An empirical study of functional complexity as an indicator of overfitting in Genetic Programming

Leonardo Trujillo [a], Sara Silva [b,c],
Pierrick Legrand [d,e] and Leonardo Vanneschi [f,b]

[a] Instituto Tecnológico de Tijuana, Av. Tecnológico S/N, Tijuana, BC, México
[b] INESC-ID Lisboa, KDBIO group, Lisbon, Portugal
[c] CISUC, ECOS group, University of Coimbra, Portugal
[d] IMB, Institut de Mathématiques de Bordeaux, UMR CNRS 5251, France
[e] ALEA Team at INRIA Bordeaux Sud-Ouest, France
[f] Department of Informatics, Systems and Communication (D.I.S.Co.), University of Milano-Bicocca, Milan, Italy
leonardo.trujillo.ttl@gmail.com,sara@kdbio.inesc-id.pt,
pierrick.legrand@u-bordeaux2.fr,vanneschi@disco.unimib.it

Abstract. Recently, it has been stated that the complexity of a solution is a good indicator of the amount of overfitting it incurs. However, measuring the complexity of a program, in Genetic Programming, is not a trivial task. In this paper, we study the functional complexity and how it relates with overfitting on symbolic regression problems. We consider two measures of complexity, Slope-based Functional Complexity, inspired by the concept of curvature, and Regularity-based Functional Complexity based on the concept of Hölderian regularity. In general, both complexity measures appear to be poor indicators of program overfitting. However, results suggest that Regularity-based Functional Complexity could provide a good indication of overfitting in extreme cases.

1 Introduction

In the field of Genetic Programming (GP), a substantial amount of research focuses on the bloat phenomenon Bloat is an excess of code growth without a corresponding improvement in fitness [10], and it can cause several types of problems during a GP run. For instance, bloat can stagnate a GP search because when program trees become excessively large then fitness evaluation can turn into a computational bottleneck. Moreover, it is also assumed that bloat is related to overfitting, one of the most important problems in machine learning. It is often stated that simpler solutions will be more robust and will generalize better than complex ones, with the latter being more likely to overfit the training data [6, 14]. The GP community has tacitly assumed that *simple* solutions can be equated with small program trees, and that very large programs correspond to *complex* solutions [8]. Therefore, bloat was assumed to be a good indicator of program overfitting. However, recent experimental work suggests that this assumption is not reliable [14]. In particular, [14] showed that a causal

link between bloat and overfitting did not exist on three test cases. From this it follows that bloated programs should not be a priori regarded as complex. This leads us towards two important questions. First, how can program complexity be measured? And second, can program complexity be used as an indicator of program overfitting? Here, we study the measure of *functional complexity* proposed in [14], which we call Slope-based Functional Complexity (SFC). SFC in inspired by the concept of curvature, and represents the first measure of complexity that is explicitly intended to be an indicator of program overfitting. While SFC is based on a reasonable intuition, the methodological approach it requires can become complicated for multidimensional problems. Therefore, we propose a measure of complexity based on the concept of Hölderian regularity, and we call it Regularity-based Functional Complexity (RFC), which captures the underlying justification of SFC without being hampered by some of its practical difficulties. Both measures are experimentally tested on two real world problems and compared based on their correlation with overfitting [14].

2 Overfitting

Based on [6, 14], for a GP search in program space P an individual program $p \in P$ overfits the training data if an alternative program $p' \in P$ exists such that p has a smaller error than p' over the training samples, but p' has a smaller error over the entire distribution of instances. In the above terms, overfitting is practically impossible to measure because it requires an exhaustive search over P and the entire distribution of instances. Nevertheless, [14] argues that a good approximation can be obtained with the relationship between fitness computed on the training set and fitness computed on an independent test set. In our work, we use the measure for overfitting proposed in [14]; which proceeds as follows for a minimization problem. If at generation g test fitness is better than training fitness, then $(overfitting(g) = 0)$; if test fitness is better than the best test fitness obtained thus far then $(overfitting(g) = 0)$; otherwise, overfitting is quantified by the difference of the distance between training and test fitness at generation g and the distance between training and test fitness at the generation when the best test fitness was obtained. In this procedure, test fitness is computed only for the best individual of the population, the individual with the best training fitness. This individual is chosen because we want to measure the overfitting of the best GP solution, since this is the individual that will be returned at the end of the search. If elitism is used then training fitness will be monotonically decreasing, thus $overfitting(g) \geq 0 \; \forall g$.

