A New Implementation of Geometric Semantic GP and Its Application to Problems in Pharmacokinetics

Leonardo Vanneschi[1,2,3], Mauro Castelli[1,2], Luca Manzoni[3], and Sara Silva[2,4]

[1] ISEGI, Universidade Nova de Lisboa, 1070-312 Lisboa, Portugal
[2] INESC-ID, IST / Universidade Técnica de Lisboa, 1000-029 Lisboa, Portugal
[3] D.I.S.Co., Università degli Studi di Milano-Bicocca, 20126 Milano, Italy
[4] CISUC, Universidade de Coimbra, 3030-290 Coimbra, Portugal
lvanneschi@isegi.unl.pt

Abstract. Moraglio et al. have recently introduced new genetic operators for genetic programming, called geometric semantic operators. These operators induce a unimodal fitness landscape for all the problems consisting in matching input data with known target outputs (like regression and classification). This feature facilitates genetic programming evolvability, which makes these operators extremely promising. Nevertheless, Moraglio et al. leave open problems, the most important one being the fact that these operators, by construction, always produce offspring that are larger than their parents, causing an exponential growth in the size of the individuals, which actually renders them useless in practice. In this paper we overcome this limitation by presenting a new efficient implementation of the geometric semantic operators. This allows us, for the first time, to use them on complex real-life applications, like the two problems in pharmacokinetics that we address here. Our results confirm the excellent evolvability of geometric semantic operators, demonstrated by the good results obtained on training data. Furthermore, we have also achieved a surprisingly good generalization ability, a fact that can be explained considering some properties of geometric semantic operators, which makes them even more appealing than before.

1 Introduction

In the last few years researchers have dedicated several efforts to the definition of Genetic Programming (GP) [5,8] methods or systems based on the semantics of the solutions, where by semantics we generally intend the behaviour of a program once it is executed, or more particularly the set of its output values on input training data [9]. In particular, very recently new genetic operators, called geometric semantic operators, have been proposed by Moraglio et al. [10]. These operators have the interesting property of inducing a unimodal fitness landscape on any problem consisting in finding the match between a set of input data and a set of known outputs (like for instance in regression and classification). As a consequence, in principle all these problems should be easily solvable by GP [8], independently of how complex they are. Nevertheless, as stated by Moraglio et al. [10], these operators have a serious limitation: by construction, they always produce offspring that are approximately the double size of their parents (expressed as the total number of tree nodes), and this makes the size of the individuals in the population grow exponentially with generations. In this way, after a few

K. Krawiec et al. (Eds.): EuroGP 2013, LNCS 7831, pp. 205–216, 2013.

generations the population is composed by individuals so big that the computational cost of evaluating their fitness is unmanageable. This limitation makes these operators impossible to use in practice, in particular on complex real-life applications.

The solution suggested [10] to overcome this drawback is to integrate in the GP algorithm a "simplification" phase, aimed at transforming each individual in the population into an equivalent (i.e. with the same semantics) but smaller one. Even though this is an interesting and challenging study, depending on the language used to code individuals simplification can be very difficult, and it is often a very time consuming task. For this reason, in this paper we propose a different strategy to solve the problem: we develop a GP system incorporating an implementation of geometric semantic genetic operators that makes them usable in practice, and does so very efficiently, without requiring any simplification of the individuals during the GP run. With this system we are able, for the first time, to exploit the great potentialities of the geometric semantic operators on complex real-life problems. In order to experimentally validate our new GP system, we apply it to problems in the field of pharmacokinetics, comparing the results with the ones obtained by standard GP. The two problems addressed are the prediction of human oral bioavailability and protein-plasma binding levels of medical drugs.

The paper is organized as follows: Section 2 presents the state of the art concerning the use of semantics to improve GP. Section 3 describes the geometric semantic operators introduced by Moraglio et al., while Section 4 presents our new GP system that overcomes the current limitations of these operators, making them usable and efficient. Section 5 presents the test problems, the experimental settings and the obtained results, offering in particular a discussion about the generalization ability to out-of-sample data provided by geometric semantic operators. Finally, Section 6 concludes the paper and provides hints for future research.

