
Doctoral Thesis Research Proposal

# Genetic Programming with Semantic Equivalence Classes

PhD Program in Computer Science: XXIX cycle

*Supervisor:*

Prof. Leonardo Vanneschi
lvanneschi@isegi.unl.pt

*Author:*

Stefano Ruberto
stefano.ruberto@gssi.infn.it

*Internal advisor:*

Dr. Ivano Malavolta
ivano.malavolta@gssi.infn.it

June 8, 2017

**GSSI Gran Sasso Science Institute**
Viale Francesco Crispi, 7 - 67100 L'Aquila - Italy

# *Abstract*

This dissertation has been carried out in the context of Evolutionary Algorithms and specifically in the application of *Genetic Programming* (GP) techniques to *symbolic regression problems*. In this class of problems, given an input vector of independent numerical variables, algorithms are able to find models that approximate a desired scalar output. Among the machine learning techniques, GP has the important advantage of synthesizing as output a full intelligible model. Recently, interesting new approaches measure the computational behaviour of such models, their *semantics*, considering the output vector on all the test cases in the training set. By characterizing the behaviour of the model at hand, it is possible to improve the efficiency during exploration of the solution space, while avoiding duplicate behaviours and directing the evolution towards the desired directions. In this context, *geometric* approaches play a very important role, mainly due to their performance in terms of models accuracy and ability to generalize. However, they also suffer from a number of drawbacks, i.e., the exponential growth of the model's structure (*bloat*) and a slow convergence rate.

In this dissertation we propose a radically different approach, which is based on a new semantic framework that performs effectively against bloat, thus preserving models clarity, and reaching comparable accuracy and generalization performances starting from a common background. More specifically, we introduce the concept of semantics-based *equivalence classes*. The approach is implemented by means of two different novel genetic programming systems, in which two different definitions of equivalence are used. In both these 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 to find any solution that belongs to the same equivalence class as a global optimum. Equivalence classes generalize the use of *peseudo-distances*, as opposed to the traditional use of proper metrics in semantic analysis. Furthermore, we propose improvements to these genetic programming systems in which, once a solution belonging to a particular equivalence class is generated, no other solutions in that class are accepted anymore in the population during the evolution. We call *filtered systems* these improved versions with respect to efficiency and speed in the solution space exploration phase.

We validated the proposed approach via a experimental results obtained on seven complex real-life test problems. Experimental results show that using equivalence classes is a promising direction and that filters are generally helpful to improve the performance

of the systems. Experimental results also show that filters are useful to improve the performance of a state-of-the-art GP method. Indeed, we did an extensive experimentation on the well-known *linear scaling* technique; this contribution is rather general and can work in synergy with the aforementioned *semantic/geometric* techniques and with other machine learning frameworks.

# Declaration.

This thesis has been composed by myself and the presented work is my own under the guidance of my supervisors Leonardo Vanneschi, Mauro Castelli, Ivano Malavolta.

Moreover, Chapter 1 contains text that is a re-elaboration from my paper [1] co-authored with Leonardo Vanneschi and Mauro Castelli. Chapter 2 is essentially a internal GSSI report I presented on my 2nd year admission in PhD program. Part of Chapter 3 is a re-elaboration of [1] co-authored with Leonardo Vanneschi and Mauro Castelli. Chapters 4 and 5 are essentially sections of[1] co-authored with Leonardo Vanneschi and Mauro Castelli. Appendix A is extracted from [2] co-authored with Leonardo Vanneschi , Mauro Castelli, and Sara Silva.

# Contents

**List of Figures** **vi**

**1 Introduction** **1**
1.1 Semantic genetic programming and open problems . . . 1
1.2 Equivalence Classes in Genetic Programming . . . 4
1.3 Contributions overview . . . 5
1.4 Structure of this dissertation . . . 6

**2 Introduction to genetic programming** **7**
2.1 Genetic Programming . . . 7
2.1.1 Introduction to Genetic Programming . . . 8
2.1.2 Main features . . . 9
2.1.3 GP solutions structures : the trees . . . 11
2.1.3.1 Functional symbols . . . 11
2.1.3.2 Terminal symbols . . . 11
2.1.3.3 Tree structures . . . 12
2.1.4 The fitness function. . . . 13
2.1.4.1 Fitness function example in regression . . . 13
2.1.5 Initialization . . . 14
2.1.5.1 Initialization methods: *grow* and *full* . . . 15
2.1.5.2 Ramped Half-and-Half Method . . . 15
2.1.6 Selection . . . 15
2.1.6.1 Tournament selection . . . 16
2.1.7 Genetic operators . . . 16
2.1.7.1 Crossover . . . 16
2.1.7.2 Mutation . . . 17
2.2 Introducing Genetic Programming Open Problems . . . 18
2.2.1 Bloat . . . 18
2.2.2 Overfitting . . . 20
2.2.3 Diversity . . . 21
2.2.3.1 Genotype Diversity . . . 22
2.2.3.2 Phenotype diversity: investigating semantic. . . . 22
2.2.4 Semantic in Genetic Programming . . . 23
2.2.4.1 Semantic analysis during initialization . . . 25
2.2.4.2 Semantic in selection . . . 26
2.2.4.3 Semantic genetic operators. . . . 26

**3 Research Contribution** **27**
3.1 More on open problems and semantic Genetic Programming . . . . . 27
3.1.1 Geometric Semantic Genetic Programming . . . . . 28
3.1.2 ESAGP . . . . . 31
3.2 Proposal . . . . . 36

**4 Semantic Equivalence Classes Genetic Programming** **42**
4.1 Methodology . . . . . 42
4.1.1 GPPLUS: GP by Translation . . . . . 45
4.1.2 GPMUL: GP by proportions . . . . . 46

**5 Experimental study** **50**
5.0.1 Systems and test problems . . . . . 50
5.0.2 Experimental Results . . . . . 52

**6 Conclusions and Future Work** **80**

**A ESAGP** **83**
A.1 Alignment in the Error Space . . . . . 83
A.2 One Step Error Space Alignment GP: ESAGP-1 . . . . . 85
A.3 Two Steps Error Space Alignment GP: ESAGP-2 . . . . . 87
**Generalizing to $\mu$ dimensions.** . . . . . 89

# List of Figures

2.1 Genetic Programming general cycle schema. The big arrows indicate transitions to the next phase of the cycle starting at the top of the figure. 9
2.2 Expression 2.3 represented through two different parse trees. . . . . 12
2.3 The crossover genetic operator exchange sub-trees between the parents at the crossover points. . . . . 17

5.1 Dataset *airfoil* (training). Results are relative to: *GPPLUS* technique (5.1a), *GPMUL* (5.1b), *LS* (5.1c). Figures (5.1a), (5.1b), (5.1c), have median fitness (RMSE) on the vertical axes and computational effort on horizontal axes (calculated as the number of nodes evaluated during training). Figure (5.1d) reports the performance of GSGP as well as the ones achieved by the 3 best variants (i.e., with or without filters) of the proposed system based on equivalence classes. . . . . 54
5.2 Dataset *airfoil* (test). Results are relative to: *GPPLUS* technique (5.2a), *GPMUL* (5.2b), *LS* (5.2c). Figures (5.2a), (5.2b), (5.2c), have median fitness (RMSE) on the vertical axes and computational effort on horizontal axes (calculated as the number of nodes evaluated during training). Figure (5.2d) reports the performance of GSGP as well as the ones achieved by the 3 best variants (i.e., with or without filters) of the proposed system based on equivalence classes. . . . . 55
5.3 Dataset *concrete* (training). Results are relative to: *GPPLUS* technique (5.3a), *GPMUL* (5.3b), *LS* (5.3c). Figures (5.3a), (5.3b), (5.3c), have median fitness (RMSE) on the vertical axes and computational effort on horizontal axes (calculated as the number of nodes evaluated during training). Figure (5.3d) reports the performance of GSGP as well as the ones achieved by the 3 best variants (i.e., with or without filters) of the proposed system based on equivalence classes. . . . . 56
5.4 Dataset *concrete* (test). Results are relative to: *GPPLUS* technique (5.4a), *GPMUL* (5.4b), *LS* (5.4c). Figures (5.4a), (5.4b), (5.4c), have median fitness (RMSE) on the vertical axes and computational effort on horizontal axes (calculated as the number of nodes evaluated during training). Figure (5.4d) reports the performance of GSGP as well as the ones achieved by the 3 best variants (i.e., with or without filters) of the proposed system based on equivalence classes. . . . . 57

5.5 Dataset *motor* (training). Results are relative to: *GPPLUS* technique (5.5a), *GPMUL* (5.5b), *LS* (5.5c). Figures (5.5a), (5.5b), (5.5c), have median fitness (RMSE) on the vertical axes and computational effort on horizontal axes (calculated as the number of nodes evaluated during training). Figure (5.5d) reports the performance of GSGP as well as the ones achieved by the 3 best variants (i.e., with or without filters) of the proposed system based on equivalence classes. . . . . 58

5.6 Dataset *motor* (test). Results are relative to: *GPPLUS* technique (5.6a), *GPMUL* (5.6b), *LS* (5.6c). Figures (5.6a), (5.6b), (5.6c), have median fitness (RMSE) on the vertical axes and computational effort on horizontal axes (calculated as the number of nodes evaluated during training). Figure (5.6d) reports the performance of GSGP as well as the ones achieved by the 3 best variants (i.e., with or without filters) of the proposed system based on equivalence classes. . . . . 59

5.7 Dataset *total* (training). Results are relative to: *GPPLUS* technique (5.7a), *GPMUL* (5.7b), *LS* (5.7c). Figures (5.7a), (5.7b), (5.7c), have median fitness (RMSE) on the vertical axes and computational effort on horizontal axes (calculated as the number of nodes evaluated during training). Figure (5.7d) reports the performance of GSGP as well as the ones achieved by the 3 best variants (i.e., with or without filters) of the proposed system based on equivalence classes. . . . . 60

5.8 Dataset *total* (test). Results are relative to: *GPPLUS* technique (5.8a), *GPMUL* (5.8b), *LS* (5.8c). Figures (5.8a), (5.8b), (5.8c), have median fitness (RMSE) on the vertical axes and computational effort on horizontal axes (calculated as the number of nodes evaluated during training). Figure (5.8d) reports the performance of GSGP as well as the ones achieved by the 3 best variants (i.e., with or without filters) of the proposed system based on equivalence classes. . . . . 61

5.9 Dataset *protein* (training). Results are relative to: *GPPLUS* technique (5.9a), *GPMUL* (5.9b), *LS* (5.9c). Figures (5.9a), (5.9b), (5.9c), have median fitness (RMSE) on the vertical axes and computational effort on horizontal axes (calculated as the number of nodes evaluated during training). Figure (5.9d) reports the performance of GSGP as well as the ones achieved by the 3 best variants (i.e., with or without filters) of the proposed system based on equivalence classes. . . . . 62

5.10 Dataset *protein* (test). Results are relative to: *GPPLUS* technique (5.10a), *GPMUL* (5.10b), *LS* (5.10c). Figures (5.10a), (5.10b), (5.10c), have median fitness (RMSE) on the vertical axes and computational effort on horizontal axes (calculated as the number of nodes evaluated during training). Figure (5.10d) reports the performance of GSGP as well as the ones achieved by the 3 best variants (i.e., with or without filters) of the proposed system based on equivalence classes. . . . . 63

5.11 Dataset *slump* (training). Results are relative to: *GPPLUS* technique (5.11a), *GPMUL* (5.11b), *LS* (5.11c). Figures (5.11a), (5.11b), (5.11c), have median fitness (RMSE) on the vertical axes and computational effort on horizontal axes (calculated as the number of nodes evaluated during training). Figure (5.11d) reports the performance of GSGP as well as the ones achieved by the 3 best variants (i.e., with or without filters) of the proposed system based on equivalence classes. . . . . 64

5.12 Dataset *slump* (test). Results are relative to: *GPPLUS* technique (5.12a), *GPMUL* (5.12b), *LS* (5.12c). Figures (5.12a), (5.12b), (5.12c), have median fitness (RMSE) on the vertical axes and computational effort on horizontal axes (calculated as the number of nodes evaluated during training). Figure (5.12d) reports the performance of GSGP as well as the ones achieved by the 3 best variants (i.e., with or without filters) of the proposed system based on equivalence classes. . . . . 65
5.13 Dataset *yacht* (training). Results are relative to: *GPPLUS* technique (5.13a), *GPMUL* (5.13b), *LS* (5.13c). Figures (5.13a), (5.13b), (5.13c), have median fitness (RMSE) on the vertical axes and computational effort on horizontal axes (calculated as the number of nodes evaluated during training). Figure (5.13d) reports the performance of GSGP as well as the ones achieved by the 3 best variants (i.e., with or without filters) of the proposed system based on equivalence classes. . . . . 66
5.14 Dataset *yacht* (test). Results are relative to: *GPPLUS* technique (5.14a), *GPMUL* (5.14b), *LS* (5.14c). Figures (5.14a), (5.14b), (5.14c), have median fitness (RMSE) on the vertical axes and computational effort on horizontal axes (calculated as the number of nodes evaluated during training). Figure (5.14d) reports the performance of GSGP as well as the ones achieved by the 3 best variants (i.e., with or without filters) of the proposed system based on equivalence classes. . . . . 67
5.15 Dataset *airfoil*. Figures (a), (c) and (e) show test fitness (RMSE) for different values of the filter parameter. The first boxplot of each series shows the results without filtering: *GPPLUS* on figure (a), *GPMUL* on (c) and *LS* on (e). Figures (b), (d) and (f) show the average number of nodes in the population for the corresponding filter settings. Also in this case, the first boxplot of each series shows the results without filtering: *GPPLUS* on figure (b), *GPMUL* on (d) and *LS* on (f). . . . . 71
5.16 Dataset *concrete*. Figures (a), (c) and (e) show test fitness (RMSE) for different values of the filter parameter. The first boxplot of each series shows the results without filtering: *GPPLUS* on figure (a), *GPMUL* on (c) and *LS* on (e). Figures (b), (d) and (f) show the average number of nodes in the population for the corresponding filter settings. Also in this case, the first boxplot of each series shows the results without filtering: *GPPLUS* on figure (b), *GPMUL* on (d) and *LS* on (f). . . . . 72
5.17 Dataset *motor*. Figures (a), (c) and (e) show test fitness (RMSE) for different values of the filter parameter. The first boxplot of each series shows the results without filtering: *GPPLUS* on figure (a), *GPMUL* on (c) and *LS* on (e). Figures (b), (d) and (f) show the average number of nodes in the population for the corresponding filter settings. Also in this case, the first boxplot of each series shows the results without filtering: *GPPLUS* on figure (b), *GPMUL* on (d) and *LS* on (f). . . . . 73
5.18 Dataset *total*. Figures (a), (c) and (e) show test fitness (RMSE) for different values of the filter parameter. The first boxplot of each series shows the results without filtering: *GPPLUS* on figure (a), *GPMUL* on (c) and *LS* on (e). Figures (b), (d) and (f) show the average number of nodes in the population for the corresponding filter settings. Also in this case, the first boxplot of each series shows the results without filtering: *GPPLUS* on figure (b), *GPMUL* on (d) and *LS* on (f). . . . . 74

5.19 Dataset *protein*. Figures (a), (c) and (e) show test fitness (RMSE) for different values of the filter parameter. The first boxplot of each series shows the results without filtering: *GPPLUS* on figure (a), *GPMUL* on (c) and *LS* on (e). Figures (b), (d) and (f) show the average number of nodes in the population for the corresponding filter settings. Also in this case, the first boxplot of each series shows the results without filtering: *GPPLUS* on figure (b), *GPMUL* on (d) and *LS* on (f). . . . . 75
5.20 Dataset *slump*. Figures (a), (c) and (e) show test fitness (RMSE) for different values of the filter parameter. The first boxplot of each series shows the results without filtering: *GPPLUS* on figure (a), *GPMUL* on (c) and *LS* on (e). Figures (b), (d) and (f) show the average number of nodes in the population for the corresponding filter settings. Also in this case, the first boxplot of each series shows the results without filtering: *GPPLUS* on figure (b), *GPMUL* on (d) and *LS* on (f). . . . . 76
5.21 Dataset *yacht*. Figures (a), (c) and (e) show test fitness (RMSE) for different values of the filter parameter. The first boxplot of each series shows the results without filtering: *GPPLUS* on figure (a), *GPMUL* on (c) and *LS* on (e). Figures (b), (d) and (f) show the average number of nodes in the population for the corresponding filter settings. Also in this case, the first boxplot of each series shows the results without filtering: *GPPLUS* on figure (b), *GPMUL* on (d) and *LS* on (f). . . . . 77

A.1 Part (a): Representation of a simple bi-dimensional error space. Individuals $A$ and $B$ are optimally aligned, i.e. their respective error vectors are directly proportional. The angle between the error vector of $A$ (as well as $B$) and the one of $C$ is $\theta$. Part (b): A simple tri-dimensional error space. We point out that it is possible to find a point $m$ that is aligned with the error vectors of any pair of individuals $A$ and $B$ and optimally aligned with a third individual $C$. . . . . 84

# Chapter 1

# Introduction

In this thesis we propose a novel framework with the intent of improve capability of *Genetic Programming* (GP) to solve complex real life problem. GP is framed within the broader family of *evolutionary algorithms* (EAs) [3]. EAs are inspired by Darwin's theory of evolution in its various aspects and, specifically for GP, on the iterative process based on *reproduction, mutation, competition and selection*. Out of the natural evolution metaphor real GP systems implements the cycle in figure 2.1. To complete the various phases described there many specific algorithm exist. These algorithms manipulate structures representing a population of solutions. Contrary to other machine learning techniques GP is able to produce as output model or algorithm that are fully understandable by humans and are not a mere black box tool with obvious advantages in term of knowledge. The work introduced here develops techniques that are applied to the symbolic regression domain, but their generality, and the possibility of broader application, will be clear to the reader, and potentially not limited to the EAs machine learning field.

## 1.1 Semantic genetic programming and open problems

This thesis is focused on a promising trend recently established in *Genetic Programming* (GP), the *Semantic GP*. In this framework the individuals forming the population undergoing selection-evolution cycles are analysed from a point of view centred on their behaviour more than on the solutions' structure itself. Indeed we are interested in an algorithm or model solving our problems, and obviously many formulation of the same solution are possible, and in practice this redundancy is a real issue [4]. If we are concerned in enhancing the efficiency of the exploration of the solution space we are not

interested in duplicates behaviours or, in other words we are not interested in algorithms that are computing the same solutions or that have the same semantic.

Many criterion have been adopted to measure diversity and usually evolutionary systems that preserve this population's characteristic show increments in their performances, see for examples [5],[6] and [7].

The new semantic perspective have introduced new metrics aiming at measuring diversity. The semantic itself is defined as in [8] and [9] where it is the vector whose elements are obtained evaluating every fitness case in the training set. If we call such set $\mathbf{X} = \{\overrightarrow{x_1}, \overrightarrow{x_2}, ..., \overrightarrow{x_n}\}$, in a symbolic regression problem, an GP individual is a function or program $P$ that, for each vector $\overrightarrow{x_i}$ in $\mathbf{X}$ returns the scalar value $P(\overrightarrow{x_i})$. We define *semantics* of $P$ the vector $\overrightarrow{sP} = [P(\overrightarrow{x_1}), P(\overrightarrow{x_2}), ..., P(\overrightarrow{x_n})]$. This vector can be represented as a point in an $n$-dimensional metric space, that is generally called *semantic space* where the well known properties characterizing a metric space hold thus having defined a distance $d$ with the usual properties:

$$\begin{aligned} d(s_{P_i}, s_{P_j}) &\geq 0 \\ d(s_{P_i}, s_{P_j}) = 0 &\Leftrightarrow s_{P_i} = s_{P_j} \\ d(s_{P_i}, s_{P_j}) &= d(s_{P_j}, s_{P_i}) \\ d(s_{P_i}, s_{P_j}) + d(s_{P_j}, s_{P_k}) &\geq d(s_{P_i}, s_{P_k}) \end{aligned} \tag{1.1}$$

Differently from the *syntactic* or *genotypic space*, where individuals are represented by programs syntax, the semantic space give a numerical representation that is strictly related to the program behaviours during the test. In this scenario the syntax of the selected representations doesn't matter any more and the genetic operators designed in this space are almost independent from such phenotipic details. New similarity measure have been defined based on the metrics imposed on the semantic vector space, see equations 1.1. Semantics methods have been used in three different phases of the evolutionary process trying to improve diversity and therefore exploration and performances. Many approaches try to perform a semantic analysis during *initialization* phase like for example [10] and [11] here the attempt is to have an high variety of phenotypes-genotypes already at the beginning of the evolutionary process. Other approaches also work during the *selection* phase discarding too similar semantics, see [12] and [13], or trying to match individuals with different semantics [14] always with the aim of obtaining more diversity in the population. Another group of approaches tries to introduce semantic in the *genetic operators*. These techniques establish the semantic properties of the sub-trees from two individuals trying to figure out which is the best location to

perform crossover. Many proposal have been formulated targeting similar semantics to promote gradual changes, e.g. in [9], or at the opposite targeting different semantics to obtain more diversity,e.g. in [15] or also intermediate approaches like in [16].

In this thesis we are interested in some of the major issues still open in GP, indeed we believe that our proposal is positively affecting *bloat* and *generalization* problems. We observe bloat when the growth of the solution complexity is non-functional to performance improvement [17]. This may arise from many causes and we present the most relevant theories in section 2.2 but then we focus on the idea that neutral mutations have a main role both in the code growth and the generalization issue. We believe that also this last problem is related to the size of the solutions and their complexity: it is possible that the program incorporates data from the training set rather than learning a theory of general validity. In accordance with the principle of *minimum length of the assumptions*, see eg [18] , there is general consensus that the assumptions that require less information to be encoded are also the best ones. It is believed that the simplest solutions are able to generalize better and resist more effectively to noise.

We highlight important relationships among *bloat* and *overfitting* problems and two GP frameworks that inspired us and are the foundations of what we introduce here. In the fields of semantic GP there is a *Geometric* framework, a GP branch called *Geometric Semantic Genetic Programming* (GSGP) that has really interesting properties [19]. Using the same kind of semantic space and metrics we have seen above (see equations 1.1) geometric techniques introduce genetic operators that, by construction, induce a unimodal fitness landscape, in other words there is a perfect correspondence between fitness and distance from the target because new individual can be constructed deterministically to stay in a known position of the semantic space. GSGP techniques have a good generalization properties even if, in the canonical form described in [19], they are subjected to an exponential growth of the complexity of the solutions. We discuss how geometric properties are preserving generalization despite the complexity of the solutions. From the GSGP framework we take the ability of build solutions in a deterministic way but, contrary to GSGP, we are not combining solutions with geometric operators and we do not rely on geometric properties to guarantee optimal generalization against overfitting.

There is another framework from our previous work that has an important role in the present thesis and is based on the alignment of individuals in the populations and the target in the semantic space: the *Error Space Alignments Genetic Programming* (ESAGP) [2]. The reader can find a complete description of this approach in appendix A. One of the interesting point of ESAGP is that maintain semantic diversity at the population level, no duplicate is admitted, even considering the past population. A key feature of ESAGP is that to discriminate among semantics it is not using a usual metric

to compute a distance but rather shows the utility of using a pseudo-distance approach in particular ESAGP relay on *Semantic Angles*. In this thesis we are building on the following characteristics of ESAGP: (i)it uses geometrical properties to reconstruct the target(ii) it shows a good generalization ability thanks to its property of filtering out duplicate semantics and thus keeping solution complexity low (Occam razor principle strongly enforced) (iii) it uses pseudo-distance to compare semantics thus enabling an equivalence class approach: all the semantic vectors aligned in the error space are equivalent from the point of view of target approximation with a geometric approach because they al stay in te same vector subspace. We want to generalize ESAGP's properties (i,ii,iii) keeping in consideration the geometric semantic approach of GSGP and thus we propose a novel flexible framework based on *Equivalence Classes*.

## 1.2 Equivalence Classes in Genetic Programming

The work presented here is strongly related to contributions discussed in the previous sections and it directly incorporates semantic awareness in GP. Nevertheless, it does so from a different perspective. In this thesis, we propose a novel idea to exploit semantic awareness in GP: *semantic based equivalence classes* (SECGP). The context of SECGP is still the semantic genetic programming and the semantic vector space that we are using is the same one introduced in the previous section. 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, finding one solution in the same equivalence class as a globally optimal solution allows us to solve the problem by reconstructing the global optimum analytically. Using direct semantic manipulations, similarly to GSGP, we compare entire classes of individuals. Instead of the operators proposed in [20], we use traditional genetic operators to build new solutions. SECGP come naturally equipped with the ability of discriminating class of semantics and this further enhance the type of redundancy that can be avoided, not just a single point in the semantic space but rather entire classes. Any single individual is representative of a class and using the equivalence relationship it's easy to compute any other individual in the same class, this imply that we have a new *pseudometric* that has the same advantages discussed for ESAGP semantic angles, but is also a generalization of that concept, indeed *semantic angles* are a particular implementation of an SECGP system. Thanks to its enhanced capacity of discrimination among semantics we expect that a SECGP systems have the ability to maintain an higher level of diversity in the population thus exploring more effectively the solution space and at a faster rate than GSGP.

To enhance genralization ability and reduce overfitting SECGP adopt the strategy of reducing as much as possible the code growth that it's not related to any real change in fitness. Neutral changes in the solution's structures are completely forbidden and, in the proposed Filtered version F-SECGP, any duplicate or similar semantic is discarded on evaluation. This behaviour strongly enforce the *Occam Razor Principle*, any changes in the semantic can be immediately evaluated and complexity it is introduced gradually, positively affecting also the overfitting problem.

## 1.3 Contributions overview

The approach presented in this thesis proposes these original contributions from the theoretical point of view. (i) We show how to generalize the use of pseudo-distance and its usefulness for the comparison of semantic vectors. This added flexibility allows also to craft or learn useful equivalence relationships. (ii) We define a new formal concept of duplicate semantics based on equivalence classes.(iii) We shows a new effective method of filtering duplicate semantics (iv) thus permitting only a very controlled growth of the solution's structural complexity and a consequent good generalization ability. (v) We simplify the geometric approach in the context of equivalence classes, losing the guaranteed geometric properties (unimodal fitness landscape and good generalization ability) but getting a potentially compact and equally general and accurate solutions.

From a more practical perspective we define four simple and completely new techniques: two new SECGP, called *GPPLUS* and *GPMUL* and tow F-SECGP called *FGPPLUS* and *FGPMUL*. Moreover we reformulate the well known linear scaling technique in the new perspective of SECGP framework.

Experiments to test the performance of the systems proposed have been conducted on seven complex real-life applications. The results obtained can be summarized as follows: when considering the results on unseen test data, on five out of the seven studied test problems, the systems proposed are better than (or comparable to) the state-of-the-art of GP for symbolic regression (i.e. *geometric semantic GP*). Also, on all the test problems taken into account, filters are beneficial for improving the performance of the systems and show the capability to evolve models with a good generalization ability. Last but not least, the use of filters allows the studied systems to generate individuals that are significantly smaller, compared to their unfiltered counterparts.

## 1.4 Structure of this dissertation

We begin, in chapter 2, introducing Genetic Programming and the general connected background including the theoretical basis necessary to understand the problems that we are tackling in this thesis. This chapter also explains the principles of *semantic genetic programming*.

In chapter 3 we explore the recent literature that inspired our work highlighting the relationship of each described technique with both open problems and our proposal. We discuss also new interesting perspective on our past work.

Chapter 4 more formally introduces *equivalence classes* in the context of GP and explicitly formulate simple example systems developing the simple necessary mathematical tools.

Four of this new systems, a revisited Linear Scaling technique along with unchanged linear scaling and GSGP systems are the basis for the large experimental phase explained in chapter 5 on seven complex real-life applications. In the same chapter the readers find also the commented results.

In chapter 6 we briefly present our conclusions and some consideration on further research directions.

In appendix A the reader has a useful recall the core concepts of the ESAGP techniques that can be considered, as extensively explained in chapter 3 an ancestor of the *equivalence classes* in semantic GP. This appendix it's an aid for the comprehension of chapter 3.

# Chapter 2

# Introduction to genetic programming

## 2.1 Genetic Programming

✎ l'header qui dice "list of figures" E' un bug noto di questo templaet della tesi .. dopo chiedo a Korenzo come lo ha risolto ...

In this section the *Genetic Programming* (GP) [21] techniques are introduced and their main flavours and character presented.

The ideas at the heart of GP have been clearly connected to "machine intelligence" very early by one the father of *Artificial Intelligence* (AI). Alan Turing was first mentioning the possibility of using the concept of *natural selection* and *evolution* to build some sort of intelligence in his seminal work titled "Computing machinery and intelligence" already in 1950 [22]. This first formulation discusse the main steps, inspired from Charles Darwin theory of evolution, that are common to all *evolutionary algorithms* EA not excluding GP. Modern EA have been progressed until the point that we find documented in literature numerous successful applications to real world problems. Many of these have been defined by the scientific community "human competitive" [23], meaning that the quality and originality of solutions synthesized by this class of algorithms are very high and comparable to solutions proposed by humans. We easily find numerous lists of practical applications embracing a wide variety of fields [1] where EAs have obtained

[1] e.g. citation from [23]: "quantum computing circuits, analog electrical circuits, antennas, mechanical systems, controllers, game playing, finite algebras, photonic systems, image recognition, optical lens systems, mathematical algorithms, cellular automata rules, bioinformatics, sorting networks, robotics, assembly code generation, software repair, scheduling, communication protocols, symbolic regression, reverse engineering, and empirical model discovery"

important results, this indicate that EAs methods are really general and thus can be classified as full flagged AI methods.

### 2.1.1 Introduction to Genetic Programming

The Genetic Programming is framed within the broader family of *evolutionary algorithms* (EAs) [3]. EAs are inspired by Darwin's theory of evolution in its various aspects and, specifically for GP, on the iterative process based on *reproduction, mutation, competition and selection*. In general, the EAs are methods of stochastic optimization that aim to provide an approximation of the optimal solution where analytical methods and deterministic algorithms are not applicable due to the complexity of the problem. The process is iterated, starting from a population of candidate solutions, generated by a random process, until an optimal solution is found or an alternative termination condition, provided by the user, is reached. The initial population of solutions is evolved by means of operators that give rise to subsequent generations of possible solutions, the best individuals or the luckiest survive and produce offspring. Older individuals are replaced by new ones in whole or in part.

Various EAs differ in the used operators and algorithms: in this work we present the results and methods related to GP. It will be clear reading that, despite this first attempt, the proposals may have wider application to the whole EAs family. One of the main features of the GP is to output, as a result of the evolution, a real algorithm. From sets of symbols specified, GP produces software that is also meaningful for humans. Compared with other methods there is the advantage of having solutions that can be fully understood and not simply used as a black-box: this also allows to use these models as input in further optimization processes also directly by humans. Indeed in a black-box approach, like for example the Artificial Neural Networks, even if we can use the model to predict results the structure of the model itself it's not understandable by humans, and the real meaning of the model is lost.

