On the Generalization Ability of Geometric Semantic Genetic Programming

Ivo Gonçalves[1,2](✉), Sara Silva[1,2,3], and Carlos M. Fonseca[1]

[1] CISUC, Department of Informatics Engineering, University of Coimbra, 3030-290 Coimbra, Portugal
{icpg,cmfonsec}@dei.uc.pt

[2] BioISI - Biosystems and Integrative Sciences Institute, Faculty of Sciences, University of Lisbon, Campo Grande, 1749-016 Lisbon, Portugal
sara@fc.ul.pt

[3] NOVA IMS, Universidade Nova de Lisboa, 1070-312 Lisbon, Portugal

Abstract. Geometric Semantic Genetic Programming (GSGP) is a recently proposed form of Genetic Programming (GP) that searches directly the space of the underlying semantics of the programs. The fitness landscape seen by the GSGP variation operators is unimodal with a linear slope by construction and, consequently, easy to search. Despite this advantage, the offspring produced by these operators grow very quickly. A new implementation of the same operators was proposed that computes the semantics of the offspring without having to explicitly build their syntax. This allowed GSGP to be used for the first time in real-life multidimensional datasets. GSGP presented a surprisingly good generalization ability, which was justified by some properties of the geometric semantic operators. In this paper, we show that the good generalization ability of GSGP was the result of a small implementation deviation from the original formulation of the mutation operator, and that without it the generalization results would be significantly worse. We explain the reason for this difference, and then we propose two variants of the geometric semantic mutation that deterministically and optimally adapt the mutation step. They reveal to be more efficient in learning the training data, and they also achieve a competitive generalization in only a single operator application. This provides a competitive alternative when performing semantic search, particularly since they produce small individuals and compute fast.

Keywords: Geometric semantic genetic programming · Generalization · Overfitting · Pharmacokinetics · Drug discovery

1 Introduction

Geometric Semantic Genetic Programming (GSGP) [8] is a recently proposed form of Genetic Programming (GP) [6] that searches directly the space of the underlying semantics of the programs. One of the most interesting properties of GSGP is that the fitness landscape seen by its variation operators is a cone by construction, and consequently easy to search. Despite this advantage, the individuals produced by these operators are always bigger than their parents. Since this growth is rather quick, GSGP ends up being hard to use in practice, specially in real-life multidimensional datasets. To counteract this, a new implementation of the same geometric semantic operators was proposed by Vanneschi et al. [10]. In this implementation, the semantics of the offspring can be determined without having to explicitly build their syntax. This allowed GSGP to be used for the first time in real-life multidimensional datasets. Results have shown that besides the expected good performance on the training data, GSGP also presented a surprisingly good generalization ability. This generalization ability was justified by the authors as a result of some properties of the geometric semantic operators. However, their implementation of the mutation operator [10] presented a small deviation from the original definition [8] that is still valid under the geometric semantic framework. This implementation is available in the free open-source GSGP C++ library [2]. In our work we study the effect of both implementations of the geometric semantic mutation on the generalization ability of GSGP. We also propose and test two new variations of the geometric semantic mutation which are able to provide an optimal mutation step adaptation.

The paper is organized as follows. Section 2 contextualizes GSGP. Section 3 describes the experimental setup. Section 4 presents and discusses the effect of both implementations of the mutation operator on the generalization ability of GSGP. Section 5 proposes and discusses the results of the two new geometric semantic mutation operators, and Sect. 6 concludes.

2 Geometric Semantic Genetic Programming

Moraglio et al. [8] recently proposed a new GP formulation called Geometric Semantic Genetic Programming (GSGP). GSGP derives its name from the fact that it is formulated under a geometric framework [7] and from the fact that it operates directly in the space of the underlying semantics of the individuals. In this context, semantics is defined as the outputs of an individual over a set of data instances. Perhaps the most interesting property of GSGP is that the fitness landscape seen by its variation operators is always unimodal with a linear slope (cone landscape) by construction. This implies that there are no local optima, i.e., with the exception of the global optimum, every point in the search space has at least one neighbor with better fitness and that neighbor is reachable through the application of the variation operators. The immediate consequence of this type of landscape is that it is easy to search. A drawback of GSGP is that its operators always produce offspring bigger that their parents. Since our work is on regression problems, the GSGP operators presented here are for real-value semantics. For proofs and further details the reader is referred to [8].