Therefore, overfitting depends on the performance of a program on the test dataset. In the best case scenario, the test set is a representative sample of the unseen data that a program might encounter during on-line operation. However, it might also be true that an independent test set could be difficult to produce or might be unavailable. Therefore, an indicator of program overfitting would be a useful practical tool. In this sense, [14] suggests that program complexity could indicate if overfitting is present during a GP run.

3 Program complexity

Program complexity could be measured in different ways. For instance, [15] proposed two measures. The first is to equate complexity with tree size, based on the assumption that bloated programs are also complex. The second addresses the complexity of the program output or functional complexity, measured as the degree of the Chebyshev polynomial that approximates the output. In [15] complexity is studied as an objective that should be optimized, not as an indicator of program overfitting, which could be studied in future work.

3.1 Functional complexity

If we are describing the complexity of a program, focusing on the functional output seems to be the best approach for several reasons. First, the size and shape of a program might be misleading if many nodes are introns, in which case they have no bearing on program output. Moreover, even a large program might be simplified and expressed as a compact expression. The size of the program influences the dynamics of the evolutionary process, but tells us little with respect to the output of each program. Secondly, the genotype-phenotype distinction is not clear in GP, where an explicit phenotype is normally not defined [5]. Therefore, it is easier to focus on the functional output of a program in order to describe its behavior. Here, we describe two measures of functional complexity. We begin with the SFC measure proposed in [14], and afterwards present our proposed RFC measure. It is important to state that despite the differences between SFC and RFC, it should become clear that both are concerned with measuring the same underlying characteristic of functional behavior.

Slope-based Functional Complexity The complexity measure proposed in [14] is inspired by the concept of function curvature. Curvature is the amount by which the geometric representation of a function deviates from being flat or straight. In [14], the authors correctly point out that a program tree with a response that tends to be flat, or smooth, should be characterized as having a low functional complexity, and that such a function will tend to have a lower curvature at each sampled point. Conversely, a complex functional output would exhibit a larger amount of curvature at each point. This basic intuition seems reasonable, but computing a measure of curvature is not trivial [7], especially for multidimensional real-world problems. In order to overcome this, [14] proposes a heuristic measure based on the slope of the line segments that join each pair of adjacent points. Consider a bi-dimensional regression problem, for which the graphical representation of the output of a program is a polyline. This polyline is produced by plotting the points that have fitness cases in the abscissa, the corresponding values of the program output as ordinates, and sorting the points based on the fitness cases. Then, these points are joined by straight line segments and the slope of each segment is computed. The SFC is calculated by summing the differences of adjacent line segments. If the slopes are identical,

the value of the measure is zero, thus a low complexity score. Conversely, if the sign of the slope of all adjacent segments changes then complexity is high.

The intuition behind SFC seems to be a reasonable first approximation of a complexity measure that could serve as a predictor of GP overfitting. However, the formal definition of SFC can become complicated for multidimensional problems and require a somewhat heuristic implementation for the following reasons [14]. First, the SFC considers each problem dimensions independently by projecting the output of a function onto each dimension and obtaining a complexity measure for each dimension separately. In [14] the final SFC was the average of all such measures, while in this paper we test the average, median, max and min values. The decision to consider each dimension separately might be problematic since the complexity of a function in a multidimensional space could be different from the complexity of the function as projected onto each individual dimension. Second, by employing such a strategy, the data might become inconsistent. Namely, if we order the data based on a single dimension then the case might arise in which multiple values at the ordinates (program output) correspond with a single value on the abscissa (fitness cases). In such a case a slope measure cannot be computed, and an ad-hoc heuristic is required.

