Pruning Techniques for Mixed Ensembles of Genetic Programming Models

Mauro Castelli[1], Ivo Gonçalves[2,3], Luca Manzoni[4], Leonardo Vanneschi[1]

[1] NOVA IMS, Universidade Nova de Lisboa, 1070-312 Lisboa, Portugal
{mcastelli,lvanneschi}@novaims.unl.pt

[2] INESC Coimbra, DEEC, University of Coimbra, Pólo 2, 3030-290 Coimbra, Portugal
icpg@dei.uc.pt

[3] CISUC, Department of Informatics Engineering, University of Coimbra, 3030-290 Coimbra, Portugal
[4] Dipartimento di Informatica, Sistemistica e Comunicazione, Universit`a degli Studi di Milano-Bicocca, 20126 Milano, Italy
luca.manzoni@disco.unimib.it

A Multiple Expression Alignment Framework for Genetic Programming

Leonardo Vanneschi, Kristen Scott, and Mauro Castelli

NOVA IMS, Universidade Nova de Lisboa, 1070-312 Lisboa, Portugal
{lvanneschi,kscott,mcastelli}@novaims.unl.pt

Abstract. Alignment in the error space is a recent idea to exploit semantic awareness in genetic programming. In a previous contribution, the concepts of optimally aligned and optimally coplanar individuals were introduced, and it was shown that given optimally aligned, or optimally coplanar, individuals, it is possible to construct a globally optimal solution analytically. As a consequence, genetic programming methods, aimed at searching for optimally aligned, or optimally coplanar, individuals were introduced. In this paper, we critically discuss those methods, analyzing their major limitations and we propose new genetic programming systems aimed at overcoming those limitations. The presented experimental results, conducted on four real-life symbolic regression problems, show that the proposed algorithms outperform not only the existing methods based on the concept of alignment in the error space, but also geometric semantic genetic programming and standard genetic programming.

1 Introduction

In the last few years, the use of semantic awareness for improving Genetic Programming (GP) [1,2] and other heuristic methods [3] became popular. A survey discussing large part of the existing semantic approaches in GP can be found in [4]. In that survey, the existing work was categorized into three broad classes: approaches based on semantic diversity, on semantic locality and on semantic geometry. Among several other references, semantic diversity and semantic locality, and their relationship with the effectiveness of GP, were investigated in depth in [5,6]. On the other hand, the idea of studying semantic geometry revealed itself about a decade ago (see for instance [7,8]), and became a GP hot topic in 2013, when a new version of GP, called Geometric Semantic GP (GSGP) was introduced [9]. GSGP uses new operators, called geometric semantic operators (GSOs), instead of traditional crossover and mutation, and it owes part of its successes to the fact that GSOs induce a unimodal fitness landscape [10,11,12] for any supervised learning problem. In the last six years, a large number of contributions showed that GSGP is competitive with the state of the art in many applicative domains (see for instance [13,14,15]). Few years after the introduction of GSGP, a new way of exploiting semantic awareness was presented in [16] and further developed in [17,18]. The idea, which is also the focus of this paper, can be sketched as follows.

We define semantics of an individual as the vector of its output values on the training cases [9]. Hence, semantics can be represented as a point in a space that we call *semantic space*. In supervised learning, the target is also a point in the semantic space, but usually (unless the rare case where the target value is equal to zero for each training case) it does *not* correspond to the origin of the Cartesian system. Then, we translate each point in the semantic space by subtracting the target from it. In this way, for each individual, we obtain a new point, that we call *error vector*, and we call the corresponding space *error space*. The target, by construction, corresponds to the origin of the Cartesian system in the error space.

In [16], it was proven that, given sets of individuals with particular characteristics of alignment in the error space (called optimally aligned, and optimally coplanar, individuals), it is possible to analytically reconstruct a globally optimal solution (see Section 2.1). Keeping this in mind, it makes sense to develop GP systems whose objective is looking for optimally aligned, or optimally coplanar, individuals (instead of looking directly for an optimal solution, as in traditional GP). The first attempt at developing a system aimed at searching for optimally aligned, or optimally coplanar, individuals was presented in [16], where the ESAGP method was proposed. While ESAGP reported interesting results, it has the important limitation of constraining the alignment only in one particular direction in the error space, that is prefixed *a priori*. In order to overcome this limitation, a particular version of GP must be defined, that evolves individuals that are sets of programs, instead of just one program as in traditional GP. The first preliminary attempt was made in [17], where the POGP system was introduced. However, in [17] severe limitations of POGP, which make it unusable in practice, were reported.

The objective of this paper is to present new GP systems aimed at evolving sets of programs with the objective of generating optimally aligned individuals, and able to overcome all the limitations of POGP. The new systems (called Align, Nested_Align and Nested_Align_β) will be compared to standard GP, GSGP and ESAGP on four complex real-life symbolic regression problems.

The rest of the paper is structured as follows: in Section 2, we revise previous and related work, with particular focus on ESAGP and POGP, describing the known issues of POGP. Section 3 describes the proposed methods (Align, Nested_Align and Nested_Align_β), explaining how they overcome the previously discussed issues of POGP. In Section 4, we present our experimental study, in which the experimental settings and test problems are described and the obtained results discussed. Finally, Section 5 concludes the paper, also suggesting ideas for future research.

2 Previous and Related Work

2.1 Error Space Alignment GP

In [16], the concept of optimal alignment was introduced for the first time, together with a new GP method, called ESAGP (which stands for Error Space Alignment GP), that exploits it. Two individuals A and B are *optimally aligned* if a scalar constant k exists such that $e_A = k \cdot e_B$, where e_A and e_B are the error vectors of A and B respectively. From this definition, it is not difficult to see that two individuals are optimally aligned if the straight line joining their error vectors also intersects the origin in the error space. Analogously, and extending the idea to three dimensions, three individuals are *optimally coplanar* if the bi-dimensional plane in which their error vectors lie in the error space also intersects the origin. In [16], it is proven that given any pair of optimally aligned individuals A and B, it is possible to reconstruct a globally optimal solution P_{opt}. This solution is defined in Equation (1):

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

Analogously, in [16], it is also proven that given any triplet of optimally coplanar individuals, it is possible to analytically construct a globally optimal solution (the reader is referred to [16] for the equation of the globally optimal solution in that case).

