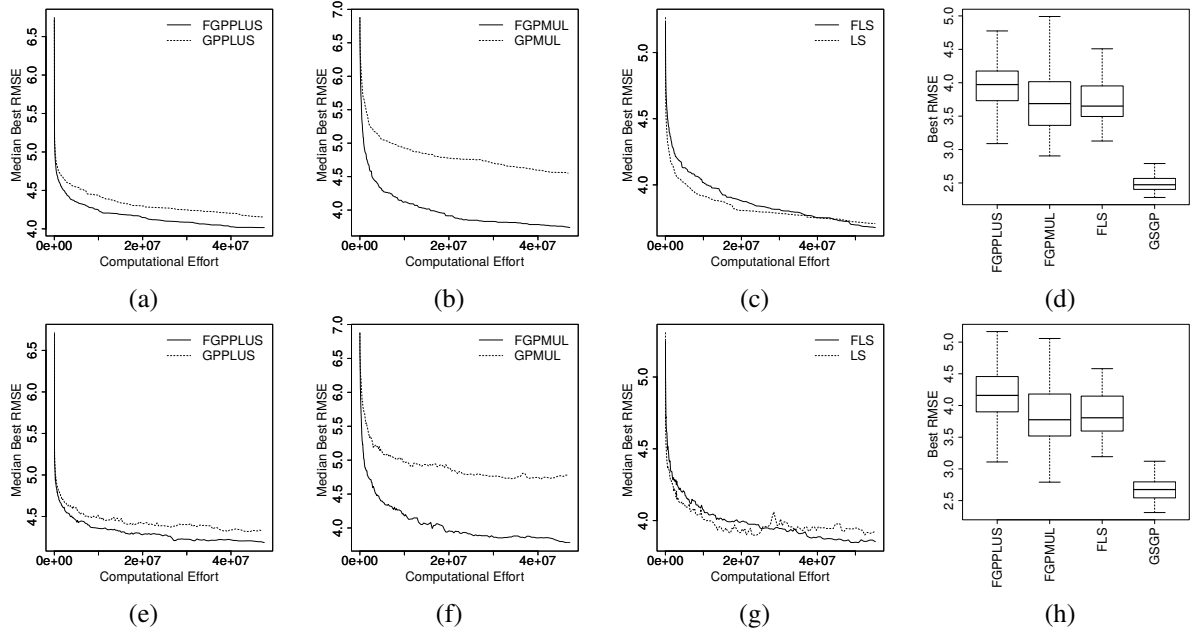


Figure 8: Dataset *airfoil*. The first line (i.e. plots (a), (b), (c) and (d)) reports the results on the training set. The second line (i.e. plots (e), (f), (g) and (h)) reports the results on the test set. Plots (a) and (e) report the results of the *GPPLUS* technique. Plots (b) and (f) report the results of the *GPMUL* technique. Plots (c) and (g) report the results of the *LS* technique. Plots-(d) and (h) report the performance of *GSQP* as well as those achieved by the 3 best variants (i.e., with or without filters) of the proposed system based on equivalence classes.

After the preliminary study to determine the ideal value of the filter parameter, the discussion of the results continues by considering a comparison between the proposed systems and *GSQP*. For each studied technique, we report the median best RMSE against the computational effort. In the first phase, we compare *GPPLUS*, *GPMUL*, and *LS* with their respective filtered counterparts (i.e. *FGPPLUS*, *FGPMUL*, and *FLS*, respectively). Then, only the methods with the best RMSE resulting from these experiments are compared with *GSQP*. This allows us to report the comparison between the studied methods and *GSQP* in a more readable way. These results are reported in Figures 8 to 14. In all these figures, the upper row of plots (i.e. the row containing plots (a), (b), (c) and (d)) reports the results on the training set, and the lower row (i.e. the row containing plots (e), (f), (g) and (h)) reports the results on the test set. The different studied methods are, instead, organized by columns; from leftmost to rightmost, we report the results of *GPPLUS*, *GPMUL*, *LS* and, finally, a comparison of *GSQP* and the three methods that achieved the best results (including the variants with and without filters). Concerning the test problems, the results are organized as follows: Figure 8 reports the results for the *airfoil* dataset; Figure 9 reports the results for the *concrete* dataset; Figure 10 reports the results for the *motor* dataset; Figure 11 reports the results for the *total* dataset; Figure 12 reports the results for the *protein* dataset; Figure 13 reports the results for the *slump* dataset; and finally, Figure 14 reports the results for the *yacht* dataset.