As for the other AE is not necessary to provide a trace of the solution of the problem a priori, indeed GP is a able to evolve programs improving results from generation to generation without requiring any prior knowledge on the system that is under optimization. [24]

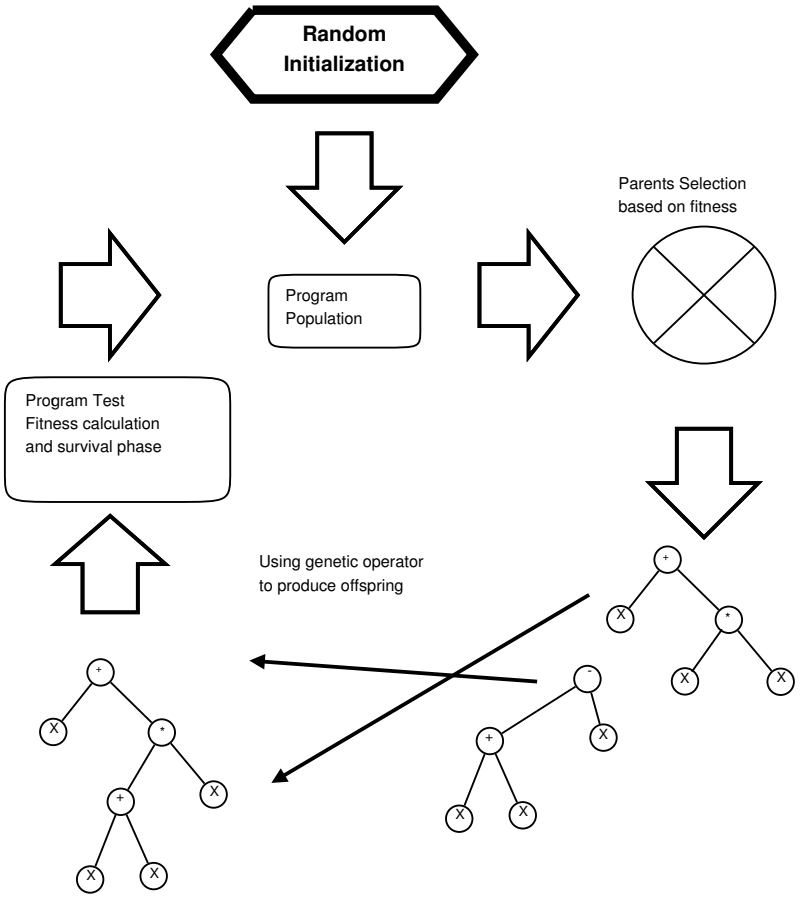



FIGURE 2.1: Genetic Programming general cycle schema. The big arrows indicate transitions to the next phase of the cycle starting at the top of the figure.

### 2.1.2 Main features

GP was designed to overcome some limitations of the *Genetic Algorithms* (for the GA, see [25] ). In particular, where the GA provide a structure of the solution of predetermined length, the GP exceeds this limit proposing structures of variable length and shape and increasing the complexity of the solutions which are subjected to the process of adaptation [24]: shape and structure are thus emerging characteristics of the iterative process of optimization. In this way, GP explore promising individuals without excluding a priori any possibility as is the case for GA [4] that uses solutions with fixed length and structures. The structure types most commonly used in GP are trees, linear chromosome and graph [4], this work consider tree structures, the most popular and most studied, but many of the concepts are easily transferable to the other cases. The tree structures are briefly described in section 2.1.3.

Figure 2.1 shows the generic GP cycle that iteratively produces refined solutions. At the beginning of the trial there is an initialization phase that generates a new population of candidate solutions: there are several possible strategies in this phase, the most

common are illustrated in section 2.1.5. In the next step parents of the next generation are selected, those who have a better fitness are more likely to be chosen: the selection process is stochastic and any individual in the population has the opportunity to participate in the subsequent process of reproduction. The fitness of an individual represents its degree of adaptation to the environment, indeed we can say that the most suitable will be selected with a greater frequency than the others.

Once the parents have been selected, the offspring is produced and a new generation is born. For this purpose genetic operators are used. These are algorithms that, starting from one or more parents, give life to different individuals. Among the most used operators generally we find *mutation*, which from a single individual gives rise to a different version of the same and the *crossover*, which from different individuals, produce children, incorporating, by *inheritance*, genetic material from all parents. Not all individuals are subjected to these processes, the best one may be allowed to pass without modification to subsequent generations, and in this case the process it is named *reproduction*, when this privilege is granted systematically to the best individuals it is named *elitism*. Some details on the genetic operators is provided in section 2.1.7.

The last cycle's step in figure 2.1 is the evaluation of the candidate that are the offspring of the parents. As in other supervised learning techniques the solutions are evaluated against the test cases and the error is measured with respect to the desired output. A *fitness function* summarizes the behaviour of each individual on the test cases by assigning them a real value that represents the degree of desirability or adaptation to the environment. This function may possibly incorporate other metrics, such as the size of the solution or other aspects of the user intends to submit to the optimization process. The fitness function is described briefly in section 2.1.4.

After assessing the fitness, and before starting over the cycle, the population of newly created individuals may be subjected to a *survival* test. At this stage it is possible that some individuals are immediately discarded based on some criteria. These filters are another type of selection that is not usually connected directly to the performance of the individual under consideration but rather to other characteristics, often structural or morphological. In this work the survival of the individual is linked to their *semantics*. The number of individuals in the population is often set at the beginning of the experiment and does not change during its development, if the surviving individuals are not sufficient to arrive at such a number in place of the discarded offspring are used parents or alternatively it is possible to proceed to create more individuals. At this point the cycle of figure 2.1 start over beginning with the selection based on fitness. Every time this cycle is iterated a new generation of candidate solutions is produced and is likely they are better than the previous ones.

### 2.1.3 GP solutions structures : the trees

Among the structures of variable complexity and morphology that have been adopted for the genetic programming there are the *trees*.

Trees are made up of basic components, nodes and leaf, chosen respectively among two disjoint set of symbols *functional* and *terminal*. The sets of symbols chosen must have sufficient expressive power to represent solutions to the problem and, at the same time, should not be extended unnecessarily to avoid a huge search space.

#### 2.1.3.1 Functional symbols

In particular, the functional symbols 2.1 represent the operations that can be performed on the input. The input of a function can be a terminal symbol or the output from another function. The arithmetic and logical operators are often part of this set but also instructions (e.g., an IF..THEN..ELSE) may be included in $F$.

$$F = \{f_1, f_2, ..., f_n\} \tag{2.1}$$

The number of arguments of the functions represented by the symbols in $F$ (arity) is determined at the beginning and can be chosen from a minimum of one up to a maximum which usually does not exceed the dimensionality of the input of test cases in the dataset [4]. An important feature of the functions included in $F$ is to be able to manage in a consistent manner any of the possible inputs. The GP, as we have seen, constructs candidate solutions through a stochastic process that has the potential to generate any type of input for each of the functions in use. A significant example is that concerning the arithmetic division: it is important to take care of division by zero. If we don't desire to stop the evaluation of such an individual we can establish a conventional output, for example zero or a very large value. The choice of how to protect the operators, adopting a modified version, it is left to the user, these choices are part of the system design : this property is named *closure* of operators.

#### 2.1.3.2 Terminal symbols

The *terminal* symbols 2.2 represent the input, the constants supplied to the system and the functions of arity zero used for their side effects. Having always a null arity these types of symbols are necessarily used as leaves of the tree representation.

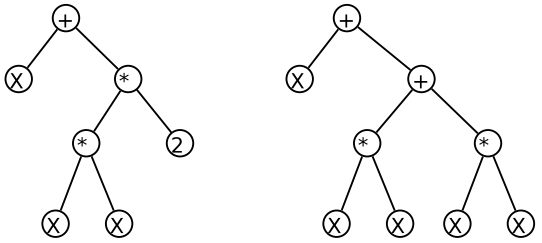



FIGURE 2.2: Expression 2.3 represented through two different parse trees.

$$T = \{t_1, t_2, ..., t_n\} \tag{2.2}$$

#### 2.1.3.3 Tree structures

The trees are the syntactic structure that represents the genome of each individual. Internal nodes are all from the functional symbols set while the leaves are all taken from the terminal symbols.

The parse tree that results is easy to understand and is flexible enough to solve regression problems such as those addressed in this work, however it is not a *turing-complete* approach, indeed it is not possible to have loops and recursions or explicitly address memory areas, see [26] .

To tackle this problem have been produced a number of languages that, considering appropriate functions and structures other than trees, are able to evolve complete programs with *Turing-complete* expressive power, such as the Push language and its evolutions [27]. Although in this thesis only the parse trees have been considered they are suitable for regression problems that are the centre of this work.

$$y = 2 * x^2 + x \tag{2.3}$$

In Figure2.2 the expression 2.3 is represented by two possible parse trees. The genetic operators are able to operate on the structure of the trees to produce new specimens from one or more parents. The choice of a particular type of structure also determines the possibility of reaching different "near" structures configurations. Structures and genetic operators determine the possible future evolutions but, due to their expressive power, structures are not unambiguously related to semantic, indeed two syntactically different individuals may have the same semantic. In other words, the *phenotype* can be expressed through different *genotypes* . The selection mechanisms described previously act taking into consideration the degree of adaptation of an individual, and therefore its phenotype, while the genetic operators operate to 'blind' on the structure of the

genome reproducing the same dualism that is observed in nature: this difference is often overlooked and is the origin of semantic techniques described in this work.

### 2.1.4 The fitness function.

The selection of individuals is made on the basis of their level of adaptation to the environment and this is the fundamental concept on which a GP system is built. The level of adaptation is evaluated by means of a *fitness function*. The fitness function is not necessarily related to the specific problem that we are trying to solve and in any case is the metric by which we measure the goodness of candidate solutions [28] . The fitness function should reflect proportionally the magnitude of the improvement that has taken place between an individual and the other in such a way that small improvements correspond to small changes in fitness and that great improvements correspond to large variations, in this case the fitness function is *continuous*. The paradigm of GP says that fittest individuals will be more likely to reproduce, for this reason having a fitness function designed meticulously is very important. Depending on the goals in the fitness function can also be incorporated assessments of the quality of the solution in a broader sense. For example, you can reward the compact size of the solutions (*parsimony pressure* ) or reward the novelty of a solution compared to the rest of the population (*fitness sharing* ) etc.

#### 2.1.4.1 Fitness function example in regression

This work deals mainly with *regression* problems, thus we show an example of fitness function suitable for this purpose.

In the supervised learning framework there are a number of example form which the system is learning, also called *fitness* cases. Writing this examples in the form of matrix 2.4 we have a generic line of inputs $x_{m1}, x_{m2}, \cdots, x_{mn}$ that produce as output the value $y_m$

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} & y_1 \\ x_{21} & x_{22} & \cdots & x_{2n} & y_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} & y_m \end{bmatrix} \tag{2.4}$$

In a regression task the problem is to find the function $g$ such that $\forall i = 1, 2, \cdots, m$ we have that $g(x_{i1}, x_{i2}, \cdots, x_{in}) = y_i$

Each program in the population is therefore a function $g$ giving as input the line $X$ and comparing the output result with the corresponding $y$.

In this case, a fitness function, $f$, may be the standard deviation in the equation 2.5.

$$f = \sqrt{\sum_{i=1}^{m}(y(x_{i1}, x_{i2}, .., x_{in},) - y_i)^2} \tag{2.5}$$

There is not a unique way to properly define a fitness function so for example, also the equation 2.6 is suitable.

$$f = \sum_{i=1}^{m}|(y(x_{i1}, x_{i2}, .., x_{in},) - y_i)^2| \tag{2.6}$$

Both functions are *continue* in the sense described above and represent the quality of the solution found in a coherent manner.

### 2.1.5 Initialization

As can be seen from Figure 2.1 the first step in a GP system is to initialize the population. There are various techniques normally used for this purpose but almost all are based on the creation of random individuals.

The construction of the trees is done by randomly pulling sets $F$ and $T$ (equations 2.1 and 2.2), the extracted functional and terminals symbols are then used to build new trees. Before proceeding we define two parameters that are widely used to characterize the shape of the trees. We define as *depth* or *level* of a node the number of nodes that must be crossed to reach him from the root of the tree. One of the most important parameters is precisely the maximum allowable depth in the tree. Another simple parameter is the maximum number of nodes that can be added to the tree. By modulating these variables it is possible to determine the shape of trees that are created, with forms ranging from the bush to the more elongated and threadlike.

If we consider individuals in the population as representing points in the solution space the hope is that different shapes and size of the trees are able to sample large areas, or at least heterogeneous areas, of this space to enhance te probability of achieving a good solution. It is assumed, therefore, that a greater variety of the genome is a desirable feature of the general population and even more for the initial population.

For this reason, when the population is initialized, individuals who are syntactically identical are generally removed to leave room for different genetic material [29] : in fact, with small trees or a limited number of symbols, the probability that two random trees are in fact the same is not remote.

#### 2.1.5.1 Initialization methods: *grow* and *full*

The initialization technique called *grow* starts assigning a functional symbol extracted at random to the root of the tree and proceeds, extracting indifferently terminal symbols and functional ones, until the predetermined maximum depth is reached.

When is extracted a terminal symbol the growth of the branch stops and the process continue with another branch until exhaustion. Once the maximum depth is reached a terminal symbol is extracted in order to be sure to not cross that limit. The shape of these trees is irregular indeed the branches often have very different depth. The method *full* uses only functional symbols until the maximum level is reached and then terminate the branch with a terminal symbol. The trees constructed in this way have all branches of the same depth and are therefore more regular and with greater number of nodes.

#### 2.1.5.2 Ramped Half-and-Half Method

Among the various methods of initialization one of the most used is *Ramped Half-and-Half*. Once the maximum depth has been established, the population is divided equally among the lengths between 1 and MAXD[2].

### 2.1.6 Selection

After passing the evaluation through fitness function each program has its own assessment. It is in the selection process that the most deserving individuals are more likely to reproduce.

For this reason there are several algorithms that assign to every individual of the population the likelihood of becoming a parent and thus have the ability to transmit their genes to the next generation.

The used selection technique is important because it determines the speed with which the population converges towards a solution. If the *selection pressure* is too high you may experience the phenomenon of *premature convergence*. Excessive pressure makes

[2]MAXD is only a conventional name used in this work not a universal denomination.

only the genome relatively more suitable to have the chance to reproduce, in a few generations the other genomes are lost with the result that the opportunity to improve performance through a sexual reproduction are very scarce and the progress of optimization is stopped prematurely. On the other hand a selection pressure too bland does not allow the stabilization of the best solutions and the evolution proceeds too slowly or stops altogether [28]. Among the main selection algorithms are *Fitness Proportional Selection*, *Ranking Selection*, *Tournament Selection*. Tournament Selection is described in some detail in the next section because served as the standard starting point for the experiments presented in this work.

#### 2.1.6.1 Tournament selection

In tournament selection competition for reproduction is not extended to the entire population, but is contained within a subset. Among the participants in the tournament the winners are those with the best fitness. These individuals are allowed to reproduce by replacing the losers. The number of individuals participating in each tournament is called *size* and the tournaments are repeated with participants drawn at random from the population [3] until a sufficient number of parents to generate the programmed progeny is reached. The minimum size of the tournament is obviously two adjusting this parameter it is possible to increase or decrease the selection pressure: indeed the larger the size the less is the probability that scarce individuals can become parents.

### 2.1.7 Genetic operators

By means of genetic operators individuals in the population are transformed into others thus exploring the solution space. There are many genetic operators variants but they can be roughly classified into three main families: *reproduction*, *crossover*, *mutation*. The reproduction is the trivial case in which the individuals selected are copied without changes in the next generation. The other two types are described in the following sections.

#### 2.1.7.1 Crossover

The *crossover* genetic operators type construct a new individual by exchanging genetic material of the parents. Generally, two sub-trees are chosen, one for each parent, and then they are exchanged, see figure 2.3. The choice of the exchanged sub-trees is a peculiar characteristic of each different crossover algorithm. In the more traditional

[3] The individuals from the population are then placed back with each new draw.

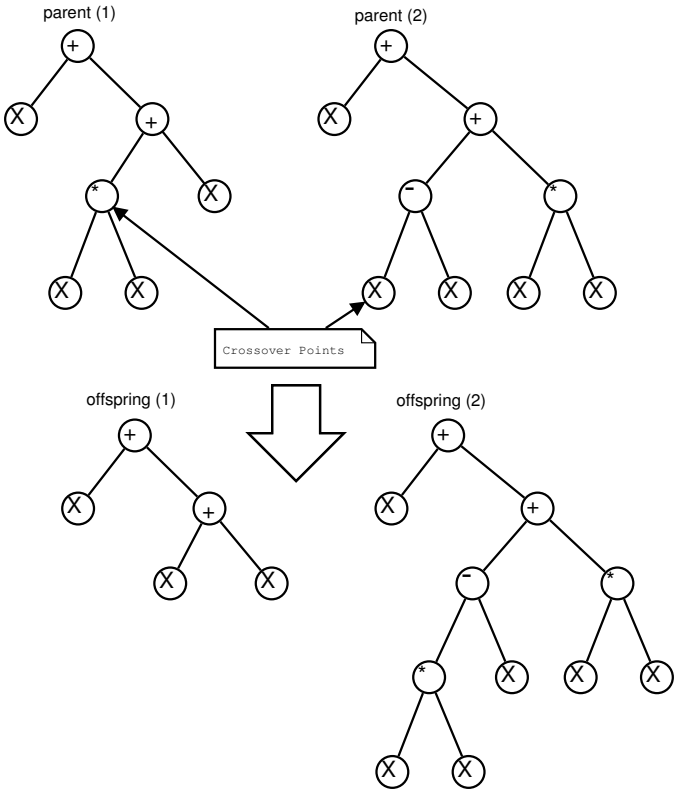



FIGURE 2.3: The crossover genetic operator exchange sub-trees between the parents at the crossover points.

procedure sub-trees are simply chosen at random. In this last case, since the number of leaves is larger than the number of internal nodes of the tree, you may choose to alter the random process in order to facilitate the extraction of an internal nodes thus having more chance of radical changes in the trees structures.

#### 2.1.7.2 Mutation

The mutation operator introduces completely new genetic material into the population, this material is not coming from other individuals. Even in this case there are several algorithms like for example *sub-tree mutation*. This operator will replace a node of the tree chosen at random with a sub-tree generated using one of the methods discussed in the section 2.1.5 about initialization. Usually the maximum length of the new sub-tree is supplied as a parameter. Other mutation operators examples are *swap mutation* that swaps two randomly chosen sub-trees and *point mutation* that replaces a single node of the tree with a symbol of the same arity.

## 2.2 Introducing Genetic Programming Open Problems

In this section we give an overview of the issues that arise in genetic programming still representing a challenge both from a theoretical and practical point if view. The phenomenons illustrated here are typical of GP, e.g. the *bloat* issue, but also more general, like the *diversity* problem affecting all the Evolutionary Algorithms, or the *overfitting*, this last one being a problem common to machine learning techniques in general.

The themes introduced in this section are strictly related to the experiments' results presented in this work indeed we are tackling also these important issues. Starting from a theoretical framework we arrive, as expected, to results that influence positively both *bloat* and *overfitting* problems giving some hints on the importance of *diversity* in GP populations. In this section we just introduce definitions within a general overview while we deepen the analysis of our contribution in chapter 3.

### 2.2.1 Bloat

The *bloat* phenomenon is challenging and also far from a complete comprehension: it is still an open problem. We introduce this issue with specific reference to the trees solutions' structure introduced in section 2.1.3 but the reader should consider that bloat happens also with other structures. At some point in the evolution of the GP system the number of nodes of the trees may begin to grow in a manner no more proportional to the improvement of fitness: it is at this stage that *bloat* appears. The trees normally start from a size determined a priori by the user which may be too small to give a satisfactory solution, so some growth of the trees is natural and desirable, as we pointed out in section 2.1.1, indeed it is just to allow the growth of the complexity of the solutions that GP paradigm has been developed.

Excessive growth of the trees, however, poses several problems. From a computational point of view the task quickly becomes intractable, in fact the amount of computing resources consumed for the evaluation of each individual is likely to become excessive. But there is also another problem, the intelligibility of the solutions by a human user: indeed one of the advantages of GP over other *machine learning* techniques [4] is to provide solutions that are not *black boxes* but fully analysable models where relationships between the variables are explained clearly by evolved algorithms or equations.

By increasing the size of the program, however, the solutions' semantics becomes proportionally more difficult to understand until it is lost completely for the human user

[4] See, for example, the models provided by the neural networks that are difficult to interpret.

when it gets too large. In this manner one of the great advantages that the GP is able to offer is lacking.

Although there are several methods to reduce or simplify the dimension of the solutions, especially for logic or symbolic regression domains, this approach is not always effective or it is not possible at all in other domains. We observe bloat when the growth of the solution complexity is non-functional to performance improvement [17] .

We now briefly review some of the theories found in literature addressing this problem introducing the key concepts.

In a first hypothesis the bloat can be attributed to the emergence and proliferation of code in the trees that remains neutral, i.e. it does not contribute in any way to determine the fitness. Knots or whole branches that, if removed from the tree, lead to no change in the performance of the algorithm. Such code segments in the tree are referred to as *introns* by analogy with the biological system based on DNA where the non-coding sequences of nucleic acids are denominated precisely introns. These sequences may be inactive on test cases that are being used but may become active on different inputs resulting in the degradation of the performance of generalization of the model found.[30] .

In [31] we find a schematic characterization of the Code of introns divided into categories that are more or less neutral in their behaviour with respect to the input or the crossover operator (as defined in section 2.1.7):

- Segments of code where the intervention of the crossover does not change the program behaviour for any input belonging to the *problem domain*
- Segments of code where the intervention of the crossover does not change the program behaviour for any of the *fitness cases* in use.
- segments of code that can not contribute to fitness, and where each of the nodes can be replaced with a NOP [5] without affecting the output given an input in the *problem domain* .
- code segments that do not contribute to the fitness and where each of the nodes can be replaced with a NOP without affecting the output given as the input the actual *fitness cases*.

Among the first explanations attempt there is the theory of *protection against destructive crossover* [32]. Since the genetic operators act blindly with respect to the fitness there

[5] No-Operation

is an high probability that a block of working code is destroyed after a crossover affected that block.[31] Introns would thus be an emerging phenomenon to protect code fragments of high value from destruction. If a crossover takes place in one of the code fragments belonging to an intron semantics of the surviving structures is preserved without damage. In this sense, the introns increases the probability that the crossover is able to capture sub-expressions without damaging the useful parts.

The *removal bias theory* [33] observe that the inactive code is mainly located close to the leaves of the tree. In this way if a crossover occurs in that area the average length of code segments removed will be lower than the replacements generating longer trees . Moreover, given that the crossover occurred in an area of inactive code is more likely that the fitness of the parent is transmitted to the offspring. These facts therefore benefit the trees longer that have been created in this way, indeed those who have suffered a crossover in the areas closest to the root have generally suffered greater damage and although they are on average shorter will also have a worse fitness. This ultimately would explain the constant proliferation of trees increasingly longer.

Another theory of interest is the one that ascribe to the *nature of the solutions search space* the bloat [34]. It starts from the prediction that above a certain individuals size fitness distribution does not correlate any more with dimensions. This means that there are many more versions of 'long' programs that have a certain fitness, and therefore the probability of sampling a long program is higher than sampling a shorter one. Given this premise, with time, the length of the programs sampled progressively increases simply because they are more numerous.

The latest theory we consider here is the *crossover bias theory* [35]. It is shown in [36] that the limit distribution of the lengths of the programs affected only by crossover [6] approaches that of Lagrange distribution of the second order. In this distribution, the small programs are much more frequent than large ones. If we apply the selection based on fitness at this point the small programs will be systematically discarded because too small trees have a reduced fitness, and larger trees will be systematically favoured. Iterating this process the average size of the selected programs increases generating the phenomenon of bloat.

### 2.2.2 Overfitting

A general definition of *overfitting* valid for the machine learning techniques in general is the following, from [37]:

[6] standard sub-tree crossover with uniform selection of the points of application

"Given a space of hypotheses $H$ a hypothesis $h$ $inH$ is *overfitting* to *trainign data* if there is any other alternative hypothesis $h'$ $inH$, such that $h$ has an error smaller than $h'$ on test cases but that $h'$ has in turn an error smaller than $h$ on the whole distribution of cases" [37].

The overfitting phenomenon has often been associated with the bloat explained in section 2.2.1. The size of the solutions grows and their complexity increases: in this way it is possible that the program incorporates data from the training set rather than learning a theory of general validity. In accordance with the principle of *minimum length of the assumptions*, see e.g. [18] , there is general consensus that the assumptions that require less information to be encoded are also the best ones. It is believed that the simplest solutions are able to generalize better and resist more effectively to noise. However, recently it has been shown that the overfitting phenomenon can occur independently from the bloat. With the new technique of *operator equalization* and its variants in [38] is shown that, although the bloat is effectively controlled in the proposed experiments, overfitting still occurs.

### 2.2.3 Diversity

Diversity of genotype and phenotype is a key concept of evolutionary algorithms and the GP is not an exception. The evolutionary process is often characterized by a loss of diversity with the progress of generations. It happens often that the solution is the result of the mixing of the genes of an elite, even very restricted, coming from progenitors at the generation zero [39].

This genetic poverty may prevent the finding of solutions really different from the few already present in the population, and thus to escape from a fitness landscape local optimum. The loss of genetic diversity in a population trapped in a local optimum is called the *premature convergence*. The phenomenon of premature convergence characterizes also other evolutionary algorithms, but it was thought that the variable-length structure of the GP solutions was sufficient to ensure adequate variability: the problem still persists. The variable-length structure, like that of a tree, also introduces an additional level of complexity that resides in the broken relationship between syntax and semantics. In fact, the link between syntactic diversity and semantic differences in GP is typically more complex and can be summed up in the fact that structures also significantly different in reality may have the same semantics [11] .

Two different approaches have been developed to tackle the diversity problem: the first is the older one and look for syntactic diversity neglecting the semantic of the programs

itself that, instead, is the core of the second approach. In [40] a review of the most used methods is found.

Recently, moving forward from the literature presented here an entire new class of semantic approaches have been developed, these works are strictly related to this thesis and will be presented in chapter 3.

#### 2.2.3.1 Genotype Diversity

Realizing the importance that the diversity of the population could play in the evolutionary process Koza ([29] ) himself defined the concept of *variety* as the percentage of individuals who do not have an exact duplicate somewhere else in the population. To measure *structural similarity* have been developed alternative strategies.

Many types of *edit-distance* have been defined: the use of this type of distance is inherited from GA where a typical example of this measure is the *Hamming distance*. In GP other types of distances have been established, measuring how many primitives needed for transforming a tree to another. Given a table with the cost of the primitive the procedure continues by calculating the cost of this transformation, and then the relative distance between two trees. These measures are necessarily more complex than those used on fixed length GA linear structures.

Among the various definitions there is the Levenshtein distance, and others that were defined by aligning the common parts of trees and by scoring the differences: in all cases a similarity value is computed on the basis of the differences between individual nodes [41] [42],[43].

There are also measures based on sub-trees, *subtree-distance*, where the difference between two individuals is given by the number of common sub-trees among them [44].

Finally in the aforementioned [39] the genome that is actively contributing to the population is tracked down to the original individuals allowing the calculation of the population diversity. The more individuals contributed to that genome the more diversity score increases.

#### 2.2.3.2 Phenotype diversity: investigating semantic.

More recently new approaches in literature give more importance to the difference in the behaviour of candidate solutions rather than mere structural one. As has been mentioned before in section 2.1.3, the introduction of variable-length structures, such as

trees, has led to the possibility that structurally different individuals actually represent the same behaviour, see for a simple example Figure 2.2.

A new perspective on diversity is therefore to measure the differences in the programs behaviour to maximize the number of strategies that the GP is browsing before converging towards the optimal one. Having in the population, for the longest period possible, a large number of different strategies obviously maximizes the probability to explore the best strategy and not get stuck in a local maximum. Applying this practice means to characterize and compare the *phenotypes* present in the population through the investigation and representation of *semantics* of programs represented by trees.

In [11], is shown that a population structurally different, which does not admit duplicates from the structural point of view, in reality does not guarantee the variety of observable behaviours and strategies. Special techniques have been developed specifically to address this problem. The work presented in this thesis starts from this need.

### 2.2.4 Semantic in Genetic Programming

A recent trend in GP is to characterize individuals not only using the structure of the tree that encodes them, but also looking at shown behaviour. What is desirable, in the end, is the variety of exhibited behaviours . As we saw in section 2.2.3, ensuring the structural diversity does not ensure differences in behaviour. Various structures may in fact encode for the same behaviour, this is a rather common eventuality [4].

We may notice also that attempts to expand the range of behaviours by increasing the trees size turns out to be very inefficient, see [45], this once again confirms the need for specific strategies that don't relay on genotype evaluation.

Although the GP allows multiple mappings between genotype and phenotype, the selection, normally operated in the GP, will reward a specific pair phenotype-genotype significantly depleting the available genetic potential. A single genotype-phenotype couple will eventually prevail over all others in a way too fast, see also [39]. Usually performance with better fitness are associated to experiments preserving a high diversity in phenotypes and fitness entropy (see equation 2.7)[5].

The evaluation of the phenotype must necessarily be based on the functionality of the individuals rather than on their structure, a static evaluation it's not enough to capture the behaviours. For this reason fitness homogeneity is used to determine phenotype variability the in the population. A more uniform distribution of the fitness in a certain range would also indicate a uniform distribution of phenotypes. [6]

Considering this principle the population diversity is also measured by calculating the fitness entropy. In the context of GP this entropy represents the degree of disorder of the population and has been defined in [7] as:

$$E(P) = -\sum_{k} p_k * \log p_k \tag{2.7}$$

In this case the whole population has been divided among $k$ classes based on fitness value and $p_k$ is the proportion of individuals belonging to class $k$ .

Such measures, however, is not particularly precise because individuals with similar fitness can actually reach that same result by adopting different strategies, and on the contrary, have two fitness rather distant even if their strategy is quite similar: for this reason using directly the fitness to measure the distance between two individuals is not very accurate.

More recent attempts try to obtain a more precise semantic measure. Trying to establish the semantics of an individual without access to its fitness function, there was a return to a type of structural analysis: an attempt was made to find a normal form the irreducible structure of individuals so that the genotype could be linked with his phenotype without ambiguities. Finding the normal forms for programs strictly depends on the considered application domain, thus at a certain extent, universal validity of this method it is compromised. Attempts of this kind are for example those in [13],[10]. In this case, the domain of Boolean functions is considered and the canonical representation used is called *ROBDD* (reduced ordered binary decision diagram). Once an individual has been reduced to its equivalent diagram is easy to check if there is a correspondence to other individuals in the population and so dealing with two syntactically different representation of the same semantics. Again the aim is to preserve the diversity within the population during the evolution and in this case it is used a modified version of crossover (SDC) that discard the generated offspring if it is semantically equivalent to the parents. This approach prevents returning to explore already sampled areas of the solutions space , or more precisely prevents to propose a solution already exploited.

Another study in the boolean domain exploits the semantics to investigate the nature of the sub-trees in the population [8]. Even in this case fitness is not accessed to determine the semantics but rather the procedure builds the truth table corresponding to the Boolean function in the sub-tree. This study determined the existence of a large proportion of fixed sub-trees that, if addressed in the crossover, can not generate the target function because their output is completely or in part independent of the input. This lead to conclude that a large proportion of the crossover is unproductive from the point

of view of semantic solution space exploration , in agreement with the results already shown in [13].

An interesting development in the measurement of semantics is among the most recent, see eg [9], and associate the semantics of the individuals with the resulting vector from the evaluations of the fitness cases in the training set.

The study of semantics has been applied to various stages of GP aiming to promote diversity in order to increase the probability of finding the optimal solution. Despite the common purpose, the techniques used are very different from each other also in relationship to the application domains.

Semantic analysis has been applied to different point of the evolutionary process, below is given a brief review ordered by application phases: *initialization*, *selection* and during *genetic operators* application. We illustrate the general research directions and some detail that is significant to this thesis.

#### 2.2.4.1 Semantic analysis during initialization

As already mentioned the purpose of semantics in the initialization is to provide a starting population with the widest possible behaviour variability. For example in [10] are considered the initializations in two domains, Boolean one and Artificial Ant. The employed methods include the use of *ROBDD* for the Boolean domain, while for the Artificial Ant was considered the moves sequence oriented on the grid as semantic indicator. During initialization the new individuals are constructed using growth algorithm starting from a nucleus trying to avoid behaviour already present in the population, [7]: it is not allowed to add behaviour identical to the population. It has been shown that the distribution of the programs in the search space has a noticeable impact on the performance of the algorithm, although the influence is positive or negative depending on the test problem considered. A similar attempt to improve algorithm performances improving the initialization phase has been implemented in [11], in this case it was considered the domain of regression as well as navigation (artificial ant and maze navigation). The output of programs was used like semantic indicator, thus having a more general approach than that used ROBDD. Also in this case the behaviour duplicates were not allowed. The results are encouraging and show an increase in the fitness of the evolutionary process for the initialized semantically populations.

[7] a variant in the same work uses instead the FULL method to generate the individuals of a certain complexity

#### 2.2.4.2 Semantic in selection

During the evolution phase, at selection time, a similar idea is to filter out or not admit to the next stage breeding that show a similar semantics to individuals already present in population, see for example [12] and [13] .