2 Previous Work on Semantics in GP

Several recent contributions have been aimed at using the notion of semantics to study, or improve, GP. McPhee et al. [9] showed that many applications of crossover often do not have any effect on semantics (i.e., basically crossover tends to produce offspring that have the same behaviour as their parents). These results have cast a shadow on the use of traditional genetic operators, and paved the way to the definition of new, semantic-based, operators. A first step in this direction was made by Beadle and Johnson [2], where semantics is used to define an algorithm called Semantically Driven Crossover. With this method, if the offspring are semantically equivalent to their parents, the children are discarded and the crossover is repeated. This process is iterated until semantically different children are found. The authors argue that this results in increased semantic diversity in the evolving population, and a consequent improvement in the GP performance. Nguyen et al. [13] investigated the role of syntactic and semantic locality of crossover in GP. The results showed that improving syntactic locality reduces code growth, which leads to a slight improvement of the ability to generalize. By comparison, improving semantic locality significantly enhances GP performance, reduces code growth and substantially improves the ability of GP to generalize. This work was the starting point in the search for new operators to directly act on semantics.

Under this perspective, Nguyen et al. [11] proposed Semantics Aware Crossover (SAC), a crossover operator promoting semantic diversity, that was subsequently extended to Semantic Similarity based Crossover (SSC) [14] and to Semantic Similarity based Mutation (SSM) [12]. Krawiec [6] proposed a class of *geometric* crossover operators for GP, i.e. operators aimed at making offspring programs semantically intermediate (medial) with respect to parent programs (a property shared also by the operators considered here). Krawiec and Lichocki [7] have also used a notion of semantic distance to propose a crossover operator for GP that is approximately a geometric crossover

3 Geometric Semantic Operators of Moraglio et al.

While the semantically aware methods cited in the previous section often exhibited superior performance with respect to traditional methods, most of them are indirect: search operators act on the syntax of the parents to produce offspring that are only accepted if some semantic criterium is satisfied. To provide operators able to work directly on the semantic, Moraglio et al. introduced new operators [10] To explain the idea, we first provide an example using Genetic Algorithms (GAs). Let us consider a GA problem in which the target solution is known and the fitness of each individual corresponds to its distance to the target (our reasoning holds for any distance measure used). This problem is characterized by a very good evolvability and it is in general easy to solve for GAs. In fact, for instance, if we use point mutation, any possible individual different from the global optimum has at least one neighbor (individual resulting from its mutation) that is closer to the target than itself, and thus is fitter. So, there are no local optima. In other words, the fitness landscape is unimodal and the fitness-distance correlation [3] is equal to 1, because fitness and distance to the goal are identical, which indicates the problem is easy to solve. Similar considerations hold for many types of crossover, including various kinds of geometric crossover [7].

Now, let us consider the typical GP problem of finding a function that maps sets of input data into known target outputs (regression and classification are particular cases). The fitness of an individual for this problem is typically a distance between its predicted output values and the expected ones (error measure). Now let us assume that we are able to find a transformation on the syntax of an individual whose effect is just a random perturbation of one of its predicted output values. In other words, let us assume that we are able to transform an individual G into an individual H whose output values are like the outputs of G, except for one value, that is randomly perturbed. Under this hypothesis, we are able to map the considered GP problem into the GA problem discussed above, in which point mutation is used. So, this transformation, if known, would induce a unimodal fitness landscape on every problem like the considered one (e.g. regressions and classifications), allowing GP to have a good evolvability on those problems, at least on training data. The same also holds for transformations on pairs of solutions that correspond to GA semantic crossovers.

This idea of looking for such operators is very ambitious and extremely challenging: finding those operators would allow us to directly search the space of semantics, at the same time working on unimodal fitness landscapes. Although not without limitations, the work of Moraglio et al. [10] accomplishes this task, defining new operators that have

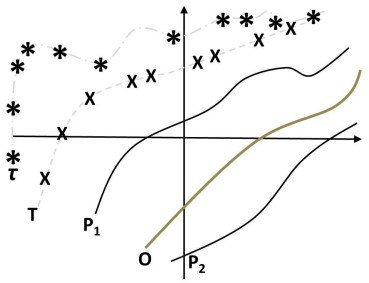

Fig. 1. An illustration of the fact that geometric semantic crossover creates an offspring that is at least not worse than the worst of its parents. In this example, offspring O (which stands between parents P_1 and P_2 in the semantic space by construction) is clearly closer to target T (training points represented by "$\times$" symbols) than parent P_2. In Section 5 we also discuss the geometric properties of this operator on test data, represented by τ (test points represented by "$*$" symbols).

exactly these characteristics. Here we report the definition of the geometric semantic operators as given by Moraglio et al. for real functions domains, since these are the operators we will use in the experimental phase. For applications that consider other types of data, the reader is referred to [10].

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* $[0, 1]$.