Keeping all this in mind, the ESAGP method introduced in [16] was composed by two GP systems: ESAGP-1, whose objective is looking for optimally aligned pairs of individuals, and ESAGP-2 whose objective is looking for triplets of optimally coplanar individuals. The biggest difference between these systems and traditional GP is that the search in ESAGP-1 and ESAGP-2 is not guided by the quality of the single solutions, but only on their alignment properties. Several possible ways of searching for alignments can be imagined. In ESAGP, one direction, called *attractor*, is fixed and all the individuals in the population are *pushed* towards an alignment with the attractor. In this way, ESAGP-1 and ESAGP-2 can maintain the traditional representation of solutions where each solution is represented by one program. The other face of the coin is that ESAGP-1 and ESAGP-2 strongly restrict what GP can do, forcing the alignment to necessarily happen in just one prefixed direction, i.e. the one of the attractor. The ESAGP systems were also studied in [18], where the operators used to reach the alignment with the attractor were GSOs. The authors of [18] report severe overfitting for this new ESAGP version. The objective of this paper is to relieve the constraint of ESAGP by defining a new GP system that is generally able to evolve *vectors* of programs (even though only vectors of size equal to 2 will be used in this paper). As already mentioned, a preliminary attempt is represented by POGP [17], described below.

2.2 Pair Optimization GP

In [17], Pair Optimization GP (POGP) was introduced. Limiting itself to the bi-dimensional case (i.e. to the case in which pairs of optimally aligned individuals are sought for), POGP extends ESAGP-1, releasing the limitation of forcing alignments in a prefixed direction. In POGP, individuals are pairs of programs, and fitness is the angle between the respective error vectors. From now on, for the sake of clarity, this type of individual (i.e. individuals characterized by more than one program) will be called *multi-individuals*. In [17], the following problems of POGP were reported: (i) generation of semantically identical, or very similar, expressions; (ii) k constant in Equation (1) equal, or very close, to zero; (iii) generation of expressions with huge error values. These problems are discussed here:

Issue 1: generation of semantically identical, or very similar, expressions. A simple way for GP to find two expressions that are optimally aligned in the error space is to find two expressions that have exactly the same semantics (and consequently the same error vector). However, this causes a problem once we try to reconstruct the optimal solution as in Equation (1). In fact, if the two expressions have the same error vector, the k value in Equation (1) is equal to 1, which gives a denominator equal to zero. Experience tells us that GP tends very often to generate multi-individuals that have this kind of problem. Also, it is worth pointing out that even preventing GP from generating multi-individuals that have an identical sematics, GP may still push the evolution towards the generation of multi-individuals whose expressions have semantics that are very similar between each other. This leads to a k constant in Equation (1) that, although not being exactly equal to 1, has a value that is very close to 1. As a consequence, the denominator in Equation (1), although not being exactly equal to zero, may be very close to zero and thus the value calculated by Equation (1) could be a huge number. This would force a GP system to deal with unbearably large numbers during all its execution, which may lead to several problems, including numeric overflow.

Issue 2: k constant in Equation (1) equal, or very close, to zero. Looking at Equation (1), one may notice that if k is equal to zero, then expression B is irrelevant and the reconstructed solution P_{opt} is equal to expression A. A similar problem also manifests itself when k is not exactly equal to zero, but very close to zero. In this last case, both expressions A and B contribute to P_{opt}, but the contribution of B may be so small to be considered as marginal, and P_{opt} would *de facto* be extremely similar to A. Experience tells us that, unless this issue is taken care of, the evolution would very often generate such situations. This basically turns a multi-individual alignment based system into traditional GP, in which only one of the expressions in the multi-individual matters. If we really want to study the effectiveness of multi-individual alignment based systems, we have to impede these kind of situations.

Issue 3: generation of expressions with huge error values. As previously mentioned, systems based on the concept of alignment in the error space could limit themselves to searching for expressions that are optimally aligned, without taking into account their performance (i.e. how close their semantics are to the target). However, experience tells us that, if we give GP the only task of finding aligned expressions, GP frequently tends to generate expressions whose semantics contain unbearably large numbers. Once again, this may lead to several problems, including numeric overflow, and a successful system should definitely prevent this from happening.

One fact that should be remarked is that none of the previous issues can be taken into account with simple conditions that prevent some precise situations from happening. For instance, one may consider solving Issue 1 by simply testing if the expressions in a multi-individual are semantically identical between each other, and rejecting the multi-individual if that happens. But, as already discussed, expressions that have very similar semantics between each other may also lead to problems. Furthermore, the idea of introducing a threshold ϵ to the semantic diversity of the expressions in a multi-individual, and rejecting all the multi-individuals for which the diversity is smaller than ϵ does not seem a brilliant solution. In fact, experience tells us that GP would tend to generate multi-individuals with a diversity equal, or very close to ϵ itself. Analogously, if we consider Issue 2, neither rejecting multi-individuals that have a k constant equal to zero, nor rejecting individuals that have an absolute value of k larger than a given threshold would solve the problem. Finally, considering Issue 3, also rejecting individuals that have the coordinates of the semantic vector larger than a given threshold δ_{max} would not solve the problem, since GP would tend to generate expressions in which the coordinates of the semantic vector are equal, or very close, to δ_{max} itself.

In such a situation, we believe that a promising way to effectively solve these issues is (besides defining the specific conditions mentioned above) to take the issues into account in the selection process, for instance giving more probability of being selected for mating to multi-individuals that have large semantic diversity between the expressions, values of k that are, as much as possible, far from zero and expressions whose semantics are, as much as possible, close to the target. These ideas are implemented in the proposed systems, which are described below.

3 Description of the Proposed Methods

In order to introduce the proposed methods in a compact way, we describe first the Nested_Align method, and then we discuss the other methods by simply pointing out the differences between them and Nested_Align.

3.1 Nested_Align

Here, we describe selection, mutation and population initialization of Nested_Align, keeping in mind that no crossover has been defined yet for this method.