The idea of selecting individuals for mating relaying on semantics is less present in the literature. One example is [14] that shows how to increase the genetic variability by preventing the coupling of individuals with the same fitness, thus having beneficial results, other example are more related to *Geometric Semantic Genetic Programming* and are introduced in chapter 3, where we discuss methods that directly manipulate semantics during crossover phases.

#### 2.2.4.3 Semantic genetic operators.

Attempts to use semantic to improve the performance of the genetic operators are the more numerous. Usually these methods establish the semantic properties of the sub-trees from two individuals trying to figure out which is the best location to perform crossover.

In particular,the semantic values of the sub-trees designated for crossover are calculated using techniques similar to those described previously, based on these values the crossover proceed or a new attempt is made.

Among the proposal the ones exchanging sub-trees with similar semantics to maintain a certain operator locality having a more gradual dynamic evolution, see e.g. [9]. Opposite to this there is the proposal to proceed only in case the sub-trees are semantically different to promote a more extensive semantic search, the results of these opposite tendencies are compared in [15]. There is also the intermediate proposal that selects sub-trees semantically different but nonetheless within a certain threshold, which according to the comparisons presented in [16], turns out to be the most promising.

There is also a semantic version of the mutation operator. Similar considerations that hold for initialization about difficulty of creating a new behaviour are valid also in mutation. In this case the semantics is used to ensure that the new mutation produce changes in the program leading to a truly new behaviour in the population, in this regard, see [46] .

Interesting genetic operators able to directly manipulate the semantic of the solutions are the methods proposed by the *Geometric Semantic Genetic Programming* that we feel are more connected to this thesis and are presented with some details in chapter 3.

# Chapter 3

# Research Contribution

## 3.1 More on open problems and semantic Genetic Programming

In chapter 2.2 we introduced the reader to evolutionary techniques and specifically to *Genetic Programming* (GP). We already explored open issues related to GP in section 2.2 and now we introduce literature that is more connected to our point of view, addressing the themes that we are tackling within this thesis. Indeed, in this chapter, with the help of related literature, we focus on specific themes clarifying the problems themselves and then we further discuss similarities and differences among our proposals and the literature.

In particular there is a category of methodologies that is able to directly manipulate the semantic of a solution during sexual reproduction pushing towards the desired target. These techniques use special versions of crossover that are able to determine an average semantic between two others. In the next section, 3.1.1, we are going to explain how this has been accomplished and why is connected to our contribution in this thesis.

Another important related works is our previous investigation on semantic techniques (ESAGP algorithm) that introduces some of the concepts that we are going to generalize and extensively explore here, see section 3.1.2.

Finally, in section 3.2, we introduce our proposal as a general framework discussing the relationships with the literature. A more formal description of our proposal with detailed algorithms is in chapter 4.

### 3.1.1 Geometric Semantic Genetic Programming

We consider the semantic space already introduced in [8] and [9] where the semantic of a program is the vector whose elements are obtained evaluating every fitness case in the training set. If we call such set $\mathbf{X} = \{\overrightarrow{x_1}, \overrightarrow{x_2}, ..., \overrightarrow{x_n}\}$, in a symbolic regression context, a GP individual is a function or program $P$ that, for each vector $\overrightarrow{x_i}$ in $\mathbf{X}$ returns the scalar value $P(\overrightarrow{x_i})$. We define *semantics* of $P$ the vector $\overrightarrow{s_P} = [P(\overrightarrow{x_1}), P(\overrightarrow{x_2}), ..., P(\overrightarrow{x_n})]$. This vector can be represented as a point in an $n$-dimensional metric space, that is generally called *semantic space* where the well known properties characterizing a metric space hold thus having defined a distance $d$ with the usual properties:

$$\begin{aligned} d(s_{P_i}, s_{P_j}) &\geq 0 \\ d(s_{P_i}, s_{P_j}) = 0 &\Leftrightarrow s_{P_i} = s_{P_j} \\ d(s_{P_i}, s_{P_j}) &= d(s_{P_j}, s_{P_i}) \\ d(s_{P_i}, s_{P_j}) + d(s_{P_j}, s_{P_k}) &\geq d(s_{P_i}, s_{P_k}) \end{aligned} \tag{3.1}$$

Differently from the *syntactic* or *genotypic space*, where individuals are represented by programs, the semantic space give a numerical representation that is strictly related to the program behaviours during the test. In this scenario the syntax of the selected representations doesn't matter any more and the genetic operators designed in this space are almost independent from such phenotipic details. The purest form of this operators has been introduced by Moraglio et al. in [19] and named *Geometric Semantic Genetic Programming* (GSGP).

While each program has only one semantics, as already discussed in section 2.2.3.2,the mapping between the semantic space and the genotypic space is in general not a bijection, because several programs can have the same semantics. The target vector $\overrightarrow{t}$, defined as $\overrightarrow{t} = [t_1, t_2, ..., t_n]$ and containing the respective expected output, is a point in the semantic space. In general, the objective of GP is to find at least one program in the genotypic space that maps into $\overrightarrow{t}$ in the semantic space.

The really interesting point in GSGP methods is that the genetic operators are not blindly directing evolution of the individuals as the traditional operators do, instead they are reaching a known in advance point in the semantic space. Given the parents, the crossover operator is targeting an individual that is on the segment between its parents under the metric $d$ while the mutation guarantee that its results stay in the ball of radius $r$ again considering the metric $d$.

Segments and balls in the semantic space $S$ are defined like in the following points:

- the ball $B$ centred in point $x \in S$ is
  $B(x; r) = \{y \in S | d(x, y) \leq r\}$
- a point $m$ on the segment between $x, y \in S$ is
  $[x, y] = \{m \in S | d(x, m) + d(m, y) = d(x, y)\}$

We repeat here the GSGP mutation and crossover operators definitions:

**Definition 1** (Geometric Semantic Crossover)**.** *Considered two parents Programs* $P_1, P_2 : \mathbb{R}^n \to \mathbb{R}$ *the resulting program is* $P_x = (P_1 \cdot P_r) + ((1 - P_r) \cdot P_2)$ *where* $P_r : \mathbb{R}^n \to [0, 1]$ *is a random program.*

**Definition 2** (Geometric Semantic Mutation)**.** *Considered a single Program* $P : \mathbb{R}^n \to \mathbb{R}$ *the mutation operator produce a new mutated Program* $P_m = P + ms \cdot (P_{r1} - P_{r2})$ *where* $P_{r1}, P_{r2} : \mathbb{R}^n \to \mathbb{R}$ *are a random programs and* $ms \in \mathbb{R}$ *is a mutation step.*

Being in GSGP the distance of a program $P$ from the target $\overrightarrow{t}$ also its fitness we have a perfect fitness distance correlation so inducing a unimodal fitness landscape very easy to search or better to descend until the target. Moreover the offspring from the crossover is constrained in a predefined space, the segments between the parents, so, from definition 1, it's easy to argue that the offspring's semantic is never worse then the worst of the parents. Slightly different considerations holds for the mutation operator since in the original version (Moraglio et al. [19]) showed in definition 2 the radius of the semantic ball, where the mutation is free to arrive, it's not constrained by a maximum range. Vanneschi et al. in [47] and Gonçalves et al. in [48] stress the analysis on this point arriving at the conclusions, supported by the experimental evidence, that limiting the mutation to a maximum radius help generalization: indeed the geometrical properties of the semantic operators holds independently of the semantic space, during training or test session, so having a maximum worsening rate for the mutation and a guarantee of not worsening for the crossover give to GSGP a good generalization property when presenting to the model unseen test data. The good generalization ability of GSGP is further analyzed in [48] and a part of this property is attributed to the character of GSGP of reproducing to some extent the mechanism of ensemble learning. From definition 1 is evident that GSGP crossover is a linear combination of models approximating the target so having a common strategies with ensemble learning techniques. Ensemble learning has the proved quality of a good generalization thus, probably, this characteristic also contribute to the interesting performances of GSGP.

As one can easily observe form definitions 1 and 2 the offspring resulting from GSGP genetic operators is always bigger than the parents and, in particular, we notice that crossover output a program that incorporates the whole structure of both parents thus

resulting in an exponential growth of the program sizes during the evolution. To overcome this problem a compressed representations of the trees has been developed. As an example in [49], [50] and [51] we found efficient implementations of the GSGP operators that permits the fast evolutions of a solution as well as the use of the final model to calculate output of the model on previously unknown input, despite its structure has an exponential size. Because of this GSGP still remain a sort of black box approach. To get over this last problem numerous hybrid approaches have been developed that use the GSGP approach but just to a limited portion of the program structures. In these approaches the genetic operators substitutes sub-trees in the parents with others different trees having the desired semantics determined as one intermediate between parents sub-trees or as an optimal semantic with respect to the target, i.e. the sub-tree with the semantic values that make the whole tree to compute the target in all the training cases. Examples of literature that follows this research directions, introducing semantic-aware search operators with geometry built-in, are[52], [53], [54], [55], [20] [56] where often the optimal sub-tree semantics is calculated using a back-propagation algorithm.In the GSGP methods described here or in the semantics methods we analyzed in section 2.2.4 we find issues related to the effectiveness at exploring new semantics and solutions. In 2.2.4 we already described methods that try to prevent offspring too semantically similar to their parents and also in GSGP framework, in general, still persist this problem: the neutrality of the offspring. In other words, also the innovative semantic methods presented here have a probability of producing new individuals with the same semantics of the parents [57] and, while it is believed that a certain amount of neutral code, see section 2.2.1, in GP is necessary to the evolution, see for example [58], the GSGP techniques try to reach a good balance avoiding redundant solutions and maximizing exploration of the solution space. A good example of how this problem has been tackled is [54] where the geometric genetic operators are also "semantically-effective" meaning that there is a guarantee that the offspring is different from both the parents. Preventing duplicate semantics within parents and offspring looks in general beneficial from the point of view of the performance in the final model usually resulting in model with reduced errors at least in the training phase. Another important consideration, relevant to the geometric approach, is that the initial semantic distributions of the populations are very important because the geometric crossover produces offspring that is in the convex hull of the actual population. With the generations the convex hull will collapse eventually to a point, but if the target is not in the convex hull since the beginning the algorithm can not converge to the target. The geometric mutation can move independently from the actual convex hull mitigating this problem, but it is clear that many geometric implementation could suffer form this problem and indeed Gonçalves et al. [48] have experimented, with good results, GP algorithms where only the mutation is allowed and an Hill Climber with bounded and unbounded mutation operators. Geometric approaches have the capacity

of exploiting effectively the semantics already discovered but they could suffer from the pointy of view of exploration. Also our previous work [2] had one of the focus points on the semantic diversity. It was introducing for the first time an interesting methods based on *semantic angles* to measure diversity, this works is extensively discussed in section 3.1.2. The concept of semantic angle as diversity measure has been subsequently used in [59] where we find a comparison with other techniques to preserve diversity. Angular diversity results to be always beneficial in conjunction with both traditional and geometric genetic operators. The performance are always enhanced but the same results doesn't hold if the diversity is measured with a usual metrics respecting properties 3.1. Indeed traditional metrics maintain diversity at the cost of reducing optimization in the final stages of the evolution. Consider that while the GP cycle push the population towards the targets, the distance among the solutions is reduced till the points that they are too near each other, and many useful optimization are rejected. This situation is avoided using angular diversity because even if the solution are very near each other from the point of view, for example, of the euclidean distance the angle among solutions doesn't vary in the reference system centered on the target, like explained in [2] and after in [59].

### 3.1.2 ESAGP

The idea of directly manipulating individuals to obtain a specific semantic result is also present in our previous work [2], 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). We give the details to the interested reader in appendix A that is a useful complements also for the comprehension of this section.

ESAGP can be seen as one of the attempt to reduce the problem of the exponential bloat that we observe using pure geometric operators. We already seen such attempts in the previous section (3.1.1) indeed in [52], [53], [54], [55], and [56] we observe techniques that use the geometry in such a way to avoid the bloat phenomenon limiting the application of the crossover to subtrees or just approximate a geometric output.

ESAGP syetm it is built around the concept of error space, that can be seen as a semantic space centered or translated in to the point representing the semantic of the target, that, this way, become the center of this new reference system.The error vector of a GP individual $P$ is the vector $\overrightarrow{e_P} = \overrightarrow{s_P} - \overrightarrow{t}$. But ESAGP it's geometric in a different way compared with GSGP like techniques, the geometrical step it's applied only once at the end of the evolutionary process to produce the target approximation while during

the evolution the system try to find individuals spanning a semantic subspace that include the origin of the system (the target). The generalized version of ESAGP called $\mu-$ESAGP need $\mu + 1$ semantics or individuals. The first $\mu$ should represent linearly independent vectors while the last one is already in the sub-space spanned by the first $\mu$ vectors.

To be clear $\mu$-ESAGP try to linearly combine semantics of few individuals (one or two in [2]) calculating intermediate points on the segments connecting these semantics such that the last linear combination in this chain give as result the desired target. In the simple case we consider, as an example, the semantic error space of just one dimension, this means that we want to find two individuals in the population that have semantics aligned with the target or , equivalently, two semantics such that their error vectors stay in the same subspace. Thus ESAGP can be formulated as a liner algebra system that shows the alignment conditions of the semantic vectors and theirs intermediate points, see equation 3.2.

$$M_A \times M_X = M_B \tag{3.2}$$

The explicit matricial form of equation 3.2 is shown in 3.3 .

$$\begin{bmatrix} (1-k_1) & 0 & 0 & \cdots & 0 \\ -1 & (1-k_2) & 0 & \cdots & 0 \\ 0 & -1 & (1-k_3) & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -1 & (1-k_n) \end{bmatrix} \cdot \begin{bmatrix} \overrightarrow{Ip_1} \\ \overrightarrow{Ip_2} \\ \overrightarrow{Ip_3} \\ \vdots \\ \overrightarrow{Target} \end{bmatrix} = \begin{bmatrix} (\vec{p_1} - \vec{p_2}) * k1 \\ -\vec{p_3} * k2 \\ -\vec{p_4} * k3 \\ \vdots \\ -\vec{P_c} * k_n \end{bmatrix} \tag{3.3}$$

In equation 3.3 the $M_X$ entries $\overrightarrow{Ip_1} \cdots \overrightarrow{Ip_{n-1}}$ are the intermediate semantic points calculated to have a good alignment while the last entry of $M_X$ is the target itself, the last point obtained with the chain of linear combinations. The $M_B$ elements $\vec{p_1} \cdots \vec{p_n}$ are the semantic points representing individuals in the populations and the final entry, $\vec{P_c}$, is the one whose error $e_{P_c}$ is linearly dependent upon the previous $e_{p_i}$ ones. Thanks to this closing point ($\vec{P_c}$), using the alignment matrix $M_A$, we can calculate the explicit formula of the target and solve with respect to the $k_i$ unknown coefficients. Matrix $M_A$ represent the proportion, with respect to the segment connecting semantic points, where the next intermediate point is going to be placed, so it could be considered an alignment matrix. For the details of how calculate the $k_i$ coefficient in the case of one or two dimension the

reader can consult Appendix A, but the principle used there of course remain valid for the more general case shown here.

Using the formulation of equation 3.3 we can express, differently from [2], a generic algorithm to solve the regression problem in the semantic space.

**Algorithm 1** Naive subspace saturation algorithm.

```
sub_space_vector_base = ∅
repeat
  p := generate_new_individual()
  if e_p⃗ is a linear combination of error vectors e_{p_i}⃗ ∈ sub_space_vector_base then
    M_B = sub_space_vector_base ⋃ p
    use M_A × M_X = M_B to approximate the target
    return
  else
    sub_space_vector_base = sub_space_vector_base ⋃ p
  end if
until TRUE
```

In algorithm 1 new individuals are generated continuously with the aim of constructing a vector base spanning a subspace, in the error space, if the new individual's error $\vec{e_p}$ is linearly independent from the all the previous errors vectors then we can add it to the base spanning the subspace, $M_B$, otherwise we can directly use equation 3.3 to approximate the target and the search process is concluded. From this point of view algorithm 1 looks similar to a novelty search process [60] using angular distance as novelty criterion.

Despite the number of independent possible vectors in the base is at most equal to the number of dimension of the semantic space (usually a tractable number), this way of proceeding is not feasible in practice without specific precautions. Practical issues in using algorithm 1 are the computational complexity and the numerical stability of the procedures. Indeed to find the actual values of the $k_i$ it is necessary to solve symbolically the last entry of $M_X$, the target, but this expression of course grows exponentially in its complexity. We want to keep as low as possible the complexity of the final linear combination: we reach this goal combining at most a dozens of individuals. At the same time all the complexity is migrating in the calculation of the intermediate points in $M_X$, where the size of the expressions is almost doubling every time we solve a successive entry in the $M_X$ column, in a process resembling the geometric crossovers. Moreover establishing the independence of a vector from a subspace can become not trivial if the subspace counts thousands of vectors. These drawbacks have been tackled effectively in [2] where, using the same theoretical framework, a different algorithm, exploiting GP, has been shown to be effective in solving some real life problem.

Having highlighted the critical points in algorithm 1 we now show why this new perspective on ESAGP framework is interesting for this thesis. GP theoreticians and practitioners have been always interested in the so called *building blocks* as a mechanism for GP to reuse useful knowledge and gradually evolve toward a good solution. Different definitions of building blocks, often based on syntactic properties of the solutions, have been considered in the literature. At the same time, genetic operators with different level of awareness regarding the existence of these useful blocks, have been implemented or analyzed. Different crossover operators have been developed with the aim of preserving useful blocks and, at the same time, developing the ability of combining them in new solutions. Starting from the early attempts [61] passing through several investigations, for example [62],[63],[64], [65], [66] ,[67],[68], we arrive at actualization of the building blocks concept in the semantic GP framework for example in [8] where you can find interesting references also to other papers on this subject. Building blocks have been always conceived mainly as constituent of a single individual in the population. In this classical view useful building blocks are incorporated in the individuals improving their performances, the focus stays on the single solution even if the population has often been seen as source of such blocks. An interesting example, developing the building block scenario in the context of GSGP, is [53] where the crossover operator try to identify and preserve the building blocks. Indeed in that work they identify homologous regions of the parents using a structural analysis and use GSGP techniques so that the offspring can substitutes that functional block with another one converging to a convenient semantics, in other words they are identifying building blocks in the sub-trees structure and manipulate them to converge to a desired result. From the formulation analyzed above, equation 3.3 and algorithm 1, ESAGP can be considered a new building block framework where the focus has been shifted from the performance of a single individual to the ones of an entire population. In our view ESAGP vector bases in algorthm 1 are the semantic equivalent of building blocks. Every vector (individual) in that base is useful as a whole and let us move in the error space until a complete base bring us near the target. While in the usual view building blocks are combined by genetic operators within a single individual in ESAGP multiple individuals are combined via geometric crossovers like operators in a final solution. Note that differently from pure geometric approach ESAGP focus on the minimalism of the representation that, moving from a vector base, doesn't contain redundancy at all. In this new perspective semantic building blocks maps one to one with individuals and now the entire population can contribute to the solution in a way that looks similar to an ensemble learning but with a real "geometrical specialization" of each individual participating in the final solution. Another relevant aspect of ESAGP is that it introduces a new concept of similarity between two individuals. In section 3.1.1 we have already analyzed as the angles between individuals in the error space can be considered a good diversity measure. In [2] angles are used

to promote diversity during mate, indeed parents are selected in a way that maximize their orthogonality in the error space so that the offspring more likely explore a new semantic: this process has been named orthogonal selection. But the angular distance has also been used to discard offspring's individuals that are in the same subspace of any other individual already generated. Indeed considering equation 3.3 and algorithm 1, it is useless to have semantics that are in the same subspace or, that is the same, have angle of zero degree, because they are *equivalent* from this point of view, one or the other of the vectors in the same subspace will work equally well in approximating the target. ESAGP introduce the idea that, in a semantic and geometric framework, there exist a useful way of declaring two individuals equivalent and this way is to abandon the usual concept of distance characterized by properties 3.1 and adopt instead the concept *pseudo-distance* where instead it is possible that $s_{P_i} \neq s_{P_j}$ even if $d(s_{P_i}, s_{P_j}) = 0$. In section 3.2 we propose a novel framework trying to generalize these concepts behind the particular case of angles in the semantic. At last we observe that ESAGP, while is actively preventing the solution growth, is also fighting against issues in generalization. ESAGP, also in the [2] implementation does not admit offspring that are in the same subspace of any other individual ever generated. Arriving to this extreme we admit only individuals that are really different one from each other, denying completely the possibility of having neutral code in the solution structure. It looks similar to a tabu search algorithm working at this semantic, geometric level. One of the effects of the neutral code's absence is that the solution structural complexity can grow only if is contributing in a visible, controllable, manner to the fitness. This contribution can be positive or negative, worsening the performance of the individual or improving them. If the diversity has negative effects then natural selection pressure will soon discard that genotype while, on the contrary, if the effect is positive it will be preserved. In this last case the new structure of the offspring is likely to be more complex than the ones of the parents but this complexity is justified by a better performance. We are adding complexity only if is really useful while more traditional GP approaches allow the growth of neutral code at some degree. Another important point is that the complexity that we add, is more controllable, meaning that we allow only changes whose effect on the phenotype is immediately evident and measurable: this way we minimize the probability that explosion of complexity can derive from bubbles of neutral code growing without control, hidden. Such bubbles can, when rearranged by genetic operators, fit better the data point in the training set, but if we didn't improve first on simpler hypothesis then we don't respect the Occam's razor principle and we are introducing unnecessary complexity and going fast toward overfitting. Imposing an immediate evaluation of the new structures we introduce complexity with much more granularity minimizing extemporary jumps . Even when "competent" techniques have been developed in the geometric framework we observe that the semantic similarity is compared using a usual metric thus

excluding only one single solution at a time while ESAGP, using angle in the error space, can exclude entire classes of individuals already represented in the past population: in other words we are exploring the solution space per classes rather than per individuals. Geometric operators can achieve good generalization exploiting their geometrical properties, as explained in section 3.1.1, e.g. imposing that the offspring is no worse than the worst of the parents, so always justifying the increase in complexity, even the exponential growth. At the other extreme ESAGP allow an unconstrained geometric jump toward the target but increase the complexity of the solutions at the minimum allowed rate, controlling this other way the overfitting. Mixing the two approaches is possible only having in mind these two different solution to the bloat and overfitting problem, e.g. using a GSGP approach to align individuals in the same subspace, without taking care of their complexity, and then using ESAGP to jump toward the target without constraints shouldn't work and would results in generalization issues, indeed it does as shown in [69]. On the contrary use semantic angles to preserve diversity and GSGP to have constrained moves toward the target should produce a useful synergy, indeed it does as showed in [59].

## 3.2 Proposal

In this section we give a very high level overview of the framework we introduce along with four concrete examples to help the comprehension. The same four example are then formalized and experimented in the next chapters. After this brief introduction we discuss what are the relationships with the literature presented in this same chapter, explaining the motivations of our choices. Indeed the work presented here is strongly related to contributions discussed in the previous sections and it directly incorporates semantic awareness in GP. Nevertheless, it does so from a different perspective. In this thesis, we propose a novel idea to exploit semantic awareness in GP: *semantic based equivalence classes* (SECGP). We remain in the context of semantic techniques so we define a semantic space similar to the ones already introduced in sections 3.1.1 and 3.1.2. Let $\mathbf{X} = \{\overrightarrow{x_1}, \overrightarrow{x_2}, ..., \overrightarrow{x_n}\}$ be the set of input data, or fitness cases, of a symbolic regression problem, and $\overrightarrow{t} = [t_1, t_2, ..., t_n]$ the vector of the respective expected output or target values (in other words, for each $i = 1, 2, ..., n$, $t_i$ is the expected output corresponding to input $\overrightarrow{x_i}$). A GP individual (or program) $P$ can be seen as a function that, for each input vector $\overrightarrow{x_i}$ returns the scalar value $P(\overrightarrow{x_i})$. Following [20], we call *semantics* of $P$ to the vector $\overrightarrow{s_P} = [P(\overrightarrow{x_1}), P(\overrightarrow{x_2}), ..., P(\overrightarrow{x_n})]$. This vector can be represented as a point in an $n$-dimensional space, that we call *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 in general not a bijection, because several programs can have the same semantics. The target vector $\overrightarrow{t}$ 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 $\overrightarrow{t}$ in the semantic space.

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, finding one solution in the same equivalence class as a globally optimal solution allows us to solve the problem by reconstructing the global optimum analytically.

Even though the concept of semantic based equivalence classes is general, and many different implementations can be imagined, here we instantiate this idea by defining two particular GP systems, corresponding to two particular definitions of equivalence.

In the first system, two individuals $P_1$ and $P_2$ are in the same equivalence class if the two semantic vectors of these two individuals are directly proportional, i.e. $\overrightarrow{s_{P_1}} = k \cdot \overrightarrow{s_{P_2}}$. In other words, there must exist a scalar constant $k$ such that, for each fitness case, the output of $P_1$ is equal to $k$ multiplied by the output of $P_2$. 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 becomes to find any solution whose semantics is directly proportional to the target $\overrightarrow{t}$ (i.e. that stands in the same straight line as the one intersecting the target and the origin). As such, it makes sense to define a new fitness function, corresponding to the variance, or any other measure of dispersion, of the ratios between the coordinates of the semantics of an individual and $\overrightarrow{t}$. When this variance is equal to zero, the individual is in the same equivalence class as the target, and the search can terminate, reconstructing analytically a globally optimal solution.

The second GP system that we propose is similar to the first one, but this time two individuals $P_1$ and $P_2$ are considered in the same equivalence class if $\overrightarrow{s_{P_1}} = k + \overrightarrow{s_{P_2}}$. This time, 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 between each other. In such a system, it makes sense to use the variance, or any other measure of dispersion, of the difference between the coordinates of the semantic vector and the target as fitness. Once again, when this variance is equal to zero, the semantic vector of the individual is in the same equivalence class as the target, and so it is possible to reconstruct the target analytically.

The first one of these two systems is called *GPMUL* (given that multiplication is the operator that allows us to reconstruct the target), while the second one is called *GPPLUS* (given that the target can be reconstructed using addition). Furthermore, we introduce two new GP systems that are similar to *GPMUL* and *GPPLUS*, but a filter that allows us to reject an individual 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 *FGPMUL* and *FGPPLUS* respectively. We refers generically to a SECGP system using filters with the name of F-SECGP. The idea of SECGP is to explore the solution space by classes of solutions rather then single individuals, this is an attempt to improve the efficiency of the search process that, behind the potential obvious advantages, has many interesting implications that we discuss in the following.

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 [20], we use traditional genetic operators to build new solutions. Doing so, we lose the interesting property of a unimodal fitness landscape induced by GSGP operators. Moreover when we transform any individual in to another of the same class we are not establishing any bound on the distance of the new semantic we are targeting and this, as discussed in section 3.1.2 is a potential treat to the generalization ability of the algorithm. Also the ensemble like effect of GSGP and ESAGP is lost because we are not combining, at this level, multiple individuals, we just use a single semantic and we eventually manipulate that one just a single time in order to approximate the target.

We lose these important properties in trade of the possibility of obtaining human readable solutions of compact size. Indeed, using a F-SECGP system, we extremely reduce the possibility of introducing neutral code and this, as discussed in previous sections, limits the useless growth of the individuals' structures. As a matter of fact SECGP come naturally equipped with the ability of discriminating class of semantics and this further enhance the type of redundancy that can be avoided, not just a single point in the semantic space but rather entire classes. Any single individual is representative of a class and using the equivalence relationship it's easy to compute any other individual in the same class, this imply that we have a new *pseudometric* that has the same advantages discussed for ESAGP semantic angles, but is also a generalization of that concept, indeed semantic angles are a particular implementation of an SECGP system.

Thanks to its enhanced capacity of discrimination among semantics we expect that a SECGP systems have the ability to maintain an higher level of diversity in the population so exploring more effectively the solution space and at a faster rate than GSGP.

Furthermore GSGP has a good exploitation ability by construction but at the same time we have seen that this represents a problem if the target it's far from the convex hull of the initial population, even using mutation, bounded or not, the exploration of the solution space can results slow, thus we expect SECGP systems to perform competitively on this point.

But how is an SECGP system compensating for the loss of properties that guarantee a good generalization of GSGP? SECGP adopt the strategy of reducing as much as possible the code growth that it's not related to any real change in fitness. We already discussed in section 3.1.2 the Occam razor principle applied to ESAGP. Here we are pushing further this concept formulating a generalization, the *equivalence classes*, that give more flexibility to the SECGP system designer. Another important point in favour of generalization in SECGP is that equivalence classes are themselves a method of generalization although in a slightly different way. Indeed, if the equivalence relationship, we use to construct classes, capture some useful properties of the application domain, then a structurally simple individual in the population of solution is able to fit a great number of cases in the training set. The same generic model fit different situations or instantiations of that concept in the datasets, thanks to the equivalence relationship that model become a sort of *semantic building block*. We think that the usefulness of the equivalence classes it is independent of the application domain but, at the same time, it is obvious that the knowledge related to a specific application can further enhance the effectiveness of this approach, capturing symmetries, scales and other typical patterns. If these patterns or domain properties are not known in advance then the SECGP system designer has two options.

The first one has been adopted in this thesis: use simple generic equivalence concepts that probably are valuable in many cases. Even if in this our work we use just one equivalence at a time it is simple to implement a system that take in account more than one equivalence relationship during the same evolutionary process. Indeed the main computational cost is the calculation of the individual's semantics while the verification of equivalence relationships consist in fast vector computations easily parallelizable on modern hardware.

The second option offered to the designer is to learn the equivalence relationship in a specific domain. Depending on the domain this can require a specific labeled dataset and probably it's not a trivial task nevertheless an interesting perspective. We are not investigating this possibility here.

SECGP is also solving some potential contradiction that ESAGP systems potentially suffer, discussed below. 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 thesis is more general and can incorporate ESAGP as a particular instantiation. ESAGP, as 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 the individuals that are considered redundant because of equivalence classes. This is an option exploited also in this work, but some contradictions of ESAGP are resolved here. ESAGP was maximizing semantic diversity and the exploration of 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. The filtering proposed here has been parametrized using a more understandable variable like the error's variance. This could potentially enable a deeper analysis of the filtering threshold to reach a dynamic self-tuning regulation along the evolutionary process. 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 have been evaluated as a whole: having new classes better fitness, the evolutionary process never returns on 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 to a specific equivalence class.

Our work has also important relationships with the well-known Linear Scaling technique ($LS$) introduced by Keijzer in [70]. If seen from our perspective in this work, $LS$ 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 $LS$. 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 a 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 equivalence definitions reported in Equation (4.4) and Equation (4.10) to the target, 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 the possibility of using filtering techniques (as the ones explained above) to improve *LS*.

# Chapter 4

# Semantic Equivalence Classes Genetic Programming

## 4.1 Methodology

Our objective is to use equivalence classes (EQCs) to create partitions of the solution space. In this way, once we have an individual belonging to one EQC, we are able to analytically construct any other individual in that class, including the one that is closer to the target in the semantic space (i.e. the one that has the best fitness). So, basically, all individuals in the population can be replaced by the best solution that belongs to their own EQC. In this way, considering the fitness of an EQC as the fitness of the best individual in that EQC, we can say that we explore the space of EQCs instead of the space of the single solutions. Under this perspective, we can even force GP to not create, or to reject, any individual belonging to an EQC that is already represented in the population. In the continuation, we define equivalence relationships using criteria based on semantics.

We introduce a relationship that we call *equivalence function* ($EF$). $EF$ receives as arguments two vectors in the semantic space, and return a vector of the same cardinality. We say that two individuals are equivalent (i.e. they belong to the same EQC) if, for every fitness case, $EF$ calculated for the two individuals returns the same constant value $k$. In particular, we have that an individual whose semantics is a vector that we call $\overrightarrow{gp}$ is equivalent to the target $\overrightarrow{t}$ if, in Equation (4.1), $\overrightarrow{k}$ has all the components $k_i$ identical to each other:

$$EF(\overrightarrow{t}, \overrightarrow{gp}) = \overrightarrow{k} \tag{4.1}$$

Once we have a GP individual equivalent to $\overrightarrow{t}$, we can reconstruct an individual whose semantics is identical to $\overrightarrow{t}$ (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), like in Equation (4.2):

$$\overrightarrow{t} = EF^{-1}(\overrightarrow{k}, \overrightarrow{gp}) \tag{4.2}$$

From Equation (4.2), reconstructing an optimal genotype (like for instance a tree) is straightforward. It is enough to compose the tree whose semantics is $\overrightarrow{gp}$ (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, that can be able to quantify the distance between the EQC of an individual and the EQC of the target. A simple possibility is, for instance, to measure the *dispersion* of the $k_i$ values, i.e. the components of vector $\overrightarrow{k}$ in Equation (4.1). This dispersion must be as low as possible, since we aim at obtaining a single constant value, which represents a perfect equivalence. For this purpose, we can, for instance, use the *variance* of the components of $\overrightarrow{k}$, like in Equation (4.3).

$$Fitness = var(\overrightarrow{k}) \tag{4.3}$$

If we consider $\overrightarrow{k}$ as the error vector of a solution (i.e. the vectorial difference between its semantics and the target), then the variance of $\overrightarrow{k}$ 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 (4.2).