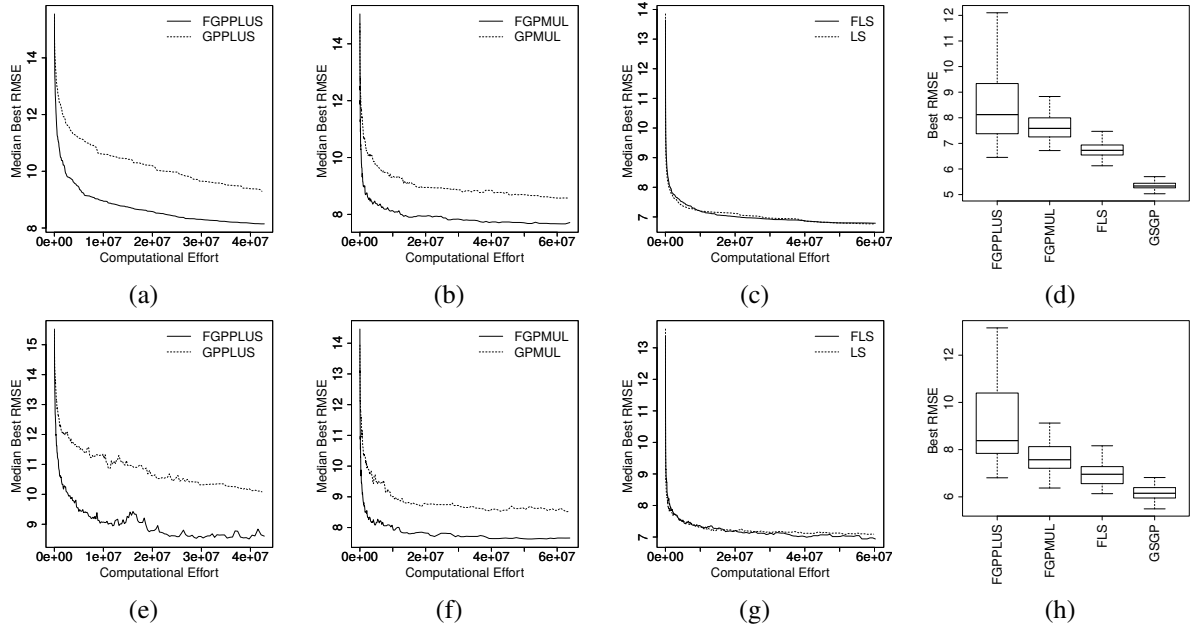


Figure 9: Dataset *concrete*. The first line (i.e. plots (a), (b), (c) and (d)) reports the results on the training set. The second line (i.e. plots (e), (f), (g) and (h)) reports the results on the test set. Plots (a) and (e) report the results of the *GPPLUS* technique. Plots (b) and (f) report the results of the *GPMUL* technique. Plots (c) and (g) report the results of the *LS* technique. Plots-(d) and (h) report the performance of GSGP as well as those achieved by the 3 best variants (i.e., with or without filters) of the proposed system based on equivalence classes.