Selection. Besides trying to optimize the performance of the multi-individuals, selection is the phase that takes into account the issues of the previous alignment-based methods discussed in Section 2.2. Nested_Align contains five selection criteria, that have been organized into a nested tournament. Let $\phi_{\mathbf{1}}, \phi_{\mathbf{2}}, ..., \phi_{\mathbf{m}}$ be the expressions characterizing a multi-individual. Once again, it is worth pointing out that only the case $m = 2$ was taken into account in this paper. But the concept is general, and so it will be explained using m expressions. The selection criteria are:

- Criterion 1: diversity (calculated using the standard deviation) of the semantics of the expressions $\phi_{\mathbf{1}}, \phi_{\mathbf{2}}, ..., \phi_{\mathbf{m}}$ (to be maximized).
- Criterion 2: the absolute value of the k constant that characterizes the reconstructed expression, as in Equation (1) (to be maximized).
- Criterion 3: the sum of the errors of the single expressions $\phi_{\mathbf{1}}, \phi_{\mathbf{2}}, ..., \phi_{\mathbf{m}}$ (to be minimized).
- Criterion 4: the angle between the error vectors of the expressions $\phi_{\mathbf{1}}, \phi_{\mathbf{2}}, ..., \phi_{\mathbf{m}}$ (to be minimized).
- Criterion 5: the error of the reconstructed expression P_{opt} in Equation (1) (to be minimized).

The nested tournament works as follows: an individual is selected if it is the winner of a tournament, that we call T_5, that is based on Criterion 5. All the participants in tournament T_5, instead of being individuals chosen at random as in the traditional tournament selection algorithm, are winners of previous tournaments (that we call tournaments of type T_4), which are based on Criterion 4. Analogously, for all $i = 4, 3, 2$, all participants in the tournaments of type T_i are winners of previous tournaments (that we will call tournaments of type T_{i-1}), based on Criterion $i - 1$. Finally, the participants in the tournaments of type T_1 (the kind of tournament that is based on Criterion 1) are individuals selected at random from the population. In this way, an individual, in order to be selected, has to undergo five selection *layers*, each of which is based on one of the five different chosen criteria. Motivations for the chosen criteria follow:

- Criterion 1 was introduced to counteract Issue 1 in Section 2.2. Maximizing the semantic diversity of the expressions in a multi-individual should naturally prevent GP from creating multi-individuals with identical semantics or semantics that are very similar between each other.
- Criterion 2 was introduced to counteract Issue 2 in Section 2.2. Maximizing the absolute value of constant k should naturally allow GP to generate multi-individuals for which k's value is neither equal nor close to zero.
- Criterion 3 was introduced to counteract Issue 3 in Section 2.2. If the expressions that characterize a multi-individual will have a "reasonable" error, then their semantics will be reasonably similar to the target, thus naturally avoiding the appearance of unbearably large numbers.
- Criterion 4 is a performance criterion: if the angle between the error vectors of the expressions $\phi_\mathbf{1}, \phi_\mathbf{2}, ..., \phi_\mathbf{m}$ is equal to zero, then Equation (1) allows us to reconstruct a perfect solution P_{opt}. Also, the smaller this angle, the smaller should be the error of P_{opt}. Nevertheless, experience tells us that multi-individuals may exist with similar values of this angle, but very different values of the error of the reconstructed solution P_{opt}, due for example to individuals with a very large distance from the target. This fact made us conclude that Criterion 4 cannot be the only performance objective, and suggested to us to also introduce Criterion 5.
- Criterion 5 is a further performance criterion. Among multi-individuals with the same angle between the error vectors of the expressions $\phi_\mathbf{1}, \phi_\mathbf{2}, ..., \phi_\mathbf{m}$, the preferred ones will be the ones for which the reconstructed solution P_{opt} has the smallest error.

Mutation. The mechanism we have implemented for applying mutation to a multi-individual is extremely simple: for each expression $\phi_\mathbf{i}$ in a multi-individual, mutation is applied to $\phi_\mathbf{i}$ with a given mutation probability p_m, where p_m is a parameter of the system. It is worth remarking that in our implementation all expressions $\phi_\mathbf{i}$ of a multi-individual have the same probability of undergoing mutation, but this probability is applied independently to each one of them. So, some expressions could be mutated, and some other could remain unchanged. The type of mutation that is applied to expressions is Koza's standard subtree mutation [1].

To this "basic" mutation algorithm, we have also decided to add a mechanism of rejection, in order to help the selection process in counteracting the issues discussed in Section 2.2. Given a prefixed parameter that we call δ_k, if the multi-individual generated by mutation has a k constant included in the range $[1 - \delta_k, 1 + \delta_k]$, or in the range $[-\delta_k, \delta_k]$, then the k constant is considered, respectively, too close to 1 or too close to 0 and the multi-individual is rejected. In this case, a new individual is selected for mutation, using again the nested tournament discussed above.

The combined effect of this rejection process and of the selection algorithm should strongly counteract the issues discussed in Section 2.2. In fact, when k is equal to 1, or equal to 0, or even close to 1 or 0 inside a given prefixed toleration radius δ_k, the multi-individual is not allowed to survive. For all the other multi-individuals, distance between k and 1 and between k and 0 are used as optimization objectives, to be maximized. This allows GP to evolve multi-individuals with k values that are "reasonably far" from 0 and 1.

Initialization. Nested_Align initializes a population of multi-individuals using multiple executions of the Ramped Half and Half algorithm [1]. More specifically, let n be the number of expressions in a multi-individual ($n = 2$ in our experiments), and let m be the size of the population that has to be initialized. Nested_Align runs n times the Ramped Half and Half algorithm, thus creating n "traditional" populations of trees $P_1, P_2, ..., P_n$, each of which containing m trees. Let $P = \{\Pi_1, \Pi_2, ..., \Pi_m\}$ be the population that Nested_Align has to initialize (where, for each $i = 1, 2, ..., m$, Π_i is an n-dimensional multi-individual). Then, for each $i = 1, 2, ..., m$ and for each $j = 1, 2, ..., n$, the j^{th} tree of multi-individual Π_i is the j^{th} tree in population P_i.