In the continuation of this paper, these general concepts will be instantiated by defining two different concrete $EF$ functions. Once we have done that, the method to reconstruct the target, starting from an individual in the same EQC, will possibly appear clearer.

An important observation is that, given that we work with continuous values, it is unlikely that two individuals will be exactly in the same EQC. For this reason, a threshold is used to establish the EQCs. Interestingly, and contrarily to what happens in standard GP, in none of the above phases an error function measuring directly the distance to the target in the semantic space, like for instance the root mean square error (RMSE), is used to drive the evolutionary process. In other terms, the sought for individuals (the

ones that are in the same equivalence class as the target, or close) can even have a very bad RMSE.

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. In the end, if we consider our system as exploring the space of EQCs (or, which is the same thing but seen from a different viewpoint, if once we generate an individual, we are immediately able to obtain analytically the best individual in terms of RMSE that belong to the same EQC as that individual), 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. More in particular, 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. Algorithm 2 contains the pseudo-code of the proposed method, where points 2.2.4. and 2.2.5. implement the filter. A crucial part of Algorithm 2, i.e. how to calculate fitness, is better specified in Algorithm 3. More specifically, Algorithm 3 contains an explanation of how fitness is calculated (point 2.2.4. of Algorithm 2) in the general case. Concrete instantiations of this algorithm will be shown in Algorithm 4 and Algorithm 5 (see the next section), reporting the process used to calculate fitness in the concrete cases of *GPPLUS* and *GPMUL*, respectively (these two methods are presented in detail in the next section).

**Algorithm 2** Genetic programming algorithm showing filtering methods, in points 2.2.4. and 2.2.4.

1. Generate a random population $\mathbf{P}$ of $N$ individuals
2. Repeat Until termination condition:
   - 2.1. Create an empty population $\mathbf{P}'$
   - 2.2. Repeat until $\mathbf{P}'$ contains $N$ individuals:
     - 2.2.1. Chose a genetic operator (crossover with probability $p_c$, mutation with probability $p_m$)
     - 2.2.2. Select one or two individuals depending on the choice in 2.2.1.
     - 2.2.3. Apply the operator chosen in 2.2.1. to the individual(s) from step 2.2.2.
     - 2.2.4. **Evaluate offspring**
     - 2.2.5. **If the new individuals' semantics are in the same equivalence class of others in $\mathbf{P}$ then they are discarded otherwise they are inserted in $\mathbf{P}'$**
   - 2.3. $\mathbf{P} = \mathbf{P}'$
3. Return best individual in $\mathbf{P}$

**Algorithm 3** Generic individual evaluation: fitness calculation in point 2.2.4. of Algorithm 2

1. Take as input an individual
2. Evaluate the individual on any dataset instance
3. Use Equation (4.1) to calculate vector $\overrightarrow{k}$ (*i.e.* $\overrightarrow{k} = EF(\overrightarrow{t}, \overrightarrow{gp})$
4. Use Equation (4.3) to calculate $Fitness = var(\overrightarrow{k})$
5. Return the $Fitness$

### 4.1.1 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 a first case. The resulting technique, that we call *GPPLUS*, is based on the concept of error vector. The error vector is 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. In other terms, two solutions are in the same equivalence class if their semantics are equivalent by translation. We define the equivalence function $EF$ like in Equation (4.4).

$$EF = \overrightarrow{t} - \overrightarrow{gp} = \overrightarrow{k} \tag{4.4}$$

At this point, the fitness function is defined as the variance of $\overrightarrow{k}$:

$$\sigma^2 = MEAN[(k_i - MEAN(\overrightarrow{k}))^2] \tag{4.5}$$

If GP finds a solution in the same EQC as the target, to build a globally optimal solution, we assume the scalar value of $k$ to be equal to the mean of $\overrightarrow{k}$:

$$k = MEAN(\overrightarrow{k}) \tag{4.6}$$

Then, a globally optimal solution (i.e. a solution whose semantics is identical to the target) can be built using Equation (4.7), where $GP$ is the notation used to identify the program that has $\overrightarrow{gp}$ as semantics:

$$T_{apx} = GP + k \tag{4.7}$$

Assuming that the final assessment of the quality of the solutions is done using the RMSE, and being the function $GP$ the target's approximation, we optimize Equation (4.8) for every point in the dataset.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(t_i - gp_i - k)^2}{n}} \tag{4.8}$$

Assigning to the first derivative of Equation (4.8) the value of zero, $\frac{\partial}{\partial k}RMSE = 0$ and calculating $k$, we obtain Equation (4.6).

Algorithm 4 describes how fitness is calculated for *GPPLUS*. Algorithm 4 can be considered an instantiation of the general process reported in Algorithm 3, applied to the concrete case of *GPPLUS*.

**Algorithm 4** Fitness calculation in point 2.2.4. of Algorithm 2 in the specific case of the *GPPLUS* technique.

1. Take as input an individual.
2. Evaluate this individual on any dataset instance.
3. Use Equation (4.4) to calculate $\overrightarrow{k} = \overrightarrow{t} - \overrightarrow{gp}$
4. Use Equation (4.5) to calculate $Fitness = MEAN[(k_i - MEAN(\overrightarrow{k}))^2]$
5. Return the $Fitness$

As previously explained, it can be useful to reject individuals that are in the same EQC as another individual in the population. To check if two individuals, with respective semantic vectors $\overrightarrow{gp_1}$ and $\overrightarrow{gp_2}$ are in the same EQC, we calculate $EF$ like in Equation (4.9):

$$EF = \overrightarrow{gp_1} - \overrightarrow{gp_2} = \overrightarrow{k} \tag{4.9}$$

To have a measure of similarity between the two classes we calculate the variance of $\overrightarrow{k}$, and we compare it to a prefixed threshold. Individuals below the threshold are considered in the same EQC and rejected. A convenient threshold's value has been found experimentally, and the results of this experimental study are discussed later in this paper. We call *FGPPLUS* this filtered version.

### 4.1.2 GPMUL: GP by proportions

In this section, we propose a second possible instantiation of the framework presented so far. The resulting GP system, that we call *GPMUL*, is based on the ratio between

the semantics of the solutions and the target, calculated for every fitness case. Two solutions are in the same EQC if the ratios of the coordinates of their semantic vectors are identical to each other. In other words, if they are equivalent by scale. We define the new $EF$ error function like in Equation (4.10):

$$EF = \frac{\overrightarrow{gp}}{\overrightarrow{t}} = \overrightarrow{k} \tag{4.10}$$

Analogously to *GPPLUS*, also for *GPMUL* fitness could be defined as the variance of the components of $\overrightarrow{k}$, as in Equation (4.5). A problem arises when GP tries to minimize Equation (4.5) by increasing the absolute value of the numerator in Equation (4.10) and by decreasing its denominator. To get a real progress during the evolution, we should express the dispersion of $\overrightarrow{k}$ in a different way. For instance, it is possible to consider a normalized version of $\overrightarrow{k}$, as in Equation (4.11).

$$k_{norm_i} = \frac{k_i}{MEAN(\overrightarrow{k})} \tag{4.11}$$

In *GPMUL*, we have artificially imposed very poor fitness values for all the programs for which $MEAN(\overrightarrow{k})$ is equal to zero, thus preventing the system from failures during the evaluation of the individuals. It is important to remark that a globally optimal solution (i.e. a solution for which the values of all the coordinates of $\overrightarrow{k}$ are equal to zero) is not the only case in which $MEAN(\overrightarrow{k})$ can be equal to zero. Thus, testing whether $MEAN(\overrightarrow{k})$ 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 to guarantee a correct functioning of the system.

Given that we have:

$$MEAN\left(\frac{k_i}{MEAN(\overrightarrow{k})}\right) = 1 \tag{4.12}$$

then Equation (4.5) can be transformed into a new fitness function, as in Equation (4.13):

$$Fitness = MEAN\left[\left(\frac{k_i}{MEAN(\overrightarrow{k})} - 1\right)^2\right] \tag{4.13}$$

Algorithm 5 describes how fitness is calculated for *GPMUL*. Algorithm 5 can be considered an instantiation of the general process reported in Algorithm 3, applied to the concrete case of *GPMUL*.

**Algorithm 5** Fitness calculation in point 2.2.4. of Algorithm 2 in the specific case of the *GPMUL* technique.

1. Take as input an individual.
2. Evaluate this individual on any dataset instance.
3. Use Equation (4.10) to calculate $\overrightarrow{k} = \frac{\overrightarrow{gp}}{\overrightarrow{t}}$
4. Use Equation (4.13) to calculate $Fitness = MEAN\left[\left(\frac{k_i}{MEAN(\overrightarrow{k})} - 1\right)^2\right]$
5. Return the $Fitness$

Analogously (although not identically) to *GPPLUS*, once GP has found a solution in the same EQC as the target, we use Equation (4.14) to build a globally optimal solution $T_{apx}$ (where $GP$ represents a program whose semantic vector is $\overrightarrow{gp}$):

$$T_{apx} = \frac{GP}{k} \tag{4.14}$$

Assuming again the RMSE as the measure that has to be minimized, we should find the scalar value $k$ that optimizes Equation (4.15):

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(t_i - \frac{gp_i}{k})^2}{n}} \tag{4.15}$$

Assigning the first derivative to zero, $\frac{\partial}{\partial k}RMSE = 0$ , we obtain the optimal value of $k$ expressed by Equation (4.16):

$$k = \frac{\sum_{i=1}^{n}(gp_i)^2}{\sum_{i=1}^{n}(t_i \cdot gp_i)} \tag{4.16}$$

The filtering process is similar to the one introduced in Section 4.1.1, but with a difference regarding the $EF$ function. Specifically, we use Equation (4.10) applied to two semantic vectors $\overrightarrow{gp_1}$ and $\overrightarrow{gp_2}$ like in Equation (4.17):

$$EF = \frac{\overrightarrow{gp_1}}{\overrightarrow{gp_2}} = \overrightarrow{k} \tag{4.17}$$

where fitness cases for which the denominator of the equation is equal to 0 are ignored. Finally, we evaluate if the new individual has to be discarded by checking if the dispersion of $\overrightarrow{k}$ is below a prefixed threshold. As previously mentioned, an appropriate threshold

value has been determined empirically, and the results of this experimental study are reported and discussed later in this paper. We call this filtered version *FGPMUL*.

# Chapter 5

# Experimental study

This chapter describes the experimental phase. More specifically, Section 5.0.1 introduces the objectives of the experimental study, describes the test problems taken into account and the experimental settings while Section 5.0.2 discusses the obtained results.

### 5.0.1 Systems and test problems

The objective of the experimental study is to compare between each other the methods *GPPLUS*, *GPMUL*, *LS* and their filtered variants, *FGPPLUS*, *FGPMUL* and *FLS*. For getting a better understanding of the quality of the performance of these systems, we also consider the performance obtained by geometric semantic genetic programming (GSGP) [20]. We also decided to not report results achieved by standard GP in this experimental study, because a preliminary study has indicated that standard GP was consistently outperformed by all the presented methods on all the test problems.

The proposed *GPPLUS*, *GPMUL* and *LS* systems, as well as their filtered variants, have been implemented on top of the ECJ framework, a well-known and freely available GP system (https://cs.gmu.edu/~eclab/projects/ecj/). To compare the performance of these systems against the one of GSGP, we have considered seven different symbolic regression test problems that have been already used in previous GP studies. Table 5.1 reports, for each dataset, the number of features and the number of instances. For a complete description of these datasets, the reader is referred to the references reported in the same table.

For all the test problems, 100 runs were performed with each technique. In each run, a different partition between training and test data was considered, built as follows: 70% of the instances, selected at random with uniform probability, were used for training the model, while the remaining 30% have been used to test its performance on unseen data.

TABLE 5.1: Description of the test problems. For each dataset, the number of features (independent variables) and the number of instances have been reported.

| Dataset | # Features | # Instances |
|---|---|---|
| Airfoil Self-Noise [71] | 5 | 1502 |
| Concrete Compressive Strength [72] | 8 | 1029 |
| Parkinson Voice Recording (MOTOR) [73] | 19 | 5875 |
| Parkinson Voice Recording (TOTAL) [73] | 19 | 5875 |
| Protein Folding [74] | 9 | 45000 |
| Concrete Slump Test [75] | 9 | 102 |
| Yacht Hydrodynamics [76] | 6 | 307 |

All the techniques share exactly the same configuration, except for the value of the filter threshold (in the various filtered versions). For this parameter, a preliminary extensive study has been performed for determining the value that allows us to obtain the best results. All the runs used populations of 200 individuals allowed to evolve for 200 generations. Trees initialization was performed using the Ramped Half-and-Half method [21], with a maximum initial depth equal to 6. During the evolution, no maximum depth was imposed for the trees. The function set contained the four binary arithmetic operators, including protected division as in [21]. The terminal set contained a number of variables equal to the number of features in the dataset. Survival from one generation to the other was always guaranteed to the best individual of the population (elitism). Crossover and mutation probabilities are equal to 0.9 and 0.1 respectively. Selection has been performed using a tournament of size 4. This configuration of the parameters has been obtained after a preliminary tuning phase. The parameter values used in the experimental phase are the ones that have allowed us to obtain the best results among the other alternative configurations tested. The experimental study that has allowed us to choose the best filter parameter for each studied technique is a part of the results that will be presented in the next section. In particular, these results are presented in Figures 5.15 -5.21.

For all the test problems, results are reported in terms of the root mean square error (RMSE) between target and predicted values. It is worth reaffirming that only GSGP, among the compared techniques, uses the RMSE as fitness function during the evolutionary process. In all the plots reported in the continuation of this section, medians of the RMSE of the best individual in the population on the training set, over 100 independent runs, have been reported. The median was preferred over the mean because of its higher robustness to outliers. Median RMSE has been studied against the computational effort instead of the number of generations, because generations do not have the same computational complexity for all the studied methods. Following [**?** ], as a measure of the computational effort we have used the accumulated number of tree nodes evaluated

until the current instant. In other words, for each generation $g$, the computational effort spent by a GP technique until generation $g$ is:

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

where, for each generation $h = g, g-1, ..., 1$, $NN_h$ is the sum of the number of tree nodes in all the individuals that have been evaluated at generation $h$. It is worth pointing out that, while calculating $CE(g)$, we do not take into account only the individuals that take part in the evolution, but also the ones that, eventually, have been 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, an evaluation of the individual is needed, 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 the ones without filters.

### 5.0.2 Experimental Results

The discussion of the results starts by considering a comparison between the proposed systems and GSGP. After this analysis, an evaluation of the beneficial effects of using filters is reported. Then, the section is concluded with an analysis of the size of the individuals in the population during the evolution of the different studied techniques.

For each technique considered, we report the median best RMSE against the computational effort. The median is calculated over 100 independent runs, where each run uses a different training-test partition. Results on the test set are relative to the RMSE of the best training individual, evaluated on the test set.

In a 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 GSGP. This allows us to report the comparison between the studied methods and GSGP in a more readable way, since only four methods will be compared to each other, instead of seven. These results are reported in Figures 5.1 to 5.14. In all these figures, plot (a) contains a comparison between *GPPLUS* and *FGPPLUS*; plot (b) contains a comparison between *GPMUL* and *FGPMUL*; plot (c) contains a comparison between *LS* and *FLS*; and finally plot (d) contains a comparison between the methods that obtained the best results in plots (a), (b) and (c) and GSGP. While plots (a), (b) and (c) report curves of the evolution of the RMSE against the computational effort, plots (d) report the final comparisons with GSGP in the form of boxplots. The motivation for this difference is

that, in this way, GSGP results can be taken directly from the literature (where they are actually reported as boxplots).

Concerning the test problems, the results are organized as follows:

- Figure 5.1 and Figure 5.2 report the results for the airfoil self-noise problem (simply *airfoil* from now on), respectively on the training and test set.
- Figure 5.3 and Figure 5.4 report the results for the concrete compressive strength problem (*concrete* from now on) (Figure 5.3 on the training set and Figure 5.4 on the test set).
- Figure 5.5 and Figure 5.6 report the results for the Parkinson's voice recording problem, using the MOTOR dataset (*motor* from now on) (Figure 5.5 on the training set and Figure 5.6 on the test set).
- Figure 5.7 and Figure 5.8 report the results for the Parkinson's voice recording problem, using the TOTAL dataset (simply *total* from now on), respectively on the training and test set.
- Figure 5.9 and Figure 5.10 report the results for the protein folding problem (simply *protein* from now on) (Figure 5.9 on the training set and Figure 5.10 on the test set).
- Figure 5.11 and Figure 5.12 report the results for the concrete slump test problem (*slump* from now on), respectively on the training and test set.
- Figure 5.13 and Figure 5.14 report the results for the yacht hydrodynamics problem (*yacht* from now on) (Figure 5.13 on the training set and Figure 5.14 on the test set).

As we can see from these figures, GSGP is, in general, the method that returned the best RMSE, except for the motor and protein problems (where the best values of RMSE were found by $FGPPLUS$ and $FLS$, both on the training and on the test set) and for the yacht problem (where the best RMSE was found by $FLS$, both on the training and on the test set). Moreover, GSGP was outperformed by $FGPMUL$, $FGPPLUS$ and $FLS$ on the test set when the slump problem was considered. Furthermore, filtering appears to be beneficial for $GPMUL$, $GPPLUS$ and $LS$ (a deeper analysis on the effect of filtering is offered in the continuation). In fact, the filtered versions always outperformed the non-filtered ones, both on the training and on the test set, 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, is related to their behavior on the

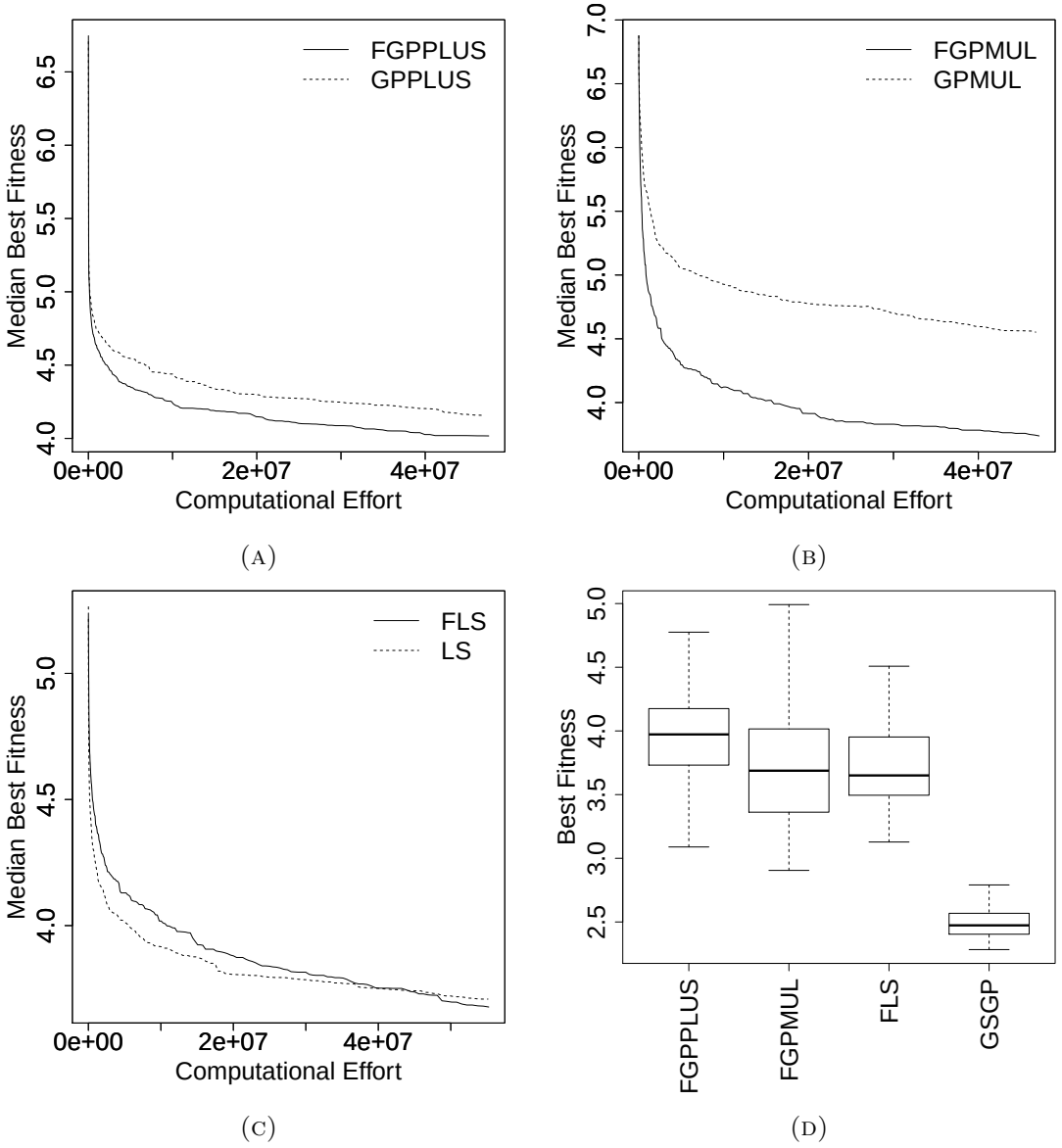



FIGURE 5.1: Dataset *airfoil* (training). Results are relative to: *GPPLUS* technique (5.1a), *GPMUL* (5.1b), *LS* (5.1c). Figures (5.1a), (5.1b), (5.1c), have median fitness (RMSE) on the vertical axes and computational effort on horizontal axes (calculated as the number of nodes evaluated during training). Figure (5.1d) reports the performance of GSGP as well as the ones achieved by the 3 best variants (i.e., with or without filters) of the proposed system based on equivalence classes.

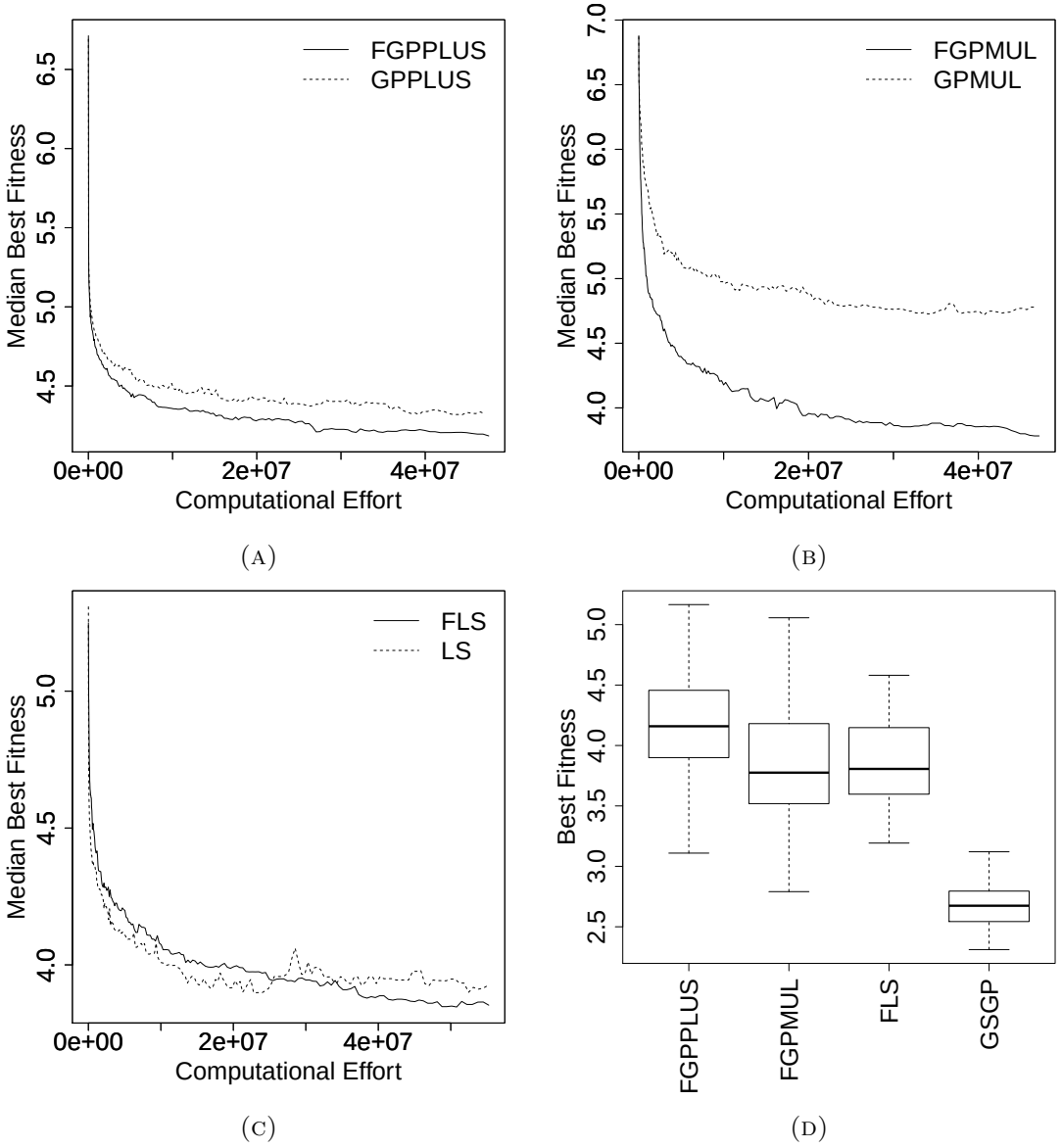



FIGURE 5.2: Dataset *airfoil* (test). Results are relative to: *GPPLUS* technique (5.2a), *GPMUL* (5.2b), *LS* (5.2c). Figures (5.2a), (5.2b), (5.2c), have median fitness (RMSE) on the vertical axes and computational effort on horizontal axes (calculated as the number of nodes evaluated during training). Figure (5.2d) reports the performance of GSGP as well as the ones achieved by the 3 best variants (i.e., with or without filters) of the proposed system based on equivalence classes.

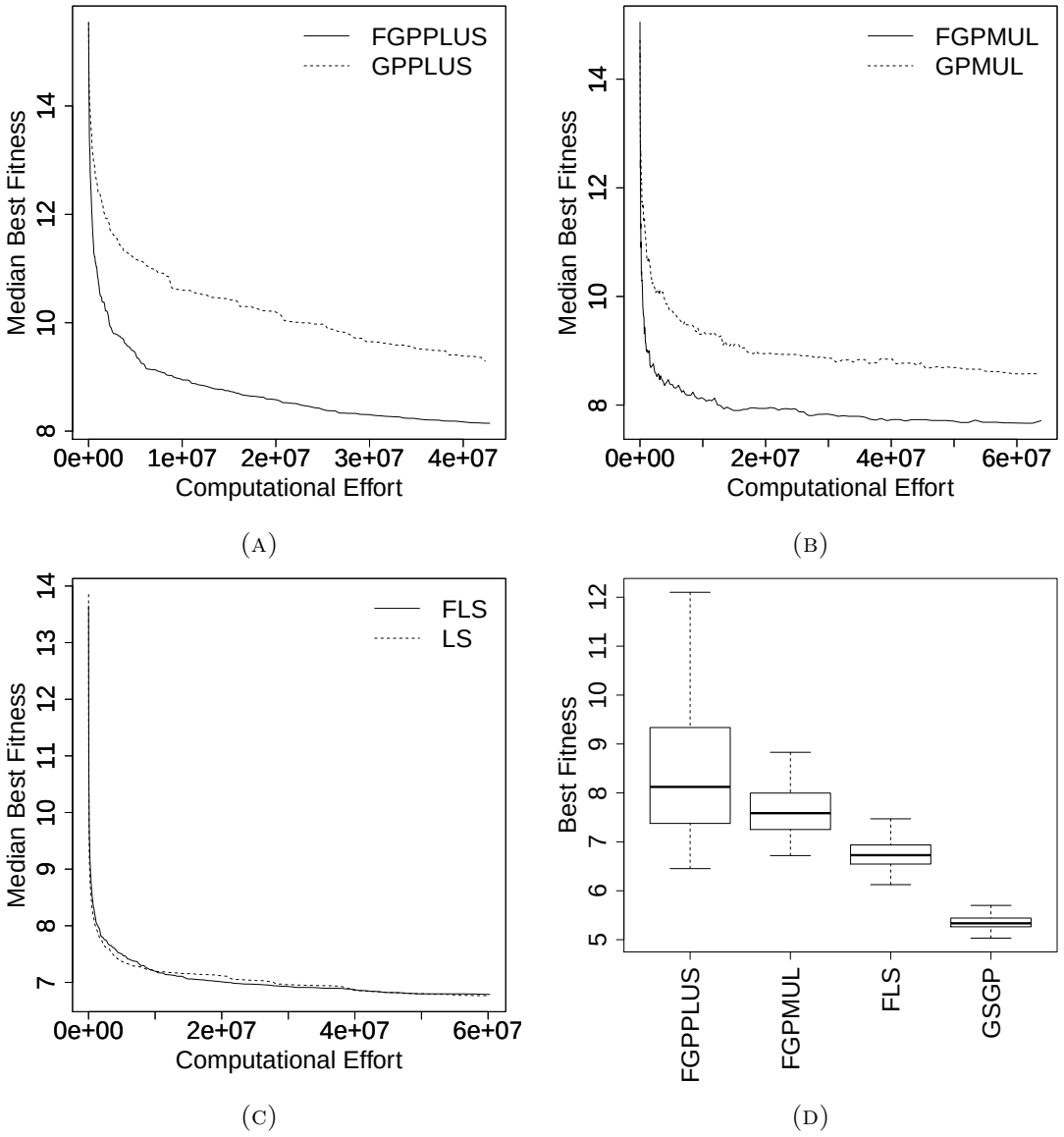
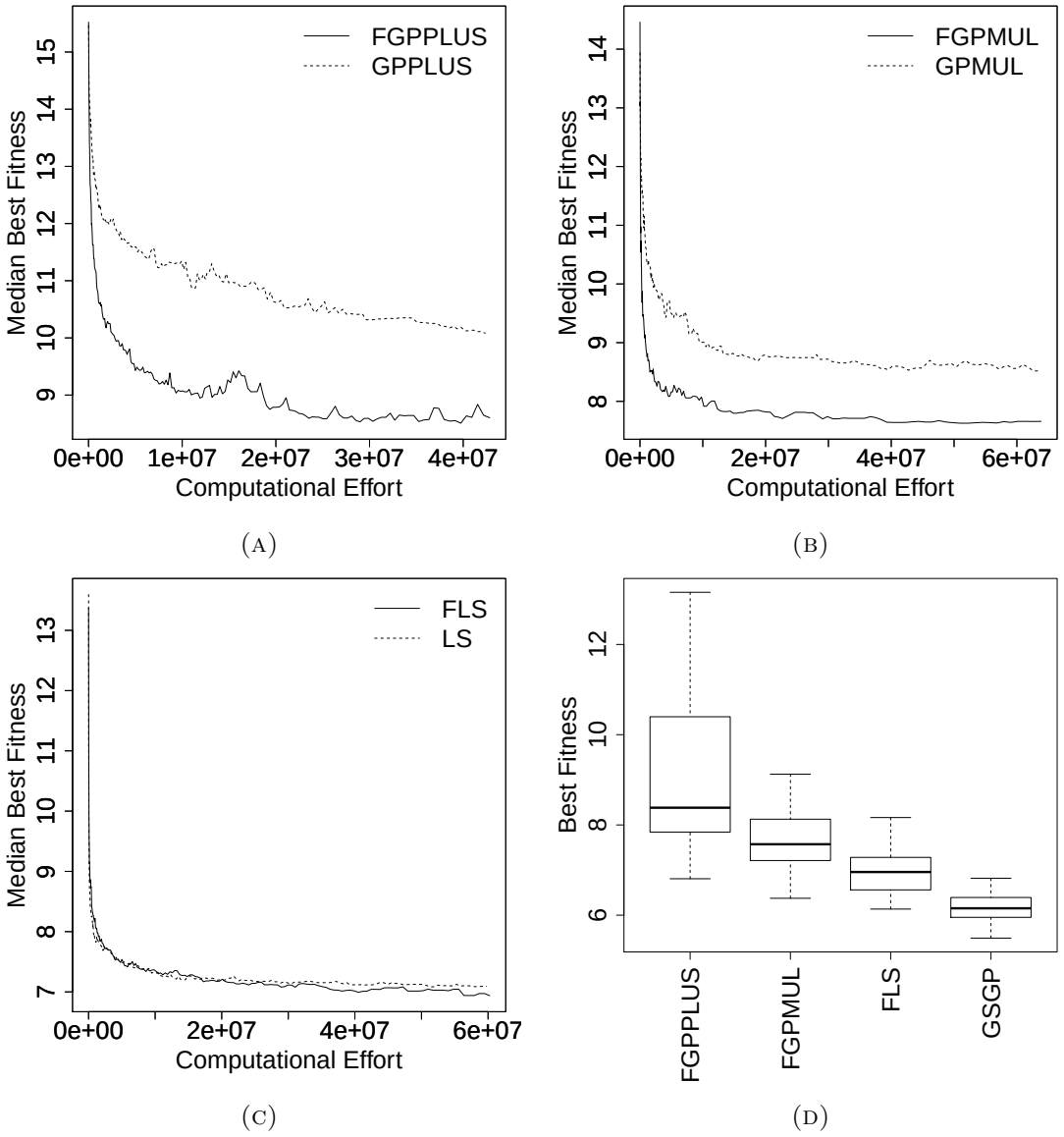