The interested reader is referred to [10] for a formal proof of the fact that this operator corresponds to a geometric crossover on the semantic space, in the sense that it produces an offspring that stands between its parents in this space. We do not report the proof here, but we limit ourselves to remark that, even without a formal proof, we can have an intuition of it considering that the (only) offspring generated by this crossover has a semantic vector that is a linear combination of the semantics of the parents with random coefficients included in $[0, 1]$ and whose sum is equal to 1. Moraglio et al. [10] also prove an interesting consequence of this fact: the fitness of the offspring cannot be worse than the fitness of the worst of its parents. Also in this case we do not replicate the proof here, but we limit ourselves to giving a visual intuition of this property: in Figure 1 we represent a simple two-dimensional semantic space in which we draw a target function T (training points are represented by "$\times$" symbols), two parents P_1 and P_2 and one of their offspring O (which by construction stands between its parents), plus a test set (composed by test points represented by "$*$" symbols) that will be discussed in the final part of Section 5. It is immediately apparent from Figure 1 that O is closer to T than P_2 (which is the worst parent in this case). The generality of this property is proven in [10].

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.*

Moraglio et al. [10] formally prove that this operator corresponds to a box mutation on the semantic space, and induces a unimodal fitness landscape. Even without a formal

proof it is not difficult to have an intuition of it, considering that each element of the semantic vector of the offspring is a "weak" perturbation of the corresponding element in the parent's semantics. We informally define this perturbation as "weak" because it is given by a random expression centred on zero (the difference between two random trees). Nevertheless, by changing parameter ms, we are able to tune the "step" of the mutation, and thus the importance of this perturbation.

We highlight the fact that these operators create offspring that contain the complete structure of the parents, plus one or more random trees and some additional arithmetic operators: the size of the offspring is thus clearly much larger than the size of their parents. The exponential growth of the individuals in the population, demonstrated by Moraglio et al. [10], makes these operators unusable in practice: after a few generations the population becomes unmanageable because the fitness evaluation process becomes unbearably slow. The solution suggested in [10] consists in performing an automatic simplification step after each generation in which the individuals are replaced by (hopefully smaller) semantically equivalent ones. However, this additional step adds to the computational cost of GP and is only a partial solution to the progressive size growth. Last but not least, depending on the particular language used to code individuals and the used primitives, automatic simplification can be a very hard task.

In the next section, we present a novel implementation of GP using these operators that overcomes this limitation, making them efficient without performing any simplification step.

4 Novel Implementation of Geometric Semantic GP

Here we describe the proposed implementation of Geometric Semantic GP. Note that, although we describe the algorithm assuming the representation of the individuals is tree based, the implementation fits any other type of representation.

In a first step, we create an initial population of (typically random) individuals, exactly as in standard GP. We store these individuals in a table (that we call P from now on) as shown in Figure 2(a), and we evaluate them. To store the evaluations we create a table (that we call V from now on) containing, for each individual in P, the values resulting from its evaluation on each fitness case (in other words, it contains the semantics of that individual). Hence, with a population of n individuals and a training set of k fitness cases, table V will be made of n rows and k columns.

Then, for every generation, a new empty table V' is created. Whenever a new individual T must be generated by crossover between selected parents T_1 and T_2, T is represented by a triplet $T = \langle \mathrm{ID}(T_1), \mathrm{ID}(T_2), \mathrm{ID}(R) \rangle$, where R is a random tree and, for any tree τ, $\mathrm{ID}(\tau)$ is a *reference* (or memory pointer) to τ (using a C-like notation). This triplet is stored in an appropriate structure (that we call $\mathcal{M}$ from now on) that also contains the name of the operator used, as shown in Figure 2c. The random tree R is created, stored in P, and evaluated in each fitness case to reveal its semantics. The values of the semantics of T are also easily obtained, by calculating $(T_1 \cdot R) + ((1 - R) \cdot T_2)$ for each fitness case, according to the definition of geometric semantic crossover, and stored in V'. Analogously, whenever a new individual T must be obtained by applying mutation to an individual T_1, T is represented by a triplet $T = \langle \mathrm{ID}(T_1), \mathrm{ID}(R_1), \mathrm{ID}(R_2) \rangle$

(stored in $\mathcal{M}$), where R_1 and R_2 are two random trees (newly created, stored in P and evaluated for their semantics). The semantics of T is calculated as $T_1 + ms \cdot (R_1 - R_2)$ for each fitness case, according to the definition of geometric semantic mutation, and stored in V'. In the end of each generation, table V' is copied into V and erased. All the rows of P and $\mathcal{M}$ referring to individuals that are not ancestors[1] of the new population can also be erased. Note that, while $\mathcal{M}$ grows at every generation, by keeping the semantics of the individuals separated we are able to use a table V whose size is independent from the number of generations.