To this "basic" initialization algorithm, we have added an adjustment mechanism to make sure that the initial population does not contain multi-individuals with a k equal, or close, to 0 and 1. More in partcular, given a prefixed number of expressions α, that is a new parameter of the system, if the created multi-individual has a k value included in the range $[1 - \delta_k, 1 + \delta_k]$, or in the range $[-\delta_k, \delta_k]$ (where δ_k is the same parameter as the one used for implementing rejections of mutated individuals), then α randomly chosen expressions in the multi-individual are removed and replaced by as many new randomly generated expressions. Then the k value is calculated again, and the process is repeated until the multi-individual has a k value that stays outside the ranges $[1 - \delta_k, 1 + \delta_k]$ and $[-\delta_k, \delta_k]$. Only when this happens, the multi-individual is accepted inside the population. Given that only multi-individuals of two expressions are considered in this paper, in our experiments we have always used $\alpha = 1$.

3.2 Differences Between Align, Nested_Align_β and Nested_Align

Align. The difference between Align and Nested_Align is that Align does not use the nested tournament discussed above. Selection in Align is implemented by a traditional tournament algorithm, using as fitness the error of the reconstructed expression P_{opt} in Equation (1). Mutation and initialization in Align work exactly as in Nested_Align. In this way, the only mechanism that Align has to counteract the issues described in Section 2.2 is to make sure that initialization and mutation only create multi-individuals with a k value outside the ranges $[1 - \delta_k, 1 + \delta_k]$ and $[-\delta_k, \delta_k]$. The motivation for the introduction of Align is that the nested tournament that characterizes Nested_Align may be complex and time-consuming. Comparing the performance of Nested_Align to the ones of Align, we will be able to evaluate to importance of the nested tournament and its impact on the performance of the system.

Nested_Align_β. This method integrates a multi-individual approach with a traditional single-expression GP approach. More precisely, the method begins as Nested_Align, but after β generations, the evolution is done by GSGP. In order to transform a population of multi-individuals into a population of traditional single-expression individuals, each individual is replaced by the reconstructed solution P_{opt} in Equation (1). The rationale behind the introduction of Nested_Align_β is that alignment-based systems are known to have a very quick improvement in fitness in the first generations, which may sometimes cause overfitting of training data (the reader is referred to [16,17,18,19,20] for a discussion of the issue). Given that GSGP is instead known for being a slow optimization process, able to limit overfitting under certain circumstances (see [21]), the idea is transforming Nested_Align into GSGP, possibly before overfitting arises. Even though a deep study of parameter β is strongly in demand, only $\beta = 50$ was tested in this paper. For this reason, from now on, the name Nested_Align_50 will be used for this method.

4 Experimental Study

4.1 Experimental Settings and Test Problems

For each model, 30 runs were performed, each on a different randomly selected split of the dataset into training set (70%) and test set (30%). The parameters used are summarized in Table 1. Besides those parameters, the primitive operators were addition, subtraction, multiplication and division protected as in [1]. The terminal symbols included one variable for each feature in the dataset, plus the following numerical constants: -1.0, -0.75, -0.5, -0.25, 0.25, 0.5, 0.75, 1.0. Parent selection was done using tournaments of size 5, with the exception of the models which use nested selection (i.e. Nested_Align and, in the first 50 generations, Nested_Align_50), which used a tournament of size 10 for each layer of the nested selection. For standard GP subtree crossover and subtree mutation were used [1], where crossover rate was equal to 0.9 and mutation rate was equal to 0.1. For all the other studied methods, crossover rate was equal to zero (i.e. no crossover was performed during the evolution). While Align, Nested_Align and Nested_Align_50 do not have a crossover operator implemented yet, the motivation for not using crossover in GSGP can be found in [17], where it is clearly shown that GSGP using only mutation often overcomes GSGP using both crossover and mutation. The test problems that we have used in our experimental study are four symbolic regression real-life applications. All these problems have already been used in previous GP studies [17,21,22,23,24]. Table 2 reports, for each dataset, the number of features (variables) and the number of instances (observations). For a complete description of these datasets, the reader is referred to the references reported in the same table.

4.2 Experimental Results

The results we have obtained are reported in Figures 1, 2, 3, 4 and 5. They are organized as follows: in Figures 1 and 2, we report the results of the best error obtained on training data (more particularly, in Figure 1 the proposed methods are compared to standard GP and GSGP, while in Figure 2 they are compared to ESAGP-1 and ESAGP-2); in Figures 3 and 4, we report the results of the best training model on unseen test

Parameter	Setting
Population size	100
Max. # of generations	200
Initialization	Ramped H-H
Max. depth for evolution	17
Max. depth for initialization	6
δ_k	0.02

Table 1. GP parameters used in our experiments.

Dataset	# Features	# Instances
Bioavailability [25]	241	206
PPB [25]	626	131
Toxicity [25]	626	234
Energy [24]	8	768

Table 2. Description of the test problems. For each dataset, the number of features (independent variables) and the number of instances (observations) are reported.