Definition 1 (Geometric Semantic Crossover). *Given two parent functions* $T_1, T_2 : \mathbb{R}^n \rightarrow \mathbb{R}$*, the geometric semantic crossover returns the real function* $T_{XO} = (T_1 \cdot T_R) + ((1 - T_R) \cdot T_2)$*, where* T_R *is a random real function whose output values range in the interval* $[0, 1]$.

From the crossover definition it follows that every offspring is bigger than its parents combined. This leads to exponential individual growth.

Definition 2 (Geometric Semantic Mutation). *Given a parent function* $T : \mathbb{R}^n \rightarrow \mathbb{R}$*, the geometric semantic mutation with mutation step ms returns the real function* $T_M = T + ms \cdot (T_{R1} - T_{R2})$*, where* T_{R1} *and* T_{R2} *are random real functions.*

For the mutation operator, the individual growth produced is linear. The continuous individual growth produced by both operators renders GSGP hard to use in practice, specially in real-life multidimensional datasets. Vanneschi et al. [10] tackled this issue by providing a different implementation of these operators. In this implementation, the semantics of the offspring can be determined without having to explicitly build their syntax. Consequently, Vanneschi et al. [10] were able to use GSGP for the first time in real-life multidimensional datasets. They reported competitive performance both on training and testing data. The arguments presented for the good performance on testing data will be presented and discussed in Sect. 4.2.

However, the implementation of the mutation operator of Vanneschi et al. [10] had a small deviation from the original definition. Their implementation imposed that the random subtrees generated (T_{R1} and T_{R2}), always had a logistic function as their root node. This implies that the output of each random subtree ranges in the interval $[0, 1]$ and that, consequently, the output resulting from subtracting these random subtrees ranges in the interval $[-1, 1]$. As the mutation operator applies a mutation step, the final output added to each parent always ranges in the interval $[-ms, ms]$. Looking back at the original definition of the geometric semantic mutation [8], there is no defined range for the outputs of the random subtrees. It should be noted that this small implementation deviation is still valid under the geometric semantic framework. This deviation was not explicit in their work but was confirmed upon contact, and it is also the implementation made available in the GSGP library [2]. For clarification purposes, we will refer to the original mutation definition as Unbounded Mutation (UM) and to the alternative mutation implementation as Bounded Mutation (BM). In the end, BM applies a structural bound on the perturbation applied to the parent. This bound holds independently of the data (training or testing). We explore the effects of using a structural bound in Sect. 4.

3 Experimental Setup

To provide a fair comparison between unbounded and bounded mutation, our experimental setup is similar to the one of Vanneschi et al. [10]. The experimental parameters are provided in Table 1. The mutation step for GSGP is set to 1 as this showed better results in the preliminary testing than the value of 0.001 used by Vanneschi et al. [10]. Experiments are run for 500 generations because that is where the statistical comparisons were made in the mentioned work. Standard GP and Semantic Stochastic Hill Climber (SSHC) [8] are used as baselines for comparison. GSGP without crossover is also tested (GSGP NC). As this work studies the effects of unbounded and bounded mutations (UM and BM respectively), each method is tested with both mutations. Therefore, the variants tested are: GSGP UM and BM; GSGP NC UM and BM and SSHC UM and BM. All claims of statistical significance are based on Mann-Whitney U tests, with Bonferroni correction, and considering a significance level of $\alpha = 0.05$. For each dataset 30 different random partitions are used. Each variant uses the same 30 partitions. Experiments are conducted on the same two multidimensional symbolic regression real-life datasets used by Vanneschi et al. [10]. These datasets are the Bioavailability (hereafter Bio) and the Plasma Protein Binding (hereafter PPB). They have, respectively, 359 instances and 241 features, and 131 instances and 626 features. For a detailed description of these datasets the reader is referred to Archetti et al. [1] and Vanneschi et al. [10].

Table 1. GSGP and Standard GP parameters used in the experiments

Parameter	Value
Runs	30
Generations	500
Population size	100
Training - Testing division	70 % - 30 %
Fitness	Root Mean Squared Error
GSGP crossover	SGXM [8], probability 0.5
GSGP mutation	SGMR [8], probability 0.5
Standard GP crossover	Standard subtree crossover, probability 0.9
Standard GP mutation	Standard subtree mutation, probability 0.1
Tree initialization	Ramped Half-and-Half, maximum depth 6
Function set	+, -, *, and /, protected as in [9]
Terminal set	Input variables, no constants
Parent selection	Tournament of size 4
Elitism	Best individual always survives
Maximum tree depth	None

4 Experimental Study

All the evolution plots presented in the next sections are based on the median over 30 runs of the training and testing error of the best individuals in the training data. The median was preferred over the mean since it is more robust to outliers. Section 4.1 presents the results and Sect. 4.2 discusses the generalization ability.