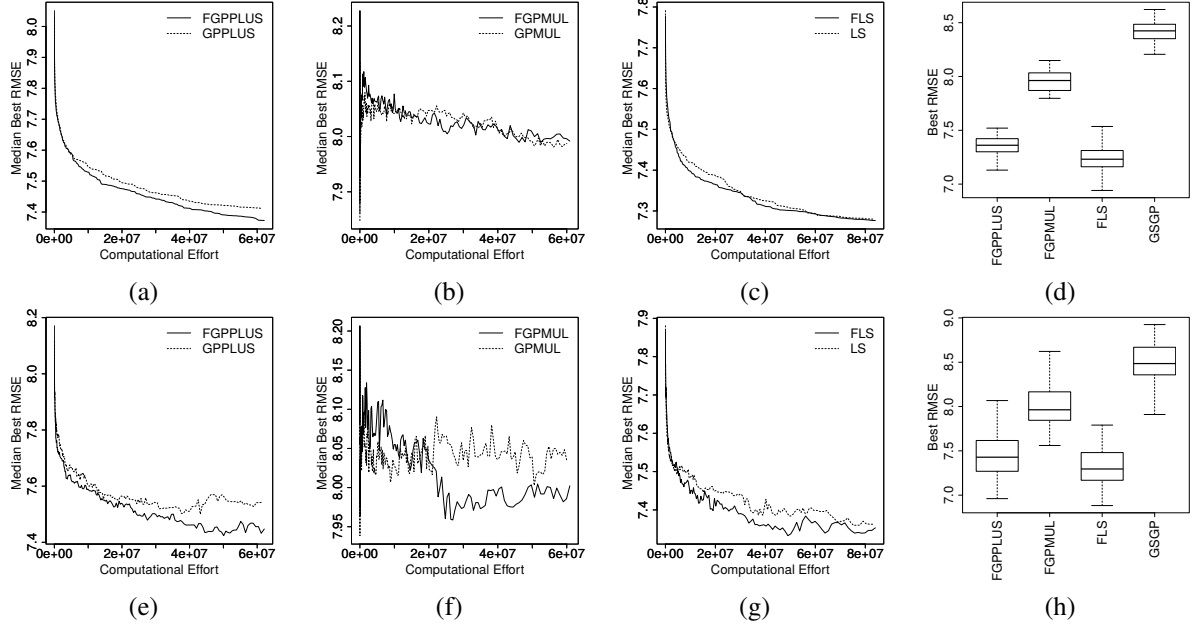


Figure 10: Dataset *motor*. The first line (i.e. plots (a), (b), (c) and (d)) reports the results on the training set. The second line (i.e. plots (e), (f), (g) and (h)) reports the results on the test set. Plots (a) and (e) report the results of the *GPPLUS* technique. Plots (b) and (f) report the results of the *GPMUL* technique. Plots (c) and (g) report the results of the *LS* technique. Plots-(d) and (h) report the performance of GSGP as well as those achieved by the 3 best variants (i.e., with or without filters) of the proposed system based on equivalence classes.

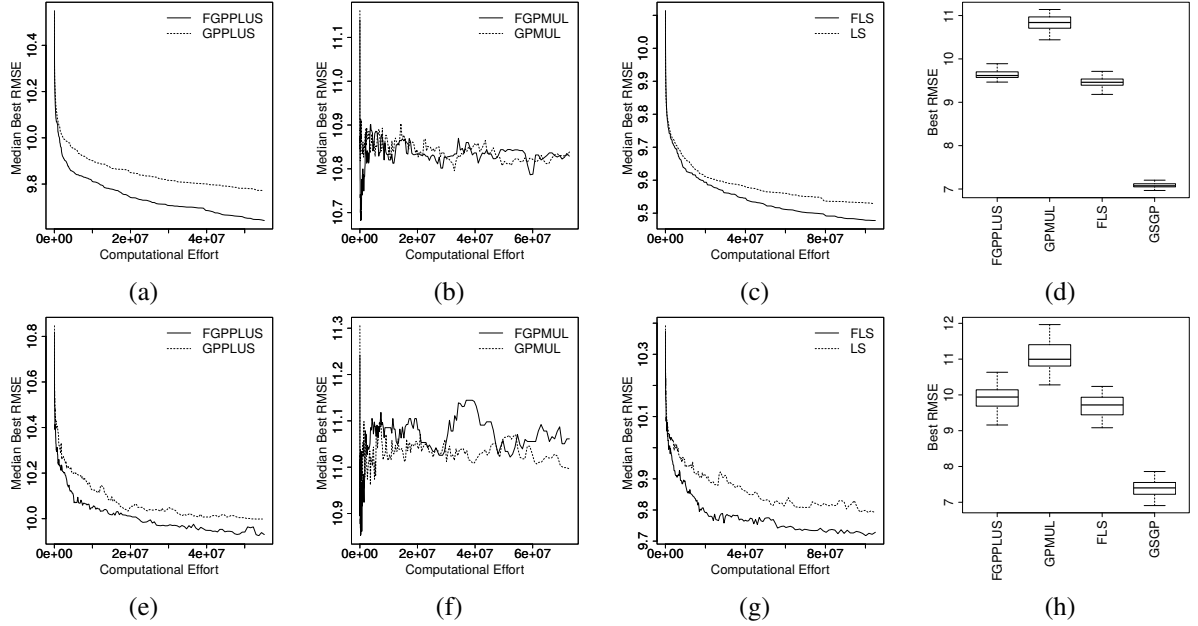


Figure 11: Dataset *total*. The first line (i.e. plots (a), (b), (c) and (d)) reports the results on the training set. The second line (i.e. plots (e), (f), (g) and (h)) reports the results on the test set. Plots (a) and (e) report the results of the *GPPLUS* technique. Plots (b) and (f) report the results of the *GPMUL* technique. Plots (c) and (g) report the results of the *LS* technique. Plots-(d) and (h) report the performance of GSGP as well as those achieved by the 3 best variants (i.e., with or without filters) of the proposed system based on equivalence classes.