Summarizing, this algorithm is based on the idea that, when semantic operators are used, an individual can be fully described by its semantics (which makes the syntactic component much less important than in standard GP), a concept discussed in depth in [10]. Therefore, at every generation we update table V with the semantics of the new individuals, and save the information needed to build their syntactic structures without explicitly building them. In terms of computational time, we emphasize that the process of updating table V is very efficient as it does not require the evaluation of the entire trees. Indeed, evaluating each individual requires (except for the initial generation) a constant time, which is independent from the size of the individual itself. In terms of memory, tables P and $\mathcal{M}$ grow during the run. However, table P adds a maximum of $2 \times n$ rows per generation (if all new individuals are created by mutation) and table $\mathcal{M}$ (which contains only memory pointers) adds a maximum of n rows per generation. Even if we never erase the "ex-ancestors" from these tables (and never reuse random trees, which is also possible), we can manage them efficiently for several thousands of generations. Let us briefly consider the cost in terms of time and space of evolving a population of n individuals for g generations. At every generation, we need $O(n)$ space to store the new individuals. Thus, we need $O(ng)$ space in total. Since we need to do only $O(1)$ operations for any new individual (since the fitness can be computed using the fitness of the parents), the time complexity is also $O(ng)$. Thus, we have a linear space and time complexity with respect to population size and number of generations.

The final step of the algorithm is performed after the end of the last generation. In order to reconstruct the individuals, we may need to "unwind" our compact representation and make the syntax of the individuals explicit. Therefore, despite performing the evolutionary search very efficiently, in the end we may not avoid dealing with the large trees that characterize the standard implementation of geometric semantic operators. However, most probably we will only be interested in the best individual found, so this unwinding (and recommended simplification) process may be required only once, and it is done offline after the run is finished. This greatly contrasts with the solution proposed by Moraglio et al. of building and simplifying every tree in the population at each generation online with the search process. If we are not interested in the form of the optimal solution, we can avoid the "unwinding phase" and we can evaluate an unseen input with a time complexity is $O(ng)$. In this case the the individual is used as a "black-box" which, in some cases, may be sufficient.

Excluding the time needed to build and simplify the best individual, the proposed implementation allowed us to evolve populations for thousands of generations with a

[1] We abuse the term "ancestors" to designate not only the parents but also the random trees used to build an individual by crossover or mutation.

Id	Individual
T_1	$x_1 + x_2x_3$
T_2	$x_3 - x_2x_4$
T_3	$x_3 + x_4 - 2x_1$
T_4	x_1x_3
T_5	$x_1 - x_3$

(a)

Id	Individual
R_1	$x_1 + x_2 - 2x_4$
R_2	$x_2 - x_1$
R_3	$x_1 + x_4 - 3x_3$
R_4	$x_2 - x_3 - x_4$
R_5	$2x_1$

(b)

Id	Operator	Entry
T_6	crossover	$\langle \mathrm{ID}(T_1), \mathrm{ID}(T_4), \mathrm{ID}(R_1) \rangle$
T_7	crossover	$\langle \mathrm{ID}(T_4), \mathrm{ID}(T_5), \mathrm{ID}(R_2) \rangle$
T_8	crossover	$\langle \mathrm{ID}(T_3), \mathrm{ID}(T_5), \mathrm{ID}(R_3) \rangle$
T_9	crossover	$\langle \mathrm{ID}(T_1), \mathrm{ID}(T_5), \mathrm{ID}(R_4) \rangle$
T_{10}	crossover	$\langle \mathrm{ID}(T_3), \mathrm{ID}(T_4), \mathrm{ID}(R_5) \rangle$

(c)

Fig. 2. Illustration of the example described in Section 4. (a) The initial population P; (b) The random trees used by crossover; (c) The representation in memory of the new population P'

considerable speed up with respect to standard GP. Future work will provide a comparison of the execution times of the different methods.

Example. Let us consider the simple initial population P shown in table (a) of Figure 2 and the simple pool of random trees that are added to P as needed, shown in table (b). For simplicity, we will generate all the individuals in the new population (that we call P' from now on) using only crossover, which will require only this small amount of random trees. Besides the representation of the individuals in infix notation, these tables contain an identifier (Id) for each individual (T_1, ..., T_5 and R_1, ..., R_5). These identifiers will be used to represent the different individuals, and the individuals created for the new population will be represented by the identifiers T_6, ..., T_{10}.