4.1 Results

Figure 1 presents the training and testing error evolution plots in both datasets. This figure also shows the adaptive mutation step variants (SSHC AUM, SSHC DAUM, SSHC ABM and SSHC DABM) that will be presented and discussed in Sect. 5.

Starting with the comparisons against Standard GP, it was confirmed that GSGP BM generalizes better in both datasets (p-values: Bio 1.794×10^{-6} and

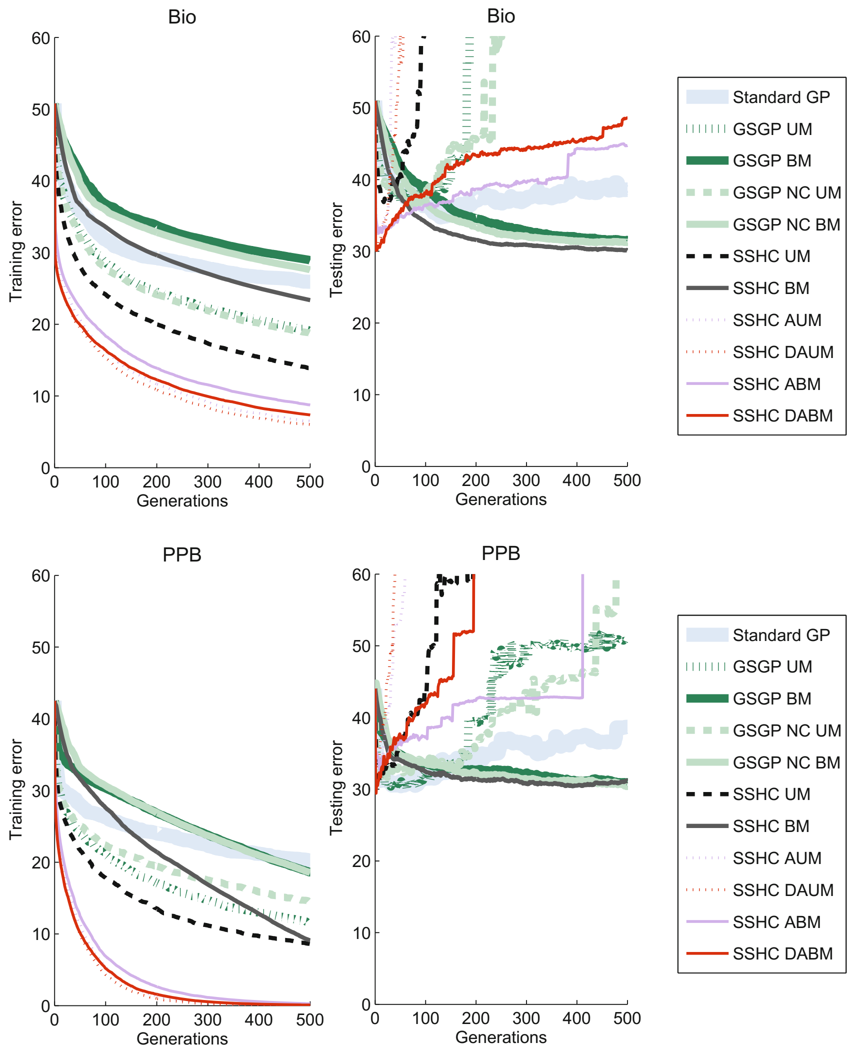

Fig. 1. Bio (top) and PPB (bottom) training and testing error evolution plots

PPB 5.121×10^{-4}). These same conclusions were already presented by Vanneschi et al. [10]. In terms of training error, Standard GP is superior to GSGP BM in the Bio dataset (p-value 5.434×10^{-5}) and no statistically significant differences were found in the PPB dataset. On the other hand, Standard GP is superior to GSGP UM in terms of testing error in the Bio dataset (p-value 9.273×10^{-4}), while no statistically significant differences were found in the PPB dataset. GSGP UM is, however, superior to Standard GP in terms of training error in both datasets (p-values: Bio 2.872×10^{-11} and PPB 2.872×10^{-11}).