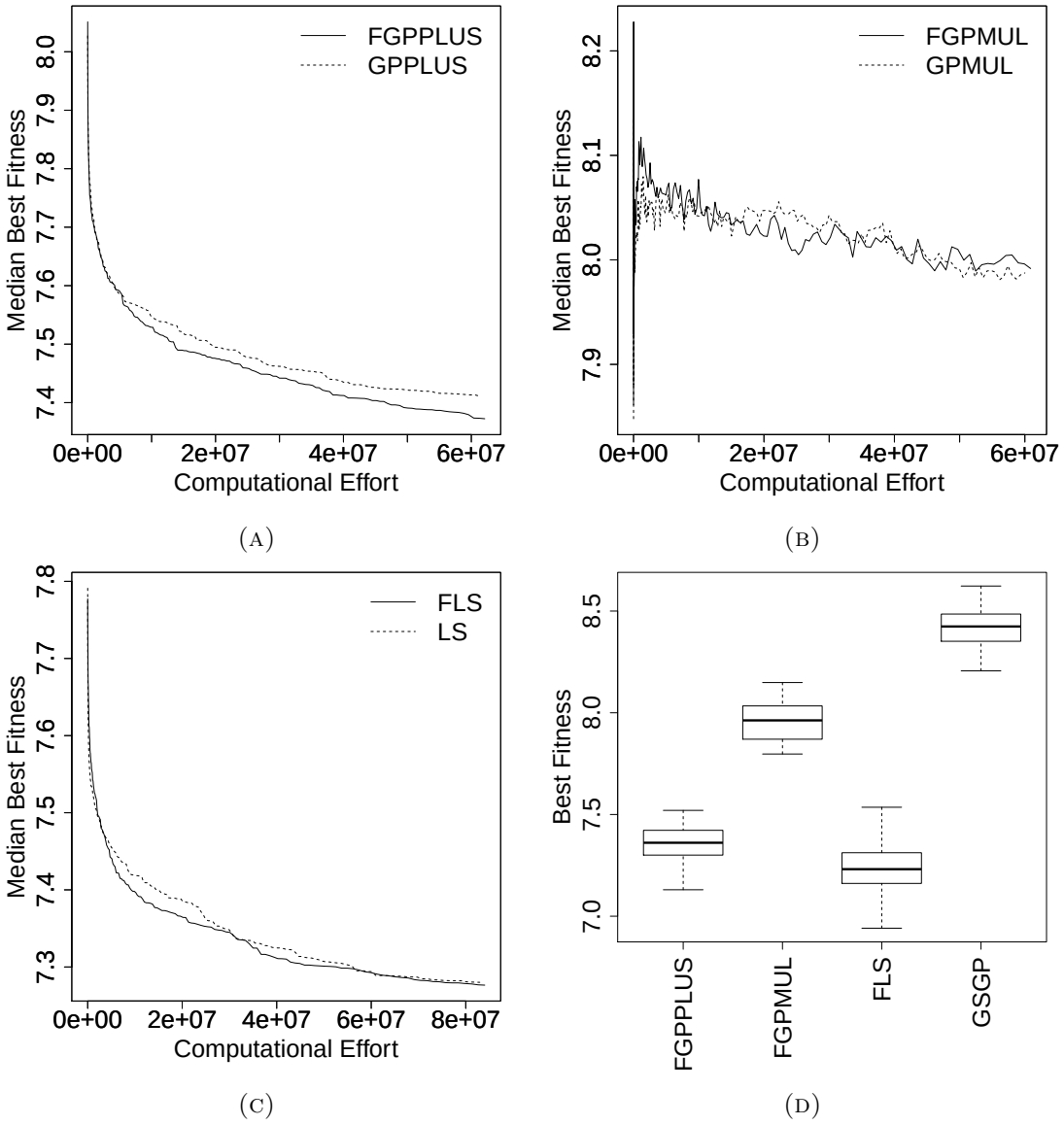

FIGURE 5.3: Dataset *concrete* (training). Results are relative to: *GPPLUS* technique (5.3a), *GPMUL* (5.3b), *LS* (5.3c). Figures (5.3a), (5.3b), (5.3c), have median fitness (RMSE) on the vertical axes and computational effort on horizontal axes (calculated as the number of nodes evaluated during training). Figure (5.3d) reports the performance of GSGP as well as the ones achieved by the 3 best variants (i.e., with or without filters) of the proposed system based on equivalence classes.




(A) (B)

(C) (D)

FIGURE 5.4: Dataset *concrete* (test). Results are relative to: *GPPLUS* technique (5.4a), *GPMUL* (5.4b), *LS* (5.4c). Figures (5.4a), (5.4b), (5.4c), have median fitness (RMSE) on the vertical axes and computational effort on horizontal axes (calculated as the number of nodes evaluated during training). Figure (5.4d) reports the performance of GSGP as well as the ones achieved by the 3 best variants (i.e., with or without filters) of the proposed system based on equivalence classes.




(A) (B)

(C) (D)

FIGURE 5.5: Dataset *motor* (training). Results are relative to: *GPPLUS* technique (5.5a), *GPMUL* (5.5b), *LS* (5.5c). Figures (5.5a), (5.5b), (5.5c), have median fitness (RMSE) on the vertical axes and computational effort on horizontal axes (calculated as the number of nodes evaluated during training). Figure (5.5d) reports the performance of GSGP as well as the ones achieved by the 3 best variants (i.e., with or without filters) of the proposed system based on equivalence classes.

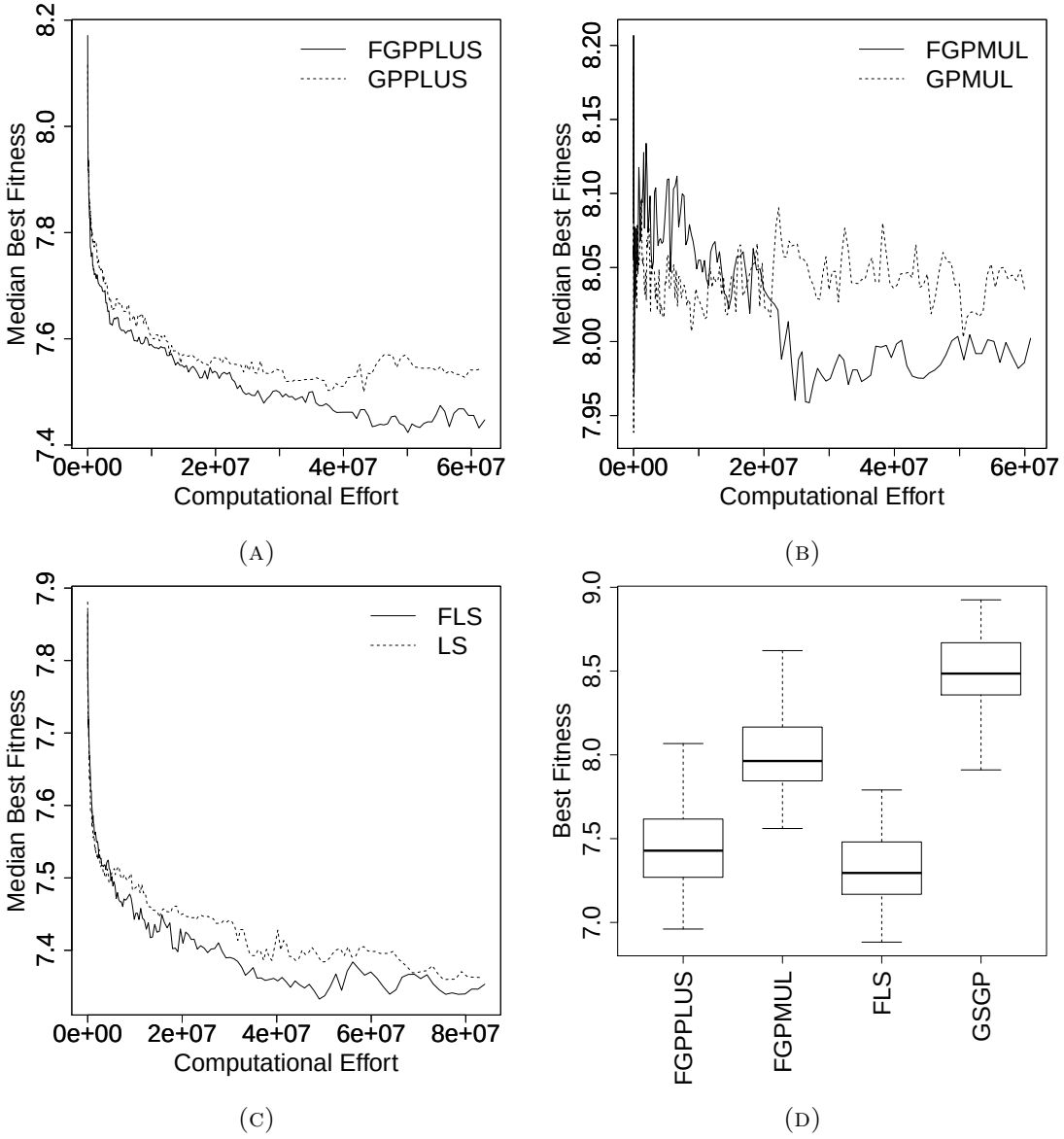
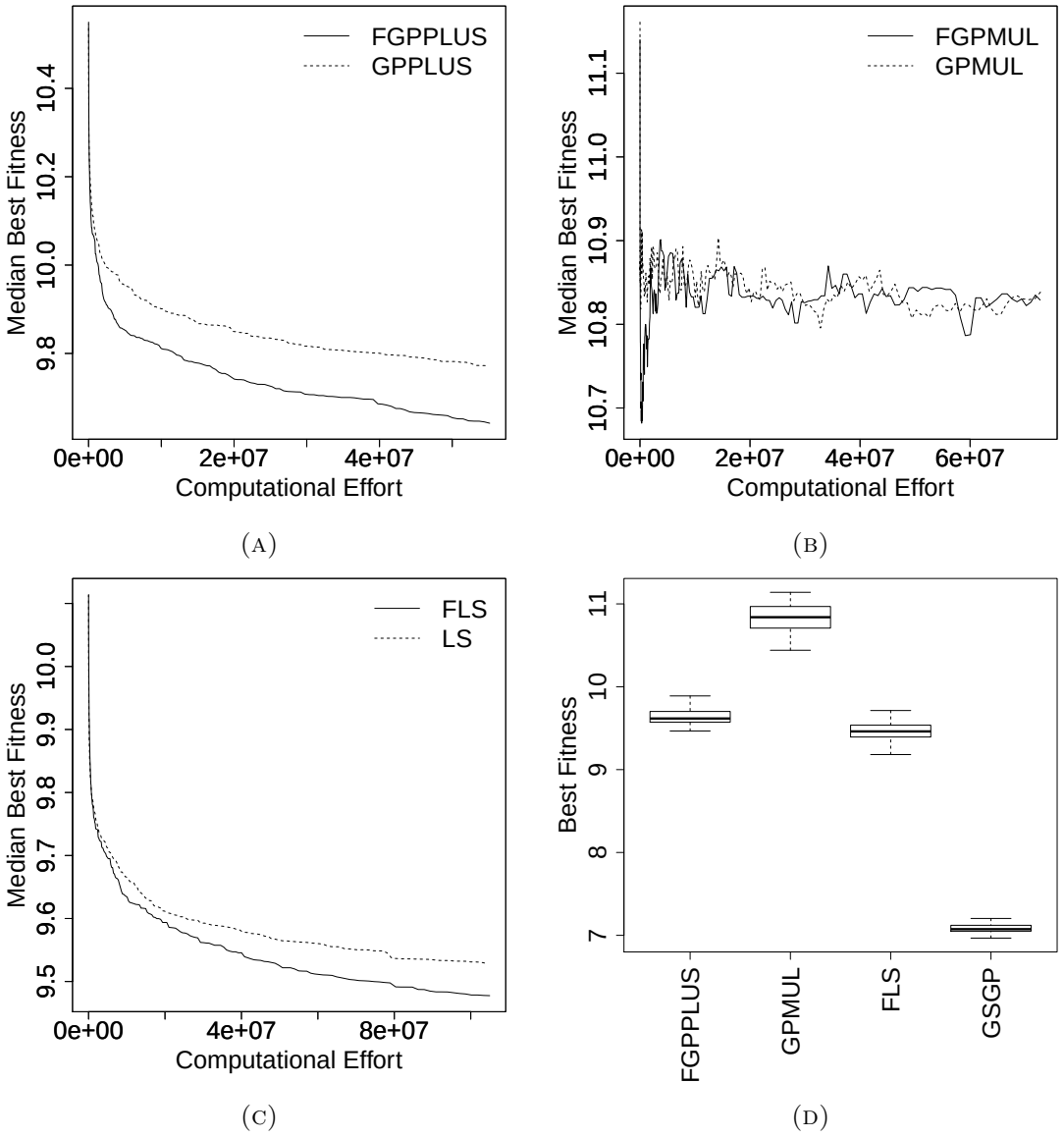


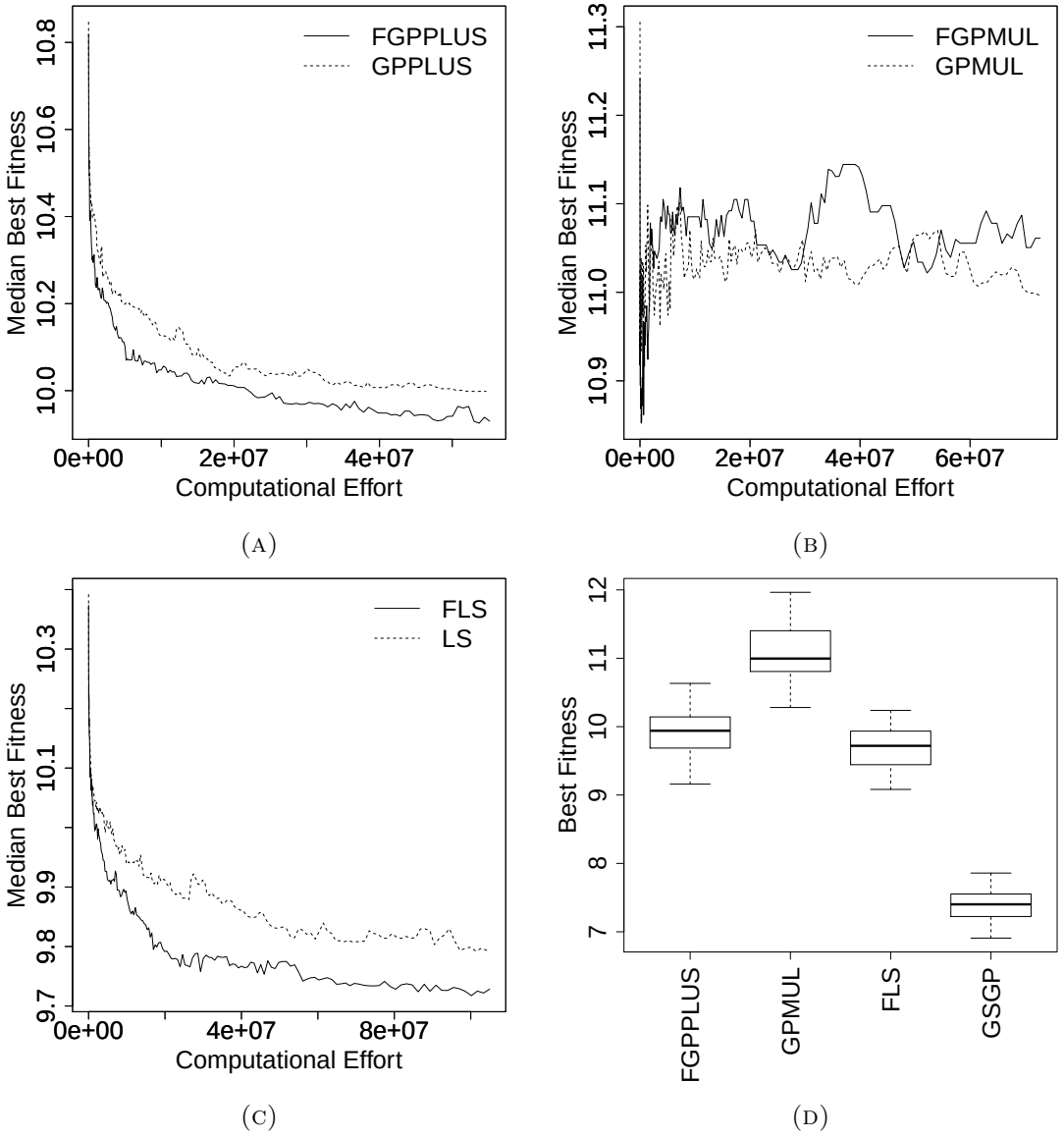

FIGURE 5.6: Dataset *motor* (test). Results are relative to: *GPPLUS* technique (5.6a), *GPMUL* (5.6b), *LS* (5.6c). Figures (5.6a), (5.6b), (5.6c), have median fitness (RMSE) on the vertical axes and computational effort on horizontal axes (calculated as the number of nodes evaluated during training). Figure (5.6d) reports the performance of GSGP as well as the ones achieved by the 3 best variants (i.e., with or without filters) of the proposed system based on equivalence classes.




(A) (B)

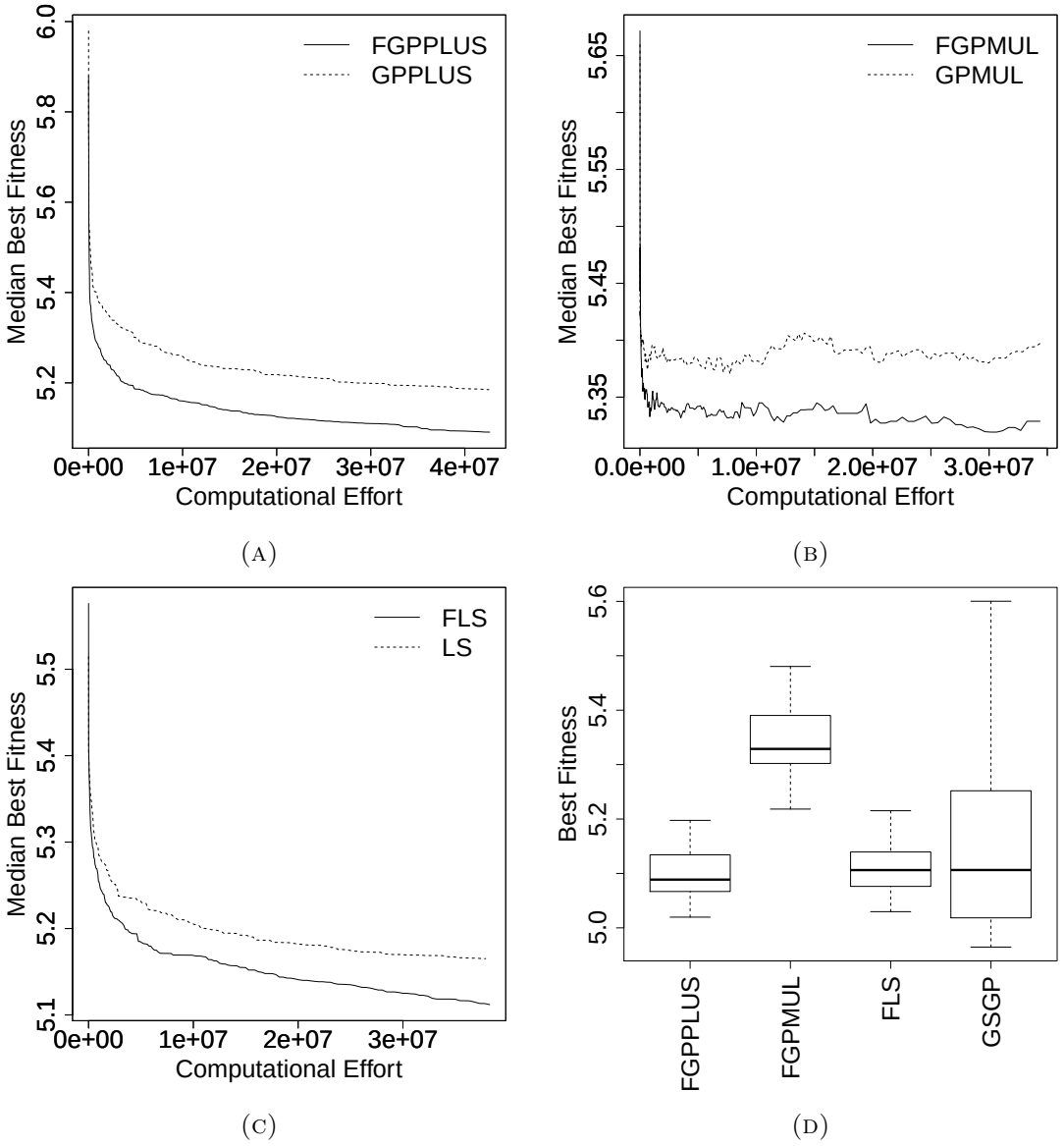
(C) (D)

FIGURE 5.7: Dataset *total* (training). Results are relative to: *GPPLUS* technique (5.7a), *GPMUL* (5.7b), *LS* (5.7c). Figures (5.7a), (5.7b), (5.7c), have median fitness (RMSE) on the vertical axes and computational effort on horizontal axes (calculated as the number of nodes evaluated during training). Figure (5.7d) reports the performance of GSGP as well as the ones achieved by the 3 best variants (i.e., with or without filters) of the proposed system based on equivalence classes.




(A) (B) (C) (D)

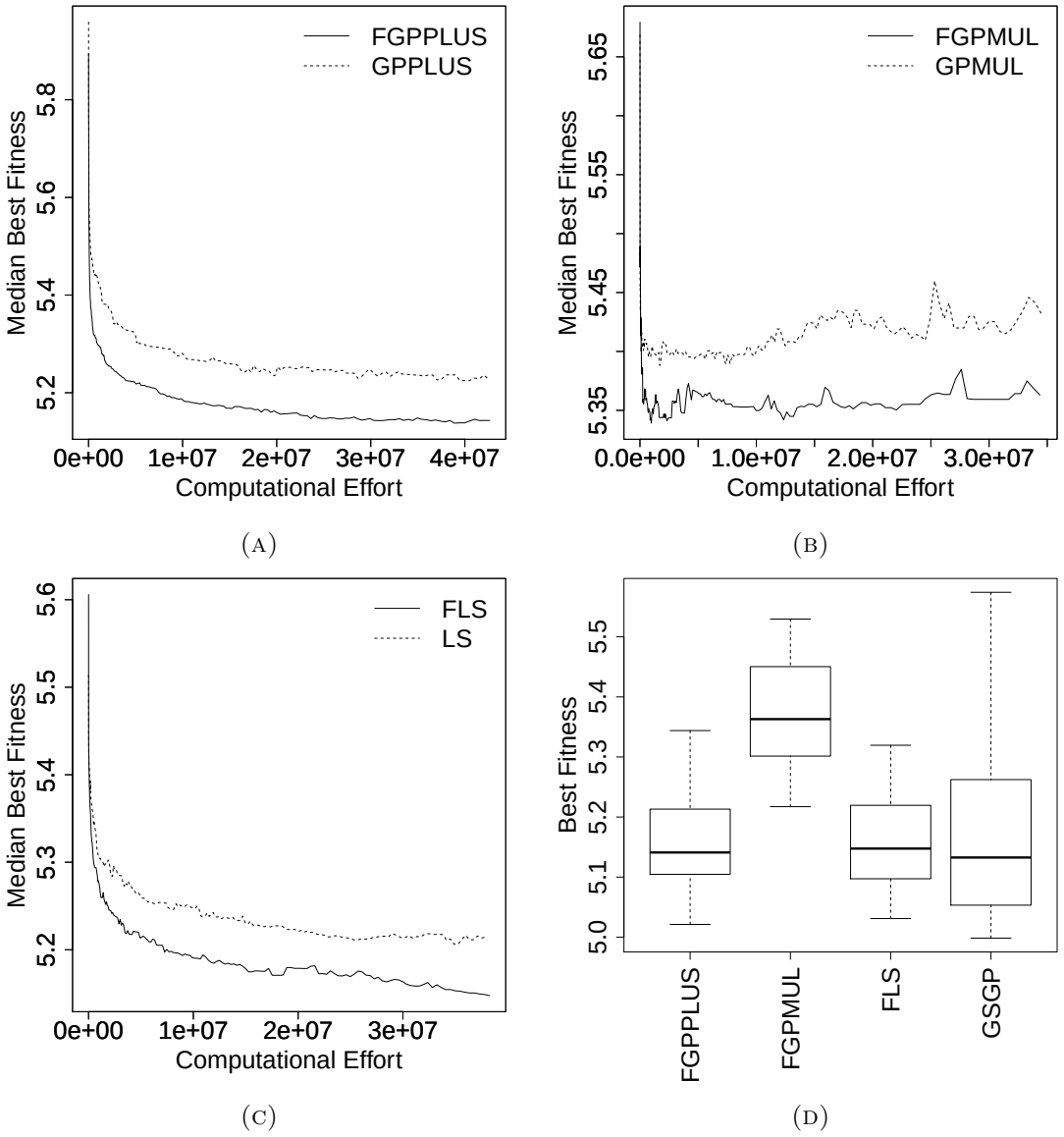
FIGURE 5.8: Dataset *total* (test). Results are relative to: *GPPLUS* technique (5.8a), *GPMUL* (5.8b), *LS* (5.8c). Figures (5.8a), (5.8b), (5.8c), have median fitness (RMSE) on the vertical axes and computational effort on horizontal axes (calculated as the number of nodes evaluated during training). Figure (5.8d) reports the performance of GSGP as well as the ones achieved by the 3 best variants (i.e., with or without filters) of the proposed system based on equivalence classes.




(A) (B)

(C) (D)

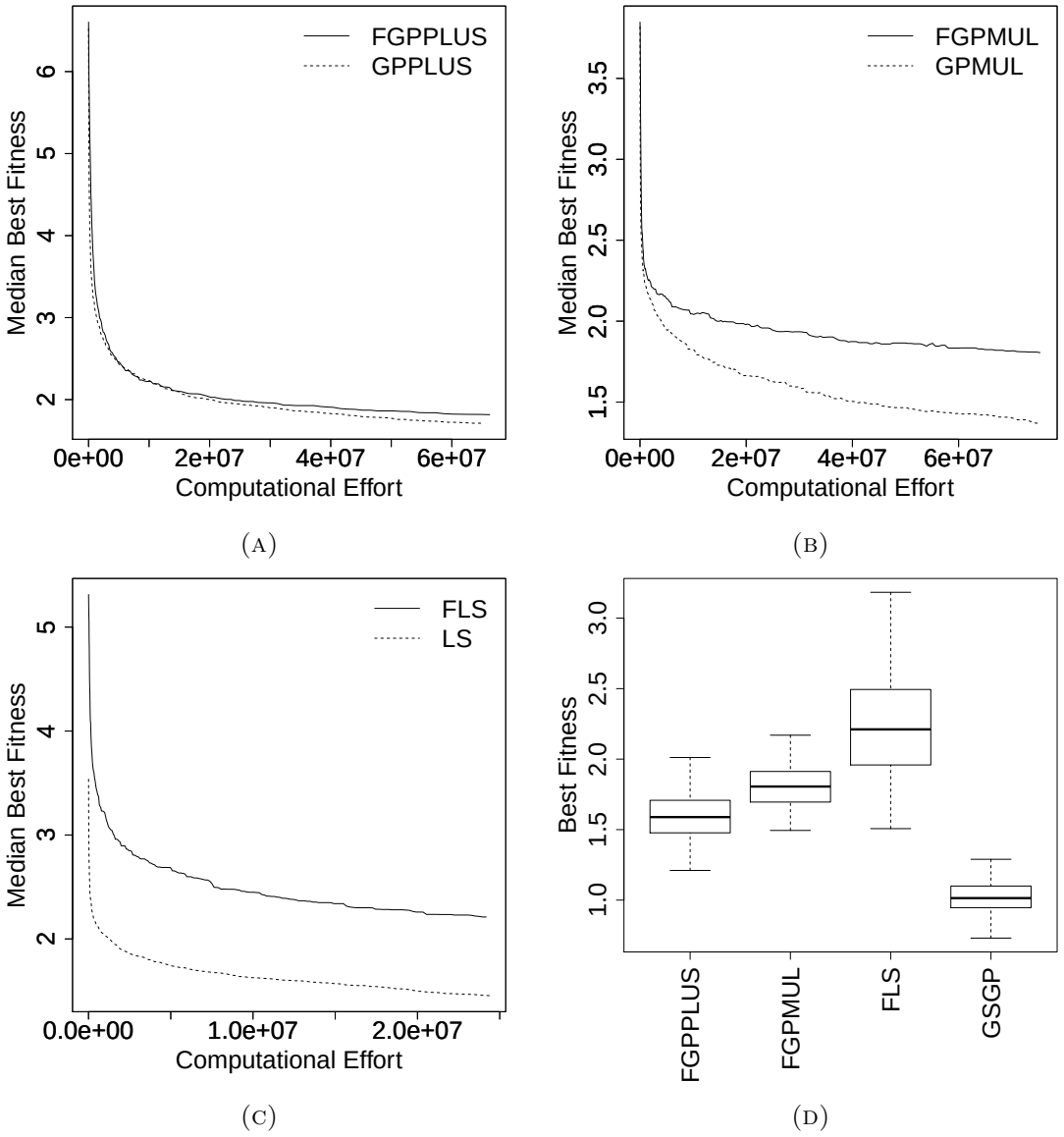
FIGURE 5.9: Dataset *protein* (training). Results are relative to: *GPPLUS* technique (5.9a), *GPMUL* (5.9b), *LS* (5.9c). Figures (5.9a), (5.9b), (5.9c), have median fitness (RMSE) on the vertical axes and computational effort on horizontal axes (calculated as the number of nodes evaluated during training). Figure (5.9d) reports the performance of GSGP as well as the ones achieved by the 3 best variants (i.e., with or without filters) of the proposed system based on equivalence classes.