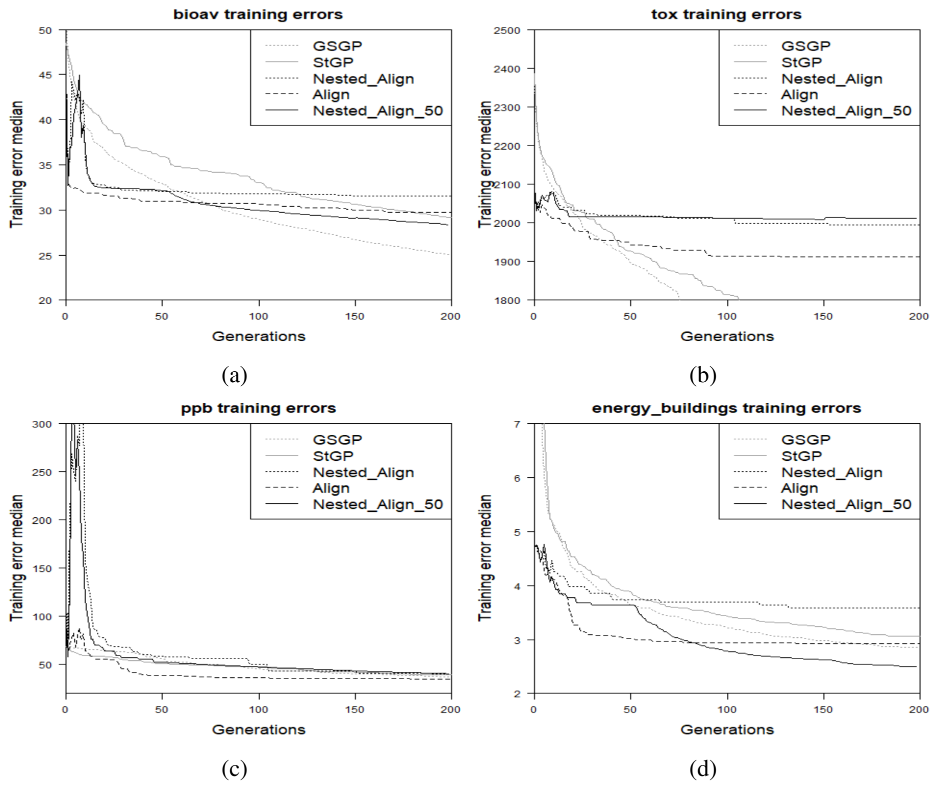
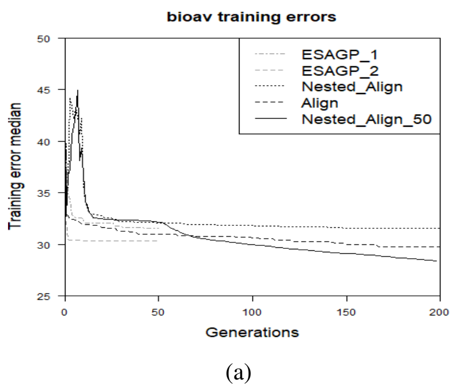


Fig. 1. Results on the *training set*. Comparison between the prososed methods (Nested_Align, Align and Nested_Align_50), standard GP and GSGP. Plot (a): Bioavailability; plot (b): Toxicity; plot (c): PPB; plot (d): Energy.

data (in Figure 3 the proposed methods are compared to standard GP and GSGP, while in Figure 4 they are compared to ESAGP-1 and ESAGP-2); in Figure 5, we report the results relative to the size of the programs (calculated as the number tree nodes). In Figures 1 and 3 (i.e. the ones where the proposed methods are compared to standard GP and GSGP), plot (a) reports the results obtained on the Bioavailability problem, plot (b) reports the ones obtained on the Toxicity problem, plot (c) on the PPB prob-



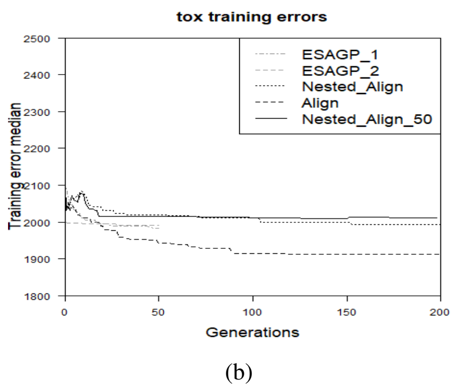
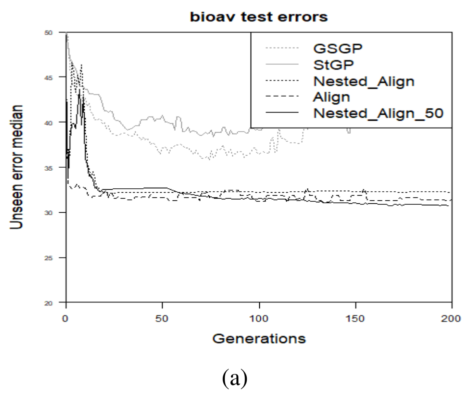
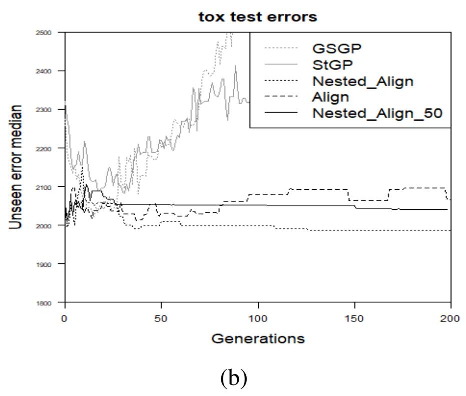
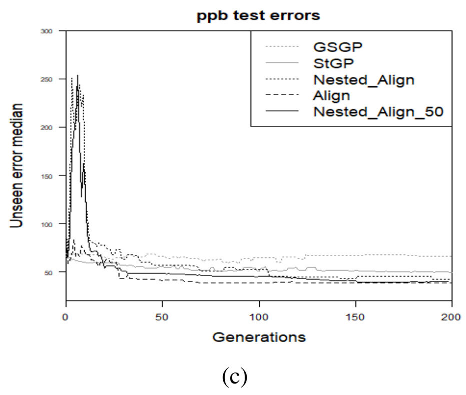


Fig. 2. Results on the *training set*. Comparison between the prososed methods (Nested_Align, Align and Nested_Align_50), ESAGP-1 and ESAGP-2. Plot (a): Bioavailability; plot (b): Toxicity;