It can be observed in evolution plots that Standard GP and GSGP UM overfit the training data, while GSGP BM generalizes well. In these datasets, Standard GP is known to overfit (e.g., [4]) and GSGP BM has been recently shown, and also confirmed here, to generalize well [10]. On the other hand, there is a clear distinction between GSGP BM and UM, as the latter quickly overfits the training data. It generalizes even worse than Standard GP in the Bio dataset. This same distinction between BM and UM occurs with GSGP NC and with SSHC. As a general trend, the BM variants (GSGP BM, GSGP NC BM and SSHC BM) generalize well, while the UM variants (GSGP UM, GSGP NC UM and SSHC UM) overfit the training data. This discrepancy between the generalization ability of the UM and BM variants is discussed in the next section.

Vanneschi et al. [10] mentioned that GSGP requires a relatively high mutation probability in order to explore the search space more efficiently. Indeed, our results show only small differences between using the crossover operator (GSGP UM and BM) or not using it at all (GSGP NC UM and BM). Statistically, there are no differences in terms of generalization, in any comparison with the same mutation operator. In terms of training error the results are not consistent: GSGP NC BM is significantly better than GSGP BM on the Bio dataset (p-value 3.955×10^{-5}); GSGP UM is significantly better than GSGP NC UM on the PPB dataset (p-value 4.734×10^{-11}); and no other statistically significant differences were found. However, a possible inefficiency of the crossover operator should be expected. This operator can only produce an offspring which improves over both parents when the target semantics are between (even if partially) the semantics of the parents. Without an explicit semantic diversity control of the population and a mate selection procedure that takes the target semantics into account, the crossover operator may be inefficient. This inefficiency may also increase with larger semantic spaces, i.e., as the number of data instances increases. From these experiments, it can be concluded that the crossover operator can be skipped altogether since it does not significantly and consistently improve the search outcome (in testing or training error). It also presents the disadvantage of exponentially increasing the size of the individuals, as opposed to the linear increase with the mutation operator.

On a final note, the evolution plots also show that the SSHC variants consistently learn faster than the GSGP and GSGP NC variants with the same mutation operator. This should be expected as the semantic space has no local optima and consequently the search can be focused around the best individual in the population. This leads to a faster decrease in the training error and, in the case of SSHC BM, to a faster generalization to unseen data.

4.2 Discussion on Generalization Ability

From the results presented in Sect. 4.1, it is clear that what differentiates the several methods in terms of generalization is the usage of a bounded or unbounded mutation (BM and UM respectively). BM variants generalize well, while UM variants overfit the training data.

The GSGP implementation of Vanneschi et al. [10] used a BM and reached the same conclusions regarding its competitive generalization. They justified this generalization ability by considering some properties of the geometric semantic operators. Particularly, they remarked that the geometric properties of those operators hold independently of the data on which the evaluation is taken place and consequently they also hold on testing data. For the crossover operator this implies that each offspring produced also stands between its parents in the testing data semantic space. Therefore, in the worst case, each offspring is not worse than the worst of its parents on testing data. The implication for the mutation operator is that the perturbation that each offspring produces is bounded, also in the testing data semantic space, by the mutation step (ms). Specifically, the semantic variation on the testing data also ranges in the interval $[-ms, ms]$. Therefore, Vanneschi et al. [10] concluded that the geometric semantic operators guarantee that a possible worsening of the testing error is bounded and consequently that these operators help control overfitting.

As seen before, the usage of a bounded or unbounded mutation was crucial in determining the generalization achieved. The BM operator was able to produce a competitive generalization by guaranteeing bounded and small perturbations on the testing data. This was crucial to generalize well. However, it is clear that perturbations that increase the testing error are always possible. It is also clear that if these perturbations were a significant majority of the applications of the operator then overfitting would be inevitable. Therefore, it can be concluded that after reaching what can be thought of as a generalization plateau (the point where it seems that no further induction can be performed with the available data), the BM operator generates about half of its perturbations in the decreasing testing error direction and the other half in the increasing testing error direction. These perturbations end up compensating each other and therefore creating the relatively smooth generalization plateau. On the other hand, the UM operator performed badly in terms of generalization. Since in this operator the perturbations produced on the testing data can be arbitrarily large, a single application of a mutation that results in overfitting (decreases training error but increases testing error) can have an arbitrarily large increase in the testing error. This results in considerable uncertainty in the testing error evolution. This effect may be more noticeable in regression problems since any data instance can have an arbitrarily large error contribution, as opposed to classification problems where normally the error is bounded for each data instance.