The individuals of the new population P' are simply represented by the set of entries exhibited in table (c) of Figure 2. This table contains, for each new individual, a *reference* to the ancestors that have been used to generate it and the name of the operator used to generate it (either "crossover" or "mutation"). For example, the individual T_6 is generated by the crossover of T_1 and T_4 and using the random tree R_1.

Let us assume that now we want to reconstruct the genotype of one of the individuals in P', for example T_{10}. The tables in Figure 2 contain all the information needed to do that. In particular, from table (c) we learn that T_{10} is obtained by crossover between T_3 and T_4, using random tree R_5. Thus, from the definition of geometric semantic crossover, we know that it will have the following structure: $(T_3 \cdot R_5) + ((1 - R_5) \cdot T_4)$. The remaining tables (a) and (b), that contain the syntactic structure of T_3, T_4, and R_5, provide us with the rest of the information we need to completely reconstruct the syntactic structure of T_{10}, which is $((x_3 + x_4 - 2x_1) \cdot (2x_1)) + ((1 - (2x_1)) \cdot (x_1x_3))$ and upon simplification becomes $-x_1(4x_1 - 3x_3 - 2x_4 + 2x_1x_3)$.

5 Experimental Study

Problems in Pharmacokinetics. The implementation described in the previous section allows the geometric semantic operators to be used, for the first time, in complex real-life applications. We have chosen two hard regression problems in the field of pharmacokinetics: prediction of human oral bioavailability and prediction of the protein-plasma binding levels of medical drugs. Both have already been tackled by GP in published literature, e.g. [1]. *Human oral bioavailability* (represented as %F) is

the parameter that measures the percentage of the initial orally submitted drug dose that effectively reaches the systemic blood circulation after passing through the liver. Being able to reliably predict the %F value for a potential new drug is outstandingly important, given that the majority of failures in compounds development from the early nineties to nowadays are due to inaccurate predictions of this pharmacokinetic parameter during the drug discovery process [4]. The %F dataset consists of 359 instances, where each instance is a vector of 242 elements (241 molecular descriptor values identifying a drug, followed by the known value of %F for that drug). This dataset is freely available from the GP Benchmarks website, `gpbenchmarks.org`. *Protein-plasma binding level* (represented as %PPB) quantifies the percentage of the initial drug dose that reaches the blood circulation and binds to the proteins of plasma. This measure is fundamental for good pharmacokinetics, both because blood circulation is the major vehicle of drug distribution into human body and since only free (unbound) drugs can permeate the membranes reaching their targets [1]. The %PPB dataset consists of 131 instances, where each instance is a vector of 627 elements (626 molecular descriptor values identifying a drug, followed by the known %PPB for that drug).

Experimental Settings. We have tested our implementation of GP with geometric semantic operators (GS-GP) against a standard GP system (STD-GP). A total of 30 runs were performed with each technique using different randomly generated partitions of the dataset into training (70%) and test (30%) sets. All the runs used populations of 100 individuals allowed to evolve for 2000 generations. It is worth noting that the goal was not to achieve the best possible results, so the parameter settings were not tuned for each technique, save one exception described below. Tree initialization was performed with the Ramped Half-and-Half method [5] with a maximum initial depth equal to 6. The function set contained the four binary arithmetic operators $+$, $-$, $*$, and $/$ protected as in [5]. Fitness was calculated as the root mean squared error (RMSE) between predicted and expected outputs. The terminal set contained the number of variables corresponding to the number of features in each dataset. Tournaments of size 4 were used to select the parents of the new generation. To create new individuals, STD-GP used standard (subtree swapping) crossover [5] and (subtree) mutation [5] with probabilities 0.9 and 0.1, respectively.For GS-GP the mutation rate was 0.5. Preliminary tests have shown that the geometric semantic operators require a relatively high mutation rate in order to be able to effectively explore the search space. The ms step used was 0.001 as in [10]. For both systems, survival was elitist as it always copied the best individual into the next generation. No maximum tree depth limit has been imposed during the evolution.