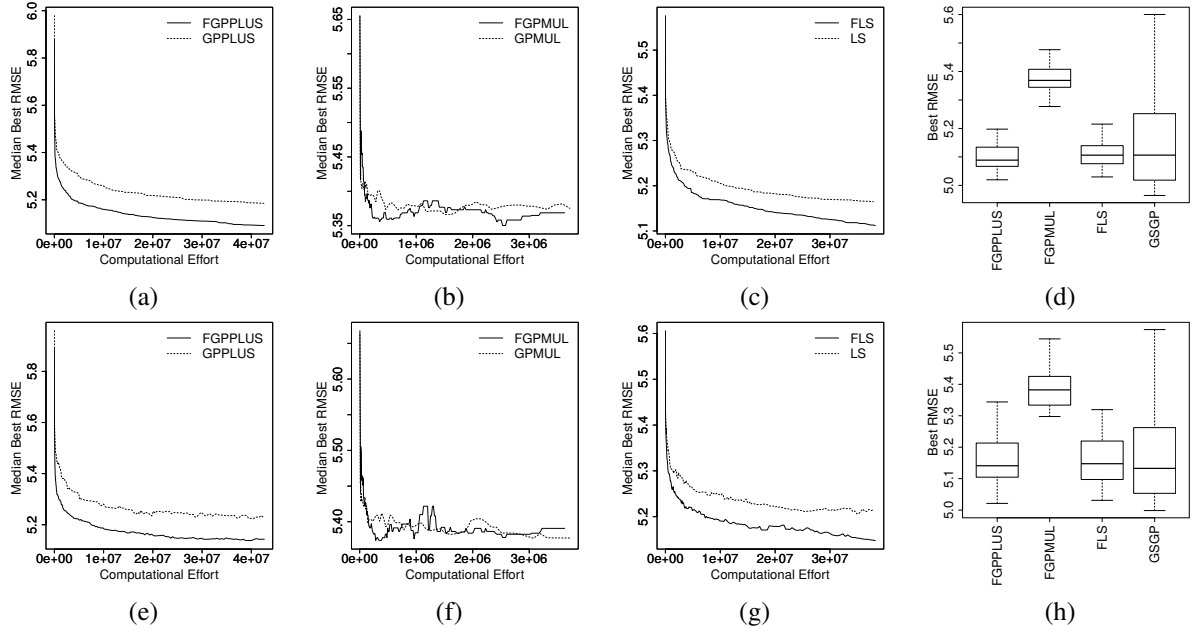


Figure 12: Dataset *protein*. The first line (i.e. plots (a), (b), (c) and (d)) reports the results on the training set. The second line (i.e. plots (e), (f), (g) and (h)) reports the results on the test set. Plots (a) and (e) report the results of the *GPPLUS* technique. Plots (b) and (f) report the results of the *GPMUL* technique. Plots (c) and (g) report the results of the *LS* technique. Plots-(d) and (h) report the performance of GSGP as well as those achieved by the 3 best variants (i.e., with or without filters) of the proposed system based on equivalence classes.

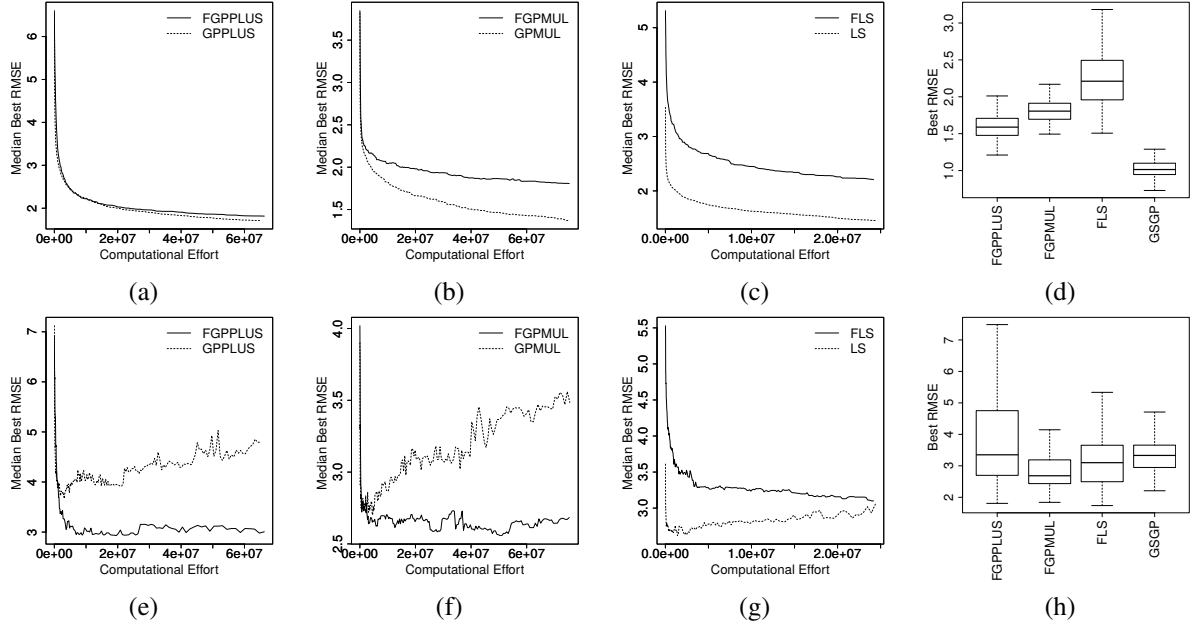


Figure 13: Dataset *slump*. The first line (i.e. plots (a), (b), (c) and (d)) reports the results on the training set. The second line (i.e. plots (e), (f), (g) and (h)) reports the results on the test set. Plots (a) and (e) report the results of the *GPPLUS* technique. Plots (b) and (f) report the results of the *GPMUL* technique. Plots (c) and (g) report the results of the *LS* technique. Plots-(d) and (h) report the performance of GSGP as well as those achieved by the 3 best variants (i.e., with or without filters) of the proposed system based on equivalence classes.