(A) (B)

(C) (D)

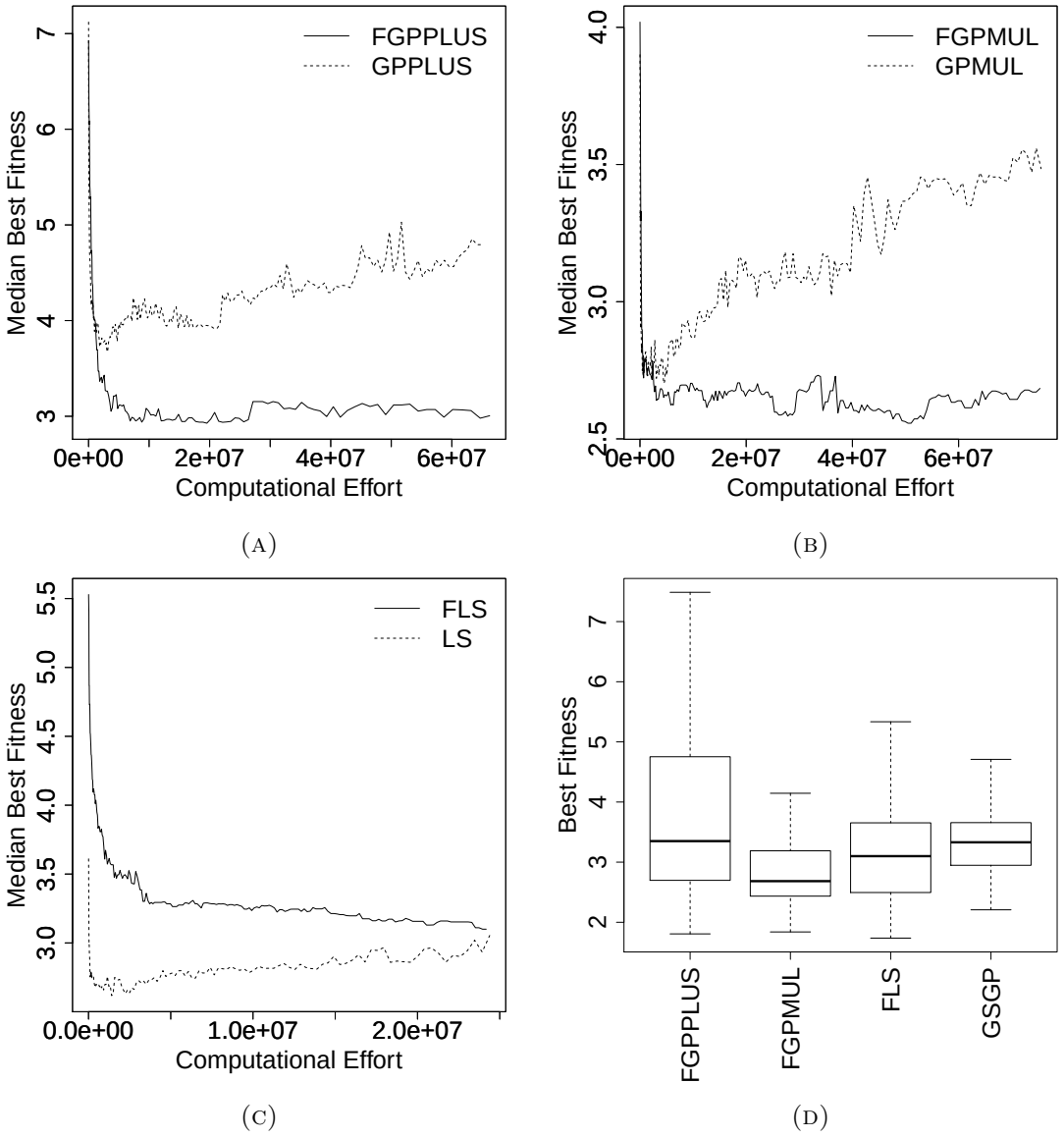
FIGURE 5.10: Dataset *protein* (test). Results are relative to: *GPPLUS* technique (5.10a), *GPMUL* (5.10b), *LS* (5.10c). Figures (5.10a), (5.10b), (5.10c), have median fitness (RMSE) on the vertical axes and computational effort on horizontal axes (calculated as the number of nodes evaluated during training). Figure (5.10d) reports the performance of GSGP as well as the ones achieved by the 3 best variants (i.e., with or without filters) of the proposed system based on equivalence classes.




(A) (B)

(C) (D)

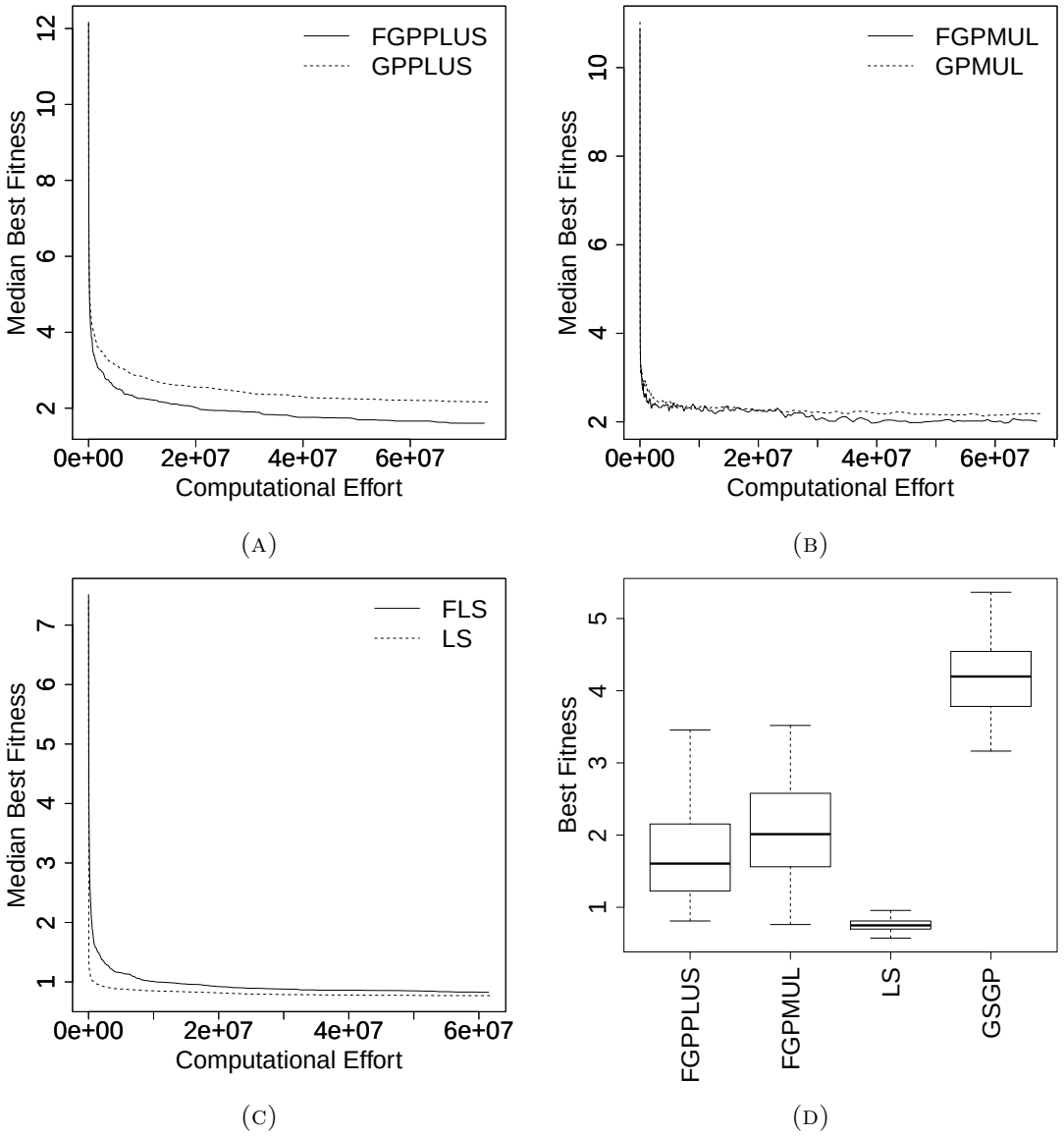
FIGURE 5.11: Dataset *slump* (training). Results are relative to: *GPPLUS* technique (5.11a), *GPMUL* (5.11b), *LS* (5.11c). Figures (5.11a), (5.11b), (5.11c), have median fitness (RMSE) on the vertical axes and computational effort on horizontal axes (calculated as the number of nodes evaluated during training). Figure (5.11d) reports the performance of GSGP as well as the ones achieved by the 3 best variants (i.e., with or without filters) of the proposed system based on equivalence classes.




(A) (B)

(C) (D)

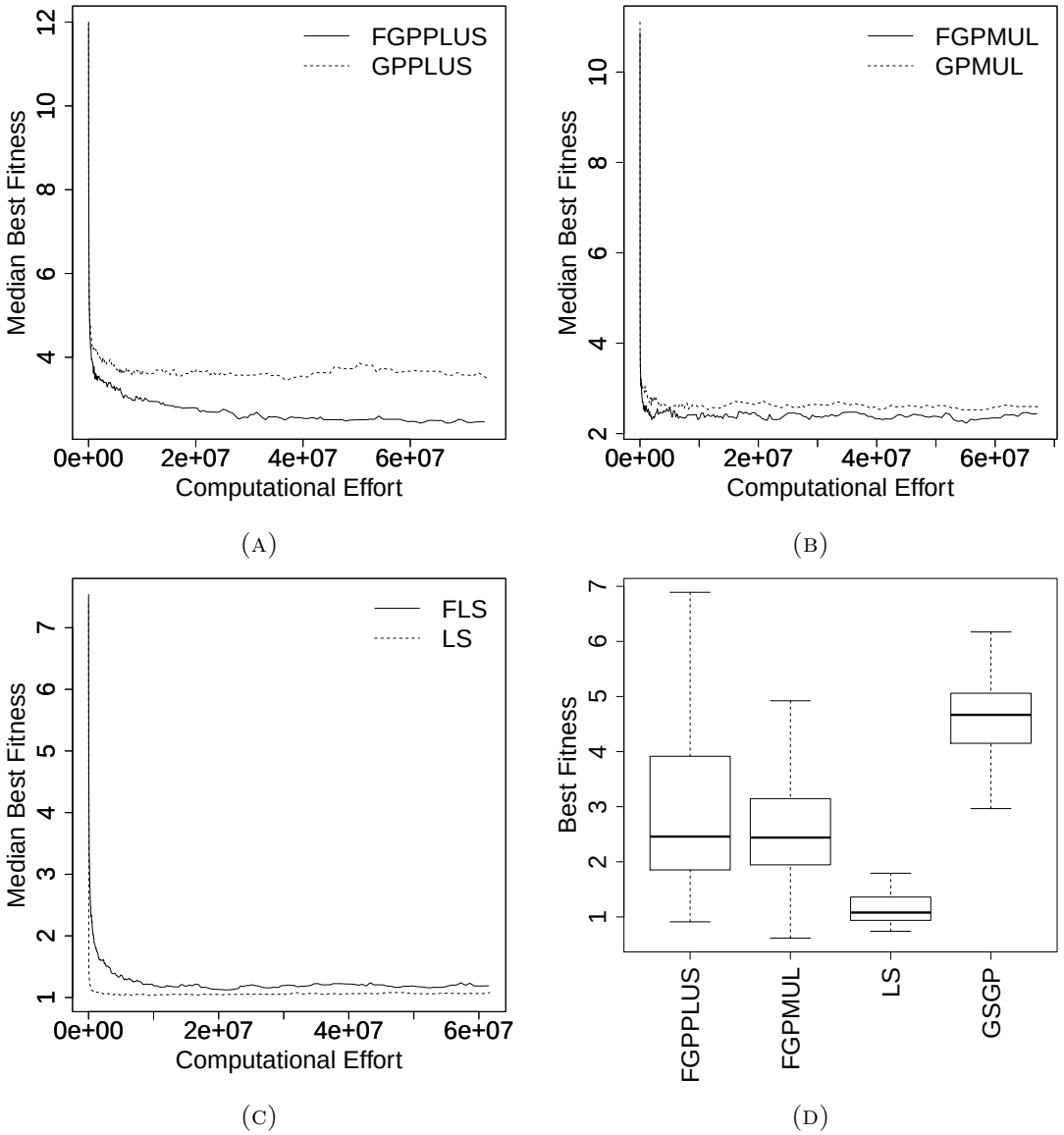
FIGURE 5.12: Dataset *slump* (test). Results are relative to: *GPPLUS* technique (5.12a), *GPMUL* (5.12b), *LS* (5.12c). Figures (5.12a), (5.12b), (5.12c), have median fitness (RMSE) on the vertical axes and computational effort on horizontal axes (calculated as the number of nodes evaluated during training). Figure (5.12d) reports the performance of GSGP as well as the ones achieved by the 3 best variants (i.e., with or without filters) of the proposed system based on equivalence classes.




(A) (B)

(C) (D)

FIGURE 5.13: Dataset *yacht* (training). Results are relative to: *GPPLUS* technique (5.13a), *GPMUL* (5.13b), *LS* (5.13c). Figures (5.13a), (5.13b), (5.13c), have median fitness (RMSE) on the vertical axes and computational effort on horizontal axes (calculated as the number of nodes evaluated during training). Figure (5.13d) reports the performance of GSGP as well as the ones achieved by the 3 best variants (i.e., with or without filters) of the proposed system based on equivalence classes.




(A) (B)

(C) (D)

FIGURE 5.14: Dataset *yacht* (test). Results are relative to: *GPPLUS* technique (5.14a), *GPMUL* (5.14b), *LS* (5.14c). Figures (5.14a), (5.14b), (5.14c), have median fitness (RMSE) on the vertical axes and computational effort on horizontal axes (calculated as the number of nodes evaluated during training). Figure (5.14d) reports the performance of GSGP as well as the ones achieved by the 3 best variants (i.e., with or without filters) of the proposed system based on equivalence classes.

test set: the RMSE values are comparable to the ones achieved on the training set, hence suggesting that the solutions returned by these methods are rather robust. As a partial conclusion, and keeping in mind that all the variants studied outperform standard GP on all the problems considered, 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 have 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 5.1 to 5.14. The $p$-values have been calculated using the Wilcoxon rank-sum test for pairwise data comparison with a significance value of $\alpha = 0.05$. Table 5.2 reports the results of this study.

Table 5.2 shows that, in the large majority of the cases, filtering brings a statistically significant advantage in terms of RMSE. This can be seen by observing that the third and the fifth columns of the table often contain negative values (i.e., lower RMSE obtained by using the filtered variants). The only exceptions are *GPMUL* on the *total* problem and *LS* on the *yacht* problem, where the non-filtered techniques present better RMSE values than the respective filtered counterparts, both on the training and on the test set. It is interesting to point out the case of the *slump* dataset: this is the only test 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. This situation is visible comparing Figures 5.11b and 5.12b.

The effect of using the semantic filters is discussed in more detail here. Specifically, to obtain a better understanding of the advantage related to the use of filters, a wide range of values for the filter parameter (i.e., the parameter that tunes the effect of the filters, as presented in Section 4.1) has been considered. We are interested in understanding how this parameter influences the quality of the generated solutions (in terms of RMSE) and their size (expressed in terms of the average number of nodes of the individuals in the population).

The results are presented in Figures 5.15 to 5.21. In all 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 have been studied. The non-filtered version of each method is also reported: it is

| Technique | Dataset | P-value Training | % diff on training | P-value Test | % diff on test |
|---|---|---|---|---|---|
| **GPMUL** | *airfoil* | 0,00 | **-18,99** | 0,00 | **-21,00** |
| | *concrete* | 0,00 | **-11,52** | 0,00 | **-11,11** |
| | *motor* | 0,02 | **-0,31** | 0,46 | -0,88 |
| | *total* | 0,77 | -0,10 | 0,75 | 0,60 |
| | *protein* | 0,01 | **-1,24** | 0,00 | **-1,41** |
| | *slump* | 0,00 | **32,90** | 0,00 | **-25,12** |
| | *yacht* | 0,12 | -6,58 | 0,16 | -5,67 |
| **GPPLUS** | *airfoil* | 0,00 | **-4,40** | 0,05 | **-3,79** |
| | *concrete* | 0,00 | **-12,60** | 0,00 | **-16,87** |
| | *motor* | 0,01 | **-0,67** | 0,01 | **-1,52** |
| | *total* | 0,00 | **-1,59** | 0,01 | **-0,58** |
| | *protein* | 0,00 | **-1,86** | 0,01 | **-1,65** |
| | *slump* | 0,00 | **-7,58** | 0,00 | **-28,82** |
| | *yacht* | 0,00 | **-22,18** | 0,00 | **-30,75** |
| **LS** | *airfoil* | 0,69 | -1,57 | 0,02 | **-3,02** |
| | *concrete* | 0,66 | -0,56 | 0,27 | -1,90 |
| | *motor* | 0,00 | **-0,67** | 0,10 | -0,91 |
| | *total* | 0,03 | **-0,71** | 0,18 | -0,76 |
| | *protein* | 0,00 | **-1,14** | 0,30 | -1,28 |
| | *slump* | 0,00 | **82,43** | 0,01 | **-1,97** |
| | *yacht* | 0,00 | **10,22** | 0,16 | 10,29 |

**Table 5.2** 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 means and advantage in using filters having an error reduction. For the filtered counterparts, we selected the filter parameter that, among the considered ones, produces the best training performance. Statistics refer to the same experiments reported in Figures 5.1 - 5.14.

the leftmost box on every figure. Plots (b), (d) and (f) show the average number of nodes in the 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 5.15 reports the results for the airfoil problem;
- Figure 5.16 reports the results for the concrete problem;
- Figure 5.17 reports the results for the motor problem;
- Figure 5.18 reports the results for the total problem;
- Figure 5.19 reports the results for the protein problem;
- Figure 5.20 reports the results for the slump problem;
- Figure 5.21 reports the results for the yacht problem;

Considering plots (a), (c) and (e) of these figures, we can see that *FGPMUL* and *FGPPLUS* generally achieve better results than *GPMUL* and *GPPLUS* also for very small values of the filtering 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 the problems, just before a steep increase happens. 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. Increasing the value of the filter parameter above a certain threshold (that is different for every dataset), the error increases and all the 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, it is possible to observe a similar trend, although it is less visible on the test data. The dataset for which the trend is less visible, in the case of *LS*, is *slump*. Still, also for that dataset, it is possible to observe a worsening of the RMSE when the value of $10^3$ is considered for the filtering parameter (see Figure 5.20e). To summarize, filtering improved the performance of all the studied techniques on almost all studied test problems. Interestingly, this holds both on training and unseen test data. Finally, we remark that the advantage of *GPPLUS* and *GPMUL* in using filters is larger than the one of *LS*.

Plots (b), (d) and (f) indicate that filters have also 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, when the values of the filtering parameter are high, the generated individuals tend to be smaller than when they are low. However, it is

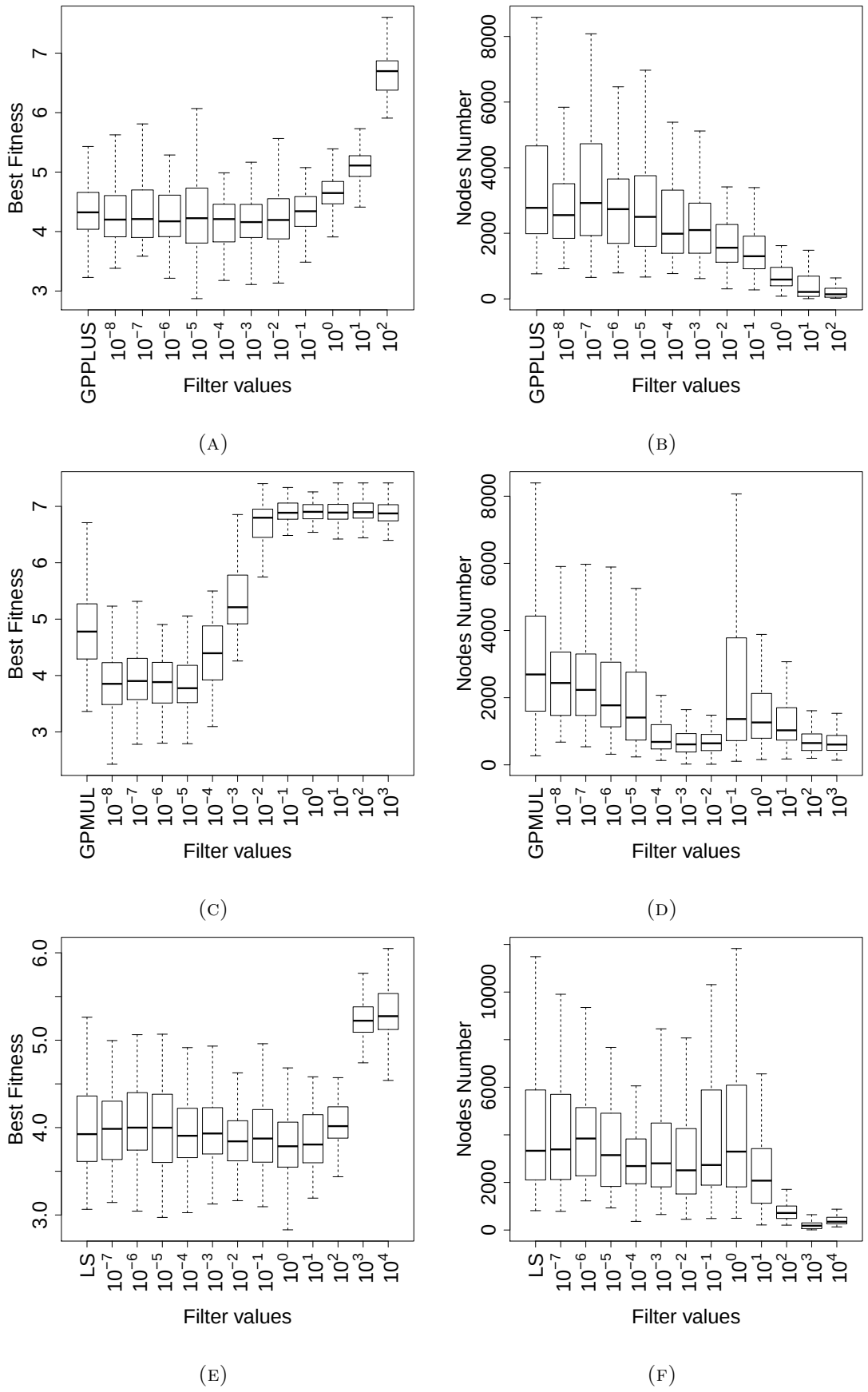



FIGURE 5.15: Dataset *airfoil*. Figures (a), (c) and (e) show test fitness (RMSE) for different values of the filter parameter. The first boxplot of each series shows the results without filtering: *GPPLUS* on figure (a), *GPMUL* on (c) and *LS* on (e). Figures (b), (d) and (f) show the average number of nodes in the population for the corresponding filter settings. Also in this case, the first boxplot of each series shows the results without filtering: *GPPLUS* on figure (b), *GPMUL* on (d) and *LS* on (f).

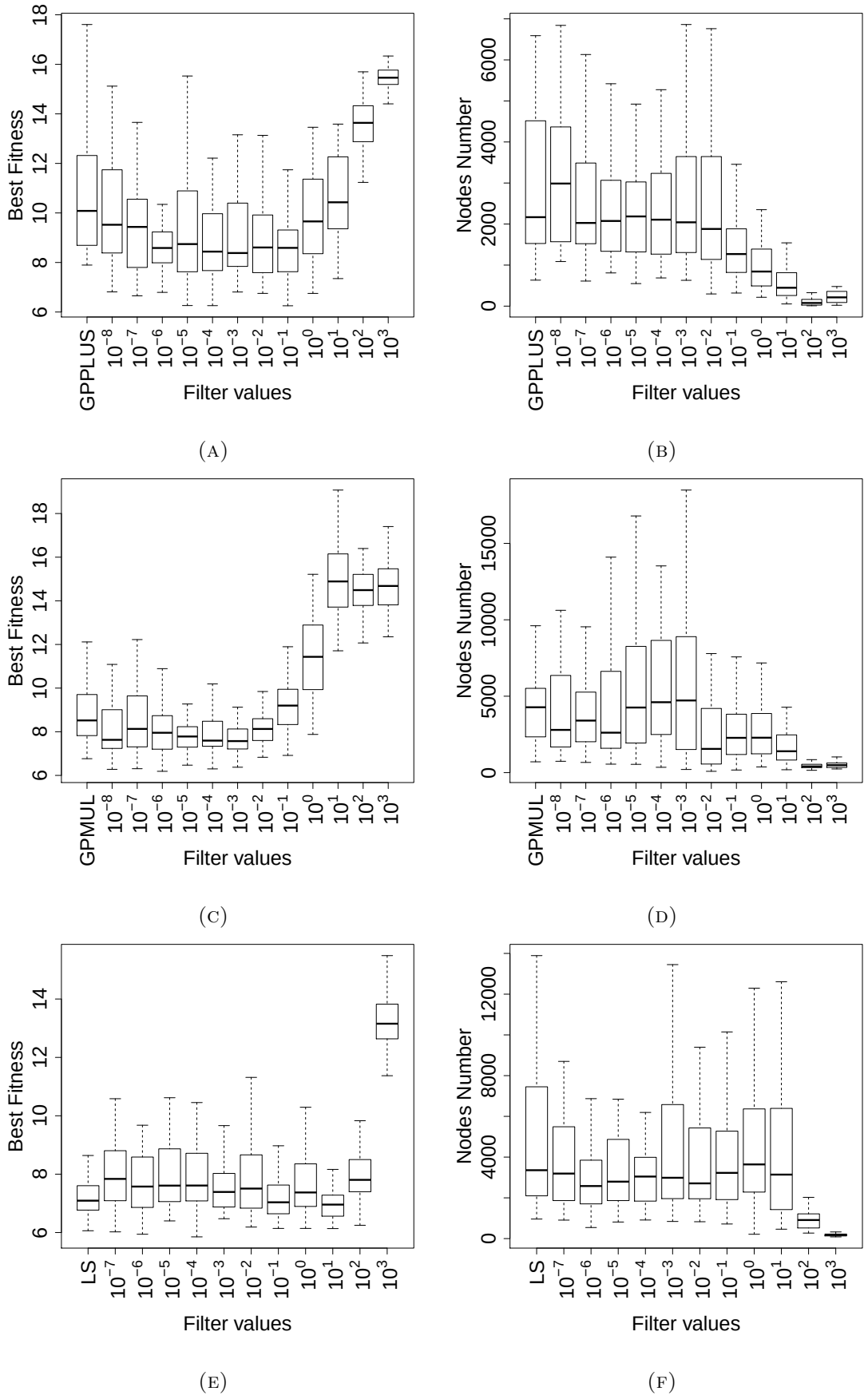



FIGURE 5.16: Dataset *concrete*. Figures (a), (c) and (e) show test fitness (RMSE) for different values of the filter parameter. The first boxplot of each series shows the results without filtering: *GPPLUS* on figure (a), *GPMUL* on (c) and *LS* on (e). Figures (b), (d) and (f) show the average number of nodes in the population for the corresponding filter settings. Also in this case, the first boxplot of each series shows the results without filtering: *GPPLUS* on figure (b), *GPMUL* on (d) and *LS* on (f).

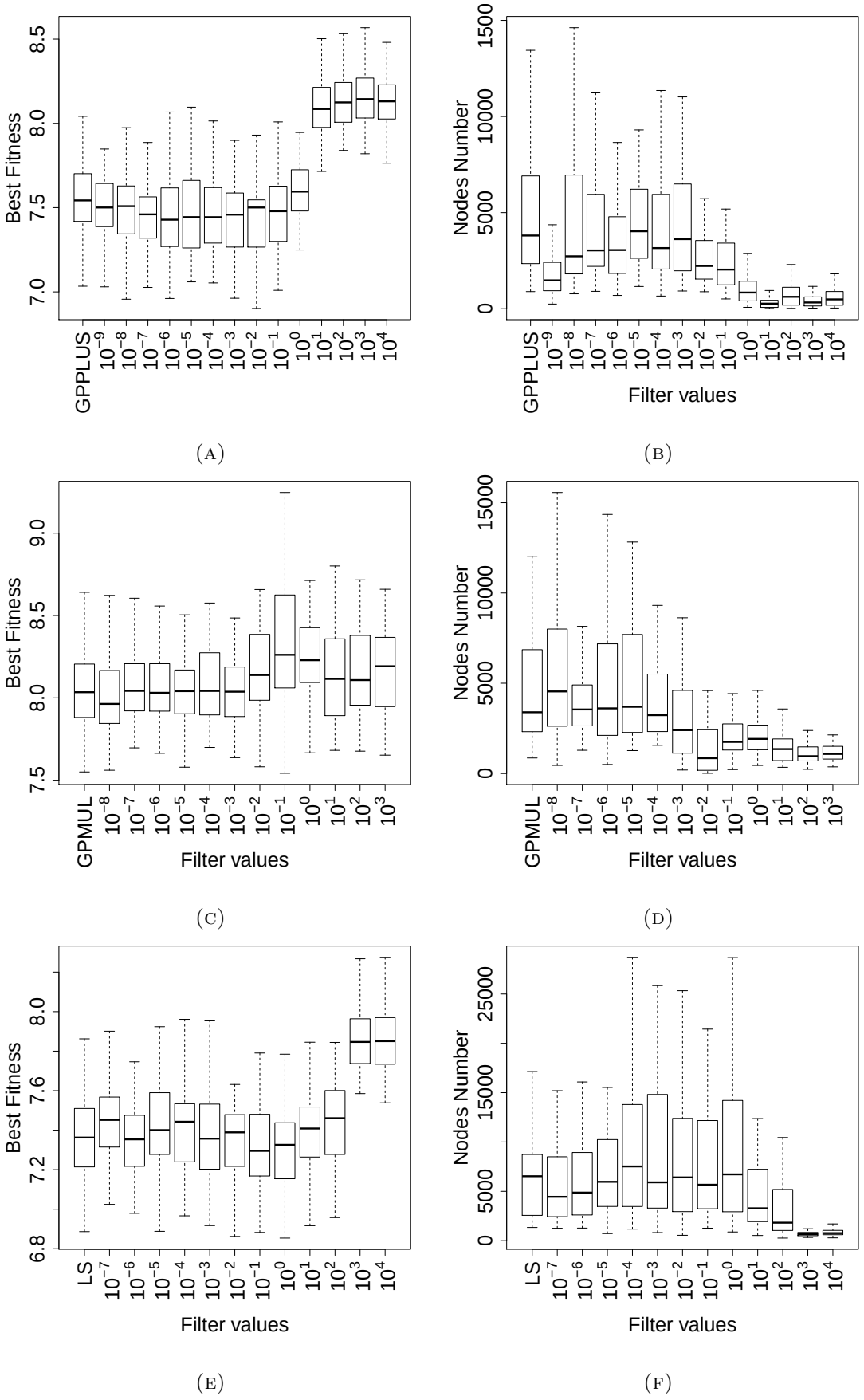



FIGURE 5.17: Dataset *motor*. Figures (a), (c) and (e) show test fitness (RMSE) for different values of the filter parameter. The first boxplot of each series shows the results without filtering: *GPPLUS* on figure (a), *GPMUL* on (c) and *LS* on (e). Figures (b), (d) and (f) show the average number of nodes in the population for the corresponding filter settings. Also in this case, the first boxplot of each series shows the results without filtering: *GPPLUS* on figure (b), *GPMUL* on (d) and *LS* on (f).