Experimental Results. The experimental results are reported using curves of the fitness (RMSE) on the training and test sets and boxplots obtained in the following way. For each generation the training fitness of the best individual, as well as its fitness in the test set (that we call test fitness) were recorded. The curves in the plots report the median of these values for the 30 runs. The median was preferred over the mean because of its higher robustness to outliers. The boxplots refer to the fitness values in generation 500, for reasons explained later. In the following text we may use the terms fitness, error and RMSE interchangeably. Plots (a) and (b) of Figure 3 show the evolution of training and test error for STD-GP and GS-GP on the bioavailability problem. They

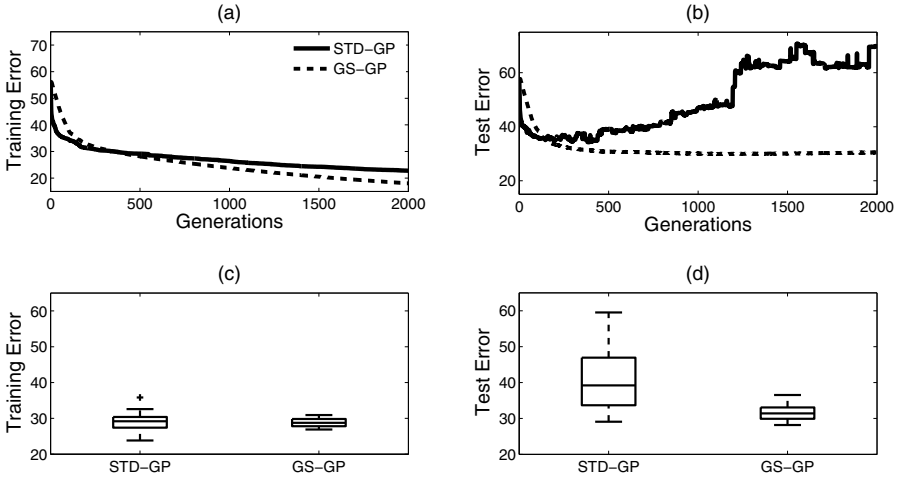

Fig. 3. Results on the bioavailability problem. Evolution of (a) training and (b) test errors for each technique, median of 30 runs. Boxplots of (c) training and (d) test fitness at generation 500. In boxplot (d) STD-GP has three outliers located at 157, 485 and 843 (not shown).

clearly show that GS-GP outperforms STD-GP on both training and test sets. We could informally say that on the training set both techniques "learn well", in the sense that the error curves in plot (a) are steadily decreasing during the whole considered runs, although GS-GP reaches lower error. On the other hand, on the test set, while STD-GP reveals a major loss of generalization ability, GS-GP exhibits a "desirable" behavior where the curve of the test error is regular and monotonically decreasing during the entire evolutionary process. We interpret these results saying that, unlike STD-GP, GS-GP does not overfit the training data on the bioavailability problem. The boxplots (c) and (d) of Figure 3 refer to the fitness values at generation 500, where both techniques have achieved more or less the same training fitness, and GS-GP is not improving test fitness anymore. The boxplots show that GS-GP has less dispersion of results than STD-GP, in particular on the test set. To analyse the statistical significance of these results, a set of tests has been performed. The Kolmogorov-Smirnov 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 has been used under the alternative hypothesis that the samples do not have equal medians. The p-values obtained were 0.70 when training fitness of STD-GP is compared to training fitness of GS-GP and 6.3×10^{-6} when test fitness of STD-GP is compared to test fitness of GS-GP. Therefore, when using the usual significance level $\alpha = 0.01$ (or even if we use a much smaller one), we can state that at generation 500 the studied techniques have comparable fitness on the training data and GS-GP has significantly lower (i.e., better) fitness than STD-GP on the test data. Plots (a) and (b) of Figure 4 show the evolution of training and test error for STD-GP and GS-GP on the protein-plasma binding problem. As in the bioavailability problem, GS-GP reveals to be superior to STD-GP, this time with a wide difference also on the training set, where GS-GP is able to reach a minimal error. The behaviour on the test set is very similar to the one reported for the bioavailability problem.