For these reasons, the BM operator seems more robust and should be preferred over the UM operator.

For the reasons already mentioned in Sect. 4.1, the crossover operator had little effect in the results. However, in principle, the crossover operator should be riskier in terms of generalization than the BM operator. This is because the variation in the testing data semantic space, although bounded by the testing semantics of the parents, can still be arbitrarily large. This results from the fact that the parents can be, in terms of testing data semantics, very far apart. Since there is no way of knowing if the parents are close or far apart in the testing data semantic space, the bounds (defined by the semantics of both parents), on testing data, are not useful in practice. This is another disadvantage of the crossover operator, following the exponential growth of the offspring produced and the low efficiency in terms of search.

Although the generalization achieved by the GSGP with bounded mutation is very competitive, the issue of the size of the solutions generated by these geometric semantic operators remains. As mentioned in Sect. 2, using crossover in GSGP translates into an exponential growth of the individuals. In our experimental study, individuals in GSGP reach several millions of nodes with only around 20 generations conducted. This raises the question: how can such large/complex individuals (models) achieve such competitive generalization? Some interpretations of theories such as Occam's razor and the Minimum Description Length principle state that smaller/less complex models generalize better. Consequently, and in light of this view, this result would be improbable, if not impossible. How can this be? A possible answer may lie in ensemble learning. Ensemble learning is a Machine Learning paradigm in which several models are created and combined to produce a final model. Dietterich [3] provided three reasons as to why constructing an ensemble of models may be superior to constructing a single model. The first two reasons are computational and representational. The computational reason is related to the difficulties in searching the search space, such as getting stuck in a local optima. The representational reason arises when the true target function cannot be represented by any of the hypotheses in the search space. These first two reasons are not discussed in detail as they are not relevant to GSGP, respectively because the semantic space has no local optima and because in GSGP (and in traditional GP) any hypothesis can be represented that could also be represented by an ensemble. The last reason is the one which is relevant to GSGP and to generalization in general. It is a statistical reason and it is related to the fact that several different models can have a similar or even the same training data performance. This is essentially a model selection problem. Which model should be chosen? There is no way of knowing which model will generalize better. Ensemble learning tackles this issue by combining several accurate models, which reduces the risk of the final model being overfitted. Even if some overfitted models are present in the ensemble, their negative contribution to the final model will be reduced since the final model will also include contributions from models which generalize well. It is a common result in ensemble learning to have large ensembles which achieve competitive generalization. Therefore, and in general, large/complex models (individuals) can also generalize well depending on how they are constructed.

GSGP can be seen as an ensemble learning method, since its operators always combine existing individuals independently to produce new individuals. The crossover operator combines two parents with a randomly generated individual and the mutation operator combines one parent with another randomly generated individual (the individual which results from subtracting the two random subtrees). We can think of these parents and randomly generated individuals as full models themselves. This interestingly relates back to ensemble learning, where a necessary condition for its positive outcome is that the ensemble has a mix of accurate and diverse models [5]. In GSGP we can think of the parents as the accurate models (as they have survived during the evolution) and the randomly generated individuals as providing the also needed diversity. GSGP may derive some of its competitive generalization from this. If, for instance, we consider the application of two sequential mutations, it follows that:

$$\mathbf{P + R1 * ms + R2 * ms}$$

where P is the initial parent, R1 and R2 are the two randomly generated individuals and ms is the mutation step. Consequently, considering only the usage of the mutation operator, GSGP can be seen as a weighted sum combination of models (we can consider that the initial parent has a weight of 1).

In the end, GSGP successfully combines elements from ensemble learning (implicitly) and from the geometric semantic framework. Combining several models to incrementally produce new models has roots in ensemble learning. This allows to reduce the model selection risk by offsetting possible bad models. On the other hand, the combination of a structurally bounded mutation (BM) and a small mutation step can further reduce the issue of adding bad models by guaranteeing that their contribution will be small.