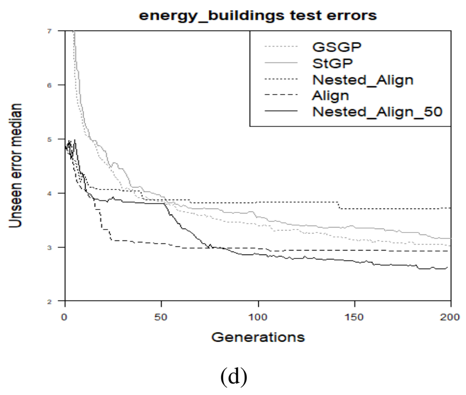
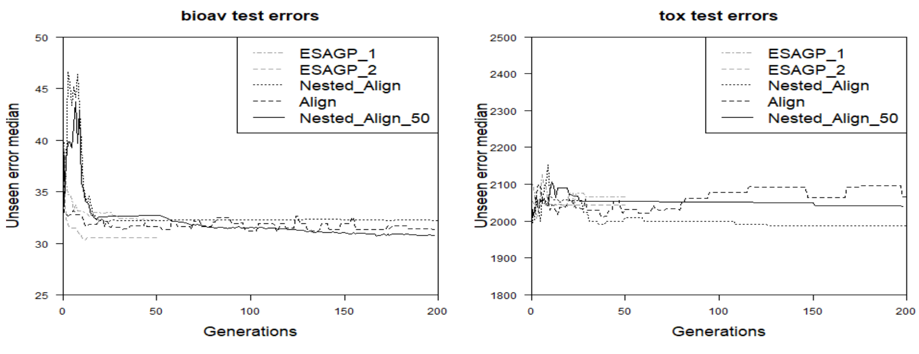
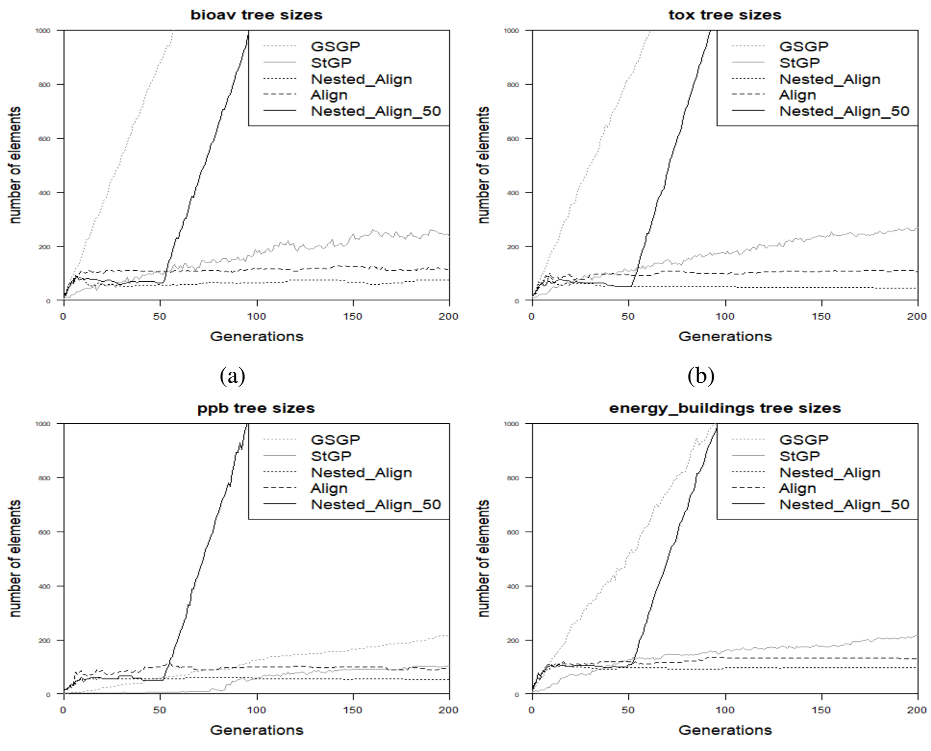


Fig. 3. Results on the *test set*. Comparison between the prososed methods (Nested_Align, Align and Nested_Align_50), standard GP and GSGP. Plot (a): Bioavailability; plot (b): Toxicity; plot (c): PPB; plot (d): Energy.



(a) (b)

Fig. 4. Results on the *test set*. Comparison between the prososed methods (Nested_Align, Align and Nested_Align_50), ESAGP-1 and ESAGP-2. Plot (a): Bioavailability; plot (b): Toxicity;



(a) (b)

(c) (d)

Fig. 5. Results concerning the *program size*. Comparison between the prososed methods (Nested_Align, Align and Nested_Align_50), standard GP and GSGP. Plot (a): Bioavailability; plot (b): Toxicity; plot (c): PPB; plot (d): Energy.

lem, and plot (d) on Energy. Concerning the ESAGP methods, we have taken the results directly from [16], for comparison. In that paper, only results relative to training and unseen error on the Bioavailability and Toxicity datasets were made available. For this reason, in Figures 2 and 4, plot (a) reports the results obtained on the Bioavailability problem and plot (b) reports the ones obtained on the Toxicity problem, and those figures do not contain any other plot. Finally, Table 3 reports the results of the statistical tests performed on the obtained unseen errors.

Table 3. p-values of the Wilcoxon rank-sum test on unseen data, under the alternative hypothesis that the samples do not have equal medians. Bold denotes statistically significant values.

			BIOAVAILABILITY		
	STGP	NESTED_ALIGN	NESTED_ALIGN_50	ALIGN	ESAGP-1
GSGP	0.133	**9.33E-05**	**1.02E-05**	**1.21E-04**	**3.18E-05**
STGP		**3.29E-04**	**1.81E-05**	**0.002**	**2.58E-04**
NESTED_ALIGN			0.289	0.438	0.624
NESTED_ALIGN_50				0.962	0.420
ALIGN					0.646
			TOXICITY		
	STGP	NESTED_ALIGN	NESTED_ALIGN_50	ALIGN	ESAGP-1
GSGP	0.035	**1.13E-09**	**3.04E-07**	**1.88E-08**	**2.39E-09**
STGP		**3.26E-06**	**1.86E-04**	**5.98E-05**	**1.93E-05**
NESTED_ALIGN			0.511	0.307	0.246
NESTED_ALIGN_50				0.704	0.678
ALIGN					0.986
			PPB		
	STGP	NESTED_ALIGN	NESTED_ALIGN_50	ALIGN	
GSGP	0.237	0.153	0.043	**0.001**	
STGP		0.474	0.124	**6.98E-04**	
NESTED_ALIGN			0.359	0.021	
NESTED_ALIGN_50				0.099	
			ENERGY		
	STGP	NESTED_ALIGN	NESTED_ALIGN_50	ALIGN	
GSGP	0.109	**1.24E-05**	**1.33E-05**	0.270	
STGP		**8.75E-04**	**3.08E-06**	0.023	
NESTED_ALIGN			**1.26E-08**	**6.03E-06**	
NESTED_ALIGN_50				**1.01E-04**	