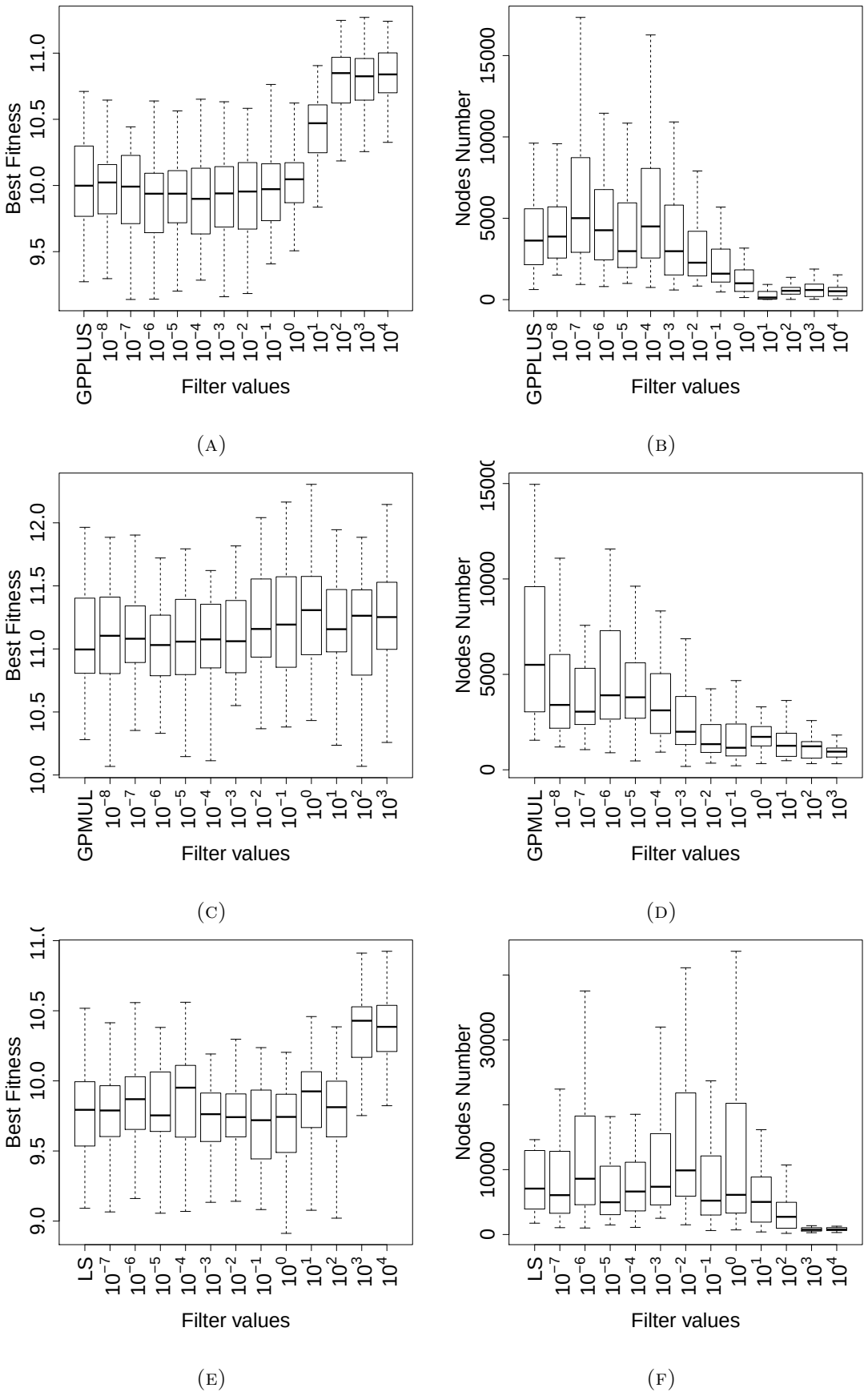



FIGURE 5.18: Dataset *total*. Figures (a), (c) and (e) show test fitness (RMSE) for different values of the filter parameter. The first boxplot of each series shows the results without filtering: *GPPLUS* on figure (a), *GPMUL* on (c) and *LS* on (e). Figures (b), (d) and (f) show the average number of nodes in the population for the corresponding filter settings. Also in this case, the first boxplot of each series shows the results without filtering: *GPPLUS* on figure (b), *GPMUL* on (d) and *LS* on (f).

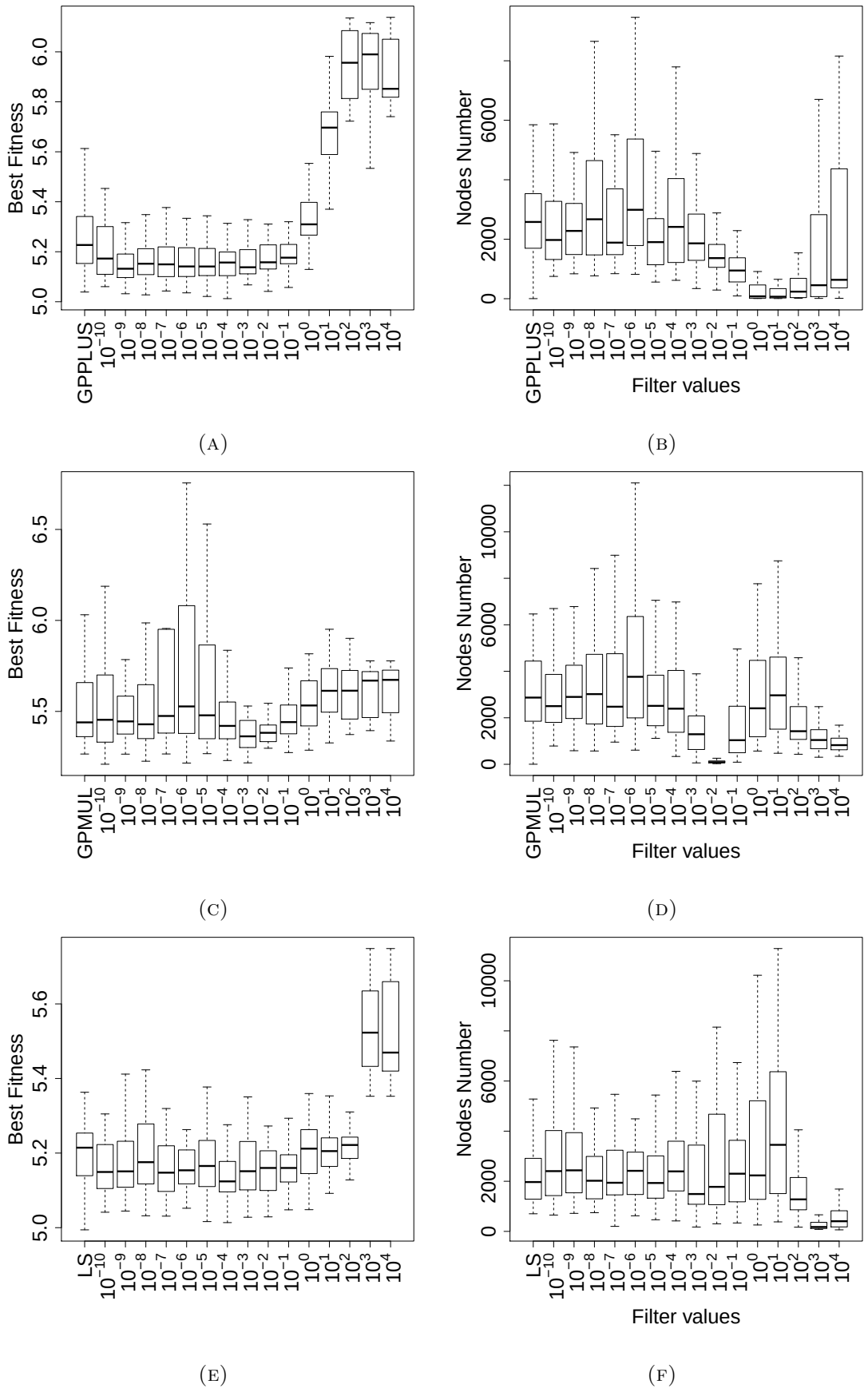



FIGURE 5.19: Dataset *protein*. Figures (a), (c) and (e) show test fitness (RMSE) for different values of the filter parameter. The first boxplot of each series shows the results without filtering: *GPPLUS* on figure (a), *GPMUL* on (c) and *LS* on (e). Figures (b), (d) and (f) show the average number of nodes in the population for the corresponding filter settings. Also in this case, the first boxplot of each series shows the results without filtering: *GPPLUS* on figure (b), *GPMUL* on (d) and *LS* on (f).

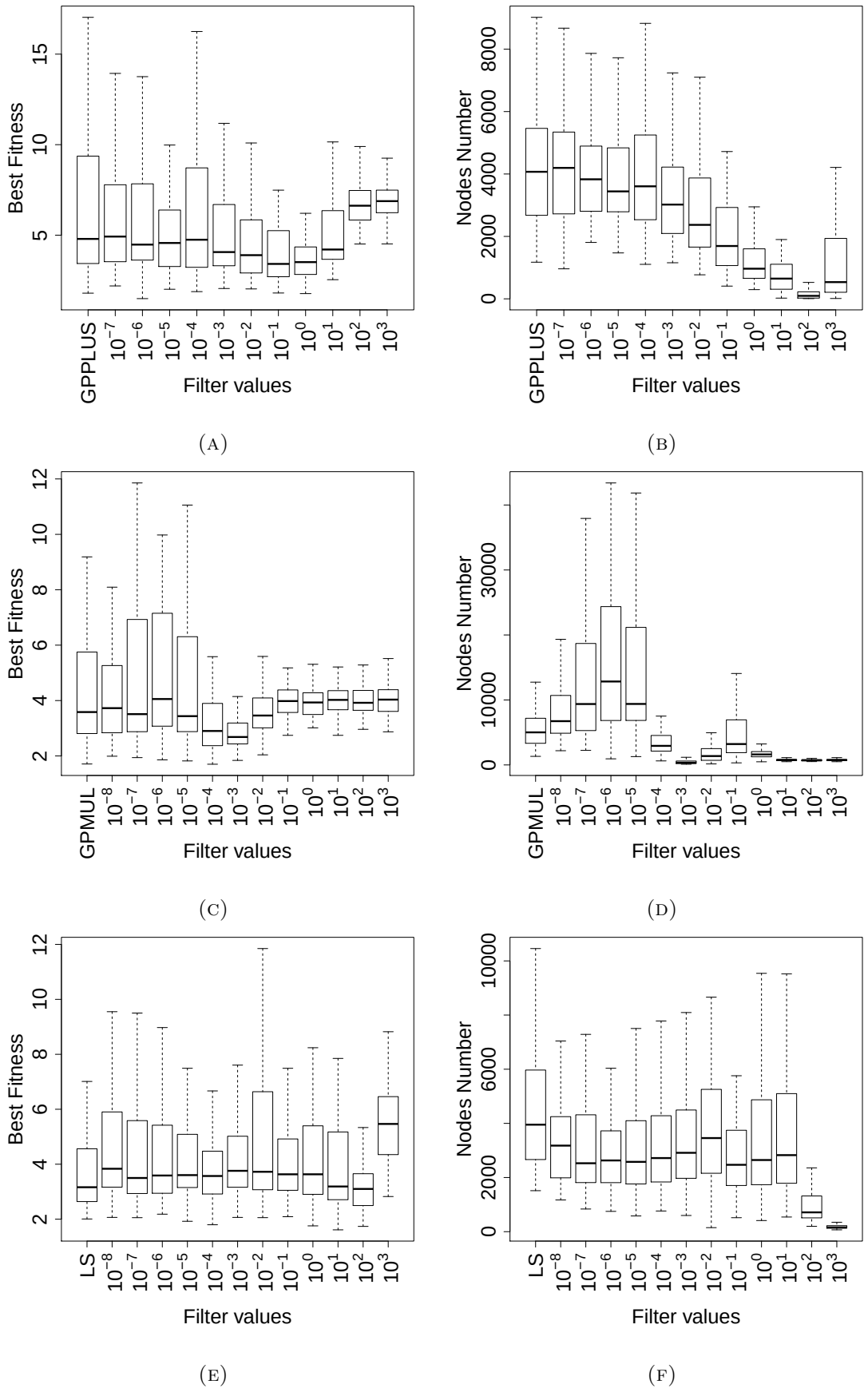



FIGURE 5.20: Dataset *slump*. Figures (a), (c) and (e) show test fitness (RMSE) for different values of the filter parameter. The first boxplot of each series shows the results without filtering: *GPPLUS* on figure (a), *GPMUL* on (c) and *LS* on (e). Figures (b), (d) and (f) show the average number of nodes in the population for the corresponding filter settings. Also in this case, the first boxplot of each series shows the results without filtering: *GPPLUS* on figure (b), *GPMUL* on (d) and *LS* on (f).

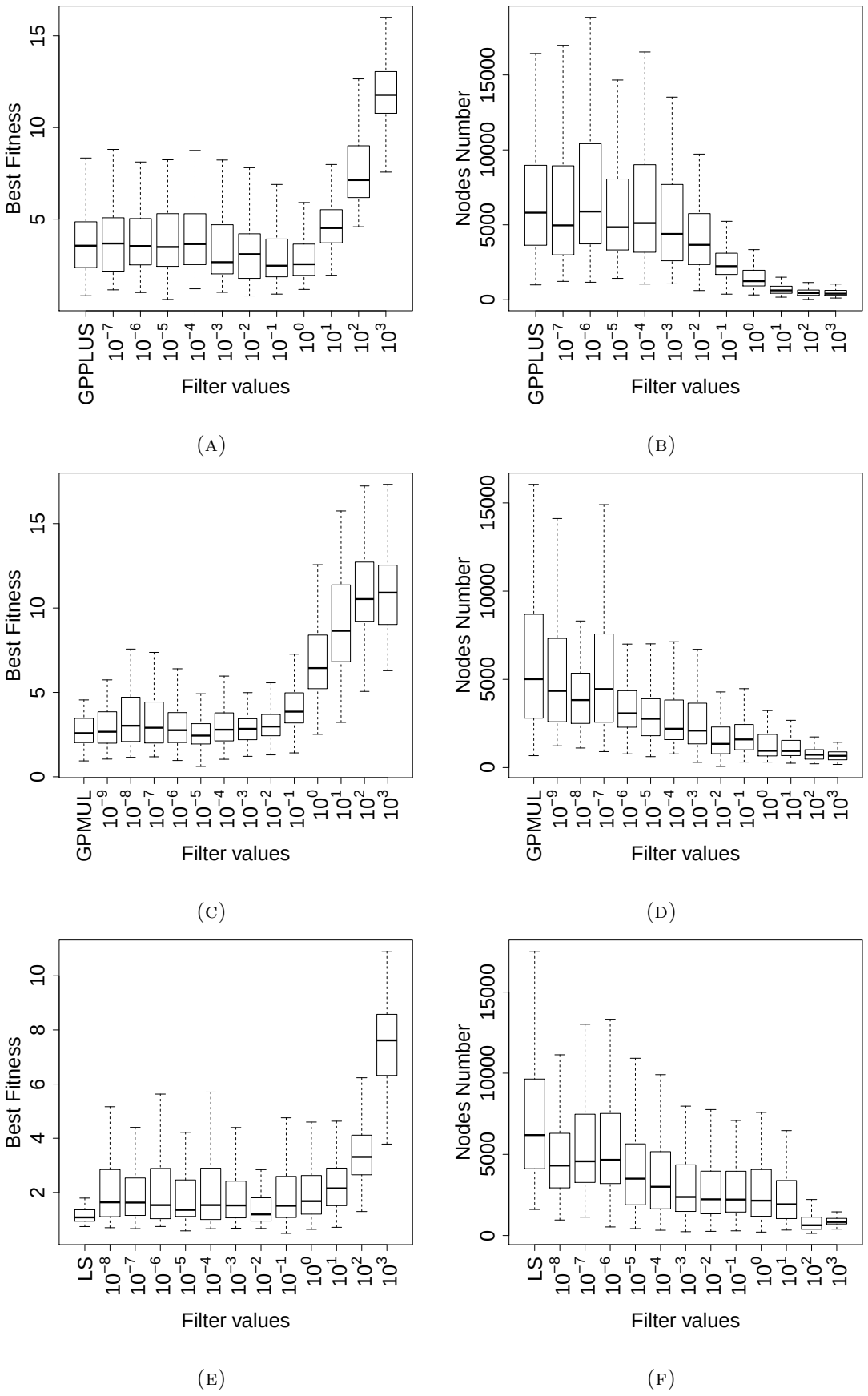



FIGURE 5.21: Dataset *yacht*. Figures (a), (c) and (e) show test fitness (RMSE) for different values of the filter parameter. The first boxplot of each series shows the results without filtering: *GPPLUS* on figure (a), *GPMUL* on (c) and *LS* on (e). Figures (b), (d) and (f) show the average number of nodes in the population for the corresponding filter settings. Also in this case, the first boxplot of each series shows the results without filtering: *GPPLUS* on figure (b), *GPMUL* on (d) and *LS* on (f).

also possible to observe that the size of the individuals does not monotonously decrease with the increase of the filtering parameter. To statistically evaluate the differences in terms of individuals' size between *GPPLUS*, *GPMUL* and *LS* and their filtered counterparts, we consider, for each problem, the best value of the filter parameter (i.e., the value that produces the best RMSE). Both the average size of the individuals in the population and the size of the best individual were studied. The Wilcoxon rank-sum test for pairwise data comparison, employing the significance level $\alpha = 0.05$, has been used under the alternative hypothesis that filtered variants produce individuals with a smaller size than the non-filtered ones. The results of this statistical study are reported, for the three systems, in Table 5.3. The table also shows the differences among filtered and non-filtered systems as a percentage.

It is possible to observe that filtered systems always produce individuals that are smaller than their non-filtered counterparts in a statistically significant way, except for *GPMUL* on the *concrete* and *motor* problems. An important remark is that, for all the systems and problems considered, it is always possible to find a parameter setting that reduces individuals' size and that also has a beneficial effect on the RMSE (as shown in Table 5.2).

| Technique | Dataset | P-value average size | % Diff pop. Average size | P-value Best size | % Diff on best ind. Size |
|---|---|---|---|---|---|
| ***GPMUL*** | *airfoil* | 0,00 | **-47,66** | 0,01 | **-43,03** |
| | *concrete* | 0,59 | 10,38 | 0,53 | 8,63 |
| | *motor* | 0,34 | 33,98 | 0,48 | 45,32 |
| | *total* | 0,00 | **-63,76** | 0,00 | **-63,00** |
| | *protein* | 0,00 | **-54,94** | 0,00 | **-68,11** |
| | *slump* | 0,00 | **-93,31** | 0,00 | **-88,70** |
| | *yacht* | 0,00 | **-44,85** | 0,00 | **-40,39** |
| ***GPPLUS*** | *airfoil* | 0,00 | **-24,39** | 0,00 | **-28,24** |
| | *concrete* | 0,27 | -5,77 | 0,94 | 0,90 |
| | *motor* | 0,06 | -18,88 | 0,05 | **-35,05** |
| | *total* | 0,74 | -17,98 | 0,99 | -15,07 |
| | *protein* | 0,05 | **-26,27** | 0,24 | -13,14 |
| | *slump* | 0,00 | **-58,43** | 0,00 | **-50,16** |
| | *yacht* | 0,00 | **-61,48** | 0,00 | **-57,62** |
| ***LS*** | *airfoil* | 0,00 | **-37,70** | 0,05 | **-19,02** |
| | *concrete* | 0,19 | -6,43 | 0,33 | -10,75 |
| | *motor* | 0,33 | -13,22 | 0,23 | 6,14 |
| | *total* | 0,23 | -26,27 | 0,27 | -27,34 |
| | *protein* | 0,70 | -1,47 | 0,34 | 14,97 |
| | *slump* | 0,00 | **-81,94** | 0,00 | **-81,09** |
| | *yacht* | 0,00 | **-63,96** | 0,00 | **-63,85** |

**Table 5.3** 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), at 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 means and advantage in using filters having a size reduction. For the filtered counterparts, we selected the filter parameter that produces the best training (RMSE) performance. Statistics refer to the same experiments reported in Figures 5.1 - 5.14.

# Chapter 6

# Conclusions and Future Work

In the field of Genetic Programming this thesis proposes a novel framework to enhance the effectiveness of this meta-heuristic techniques. We start analysing the current research with special emphasis on the so called *semantic* methods. Indeed, in our opinion, this techniques represent an important step toward both performances and comprehension of GP in general. Semantic techniques describe the behaviour of the programs representing the solutions by their own output vector when executed on the input test cases. So a semantic vector tell us what that specific solution is computing, different program can compute the same output vectors thus being completely equivalents from a solution approximation point of view.

We show in particular that a branch called Geometric Semantic Genetic Programming (GSGP) has interesting properties from which we were inspired. Interestingly GSGP techniques induce a unimodal fitness landscape that is easy to descend until the target and give rise to really interesting performances in goal approximation. Unfortunately GSGP, in its original form, has an important drawback: the dimension of the solution's structure is almost doubling at each generation resulting in an exponential growth. Even if technical advancement can deal with this issue GSGP remain essentially a black box approach. We discuss the interesting generalization properties of GSGP guaranteed by the geometrical construction and discussed alternatives to the geometric strategies to optimize this particular aspect. Indeed more traditional GP techniques have shown, thanks to extensive experimentations, that generalization is negatively affected by complex solution's structures giving some evidence that the *Occam razor* principle is valid at some extent.

In this regard we give details on a series of techniques recently introduced, in an our previous work. Using a semantic evaluation they do not admit any duplicate in the semantic itself, thus strongly limiting the amount of neutral code that can grow in a solution's

structure. In turn this is keeping the solution's growth low or at least always justified by an advancement in target approximation. Being this approximation constructed using a geometrical alignment in a translated semantic space, the error space, the framework itself has been named ESAGP Error Space Alignments Genetic Programming. One of the interesting point of ESAGP is that maintain semantic diversity at the population level, no duplicate is admitted, even the past population. A key feature of ESAGP is that to discriminate among semantics it is not using a usual metric to compute a distance but rather shows the utility of using a pseudo-distance approach in particular ESAGP relay on *Semantic Angles*. In this thesis we are building on the following characteristics of ESAGP: (i)it uses geometrical properties to reconstruct the target(ii) it shows a good generalization ability thanks to its property of filtering out duplicate semantics and thus keeping solution complexity low (Occam razor principle strongly enforced) (iii) it uses pseudo-distance to compare semantics thus enabling an equivalence class approach: all the semantic vectors aligned in the error space are equivalent from the point of view of target approximation with a geometric approach because they al stay in te same vector subspace. We want to generalize ESAGP's properties (i,ii,iii) keeping in consideration the geometric semantic approach of GSGP and thus we propose a novel flexible framework based on *Equivalence Classes*.

The approach presented in this thesis proposes these original contributions from the theoretical point of view. (i) We show how to generalize the use of pseudo-distance and its usefulness for the comparison of semantic vectors. This flexibility allows also to craft or learn useful equivalence relationships. (ii) We define a new formal concept of duplicate semantics based equivalence classes.(iii) We shows a new effective method of filtering duplicate semantics (iv) thus permitting only a very controlled growth of the solution's structural complexity and a consequent good generalization ability. (v) We simplify the geometric approach in the context of equivalence classes, losing the guaranteed geometric properties (unimodal fitness landscape and good generalization ability) but getting a potentially compact and equally general and accurate solutions.

In this thesis, 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 possible ways. In this work, it was implemented by means of two simple GP systems, called *GPPLUS* and *GPMUL*. Each one of these systems uses a different definition of equivalence. 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. Last but not least, semantic filters have been applied also to a well-known and widely used GP system, like linear scaling. Experiments to test the performance of the systems

proposed have been conducted on seven complex real-life applications. The results obtained can be summarized as follows: when considering the results on unseen test data, on five out of the seven studied test problems, the systems proposed are better than (or comparable to) the state-of-the-art of GP for symbolic regression (i.e. geometric semantic GP). Also, on all the test problems taken into account, filters are beneficial for improving the performance of the systems. Last but not least, the use of filters allows the studied systems to generate individuals that are significantly smaller, compared to their unfiltered counterparts.

In the future, we plan to develop more sophisticated definitions of equivalence. Also, we are currently investigating the concept of "weak" equivalence in which we relax the property of the equivalence for a number of fitness cases. In other words, we could admit that two individuals can stay in the same class of equivalence if their output values are directly proportional on part of the fitness cases, instead of all of them. Last but not least, we plan to develop a novel GP system, based on the concept of semantics-based equivalence classes, which is specialized in solving particular tasks of image processing and pattern recognition.

# Appendix A

# ESAGP

## A.1 Alignment in the Error Space

Let $\mathbf{X} = \{\overrightarrow{x_1}, \overrightarrow{x_2}, ..., \overrightarrow{x_n}\}$ be the set of input data, or fitness cases, of a symbolic regression problem, and $\overrightarrow{t} = [t_1, t_2, ..., t_n]$ the vector of the respective expected output or target values (in other words, for each $i = 1, 2, ..., n$, $t_i$ is the expected output corresponding to input $\overrightarrow{x_i}$). A GP individual (or program) $P$ can be seen as a function that, for each input vector $\overrightarrow{x_i}$ returns the scalar value $P(\overrightarrow{x_i})$. Following [20], we call *semantics* of $P$ to the vector $\overrightarrow{s_P} = [P(\overrightarrow{x_1}), P(\overrightarrow{x_2}), ..., P(\overrightarrow{x_n})]$. This vector can be represented as a point in a $n$-dimensional space, that we call *semantic space*. Remark that the target vector $\overrightarrow{t}$ itself is a point in the semantic space and, in general, it does *not* correspond to the origin of the Cartesian system (except for the very particular and rare case in which the expected output is equal to zero for each fitness case).

We now introduce a new notion, clearly related to the one of semantics, that we call *error vector*. The error vector of a GP individual $P$ is the vector $\overrightarrow{e_P} = \overrightarrow{s_P} - \overrightarrow{t}$. It can be represented as a point in a $n$-dimensional space, that we call *error space* (even though used for different purposes, a similar idea can be found in [**?** ]). Each vector in the semantic space is translated in the error space by subtracting $\overrightarrow{t}$. So, the target is translated in the error space into the origin of the Cartesian system. It is worth noticing that, once we have the error vector of an individual $P$, it is immediate, for instance, to calculate the root mean square error (RMSE) of $P$ on training data (RMSE $= \sqrt{\sum_{i=1}^{n} e_i^2}$, where $e_i$ is the $i^{th}$ coordinate of $\overrightarrow{e_P}$), a measure that is often used as fitness by standard GP in symbolic regression problems (see for instance [21]). We now define a new concept, whose importance will later become clear.

**Definition A.1. (Optimally Aligned Individuals).** Two GP individuals $A$ and $B$ are *optimally aligned* if it exists a scalar constant $k$ such that: $\overrightarrow{e_A} = k \cdot \overrightarrow{e_B}$

In other words, two individuals $A$ and $B$ are said to be optimally aligned if their respective error vectors are directly proportional, with a proportionality constant $k$. The reason why we use the term "aligned" for such individuals becomes clear by looking at Figure A.1(a), where a simple bi-dimensional error space is represented. In this figure, $A$ and $B$ are two optimally aligned individuals: the points that represent their respective error vectors are aligned with each other and with the origin of the Cartesian system.

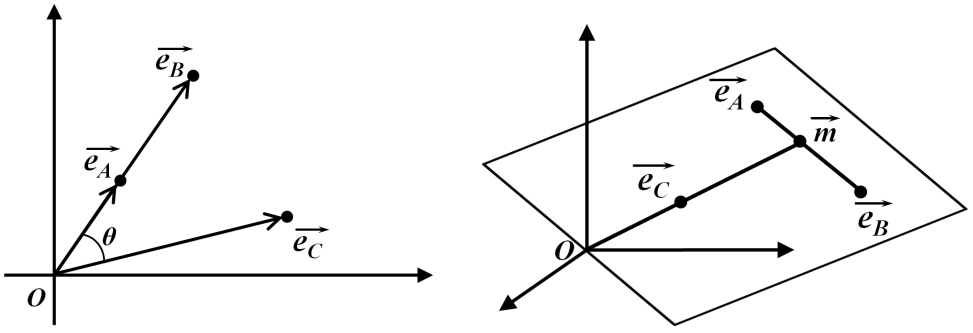



FIGURE A.1: Part (a): Representation of a simple bi-dimensional error space. Individuals $A$ and $B$ are optimally aligned, i.e. their respective error vectors are directly proportional. The angle between the error vector of $A$ (as well as $B$) and the one of $C$ is $\theta$. Part (b): A simple tri-dimensional error space. We point out that it is possible to find a point $m$ that is aligned with the error vectors of any pair of individuals $A$ and $B$ and optimally aligned with a third individual $C$.

The concept of optimally aligned individuals is important in the context of this paper because, given any two optimally aligned individuals, we can obtain a globally optimal solution in a very simple way. Let $A$ and $B$ be two optimally aligned individuals. Then, directly applying Definition A.1, we have $\overrightarrow{e_A} = k \cdot \overrightarrow{e_B}$. Applying the definition of error vector, the previous equation can be rewritten as $\overrightarrow{s_A} - \overrightarrow{t} = k \cdot (\overrightarrow{s_B} - \overrightarrow{t})$, from which it follows that $\overrightarrow{t} = \frac{1}{1-k} \cdot \overrightarrow{s_A} - \frac{k}{1-k} \cdot \overrightarrow{s_B}$. This implies that, if we find two optimally aligned individuals, whose syntactic structure we represent with $A$ and $B$, and if we know the proportionality factor $k$ between their respective error vectors, then individual whose syntactic structure is:

$$P_{opt} = \frac{1}{1-k} \cdot A - \frac{k}{1-k} \cdot B \tag{A.1}$$

has a semantic vector that perfectly corresponds to target $\overrightarrow{t}$, and thus it is a globally optimal solution. Interestingly, this property holds independently from the quality (for instance measured by means of the RMSE) of $A$ and $B$: even two extremely "bad" individuals (in terms of RMSE), if they are optimally aligned, can be used to produce a globally optimal solution. As a direct consequence, the new objective of GP can now be to find two optimally aligned individuals, instead of directly finding a globally optimal solution.

This raises at least the following two questions: (1) How can we use GP to look for a pair of optimally aligned individuals? (2) Is searching for two optimally aligned individuals easier for GP than directly searching for a globally optimal solution? The answer to question (1) is that several different strategies can be adopted. In this paper, which to the best of our knowledge represents the first effort of using GP to look for two optimally aligned individuals, we propose the ESAGP framework introduced in the following sections. Section **??** contains a discussion of possible alternative strategies. In order to answer question (2), we perform experiments where an instance of ESAGP, whose goal is to find a pair of optimally aligned individuals, is compared with ST-GP [21] and with GS-GP [77].

## A.2 One Step Error Space Alignment GP: ESAGP-1

ESAGP-1 is based on the idea that GP should work with the objective of minimizing the *angle* between the error vectors of pairs of individuals (looking for a pair for which this angle is equal to zero). Figure A.1(a) graphically represents the angle $\theta$ between the error vectors of individuals $A$ (as well as $B$) and $C$. Remembering that $\theta = \arccos((\overrightarrow{e_A} \times \overrightarrow{e_C})/(||\overrightarrow{e_A}|| \cdot ||\overrightarrow{e_C}||))$ (where $\times$ represents the scalar product between two vectors and $||\overrightarrow{v}||$ is the Euclidean norm of vector $\overrightarrow{v}$) the angle between the error vectors of two individuals is easy to calculate once we have their semantics. It is worth emphasizing that the objective of ESAGP-1 is to find optimally aligned individuals, regardless of their individual quality (for instance, as measured by the RMSE). To achieve this goal, we follow two ideas: (1) all the individuals found during the evolution, and not only the ones in the population at each generation, can be potential members of an optimally aligned pair; (2) the search cannot be driven by a measure of distance to the target in the semantic space (like the RMSE), but instead by a different fitness function that promotes the discovery of optimally aligned individuals.

To implement idea (1), ESAGP-1 maintains an archive of all the "semantically new" individuals that have been found during the GP run. Every time a new individual $P$ is generated, the algorithm checks whether it is optimally aligned with any of the individuals already in the archive. If it is not, $P$ is added to the archive, unless the archive already contains an individual with the same semantics, and the algorithm continues. Otherwise, the algorithm terminates returning the newly found pair of optimally aligned individuals. In [2] we present experimental results, reporting an RMSE value at each generation for the ESAGP framework. That error is obtained like this: at each generation, we consider the pair of individuals $(A, B)$ such that $A$ belongs to the population and $B$ belongs to the archive, and such that the angle between $\overrightarrow{e_A}$ and $\overrightarrow{e_B}$ is minimum.

Then, we construct the individual that approximates an optimal solution by applying Equation (A.1). In order to do that, we need a value for the scalar constant $k$. Given that $A$ and $B$ are not optimally aligned, $\overrightarrow{e_A}$ and $\overrightarrow{e_B}$ are not perfectly proportional, so $k$ can only be approximated. Let $a_1, a_2, ..., a_n$ be the coordinates of $\overrightarrow{e_A}$ and $b_1, b_2, ..., b_n$ the coordinates of $\overrightarrow{e_B}$. In this work, we use as $k$ the median of the values $a_1/b_1, a_2/b_2, ..., a_n/b_n$. We remark that this RMSE value is only calculated for comparing the results returned by the ESAGP framework with ST-GP and GS-GP. It is never used for selection or in any other way during the evolution.