5 Adapting the Mutation Step

As discussed in the previous section, the mutation step can play a role in reducing the risk of overfitting. When it comes to learning more efficiently, the geometric semantic mutation can be improved by adapting its step. It is possible to deterministically compute the optimal mutation step for each application of the operator. The description of how this can be accomplished is presented in Sect. 5.1. Section 5.2 presents and discusses the results.

5.1 Optimal Step Adaptation

The geometric semantic mutation can be seen as a linear combination of two elements: the parent P, and the random individual RI which results from subtracting the two random subtrees. Since RI is multiplied by the mutation step ms, we want to find a mutation step such that:

$$\mathbf{P + RI * ms = t}$$

where P and RI are semantic vectors and t is the target vector of the data. Since the parent is not influenced by any weight, we can rewrite this as:

$$\mathbf{RI} * \mathbf{ms} = (\mathbf{t} - \mathbf{P})$$

where we reach a general linear system:

$$\mathbf{Ax} = \mathbf{y}$$

The resolution of which can be performed deterministically and optimally by the application of the Moore-Penrose pseudoinverse (hereafter simply inverse). This inverse computes the mutation step which minimizes the error in the training data for each specific combination of RI, P and t. We will call this modification of the mutation operator as Adaptive Mutation (AM). As this work has studied the effects of bounded and unbounded mutations, we can divide the AM as: Adaptive Unbounded Mutation (AUM) and Adaptive Bounded Mutation (ABM).

Following a similar reasoning, another mutation operator can be devised. We can consider the possibility of adding a weight to the parent and adjusting both weights with the inverse. Let *pw* be the parent weight. Consequently:

$$\mathbf{P} * \mathbf{pw} + \mathbf{RI} * \mathbf{ms} = \mathbf{t}$$

This new semantic mutation operator will be called as Doubly Adaptive Mutation (DAM) and it can also be divided as: Doubly Adaptive Unbounded Mutation (DAUM) and Doubly Adaptive Bounded Mutation (DABM). The inverse method could also be used to perform a linear combination of more than two weighted individuals.

5.2 Results and Discussion

The newly devised operators were tested with the SSHC (more efficient than GSGP, see Sect. 4.1) and consequently its variants were named: SSHC AUM, SSHC DAUM, SSHC ABM and SSHC DABM. Figure 1 (in Sect. 4.1) shows the evolution of training and testing error for these adaptive variants. They reveal to be superior in terms of learning the training data when compared to the SSHC variants without adaptive mutation step (SSHC UM and SSHC BM). This was expected, since the step adaptation is optimal for each application of the operators. In terms of generalization, these variants quickly overfit. In light of the analysis made in Sect. 4.2, this quick overfitting should also be expected, as there is no structural bound coupled with a small mutation step and consequently no overfitting risk reduction. Since in these variants the weights can be arbitrarily large, the benefits of using a structural bound (SSHC ABM and SSHC DABM) are lost.

However, an interesting property can be found when looking closely at the initial generations. Figure 2 presents the testing error evolution on the first 10 generations. It shows that these variants achieve a competitive generalization in only a single application of the mutation operators. This is particularly clear in the SSHC DAUM and SSHC DABM variants. It was expected that these two variants fit the training data more easily when compared to the other two variants (SSHC AUM and SSHC ABM), since they have an extra degree of freedom (the parent weight). Further testing is needed to determine if this property holds across other datasets. If it holds, then these mutation variants become a competitive alternative when performing semantic search, particularly since they produce small individuals and compute fast. They also raise no issues in constructing/reconstructing large individuals, as opposed to what may happen with the GSGP variants.