Let us begin commenting the results on the training set. As Figure 1 shows, on the training set Nested_Align_50 is the method that obtains the best results on one problem over four (Energy). On two of the other problems (Bioavailability and Toxicity) the method that was able to find the best results was GSGP. Finally, on the PPB dataset all the methods returned comparable results between each other, with a slight preference for Align. Remembering that, after 50 generations, Nested_Align_50 "turns into" GSGP, our interpretation of these results is that, in general, GSGP is an appropriate method for optimizing training data, which is not surprising, given that GSOs induce a unimodal fitness landscape. In particular, the "switch" between the Nested_Align algorithm and GSGP at generation 50 seems beneficial in much of the cases. This can be seen in the Bioavailability and Energy problems, where a rapid improvement of the curve of Nested_Align_50, looking like a sudden descending "step", is clearly visible at generation 50. So, given that in the last part of the runs Nested_Align_50 and GSGP are identical, Nested_Align_50 prevails if the initial phase in which Nested_Align was executed was beneficial. On the other hand, GSGP prevails if it was not. From the above discussed results, we can conclude that it is beneficial on one problem, while it is not on two others (and it is irrelevant in the fourth of the studied problems, where Nested_Align_50 and GSGP perform comparably). Concerning a comparison between the proposed methods and ESAGP-1 and ESAGP-2 (Figure 2), two considerations have to be done: first of all, in [16] results were reported only until generation 50, and those are the only ESAGP-1 and ESAGP-2 results in our possession. Secondly, it is possible to "speculate" that both ESAGP-1 and ESAGP-2 are outperformed by other methods both on the Bioavailability and on the Toxicity datasets (more in particular, by Nested_Align_50 and Align on Bioavailability and by Align on Toxicity). In fact, even though we cannot be sure because we do not have the data of the last 150 generations, the curve of both the ESAGP methods, after a rapid decrease in the first 20 generations, seems to stabilize and to remain practically constant, approximately from generation 20 to generation 50.

Now, let us discuss the results on the test set, starting from Figure 3. On the Bioavailability, PPB and Toxicity problems, the three proposed methods clearly outperform both GSGP and standard GP, with Nested_Align_50 that is slightly preferable compared to the other two methods on Bioavailability and Align on Toxicity. On the Energy problem, the method that performs better than all the others is Nested_Align_50, and, also on the test set, we can observe a clear fitness improvement, looking like a sudden descending "step", at generation 50, where the switch between Nested_Align and GSGP takes place. In conclusion, on the test set all the three methods that we have introduced in this paper show reasonable results, improving the ones of GSGP and standard GP. Among those methods, Nested_Align_50 seems the most preferable one, corroborating our intuition that Nested_Align learns fast in the beginning, while the switch to GSGP allows us to continue the learning while limiting overfitting. Concerning a comparison between the proposed methods and ESAGP-1 and ESAGP-2 (Figure 4), what we can conclude using the data at our disposal is that both ESAGP-1 and ESAGP-2 are outperformed by Nested_Align_50 for the Bioavailability problem and by Nested_Align_50 and Nested_Align on the Toxicity problem. However, it is worth pointing out that, when discussing the results on the test set, having data only until generation 50 strongly limits our possible conclusions. In fact, we do not have any information that, later in the run, the ESAGP methods will not begin to overfit, as it happens to, for instance, to Align on the Toxicity problem. Actually, on the Toxicity problem, Align outperforms both ESAGP-1 and ESAGP-2 in the first 50 generations, and only later in the run the test error of Align starts increasing. In synthesis, we consider our conclusions (i.e. that the ESAGP methods are outperformed by Nested_Align_50 for the Bioavailability problem and by Nested_Align_50 and Nested_Align on the Toxicity problem) the most "optimistic" scenario for the ESAGP methods. If we had the results until generation 200, the picture for ESAGP could be even worse.

We finally discuss Figure 5, reporting the dimensions of the evolved programs, a very important criterion that has direct link with the models' interpretability [26]. GSGP and Nested_Align_50 generate much larger individuals compared to the other methods. This was expected, given that generating large individual is a known drawback of GSOs [9]. The fact that in the first 50 generations Nested_Align_50 does not use GSOs only partially limits the problem, simply delaying the code growth, that is, after generation 50, exactly as important as for GSGP. On the other hand, it is clearly visible that Align and Nested_Align are able to generate individuals that are smaller than the ones of standard GP. Furthermore, after a first initial phase in which the size of the individuals grow, we can see that Align and Nested_Align basically have no code growth (the curves of these two methods, after an initial phase of growth, are practically parallel to the horizontal axis). Last but not least, in all the studied problems the final models generated by Align and Nested_Align have around only 50 tree nodes.

All this considered, our conclusions are: if we are interested in performance and we can accept models that are "black boxes" (meaning with this, models that are too complicated to be interpreted and understood), then Nested_Align_50 seems the most appropriate of the proposed methods. On the other hand, if the readability of the model is an issue, then Align and Nested_Align are good compromises between performance and model simplicity.

To analyse the statistical significance of the results that we have obtained on unseen data, a set of tests has been performed. The Lilliefors test has shown that the data are not normally distributed and hence a rank-based statistic has been used. The Wilcoxon rank-sum test for pairwise data comparison with Bonferroni correction has been used, under the alternative hypothesis that the samples do not have equal medians at the end of the run, with a significance level $\alpha = 0.05$. The p-values are reported in Table 3, where statistically significant differences are highlighted with p-values in bold. As we can observe, on the Bioavailability and Toxicity datasets the differences between the proposed methods (Align, Nested_Align and Nested_Align_50) and the existing ones (standard GP, GSGP and ESAGP-1) are statistically significant, while the differences of the proposed methods between each other are not statistically significant. The same thing also holds for the Energy dataset, with the only exception of the Align method, whose results are not statistically different from the ones of GSGP and standard GP. The only dataset in which the statistical test gives us a different picture is PPB, where, among the proposed methods, Align is the only one that was able to return results that are statistically different from the ones of GSGP and standard GP.

5 Conclusions and Future Work

Three new genetic programming systems, called Align, Nested_Align and Nested_Align_50, based on the idea of alignment in the error space, were introduced in this paper. These new systems overcome some limitations of the previously existing alignment-based algorithms. On four real-life symbolic regression problems, the proposed systems have outperformed not only the state-of-the-art alignment-based methods, but also standard genetic programming and geometric semantic genetic programming. More specifically, Nested_Align_50 was the method that returned the best results, but Nested_Align_50 also generated very large programs. On the other hand, Align and Nested_Align, although returning results that are slightly worse compared to Nested_Align_50 in terms of accuracy, were able to evolve much smaller programs.