To implement idea (2), ESAGP-1 uses a fitness function that has no relationship with the distance to the target in the semantic space. To define this new fitness function, ESAGP-1 calculates a particular point in the error space, that we call *center of attraction*, or simply *attractor*. The fitness of an individual is the angle between its error vector and the attractor, and it has to be minimized (in other words, small angles are better than large ones). The attractor must be chosen in such a way to promote the evolution of optimally aligned individuals. Our idea is to choose a point that, informally, represents the majority of the individuals in a population, standing "in the middle of" an area where most of the error vectors of the individuals in the population are found. Therefore, the objective of the algorithm becomes driving the population towards this central point. We use as attractor the following vector: $\overrightarrow{a} = \sum_{P \in \text{Pop}} \overrightarrow{e_P} / ||\overrightarrow{e_P}||$ where Pop is the current population and $||\overrightarrow{v}||$ is the Euclidean norm of vector $\overrightarrow{v}$. In principle, the attractor could be calculated only once in the beginning of the run, using the initial population, or it could change dynamically during the run, for instance recalculating it at each generation (or at prefixed intervals). We have evaluated both alternatives in a set of preliminary experiments. The results suggested that modifying the attractor during the run does not significantly affect the performance of the algorithm. For this reason, in this paper we report the results obtained by fixing the attractor in the beginning of the run, using the individuals in the initial population.

Besides the novel fitness function, another interesting characteristic that distinguishes ESAGP-1 from standard GP is the procedure that forms the pairs of individuals for mating. ESAGP-1 uses a strategy that encourages semantic diversity, that we call *orthogonal coupling*. Let $d$ be the dimension of (i.e. the number of individuals belonging to) the population. Orthogonal coupling works by performing $d$ independent tournaments (using the standard tournament selection algorithm), allowing us to generate a repository of $d$ parents. Subsequently, an iterative process is performed where, at each iteration, one parent $A$ is picked at random and its partner $B$ is chosen as the individual currently in the repository such that the angle between $\overrightarrow{e_A}$ and $\overrightarrow{e_B}$ is the closest to $90°$. $A$ and $B$ are then removed from the repository and the process iterated until the repository is empty. Preliminary tests (not shown) have revealed that orthogonal

coupling does not help the performance of standard GP on the real-life problems tackled here. However, when used with a preliminary implementation of ESAGP-1, it allowed significant improvements. Thus we decided to use orthogonal coupling.

## A.3 Two Steps Error Space Alignment GP: ESAGP-2

The main information we gathered from the experiments performed with ESAGP-1 (whose results are discussed in [2] ) is that searching for two optimally aligned individuals is an easier task than directly searching for a globally optimal individual (at least for the studied problems). This opens an array of new and promising ways of easing the task of GP, and a question naturally arises: given two individuals whose error vectors are *not* aligned with the origin of the Cartesian system, can we still use them to build an optimal solution? Answering positively to this question is the main objective of ESAGP-2. The idea is shown in Figure A.1(b). Let us assume that we have two individuals, like $A$ and $B$ in the figure, for which the straight line joining the error vectors is not aligned with the origin of the Cartesian system. It is possible to find a point $\overrightarrow{m}$ that lies on the straight line joining $\overrightarrow{e_A}$ and $\overrightarrow{e_B}$ and that is aligned with the error vector of another individual $C$ and the origin. This property holds for any three points that lie on a bi-dimensional plane intersecting the origin. This allows us to extend Definition A.1 to the bi-dimensional case.

**Definition A.2. (Optimally Coplanar Individuals).** Three GP individuals $A$, $B$ and $C$ are *optimally coplanar* if the bi-dimensional plane on which $\overrightarrow{e_A}$, $\overrightarrow{e_B}$ and $\overrightarrow{e_C}$ lie also intersects the origin of the Cartesian system in the error space.

Given three optimally coplanar individuals $A$, $B$ and $C$, we can obtain an equation to express target $\overrightarrow{t}$, and consequently we can find a globally optimal solution analytically. In fact, given that $\overrightarrow{e_A}$ and $\overrightarrow{e_B}$ are aligned with each other and with $\overrightarrow{m}$, applying the same reasoning as in Section A.1, we can write: $\overrightarrow{s_A} - \overrightarrow{n} = w \cdot (\overrightarrow{s_B} - \overrightarrow{n})$, where $\overrightarrow{n}$ is the vector that corresponds to $\overrightarrow{m}$ in the semantic space (i.e. $\overrightarrow{m} = \overrightarrow{n} - \overrightarrow{t}$) and $w$ is a scalar constant. Analogously, the following relationship holds between $\overrightarrow{n}$ and the semantics of $C$: $\overrightarrow{n} - \overrightarrow{t} = k \cdot (\overrightarrow{s_C} - \overrightarrow{t})$, where $k$ is a scalar constant. Now we can obtain $\overrightarrow{n}$ from the first equation, replace it in the second one and solve it to obtain $\overrightarrow{t}$. In this way, we find:

$$\overrightarrow{t} = \frac{1}{(1-k)(1-w)} \cdot \overrightarrow{s_A} - \frac{w}{(1-k)(1-w)} \cdot \overrightarrow{s_B} - \frac{k}{1-k} \cdot \overrightarrow{s_C} \tag{A.2}$$

Equation (A.2) can also be written in the following implicit form:

$$w \cdot \overrightarrow{s_B} - w \cdot k \cdot \overrightarrow{s_C} + w \cdot k \cdot \overrightarrow{t} - w \cdot \overrightarrow{t} + k \cdot \overrightarrow{s_C} - k \cdot \overrightarrow{t} + \overrightarrow{t} - \overrightarrow{s_A} = 0 \tag{A.3}$$

Equation (A.3) can be used to find the scalar values $k$ and $w$. In particular, let $a_1$ and $a_2$ be two different coordinates of vector $\overrightarrow{s_A}$. Analogously, let $b_1$ and $b_2$ be the corresponding coordinates of vector $\overrightarrow{s_B}$ and $c_1$ and $c_2$ the corresponding ones in vector $\overrightarrow{s_C}$. We can write the following system of equations:

$$\begin{cases} w \cdot b1 - w \cdot k \cdot c1 + w \cdot k \cdot t1 - w \cdot t1 + k \cdot c1 - k \cdot t1 + t1 - a1 = 0 \\ w \cdot b2 - w \cdot k \cdot c2 + w \cdot k \cdot t2 - w \cdot t2 + k \cdot c2 - k \cdot t2 + t2 - a2 = 0 \end{cases} \tag{A.4}$$

Considering $k$ and $w$ as unknown values, we can solve the system in Equation (A.4), obtaining:

$$k = \frac{a1 \cdot b2 - a2 \cdot b1 - a1 \cdot t2 + a2 \cdot t1 + b1 \cdot t2 - b2 \cdot t1}{a1 \cdot c2 - a2 \cdot c1 - b1 \cdot c2 + b2 \cdot c1 - a1 \cdot t2 + a2 \cdot t1 + b1 \cdot t2 - b2 \cdot t1} \tag{A.5}$$

$$w = \frac{a1 \cdot c2 - a2 \cdot c1 - a1 \cdot t2 + a2 \cdot t1 + c1 \cdot t2 - c2 \cdot t1}{b1 \cdot c2 - b2 \cdot c1 - b1 \cdot t2 + b2 \cdot t1 + c1 \cdot t2 - c2 \cdot t1} \tag{A.6}$$

At this point, the values of $k$ and $w$ obtained in Equations (A.5) and (A.6) can be replaced in Equation (A.2), and this allows us to obtain a globally optimal solution (i.e. an individual whose semantics is exactly equivalent to the target).

In order to write the system in Equation (A.4), we have to choose two particular coordinates of $\overrightarrow{s_A}$, $\overrightarrow{s_B}$ and $\overrightarrow{s_C}$. It is worth pointing out that, if $A$, $B$ and $C$ are optimally coplanar, the obtained $k$ and $w$ are the same independently on the chosen pair of coordinates. But given that this event is quite rare, and more often this situation is only approximated, the choice of the coordinates may be important. Extending the approach used by ESAGP-1, we exhaustively consider all the possible pairs of coordinates; for each one of these pairs, we calculate the values of $k$ and $w$ as in Equations (A.5) and (A.6) and the values used in Equation (A.2) to approximate a globally optimal solution are the medians of these calculated values.

Basically, ESAGP-2 works as ESAGP-1 with the following two major differences: (1) the attractor, this time, is not a straight line, but a bi-dimensional plane; (2) every time a new individual $P$ is generated, it is compared with all the possible *pairs* of individuals in the archive, looking for a pair of individuals that are optimally coplanar with $P$.

To define the attractor (point (1)), ESAGP-2 calculates a plane that informally lies "in the middle" of the error vectors of the individuals in the population. To do this, we use a $k$-means clustering method [78] to partition the error vectors of the initial population into two groups. Then we use the centroids of these two clusters to calculate the attractor. In particular, the attractor is the (unique) bi-dimensional plane that intersects these two centroids and the origin of the Cartesian system in the error space. The implementation of the $k$-means algorithm we have used is the one provided by the

MATLAB environment [79], in which we have used angles, instead of Euclidean distance or other types of distances, for calculating the similarity between the vectors (this is an option that MATLAB provides). All the other parameters of the $k$-means algorithm were set to the default values of MATLAB. Once the attractor is defined, ESAGP-2 uses as fitness the angle between the error vector of the individual and the attractor. In order to calculate an angle between a vector and a plane, we use the SVD (Singular Value Decomposition) method, exactly as presented in [80].

To control whether three optimally coplanar individuals have been found (point (2)), every time a new individual $P$ is generated, all the possible pairs of error vectors in the archive are exhaustively analyzed and the bi-dimensional plane intersecting those vectors and the origin of the Cartesian system is generated. Then we check whether $\overrightarrow{eP}$ also belongs to that plane. Analogously to ESAGP-1, ESAGP-2 terminates if three optimally coplanar individuals are found. Otherwise, if the semantics of $P$ is unique, $P$ is added to the archive and the algorithm continues.

Besides the previously mentioned differences between ESAGP-2 and ESAGP-1, a third one exists: with ESAGP-2 it may happen that the size of the archive grows considerably during the evolution, and the exhaustive analysis of all the pairs may slow down the process excessively (a circumstance that we have never observed for ESAGP-1 in our experiments). Thus, in ESAGP-2 we have limited the maximum size of the archive using a predefined parameter $M$. When the number of individuals in the archive reaches $M$, every time an individual must be added to the archive another one is removed. The version we present in this paper removes the individual that has the largest angle with the attractor. In our experiments, we have empirically observed that a good compromise between computational speed and effectiveness of the method could be obtained by setting $M = 80$. Thus, we have used this value here. However, the influence of the archive size in the overall performance of the system has to be investigated more deeply in the future.

**Generalizing to $\mu$ dimensions.** If we compare the idea that inspired ESAGP-1 (graphically represented in Figure A.1(a)) with the one of ESAGP-2 (represented in Figure A.1(b)), we can informally say that the transition from ESAGP-1 to ESAGP-2 consisted in "adding one dimension": in ESAGP-1 we look for two points that must be aligned on (i.e. must belong to) a straight line intersecting the origin of the Cartesian system; in ESAGP-2 we look for three points that must belong to a bi-dimensional plane intersecting the origin. Interestingly, this last property can also be seen as a composition of the elementary property (alignment) that has to be respected on a straight line in ESAGP-1: $\overrightarrow{e_A}$, $\overrightarrow{e_B}$ and $\overrightarrow{m}$ have to be aligned; $\overrightarrow{m}$, $\overrightarrow{e_C}$ and the origin must be aligned,

too. It is not hard to convince oneself that this process can be iterated (maintaining the same informal terminology, we could say that "more dimensions can be added"), up to a point in which the number of used dimensions is equal to the number of fitness cases of the regression problem we want to solve. In other words, for any $\mu$ between 1 and the number of fitness cases, it is possible to define a GP system whose objective is to find $\mu + 1$ individuals that belong to the same $\mu$-dimensional hyperplane intersecting the origin (a property that can also be seen as the composition of $\mu$ alignments), which we hypothetically call ESAGP-$\mu$. Although in this paper we focus on ESAGP-1 and ESAGP-2, as a first step in this research path, the definition of a general strategy allowing us to obtain an ESAGP-$\mu$ for any possible number of dimensions $\mu$ is an important part of our current research. In that study, the issue of computational complexity has to be carefully considered: both the complexity of the system of equations needed to find the expression of the global optimum and the growth of the archive may become serious problems as $\mu$ increases. Thus, while ESAGP-1 has a computational complexity comparable to ST-GP and the one of ESAGP-2 can be easily controlled by limiting the size of the archive, ESAGP-$\mu$, for large values of $\mu$, may turn out to have a large computational cost, and implementation strategies to reduce it may be necessary.

✎ Devo mettere anche i risultatai sperimentali del vecchio paper ?

# Bibliography


[1] Stefano Ruberto, Leonardo Vanneschi, and Mauro Castelli. Genetic programming with semantic equivalence classes. *Swarm and Evolutionary Computation, currently under evaluation*, 2017.

[2] Stefano Ruberto, Leonardo Vanneschi, Mauro Castelli, and Sara Silva. Esagp–a semantic gp framework based on alignment in the error space. In *Genetic Programming*, pages 150–161. Springer, 2014.

[3] T. Back. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, 1996. ISBN 9780195099713.

[4] W.B. Langdon and R. Poli. *Foundations of Genetic Programming*. Springer, 2002. ISBN 9783540424512.

[5] S.M. Gustafson. *An analysis of diversity in genetic programming*. PhD thesis, University of Nottingham, 2004.

[6] Justinian P. Rosca. Genetic programming exploratory power and the discovery of functions. In John Robert McDonnell, Robert G. Reynolds, and David B. Fogel, editors, *Evolutionary Programming IV Proceedings of the Fourth Annual Conference on Evolutionary Programming*, pages 719–736, San Diego, CA, USA, 1-3 March 1995. MIT Press. ISBN 0-262-13317-2. URL ftp://ftp.cs.rochester.edu/pub/u/rosca/gp/95.ep.exploratory.ps.gz.

[7] Justinian P. Rosca. Entropy-driven adaptive representation. In Justinian P. Rosca, editor, *Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications*, pages 23–32, Tahoe City, California, USA, 9 July 1995. URL ftp://ftp.cs.rochester.edu/pub/u/rosca/gp/95.ml.gpw.ps.gz.

[8] N. McPhee, B. Ohs, and T. Hutchison. Semantic building blocks in genetic programming. *Genetic Programming*, pages 134–145, 2008.

[9] Krzysztof Krawiec and Pawel Lichocki. Approximating geometric crossover in semantic space. In Guenther Raidl, Franz Rothlauf, Giovanni Squillero, Rolf Drechsler, Thomas Stuetzle, Mauro Birattari, Clare Bates Congdon, Martin Middendorf, Christian Blum, Carlos Cotta, Peter Bosman, Joern Grahl, Joshua Knowles, David Corne, Hans-Georg Beyer, Ken Stanley, Julian F. Miller, Jano van Hemert, Tom Lenaerts, Marc Ebner, Jaume Bacardit, Michael O'Neill, Massimiliano Di Penta, Benjamin Doerr, Thomas Jansen, Riccardo Poli, and Enrique Alba, editors, *GECCO '09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 987–994, Montreal, 8-12 July 2009. ACM. doi: doi:10.1145/1569901.1570036.

[10] Lawrence Beadle and Colin G. Johnson. Semantic analysis of program initialisation in genetic programming. *Genetic Programming and Evolvable Machines*, 10(3):307–337, September 2009. ISSN 1389-2576. doi: doi:10.1007/s10710-009-9082-5. URL http://www.springerlink.com/content/yn5p45723l6tr487.

[11] D. Jackson. Phenotypic diversity in initial genetic programming populations. *Genetic Programming*, pages 98–109, 2010.

[12] D. Jackson. Promoting phenotypic diversity in genetic programming. *Parallel Problem Solving from Nature–PPSN XI*, pages 472–481, 2011.

[13] L. Beadle and C.G. Johnson. Semantically driven crossover in genetic programming. In *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 111–116. IEEE, 2008.

[14] S. Gustafson, E.K. Burke, and N. Krasnogor. On improving genetic programming for symbolic regression. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 1, pages 912–919. IEEE, 2005.

[15] Q. Nguyen, X. Nguyen, and M. O Neill. Semantic aware crossover for genetic programming: the case for real-valued function regression. *Genetic Programming*, pages 292–302, 2009.

[16] Nguyen Quang Uy, Nguyen Xuan Hoai, Michael O'Neill, R. I. McKay, and Edgar Galvan-Lopez. Semantically-based crossover in genetic programming: application to real-valued symbolic regression. *Genetic Programming and Evolvable Machines*, 12(2):91–119, June 2011. ISSN 1389-2576. doi: doi:10.1007/s10710-010-9121-2.

[17] R. Poli, L. Vanneschi, W.B. Langdon, and N.F. McPhee. Theoretical results in genetic programming: the next ten years? *Genetic Programming and Evolvable Machines*, 11(3):285–320, 2010.

[18] J. Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465 – 471, 1978. ISSN 0005-1098. doi: 10.1016/0005-1098(78)90005-5. URL http://www.sciencedirect.com/science/article/pii/0005109878900055.

[19] Alberto Moraglio, Krzysztof Krawiec, and Colin Johnson. Geometric semantic genetic programming. In Christian Igel, Per Kristian Lehre, and Carsten Witt, editors, *The 5th workshop on Theory of Randomized Search Heuristics, ThRaSH'2011*, Copenhagen, Denmark, July 8-9 2011. URL http://www.thrash-workshop.org/slides/moraglio.pdf.

[20] Alberto Moraglio, Krzysztof Krawiec, and Colin G. Johnson. Geometric semantic genetic programming. In Carlos A. Coello Coello, Vincenzo Cutello, Kalyanmoy Deb, Stephanie Forrest, Giuseppe Nicosia, and Mario Pavone, editors, *Parallel Problem Solving from Nature, PPSN XII (part 1)*, volume 7491 of *LNCS*, pages 21–31. Springer, 2012.

[21] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

[22] Alan M Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.

[23] Riccardo Poli and John Koza. *Genetic Programming*. Springer, 2014.

[24] JohnR. Koza. Genetic programming as a means for programming computers by natural selection. *Statistics and Computing*, 4:87–112, 1994. ISSN 0960-3174. doi: 10.1007/BF00175355.

[25] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, 1 edition, January 1989. ISBN 0201157675. URL http://www.worldcat.org/isbn/0201157675.

[26] P. Nordin, W. Banzhaf, et al. Evolving turing-complete programs for a register machine with self-modifying code. In *Genetic algorithms: proceedings of the sixth international conference (ICGA95)*, pages 318–325. Pittsburgh, PA, USA, 1995.

[27] L. Spector, J. Klein, and M. Keijzer. The push3 execution stack and the evolution of control. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 1689–1696. ACM, 2005.

[28] Riccardo Poli, William B Langdon, Nicholas F McPhee, and John R Koza. *A field guide to genetic programming*. Lulu. com, 2008.

[29] J.R. Koza. A genetic approach to the truck backer upper problem and the intertwined spiral problem. In *Neural Networks, 1992. IJCNN., International Joint Conference on*, volume 4, pages 310–318. IEEE, 1992.

[30] P. Nordin, W. Banzhaf, et al. Complexity compression and evolution. In *Genetic Algorithms: Proceedings of the Sixth International Conference (ICGA95)*, pages 310–317, 1995.

[31] Peter Nordin, Frank Francone, and Wolfgang Banzhaf. Explicitly defined introns and destructive crossover in genetic programming. In Justinian P. Rosca, editor, *Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications*, pages 6–22, Tahoe City, California, USA, 9 July 1995. URL http://citeseer.ist.psu.edu/nordin95explicitly.html.

[32] Peter John Angeline. Genetic programming and emergent intelligence. In Kenneth E. Kinnear, Jr., editor, *Advances in Genetic Programming*, chapter 4, pages 75–98. MIT Press, 1994. URL http://citeseer.ist.psu.edu/187189.html.

[33] Terence Soule and James A. Foster. Removal bias: a new cause of code growth in tree based evolutionary programming. In *1998 IEEE International Conference on Evolutionary Computation*, pages 781–786, Anchorage, Alaska, USA, 5-9 May 1998. IEEE Press. ISBN 0-7803-4869-9. doi: doi:10.1109/ICEC.1998.700151. URL http://citeseer.ist.psu.edu/313655.html.

[34] W. B. Langdon and R. Poli. Fitness causes bloat. Technical Report CSRP-97-09, University of Birmingham, School of Computer Science, Birmingham, B15 2TT, UK, 24 February 1997. URL ftp://ftp.cs.bham.ac.uk/pub/tech-reports/1997/CSRP-97-09.ps.gz.

[35] Riccardo Poli, William B Langdon, and Stephen Dignum. On the limiting distribution of program sizes in tree-based genetic programming. In *European Conference on Genetic Programming*, pages 193–204. Springer, 2007.

[36] S. Dignum and R. Poli. Generalisation of the limiting distribution of program sizes in tree-based genetic programming and analysis of its effects on bloat. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1588–1595. ACM, 2007.

[37] T.M. Mitchell et al. Machine learning, 1997.

[38] S. Silva, S. Dignum, and L. Vanneschi. Operator equalisation for bloat free genetic programming and a survey of bloat control methods. *Genetic Programming and Evolvable Machines*, 13(2):197–238, 2012.

[39] N.F. McPhee and N.J. Hopper. Analysis of genetic diversity through population history. In *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2, pages 1112–1120. Citeseer, 1999.

[40] N.T. Hien and N.X. Hoai. A brief overview of population diversity measures in genetic programming. In *Proc. 3rd Asian-Pacific Workshop on Genetic Programming, Hanoi, Vietnam*, pages 128–139. Citeseer, 2006.

[41] Una-May O'Reilly. Using a distance metric on genetic programs to understand genetic operators. In *IEEE International Conference on Systems, Man, and Cybernetics, Computational Cybernetics and Simulation*, volume 5, pages 4092–4097, Orlando, Florida, USA, 12-15 October 1997. ISBN 0-7803-4053-1. URL http://ieeexplore.ieee.org/iel4/4942/13793/00637337.pdf.

[42] Edwin D. de Jong, Richard A. Watson, and Jordan B. Pollack. Reducing bloat and promoting diversity using multi-objective methods. In Lee Spector, Erik D. Goodman, Annie Wu, W. B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 11–18, San Francisco, California, USA, 7-11 July 2001. Morgan Kaufmann. ISBN 1-55860-774-9. URL http://www.demo.cs.brandeis.edu/papers/rbpd_gecco01.pdf.

[43] Aniko Ekart and S. Z. Nemeth. A metric for genetic programs and fitness sharing. In Riccardo Poli, Wolfgang Banzhaf, William B. Langdon, Julian F. Miller, Peter Nordin, and Terence C. Fogarty, editors, *Genetic Programming, Proceedings of EuroGP'2000*, volume 1802 of *LNCS*, pages 259–270, Edinburgh, 15-16 April 2000. Springer-Verlag. ISBN 3-540-67339-3. URL http://www.sztaki.hu/~ekart/new_metric.ps.

[44] Maarten Keijzer. Efficiently representing populations in genetic programming. In Peter J. Angeline and K. E. Kinnear, Jr., editors, *Advances in Genetic Programming 2*, chapter 13, pages 259–278. MIT Press, Cambridge, MA, USA, 1996. ISBN 0-262-01158-1. URL http://cisnet.mit.edu/Advances-in-Genetic-Programming/276.

[45] M. Looks. On the behavioral diversity of random programs. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1636–1642. ACM, 2007.

[46] L. Beadle and C.G. Johnson. Semantically driven mutation in genetic programming. In *Evolutionary Computation, 2009. CEC 09. IEEE Congress on*, pages 1336–1342. IEEE, 2009.

[47] Leonardo Vanneschi, Mauro Castelli, Luca Manzoni, and Sara Silva. A new implementation of geometric semantic GP and its application to problems in pharmacokinetics. In Krzysztof Krawiec, Alberto Moraglio, Ting Hu, A. Sima Uyar, and Bin Hu, editors, *Proceedings of the 16th European Conference on Genetic Programming, EuroGP 2013*, volume 7831 of *LNCS*, pages 205–216, Vienna, Austria, 3-5 April 2013. Springer Verlag.

[48] Ivo Gonçalves, Sara Silva, and Carlos M. Fonseca. *On the Generalization Ability of Geometric Semantic Genetic Programming*, pages 41–52. Springer International Publishing, Cham, 2015. ISBN 978-3-319-16501-1. doi: 10.1007/978-3-319-16501-1_4. URL http://dx.doi.org/10.1007/978-3-319-16501-1_4.

[49] Mauro Castelli, Davide Castaldi, Ilaria Giordani, Sara Silva, Leonardo Vanneschi, Francesco Archetti, and Davide Maccagnola. An efficient implementation of geometric semantic genetic programming for anticoagulation level prediction in pharmacogenetics. In *16th Portuguese Conference on Artificial Intelligence (EPIA 2013)*. Springer, May 2013.

[50] Alberto Moraglio. An efficient implementation of gsgp using higher-order functions and memoization. In *SMGP workshop at PPSN*, 2014.

[51] Mauro Castelli, Sara Silva, and Leonardo Vanneschi. A c++ framework for geometric semantic genetic programming. *Genetic Programming and Evolvable Machines*, 16(1):73–81, 2015.

[52] Tomasz P Pawlak. Combining semantically-effective and geometric crossover operators for genetic programming. In *International Conference on Parallel Problem Solving from Nature*, pages 454–464. Springer, 2014.

[53] Krzysztof Krawiec and Tomasz Pawlak. Locally geometric semantic crossover: a study on the roles of semantics and homology in recombination operators. *Genetic Programming and Evolvable Machines*, 14(1):31–63, 2013.

[54] Tomasz P Pawlak. Competent algorithms for geometric semantic genetic programming. *review. PhD thesis. Poznan, Poland: Poznan University of Technology*, 2015.

[55] Tomasz P Pawlak, Bartosz Wieloch, and Krzysztof Krawiec. Semantic backpropagation for designing search operators in genetic programming. *IEEE Transactions on Evolutionary Computation*, 19(3):326–340, 2015.

[56] Quang Uy Nguyen, Tuan Anh Pham, Xuan Hoai Nguyen, and James McDermott. Subtree semantic geometric crossover for genetic programming. *Genetic Programming and Evolvable Machines*, 17(1):25–53, 2016. ISSN 1573-7632. doi: 10.1007/s10710-015-9253-5. URL http://dx.doi.org/10.1007/s10710-015-9253-5.

[57] Tomasz P Pawlak, Bartosz Wieloch, and Krzysztof Krawiec. Review and comparative analysis of geometric semantic crossovers. *Genetic Programming and Evolvable Machines*, 16(3):351–386, 2015.

[58] Candida Ferreira. Genetic representation and genetic neutrality in gene expression programming. *Advances in Complex Systems*, 5(04):389–408, 2002.

[59] Marcin Szubert, Anuradha Kodali, Sangram Ganguly, Kamalika Das, and Josh C Bongard. Reducing antagonism between behavioral diversity and fitness in semantic genetic programming. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference*, pages 797–804. ACM, 2016.

[60] Joel Lehman and Kenneth O Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation*, 19(2):189–223, 2011.

[61] John R Koza. Hierarchical automatic function definition in genetic programming. In *FOGA*, pages 297–318, 1992.

[62] Una-May O'Reilly and Franz Oppacher. The troubling aspects of a building block hypothesis for genetic programming. In *FOGA*, pages 73–88, 1994.

[63] Justinian P. Rosca. Analysis of complexity drift in genetic programming. In *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 286–294. Morgan Kaufmann, 1997.

[64] Riccardo Poli and William B. Langdon. Schema theory for genetic programming with one-point crossover and point mutation. *Evol. Comput.*, 6(3):231–252, September 1998. ISSN 1063-6560. doi: 10.1162/evco.1998.6.3.231. URL http://dx.doi.org/10.1162/evco.1998.6.3.231.

[65] Riccardo Poli. General schema theory for genetic programming with subtree-swapping crossover. In *European Conference on Genetic Programming*, pages 143–159. Springer, 2001.

[66] Nicholas Freitag McPhee and Riccardo Poli. Using schema theory to explore interactions of multiple operators. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, pages 853–860. Morgan Kaufmann Publishers Inc., 2002.

[67] Riccardo Poli and Nicholas Freitag McPhee. General schema theory for genetic programming with subtree-swapping crossover: Part ii. *Evolutionary Computation*, 11(2):169–206, 2003.

[68] Kumara Sastry, Una-May OReilly, David E Goldberg, and David Hill. Building-block supply in genetic programming. In *Genetic Programming Theory and Practice*, pages 137–154. Springer, 2003.

[69] Ivo Goncalves, Sara Silva, Carlos M. Fonseca, and Mauro Castelli. Arbitrarily close alignments in the error space: a geometric semantic genetic programming approach. In Tobias Friedrich, Frank Neumann, Andrew M. Sutton, Martin Middendorf, Xiaodong Li, Emma Hart, Mengjie Zhang, Youhei Akimoto, Peter A. N. Bosman, Terry Soule, Risto Miikkulainen, Daniele Loiacono, Julian Togelius, Manuel Lopez-Ibanez, Holger Hoos, Julia Handl, Faustino Gomez, Carlos M. Fonseca, Heike Trautmann, Alberto Moraglio, William F. Punch, Krzysztof Krawiec, Zdenek Vasicek, Thomas Jansen, Jim Smith, Simone Ludwig, JJ Merelo, Boris Naujoks, Enrique Alba, Gabriela Ochoa, Simon Poulding, Dirk Sudholt, and Timo Koetzing, editors, *GECCO 2016 Companion Volume*, pages 99–100, Denver, USA, 20-24 July 2016. ACM. doi: doi:10.1145/2908961.2908988.

[70] Maarten Keijzer. Improving symbolic regression with interval arithmetic and linear scaling. In *Genetic programming*, pages 70–82. Springer, 2003.

[71] T. Brooks, D. Pope, and A. Marcolini. Airfoil self-noise and prediction. *Technical report, NASA RP-1218*, 1989.

[72] Mauro Castelli, Leonardo Vanneschi, and Sara Silva. Prediction of high performance concrete strength using genetic programming with geometric semantic genetic operators. *Expert Systems with Applications*, 40(17):6856–6862, 2013.

[73] Mauro Castelli, Leonardo Vanneschi, and Sara 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 (10):4608 – 4616, 2014. ISSN 0957-4174.

[74] Mauro Castelli, Leonardo Vanneschi, Luca Manzoni, and Aleš Popovič. Semantic genetic programming for fast and accurate data knowledge discovery. *Swarm and Evolutionary Computation*, 26:1–7, 2016.

[75] I-C Yeh. Simulation of concrete slump using neural networks. *Proceedings of the Institution of Civil Engineers-Construction Materials*, 162(1):11–18, 2009.

[76] I. Ortigosa, R. López, and J. García. A neural networks approach to residuary resistance of sailing yachts prediction. *Proceedings of the International Conference on Marine Engineering MARINE*, 2007:250, 2007.

[77] Leonardo Vanneschi, Mauro Castelli, Luca Manzoni, and Sara Silva. A new implementation of geometric semantic GP and its application to problems in pharmacokinetics. In Krzysztof Krawiec, Alberto Moraglio, Ting Hu, A. Sima Uyar, and Bin Hu, editors, *Proceedings of EuroGP 2013*, LNCS, pages 205–216. Springer, 2013.

[78] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[79] MATLAB. *version 7.10.0 (R2010a)*. The MathWorks Inc., Natick, Massachusetts, 2010.

[80] Perake Wedin. On angles between subspaces of a finite dimensional inner product space. In Bo Kagstrom and Axel Ruhe, editors, *Matrix Pencils*, volume 973 of *Lecture Notes in Mathematics*, pages 263–285. Springer Berlin Heidelberg, 1983. ISBN 978-3-540-11983-8. doi: 10.1007/BFb0062107. URL http://dx.doi.org/10.1007/BFb0062107.