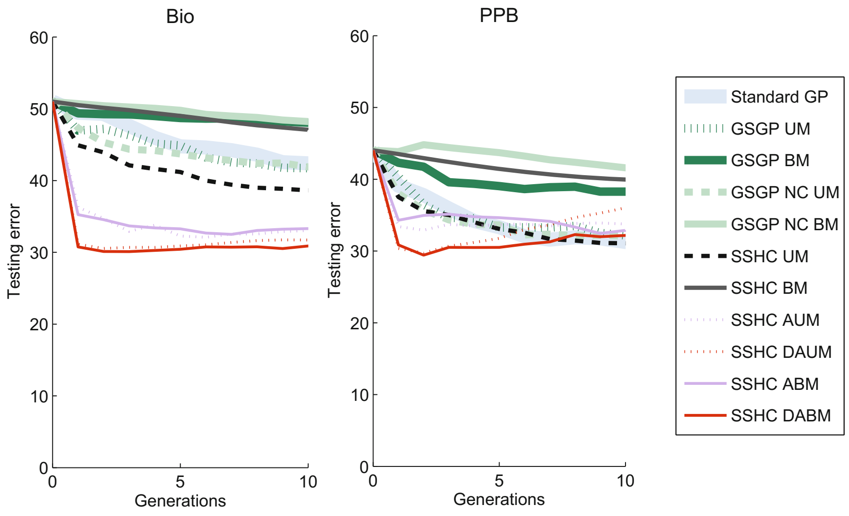

Fig. 2. Bio (left) and PPB (right) testing error evolution plots for the first 10 generations

6 Conclusions

In this work we have studied the generalization ability of Geometric Semantic Genetic Programming (GSGP), by analyzing the effects of two different implementations of the geometric semantic mutation. These implementations differ on the existence or not of a guaranteed bound on the semantic variation across both training and unseen data. Results showed that the generalization ability of GSGP differs significantly depending on whether or not this bound is used. On the tested datasets, the unbounded mutation highly overfitted the training data, while the bounded mutation achieved a competitive generalization. We have also expanded on previously reported geometric semantic arguments as to why GSGP may generalize well. Furthermore, we provided an explanation as to why such large solutions like the ones produced by GSGP can generalize competitively, by discussing how GSGP may relate with ensemble learning.

We have also proposed two new variants of the geometric semantic mutation. These new operators are able to deterministically compute the optimal mutation step for each application of the operator. They have shown to be more efficient in learning the training data, and they also achieve a competitive generalization in only a single operator application. This provides a competitive alternative when performing semantic search, particularly since they produce small individuals and compute fast.

Acknowledgments. This work was partially supported by national funds through FCT under contract UID/Multi/04046/2013 and projects PTDC/EEI-CTP/2975/2012 (MaSSGP), PTDC/DTP-FTO/1747/2012 (InteleGen) and EXPL/EMS-SIS/1954/2013 (CancerSys). The first author work is supported by FCT, Portugal, under the grant SFRH/BD/79964/2011.

References

1. Archetti, F., Lanzeni, S., Messina, E., Vanneschi, L.: Genetic programming for computational pharmacokinetics in drug discovery and development. Genet. Program. Evolvable Mach. **8**(4), 413–432 (2007)
2. Castelli, M., Silva, S., Vanneschi, L.: A C++ framework for geometric semantic genetic programming. Genet. Program. Evolvable Mach. **15**, 73–81 (2014)
3. Dietterich, T.G.: Ensemble methods in machine learning. In: Kittler, J., Roli, F. (eds.) MCS 2000. LNCS, vol. 1857, pp. 1–15. Springer, Heidelberg (2000)
4. Gonçalves, I., Silva, S.: Balancing learning and overfitting in genetic programming with interleaved sampling of training data. In: Krawiec, K., Moraglio, A., Hu, T., Etaner-Uyar, A.Ş., Hu, B. (eds.) EuroGP 2013. LNCS, vol. 7831, pp. 73–84. Springer, Heidelberg (2013)
5. Hansen, L.K., Salamon, P.: Neural network ensembles. IEEE Trans. Pattern Anal. Mach. Intell. **12**(10), 993–1001 (1990)
6. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection (Complex Adaptive Systems), 1st edn. The MIT Press, Cambridge (1992)
7. Moraglio, A.: Towards a geometric unification of evolutionary algorithms. Ph.D. thesis, Department of Computer Science, University of Essex, UK, November 2007
8. Moraglio, A., Krawiec, K., Johnson, C.G.: Geometric semantic genetic programming. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) PPSN 2012, Part I. LNCS, vol. 7491, pp. 21–31. Springer, Heidelberg (2012)
9. Poli, R., Langdon, W.B., McPhee, N.F., Koza, J.R.: A field guide to genetic programming (2008). Lulu.com
10. Vanneschi, L., Castelli, M., Manzoni, L., Silva, S.: A new implementation of geometric semantic GP and its application to problems in pharmacokinetics. In: Krawiec, K., Moraglio, A., Hu, T., Etaner-Uyar, A.Ş., Hu, B. (eds.) EuroGP 2013. LNCS, vol. 7831, pp. 205–216. Springer, Heidelberg (2013)