One of the most important limitations of this paper is that only alignments in two dimensions are considered. In other words, the proposed systems use individuals that are pairs of programs and they are only able to search for pairs of optimally aligned programs. Our current research is focused on extending the method to more then two dimensions. For instance, we are currently working on the development of systems that evolve individuals that are triplets of programs, aimed at finding triplets of optimally coplanar individuals. The subsequent step will be to further extend the method, possibly generalizing to any number of dimensions. The design of self-configuring methods, that automatically decide the most appropriate dimension, is one of the most ambitious goals of our current work.

References

1. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA, USA (1992)
2. Poli, R., Langdon, W.B., Mcphee, N.F.: A field guide to genetic programming. (March 2008)
3. Castelli, M., Silva, S., Manzoni, L., Vanneschi, L.: Geometric selective harmony search. Information Sciences **279** (2014) 468 – 482
4. Vanneschi, L., Castelli, M., Silva, S.: A survey of semantic methods in genetic programming. Genetic Programming and Evolvable Machines (2014) 1–20
5. Nguyen, Q.U.: Examining Semantic Diversity and Semantic Locality of Operators in Genetic Programming. PhD thesis, University College Dublin (2011)
6. Castelli, M., Vanneschi, L., Silva, S.: Semantic search-based genetic programming and the effect of intron deletion. IEEE transactions on cybernetics **44**(1) (2014) 103–113
7. Krawiec, K., Lichocki, P.: Approximating geometric crossover in semantic space. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation. GECCO '09, New York, NY, USA, ACM (2009) 987–994
8. Pawlak, T.P., Krawiec, K.: Competent geometric semantic genetic programming for symbolic regression and boolean function synthesis. Evolutionary Computation **15**(1) (2017) 1–28
9. Moraglio, A., Krawiec, K., Johnson, C.: Geometric semantic genetic programming. In Coello, C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M., eds.: Parallel Problem Solving from Nature - PPSN XII. Volume 7491 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2012) 21–31
10. Vanneschi, L. In: An Introduction to Geometric Semantic Genetic Programming. Springer International Publishing, Cham (2017) 3–42
11. Verel, S., Collard, P., Tomassini, M., Vanneschi, L.: Fitness landscape of the cellular automata majority problem: View from the "olympus". Theoretical Computer Science **378**(1) (2007) 54 – 77
12. Vanneschi, L., Tomassini, M., Collard, P., Vérel, S., Pirola, Y., Mauri, G.: A comprehensive view of fitness landscapes with neutrality and fitness clouds. In: Proceedings of the 10th European Conference on Genetic Programming. EuroGP'07, Berlin, Heidelberg, Springer-Verlag (2007) 241–250

13. Castelli, M., Trujillo, L., Vanneschi, L., Popovič, A.: Prediction of energy performance of residential buildings: A genetic programming approach. Energy and Buildings **102** (2015) 67 – 74
14. Castelli, M., Castaldi, D., Giordani, I., Silva, S., Vanneschi, L., Archetti, F., Maccagnola, D.: An efficient implementation of geometric semantic genetic programming for anticoagulation level prediction in pharmacogenetics. In: Portuguese Conference on Artificial Intelligence, Springer (2013) 78–89
15. Castelli, M., Vanneschi, L., De Felice, M.: Forecasting short-term electricity consumption using a semantics-based genetic programming framework: The South Italy case. Energy Economics **47** (2015) 37–41
16. Ruberto, S., Vanneschi, L., Castelli, M., Silva, S. In: ESAGP – A Semantic GP Framework Based on Alignment in the Error Space. Springer Berlin Heidelberg, Berlin, Heidelberg (2014) 150–161
17. Castelli, M., Vanneschi, L., Silva, S., Ruberto, S.: How to exploit alignment in the error space: Two different gp models. In Riolo, R., Worzel, W.P., Kotanchek, M., eds.: Genetic Programming Theory and Practice XII. Genetic and Evolutionary Computation, Ann Arbor, USA, Springer (8-10 May 2014) 133–148
18. Gonçalves, I., Silva, S., Fonseca, C.M., Castelli, M.: Arbitrarily close alignments in the error space: A geometric semantic genetic programming approach. In: Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion. GECCO '16 Companion, New York, NY, USA, ACM (2016) 99–100
19. Castelli, M., Manzoni, L., Silva, S., Vanneschi, L.: A comparison of the generalization ability of different genetic programming frameworks. In: Evolutionary Computation (CEC), 2010 IEEE Congress on, IEEE (2010) 1–8
20. Castelli, M., Manzoni, L., Silva, S., Vanneschi, L.: A quantitative study of learning and generalization in genetic programming. Genetic Programming (2011) 25–36
21. Vanneschi, L., Castelli, M., Manzoni, L., Silva, S.: A new implementation of geometric semantic GP and its application to problems in pharmacokinetics. In: Proceedings of the 16th European Conference on Genetic Programming, EuroGP 2013. Volume 7831 of LNCS., Vienna, Austria, Springer Verlag (3-5 April 2013) 205–216
22. Castelli, M., Vanneschi, L., Silva, S.: 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) (2014) 4608 – 4616
23. Castelli, M., Vanneschi, L., Silva, S.: Prediction of high performance concrete strength using genetic programming with geometric semantic genetic operators. Expert Systems with Applications **40**(17) (2013) 6856 – 6862
24. Castelli, M., Trujillo, L., Vanneschi, L., Popovič, A.: Prediction of energy performance of residential buildings: A genetic programming approach. Energy and Buildings **102** (2015) 67 – 74
25. Archetti, F., Lanzeni, S., Messina, E., Vanneschi, L.: Genetic programming for computational pharmacokinetics in drug discovery and development. Genetic Programming and Evolvable Machines **8**(4) (2007) 413–432
26. Poli, R., McPhee, N.F., Vanneschi, L.: The impact of population size on code growth in gp: Analysis and empirical validation. In: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation. GECCO '08, New York, NY, USA, ACM (2008) 1275–1282