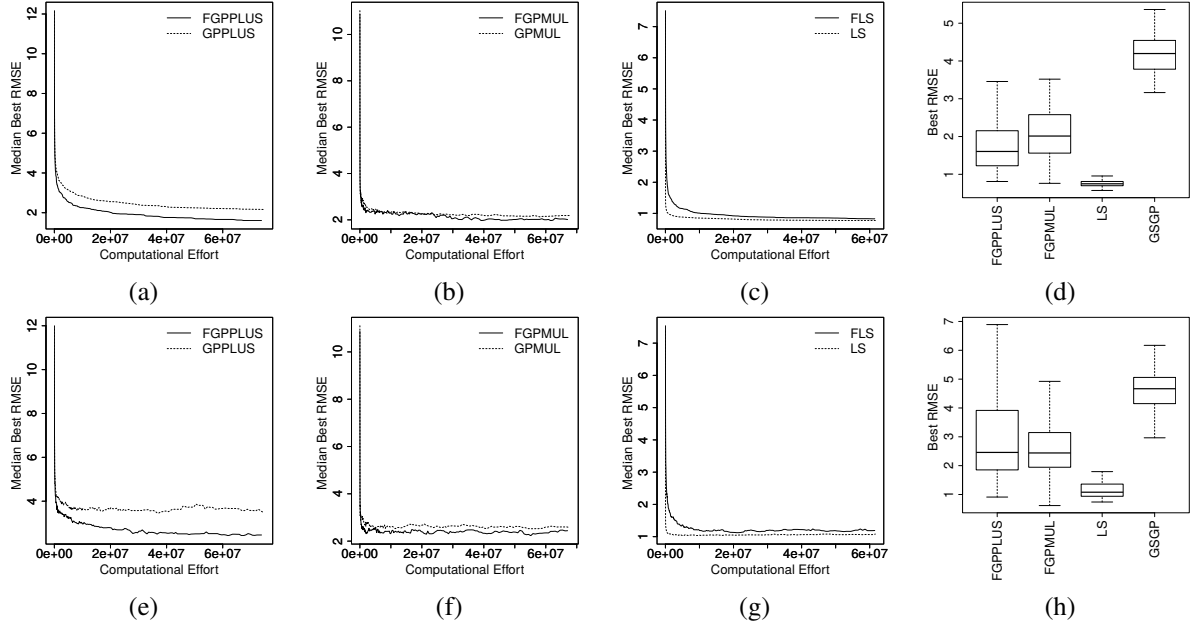


Figure 14: Dataset *yacht*. The first line (i.e. plots (a), (b), (c) and (d)) reports the results on the training set. The second line (i.e. plots (e), (f), (g) and (h)) reports the results on the test set. Plots (a) and (e) report the results of the *GPPLUS* technique. Plots (b) and (f) report the results of the *GPMUL* technique. Plots (c) and (g) report the results of the *LS* technique. Plots-(d) and (h) report the performance of *GSGP* as well as those achieved by the 3 best variants (i.e., with or without filters) of the proposed system based on equivalence classes.

As we can see from Figures 8 to 14, *GSGP* is generally the method that returned the best RMSE on the training set, except for the motor and protein problems (where the best values of RMSE were found by *FGPPPLUS* and *FLS*, for both the training and test sets) and for the yacht problem (where the best RMSE was found by *FLS*, for both training and test sets). However, *GSGP* appears to not generalize well and produces much bigger programs than the methods presented here, which is a known drawback of geometric semantic operators [3]. Moreover, *GSGP* was outperformed by *FGPMUL*, *FGPPPLUS*, and *FLS* on the test set when the slump problem was considered. Filtering appears to be beneficial for *GPMUL*, *GPPLUS*, and *LS* (as confirmed by our statistical tests presented later and by the deeper analysis on the effect of filtering that was offered in the first part of this section). In fact, the filtered versions always outperformed the non-filtered ones on both the training and test sets, except for the case of the slump problem on the training set. An interesting (and desirable) property of *GPMUL* and *GPPLUS*, as well as their filtered counterparts, relates to their behavior on the test set: the RMSE values are comparable to those achieved on the training set, hence suggesting that the solutions returned by these methods are quite robust. As a partial conclusion, and keeping in mind that all the variants studied outperform standard GP for all the problems considered (results not shown here to save space), we may assert that *GPPLUS* and *GPMUL* show interesting performance, corroborating the hypothesis that searching using EQC is a viable option.