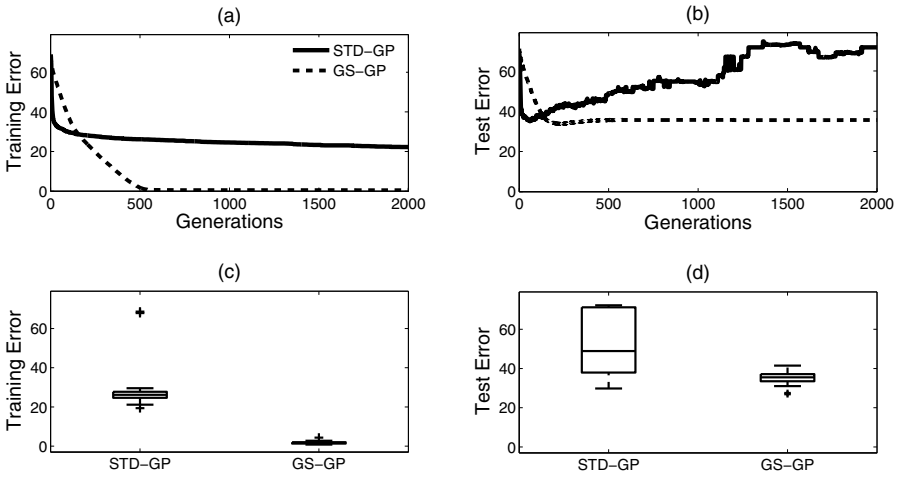

Fig. 4. Results on the protein-plasma binding problem. Evolution of (a) training and (b) test errors for each technique, median of 30 runs. Boxplots of (c) training and (d) test fitness at generation 500. In boxplot (d) STD-GP has six outliers located at 236, 259, 339, 456, 4402 and 5441 (not shown).

The boxplots (c) and (d) of Figure 4 once again refer to the values measured in generation 500, which is more or less the point when GS-GP has stabilized its fitness values both in training and test data. They show similar characteristics to the ones observed on the bioavailability problem, with GS-GP once again exhibiting a lower dispersion of results then STD-GP. They also show that GS-GP performs better than STD-GP in both training and test data. Using the same statistical tests as before, the comparative p-values obtained on the protein-plasma binding problem were 3.0×10^{-11} when training fitness of STD-GP is compared to training fitness of GS-GP and 5.1×10^{-6} when test fitness of STD-GP is compared to test fitness of GS-GP. This allows us to conclude that on the protein-plasma binding problem GS-GP outperforms STD-GP both on the training and test set in a statistically significant way.

Discussion. The good results that GS-GP has obtained on training data were expected: the geometric semantic operators induce an unimodal fitness landscape, which facilitates evolvability. On the other hand, on a first analysis, we have been surprised by the excellent results we have obtained on test data. These results even appeared a bit counterintuitive to us: we were expecting that the good evolvability on training data would entail an overfitting of those data.

However, an explanation of the excellent generalization ability shown by GS-GP on the two studied applications, we have realized one feature of geometric semantic operators that was not so obvious previously. Namely, the geometric properties of those operators hold *independently of the data* on which individuals are evaluated, and thus they hold also on test data. In other words, geometric semantic crossover produces an offspring that stands between the parents also in the semantic space induced by test data. As a direct implication, following exactly the same argument as Moraglio et al. [10],

each offspring is, in the worst case, not worse than the worst of its parents on the test set. This can be seen by looking back at Figure 1, where a simple test set τ is drawn (testing data are represented by "$*$" symbols) and where it is clear that offspring O is closer to data in τ than parent P_2. Analogously, as it happens for training data, geometric semantic mutation produces an offspring that is a "weak" perturbation of its parent also in the semantic space induced by test data (and the maximum possible perturbation is, again, expressed by the ms step). The immediate consequence for the behaviour of GS-GP on test data is that, while geometric semantic operators do not guarantee an improvement in test fitness each time they are applied, they at least guarantee that the possible worsening of the test fitness is bounded (by the test fitness of the worst parent for crossover, and by ms for mutation). In other words, *geometric semantic operators help control overfitting*. Of course overfitting may still happen, as seen in plot (b) of Figure 4 for GS-GP (slight but visible), but there are no big "jumps" in test fitness like the ones observed in plots (b) of Figures 3 and 4 for STD-GP. We remark that, without the novel implementation that allowed us to use geometric semantic GP on these complex real-life problems, this interesting property would probably remained unnoticed.

6 Conclusions and Future Work

New genetic operators, called geometric semantic operators, have been proposed for genetic programming (GP). They have the extremely interesting property of inducing a unimodal fitness landscape for any problem consisting in matching input data to known target outputs (regression and classifications are instances of this general problem). This should make all the problems of this kind easily evolvable by GP. Nevertheless, as demonstrated in the literature, in their first definition these new operators have a strong limitation that makes them unusable in practice: they produce offspring that are larger than their parents, and this results in an exponential growth of the size of the individuals in the population. In this paper we have proposed a novel implementation of GP that uses the geometric semantic operators in a very efficient manner, in terms of computational time and memory. This new GP system evolves the semantics of the individuals without explicitly building their syntax. It does so by keeping a set of trees (of the initial population and the random ones used by geometric semantic crossover and mutation) in memory and a set of pointers to them, representing the "instructions" on how to build the new individuals. Thanks to this compact representation, it was possible to explore, for the first time, the great potential of geometric semantic GP to solve complex real-life problems. We have used two problems in the field of pharmacokinetics: the prediction of human oral bioavailability and the prediction of protein-plasma binding levels of medical drugs. The experimental results demonstrate that the new system outperforms standard GP. Besides the fact that the new GP system has excellent results on training data (which was expected, given that its fitness landscape is unimodal), we were surprised by its excellent generalization ability on the studied applications, which in retrospect can be explained by considering the geometric properties of the new operators. This encourages us to pursue the study: besides additional experimental validations on new data and different applications, we plan to orient our future activity towards more theoretical studies of the generalization ability of geometric semantic GP. In particular,