To have a clearer picture of the impact of filtering on the techniques considered, we performed a statistical analysis of the results, for the experiments in which the basic techniques (*GPPLUS*, *GPMUL*, and *LS*) are compared to their filtered counterparts. Statistics refer to the experiments reported in Figures 8 to 14. The p -values were calculated using the Wilcoxon rank-sum test for pairwise data comparison with a significance value of $\alpha = 0.05$. Table 4 reports the results of this study with respect to the RMSE. As we can see, in the vast majority of cases, filtering brings a statistically significant advantage in terms of RMSE. This can be seen by observing that the third and fifth columns of the table often contain negative values (i.e., a lower RMSE obtained by using the filtered variants). The only exceptions are *GPMUL* on the total dataset and *LS* on the yacht dataset, where the non-filtered techniques provide better RMSE values than the respective filtered counterparts, for both the training and test sets. It is interesting to point out the case of the slump dataset. This is the only studied problem in which the best method on the training set does not correspond to the best method on the test set. Specifically, for this dataset, *FGPMUL* and *FLS* perform worse than their non-filtered counterparts on training data but significantly better on the test set.

The same statistical validation was performed considering the size of the individuals obtained in the experiments reported in Figures 8 to 14. Both the average size of the individuals in the population and the size of the best individual were studied. The results of this statistical study are reported in Table 5 for the three systems. One can observe that, for the majority of the datasets, filtered systems are able to evolve populations with an average size of the individuals that is smaller than their non-filtered counterparts in a statistically significant way. This is very important since the size of a model is generally strongly related to its readability and interpretability, which are very important features in many applicative domains of machine learning.

Technique	Dataset	P-value Training	% diff on training	P-value Test	% diff on test
GPMUL	<i>airfoil</i>	0.00	-18.99	0.00	-21.00
	<i>concrete</i>	0.00	-11.52	0.00	-11.11
	<i>motor</i>	0.02	-0.31	0.46	-0.88
	<i>total</i>	0.77	-0.10	0.75	0.60
	<i>protein</i>	0.97	-0.06	0.32	-0.33
	<i>slump</i>	0.00	32.90	0.00	-25.12
	<i>yacht</i>	0.12	-6.58	0.16	-5.67
GPPLUS	<i>airfoil</i>	0.00	-4.40	0.05	-3.79
	<i>concrete</i>	0.00	-12.60	0.00	-16.87
	<i>motor</i>	0.01	-0.67	0.01	-1.52
	<i>total</i>	0.00	-1.59	0.01	-0.58
	<i>protein</i>	0.00	-1.86	0.01	-1.65
	<i>slump</i>	0.00	-7.58	0.00	-28.82
	<i>yacht</i>	0.00	-22.18	0.00	-30.75
LS	<i>airfoil</i>	0.69	-1.57	0.02	-3.02
	<i>concrete</i>	0.66	-0.56	0.27	-1.90
	<i>motor</i>	0.00	-0.67	0.10	-0.91
	<i>total</i>	0.03	-0.71	0.18	-0.76
	<i>protein</i>	0.00	-1.14	0.30	-1.28
	<i>slump</i>	0.00	82.43	0.01	-1.97
	<i>yacht</i>	0.00	10.22	0.16	10.29

Table 4: For each technique, the table shows the percentage variation of RMSE, at the last generation, for both training and test, achieved using filters. The table also reports the p -values calculated using the Wilcoxon rank-sum test for pairwise data comparison (significance level was $\alpha = 0.05$). Bold numbers denote statistically significant results and negative values mean an advantage in using filters. Statistics refer to the same experiments reported in Figures 8 to 14.

Technique	Dataset	P-value average size	% Diff pop. Average size	P-value Best size	% Diff on best ind. Size
GPMUL	<i>airfoil</i>	0.00	-47.66	0.01	-43.03
	<i>concrete</i>	0.59	10.38	0.53	8.63
	<i>motor</i>	0.34	33.98	0.48	45.32
	<i>total</i>	0.00	-63.76	0.00	-63.00
	<i>protein</i>	0.00	-96.69	0.00	-93.70
	<i>slump</i>	0.00	-93.31	0.00	-88.70
	<i>yacht</i>	0.00	-44.85	0.00	-40.39
GPPLUS	<i>airfoil</i>	0.00	-24.39	0.00	-28.24
	<i>concrete</i>	0.27	-5.77	0.94	0.90
	<i>motor</i>	0.06	-18.88	0.05	-35.05
	<i>total</i>	0.74	-17.98	0.99	-15.07
	<i>protein</i>	0.05	-26.27	0.24	-13.14
	<i>slump</i>	0.00	-58.43	0.00	-50.16
	<i>yacht</i>	0.00	-61.48	0.00	-57.62
LS	<i>airfoil</i>	0.00	-37.70	0.05	-19.02
	<i>concrete</i>	0.19	-6.43	0.33	-10.75
	<i>motor</i>	0.33	-13.22	0.23	6.14
	<i>total</i>	0.23	-26.27	0.27	-27.34
	<i>protein</i>	0.70	-1.47	0.34	14.97
	<i>slump</i>	0.00	-81.94	0.00	-81.09
	<i>yacht</i>	0.00	-63.96	0.00	-63.85

Table 5: For each technique, the table shows the percentage variation related to the size of the best individual (last column) and to the average size of the population (third column), in the last generation, when the filter is used. The table also reports the p -values calculated using the Wilcoxon rank-sum test for pairwise data comparison, with a significance level of $\alpha = 0.05$. Bold numbers denote significant results and negative values mean that the filtered version has generated smaller individuals than the non-filtered one. For the filtered versions, we selected the filter parameter that produces the best training performance (RMSE). Statistics refer to the same experiments reported in Figures 8 to 14.

5. Conclusions and Future Work

In this paper, the idea of semantics-based equivalence classes for Genetic Programming (GP) was presented. This idea is general, and it can be implemented in several different ways. In this paper, it was implemented by means of two simple GP systems, called *GPPLUS* and *GPMUL*. Each of these systems uses a different definition of equivalence: *GPPLUS* identifies a collection of affine subspaces in the semantic space, while *GPMUL* defines a projective space in the semantic space. Moreover, filtered versions of these two systems, called *FGPPLUS* and *FGPMUL*, were introduced. In these versions, individuals are rejected if at least another individual belonging to the same equivalence class already exists in the population. Finally, semantic filters were also applied to a well-known and widely used GP system, like linear scaling. Experiments to test the performance of the systems proposed were conducted on seven complex real-life applications. The results indicate that the use of semantics-based equivalence classes in GP is promising. Also, filters are often beneficial for improving the systems' performance. Last but not least, the use of filters allows the studied systems to generate individuals that are significantly smaller than their unfiltered counterparts and definitely much smaller than the ones generated by geometric semantic GP. This is an important result in all the applications where the readability of the model is a fundamental factor.

In the future, we plan to develop more sophisticated definitions of equivalence. We are also currently investigating the concept of "weak" equivalence in which we relax the equivalence property for a number of fitness cases. Finally, we plan to develop a new GP system, based on the concept of semantics-based equivalence classes, which specializes in solving particular image processing and pattern recognition tasks.