we are interested in studying the "shape" of the functions produced by semantic GP with respect to the one generated by standard GP, and how this influences the generalization ability. On the more practical side, we are interested in comparing the runtime performance of geometric semantic GP with standard GP also considering the effect of a simplification phase at the end of the algorithm for when a "black-box" individual cannot be used.

Acknowledgments. This work was supported by national funds through FCT under contract Pest-OE/EEI/LA0021/2011 and by projects EnviGP (PTDC/EIA-CCO/103363/2008) and MassGP (PTDC/EEI-CTP/2975/2012), Portugal.

References

1. Archetti, F., Lanzeni, S., Messina, E., Vanneschi, L.: Genetic programming for computational pharmacokinetics in drug discovery and development. Genetic Programming and Evolvable Machines 8, 413–432 (2007)
2. Beadle, L., Johnson, C.: Semantically driven crossover in genetic programming. In: Proc. of the IEEE World Congress on Comput. Intelligence, pp. 111–116. IEEE Press (2008)
3. Jones, T., Forrest, S.: Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In: Proceedings of the Sixth International Conference on Genetic Algorithms, pp. 184–192. Morgan Kaufmann (1995)
4. Kennedy, T.: Managing the drug discovery/development interface. Drug Discovery Today 2(10), 436–444 (1997)
5. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)
6. Krawiec, K.: Medial Crossovers for Genetic Programming. In: Moraglio, A., Silva, S., Krawiec, K., Machado, P., Cotta, C. (eds.) EuroGP 2012. LNCS, vol. 7244, pp. 61–72. Springer, Heidelberg (2012)
7. Krawiec, K., Lichocki, P.: Approximating geometric crossover in semantic space. In: GECCO 2009, July 8-12, pp. 987–994. ACM (2009)
8. Langdon, W.B., Poli, R.: Foundations of Genetic Programming. Springer (2002)
9. McPhee, N.F., Ohs, B., Hutchison, T.: Semantic Building Blocks in Genetic Programming. In: O'Neill, M., Vanneschi, L., Gustafson, S., Esparcia Alcázar, A.I., De Falco, I., Della Cioppa, A., Tarantino, E. (eds.) EuroGP 2008. LNCS, vol. 4971, pp. 134–145. Springer, Heidelberg (2008)
10. Moraglio, A., Krawiec, K., Johnson, C.G.: Geometric Semantic Genetic Programming. In: Coello Coello, C.A., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) PPSN XII, Part I. LNCS, vol. 7491, pp. 21–31. Springer, Heidelberg (2012)
11. Nguyen, Q.U., Nguyen, X.H., O'Neill, M.: Semantic Aware Crossover for Genetic Programming: The Case for Real-Valued Function Regression. In: Vanneschi, L., Gustafson, S., Moraglio, A., De Falco, I., Ebner, M. (eds.) EuroGP 2009. LNCS, vol. 5481, pp. 292–302. Springer, Heidelberg (2009)
12. Quang, U.N., Nguyen, X.H., O'Neill, M.: Semantics based mutation in genetic programming: The case for real-valued symbolic regression. In: Matousek, R., Nolle, L. (eds.) 15th Intern. Conf. on Soft Computing, Mendel 2009, pp. 73–91 (2009)
13. Uy, N.Q., Hoai, N.X., O'Neill, M., McKay, B.: The Role of Syntactic and Semantic Locality of Crossover in Genetic Programming. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6239, pp. 533–542. Springer, Heidelberg (2010)
14. Uy, N.Q., Hoai, N.X., O'Neill, M., McKay, R.I., Galvan-Lopez, E.: Semantically-based crossover in genetic programming: application to real-valued symbolic regression. Genetic Programming and Evolvable Machines 12(2), 91–119 (2011)