Bibliography

- [1] J. R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, Cambridge, MA, USA, 1992.
- [2] R. A. Fisher, The goodness of fit of regression formulae, and the distribution of regression coefficients, *Journal of the Royal Statistical Society* 85 (1922) 597–612.
- [3] A. Moraglio, K. Krawiec, C. G. Johnson, Geometric semantic genetic programming, in: C. A. Coello Coello, V. Cutello, K. Deb, S. Forrest, G. Nicosia, M. Pavone (Eds.), *Parallel Problem Solving from Nature, PPSN XII (part 1)*, volume 7491 of *LNCS*, Springer, 2012, pp. 21–31.
- [4] L. Vanneschi, M. Castelli, S. Silva, A survey of semantic methods in genetic programming, *Genetic Programming and Evolvable Machines* 15 (2014) 195–214.
- [5] M. O'Neill, Semantic methods in genetic programming, *Genetic Programming and Evolvable Machines* 17 (2016) 3–4.
- [6] T. P. Pawlak, B. Wieloch, K. Krawiec, Review and comparative analysis of geometric semantic crossovers., *Genetic Programming and Evolvable Machines* 16 (2015) 351–386.
- [7] L. Vanneschi, *An Introduction to Geometric Semantic Genetic Programming*, Springer International Publishing, Cham, pp. 3–42.
- [8] M. Keijzer, Improving symbolic regression with interval arithmetic and linear scaling, in: *Genetic programming*, Springer, 2003, pp. 70–82.
- [9] M. Keijzer, Scaled symbolic regression, *Genetic Programming and Evolvable Machines* 5 (2004) 259–269.
- [10] L. Vanneschi, M. Castelli, L. Manzoni, S. Silva, A new implementation of geometric semantic GP and its application to problems in pharmacokinetics, in: K. Krawiec, A. Moraglio, T. Hu, A. S. Uyar, B. Hu (Eds.), *Proceedings of EuroGP 2013, LNCS*, Springer, 2013, pp. 205–216.
- [11] I. Gonçalves, S. Silva, C. M. Fonseca, On the Generalization Ability of Geometric Semantic Genetic Programming, Springer International Publishing, Cham, pp. 41–52.
- [12] M. Graff, E. S. Tellez, E. Villasenor, S. Miranda-Jimenez, Semantic genetic programming operators based on projections in the phenotype space, *Research in Computing Science* 94 (2015) 73–85.
- [13] S. Ruberto, L. Vanneschi, M. Castelli, S. Silva, ESAGP - a semantic gp framework based on alignment in the error space, in: *Genetic Programming*, Springer, 2014, pp. 150–161.
- [14] I. Gonçalves, S. Silva, C. M. Fonseca, M. Castelli, Arbitrarily close alignments in the error space: a geometric semantic genetic programming approach, in: T. Friedrich, F. Neumann, A. M. Sutton, M. Middendorf, X. Li, E. Hart, M. Zhang, Y. Akimoto, P. A. N. Bosman, T. Soule, R. Mikkilainen, D. Loiacono, J. Togelius, M. Lopez-Ibanez, H. Hoos, J. Handl, F. Gomez, C. M. Fonseca, H. Trautmann, A. Moraglio, W. F. Punch, K. Krawiec, Z. Vasecek, T. Jansen, J. Smith, S. Ludwig, J. Merelo, B. Naujoks, E. Alba, G. Ochoa, S. Poulding, D. Sudholt, T. Koetzing (Eds.), *GECCO 2016 Companion Volume*, ACM, Denver, USA, 2016, pp. 99–100.
- [15] T. P. Pawlak, K. Krawiec, Progress properties and fitness bounds for geometric semantic search operators, *Genetic Programming and Evolvable Machines* 17 (2016) 5–23.
- [16] Q. U. Nguyen, T. A. Pham, X. H. Nguyen, J. McDermott, Subtree semantic geometric crossover for genetic programming, *Genetic Programming and Evolvable Machines* 17 (2016) 25–53.
- [17] Ł. Bartczuk, K. Łapa, P. Koprinkova-Hristova, A New Method for Generating of Fuzzy Rules for the Nonlinear Modelling Based on Semantic Genetic Programming, Springer International Publishing, Cham, pp. 262–278.
- [18] L. O. V. B. Oliveira, L. F. Miranda, G. L. Pappa, F. E. B. Otero, R. H. C. Takahashi, Reducing Dimensionality to Improve Search in Semantic Genetic Programming, Springer International Publishing, Cham, pp. 375–385.
- [19] L. O. V. Oliveira, F. E. Otero, G. L. Pappa, A dispersion operator for geometric semantic genetic programming, in: *Proceedings of the Genetic and Evolutionary Computation Conference 2016, GECCO '16*, ACM, New York, NY, USA, 2016, pp. 773–780.

- [20] L. Spector, Autoconstructive evolution: Push, pushGP, and pushpop, in: L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, E. Burke (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, Morgan Kaufmann, San Francisco, California, USA, 2001, pp. 137–146.
- [21] M. Castelli, L. Vanneschi, S. Silva, Semantic search-based genetic programming and the effect of intron deletion, *IEEE transactions on cybernetics* 44 (2014) 103–113.
- [22] S. Silva, S. Dignum, Extending operator equalisation: Fitness based self adaptive length distribution for bloat free gp., in: *EuroGP*, Springer, pp. 159–170.
- [23] L. Beadle, C. Johnson, Semantically driven crossover in genetic programming, in: *Proceedings of the IEEE World Congress on Computational Intelligence*, IEEE Press, Hong Kong, 2008, pp. 111–116.
- [24] M. Castelli, L. Vanneschi, S. Silva, Prediction of high performance concrete strength using genetic programming with geometric semantic genetic operators, *Expert Systems with Applications* 40 (2013) 6856–6862.
- [25] M. Castelli, L. Vanneschi, S. Silva, Prediction of the unified Parkinson’s disease rating scale assessment using a genetic programming system with geometric semantic genetic operators, *Expert Systems with Applications* 41 (2014) 4608 – 4616.
- [26] M. Castelli, L. Vanneschi, L. Manzoni, A. Popovič, Semantic genetic programming for fast and accurate data knowledge discovery, *Swarm and Evolutionary Computation* 26 (2016) 1–7.
- [27] T. Brooks, D. Pope, A. Marcolini, Airfoil self-noise and prediction, Technical report, NASA RP-1218 (1989).
- [28] I.-C. Yeh, Simulation of concrete slump using neural networks, *Proceedings of the Institution of Civil Engineers-Construction Materials* 162 (2009) 11–18.
- [29] I. Ortigosa, R. López, J. García, A neural networks approach to residuary resistance of sailing yachts prediction, *Proceedings of the International Conference on Marine Engineering MARINE 2007* (2007) 250.
- [30] F. Fernández, M. Tomassini, L. Vanneschi, Studying the influence of communication topology and migration on distributed genetic programming, in: *Genetic Programming*, Springer, 2001, pp. 51–63.