Regularity-based Functional Complexity Following the basic intuition behind the SFC measure, we propose a similar measure of complexity that focuses on the same functional behavior. Another way to interpret the output we expect from an overfitted function, is to use the concept of Hölderian regularity. In the areas of signal processing and image analysis it is quite clear that the most prominent and informative regions of a signal correspond with those portions where signal variation is highest [13]. It is therefore useful to be able to characterize the amount of variation, or regularity, that is present at each point on signal. One way to accomplish this is to use the concept of Hölderian regularity, which characterizes the singular or irregular signal patterns [11]. The regularity of a signal at each point can be quantified using the pointwise Hölder exponent.

Definition 1: Let $f : \mathbb{R} \rightarrow \mathbb{R}$, $s \in \mathbb{R}^{+*} \setminus \mathbb{N}$ and $x_0 \in \mathbb{R}$. $f \in C^s(x_0)$ if and only if $\exists \eta \in \mathbb{R}^{+*}$, and a polynomial P_n of degree $n < s$ and a constant c such that

$$\forall x \in B(x_0, \eta), |f(x) - P_n(x - x_0)| \leq c|x - x_0|^s \ , \tag{1}$$

where $B(x_0, \eta)$ is the local neighborhood around x_0 with a radius η. The pointwise Hölder exponent of f at x_0 is $\alpha_p(x_0) = \sup_s \{f \in C^s(x_0)\}$.

In the above definition, P_n represents the Taylor series approximation of function f. When a singularity (or irregularity) is present at x_0 then f is non-differentiable at that point, and P_n represents the function itself. In this case, Eq. 1 describes a bound on the amount by which a signal varies, or oscillates, around point x_0 within an arbitrary local neighborhood $B(x_0, \eta)$. Hence, when the singularity is large at x_0, with a large variation of the signal, then $\alpha_p \rightarrow 0$ as $x \rightarrow x_0$. Conversely, $\alpha_p \rightarrow 1$ when the variation of the signal $(f(x) - P_n) \rightarrow 0$

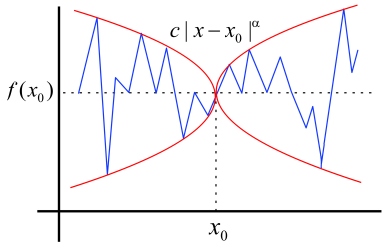


Fig. 1. Hölderian envelope of signal f at point x_0.

as $x \rightarrow x_0$, the point is smoother or more regular. Figure 1 shows the envelope that bounds the oscillations of f expressed by the Hölder exponent α_p at x_0.

The basic intuition of SFC is that a complex functional output should exhibit an irregular behavior (high curvature). Conversely, a simple function should produce a smoother, or more regular, output. Therefore, we propose to use the Hölder exponent as the basis for a complexity measure which we call Regularity-based Functional Complexity, or RFC. The idea is to compute the Hölder exponent at each fitness case and use these measures to characterize the overall regularity of the functional output. The final RFC measure could be a variety of statistics computed from the set of Hölder exponents obtained from each fitness case; here, as for SFC, we test the average, median, max and min values. There are several estimators for the Hölder exponent [1, 12]; however, here we use the oscillations method, which can be derived directly from Definition 1 [11].

Estimation of the pointwise Hölder exponent through oscillations The Hölder exponent of function $f(x)$ at point x_0 is the $sup(\alpha_p) \in [0, 1]$, for which a constant c exists such that $\forall x'$ in a neighborhood of x_0,

$$|f(x_0) - f(x')| \leq c|x_0 - x'|^{\alpha_p} \; . \tag{2}$$

In terms of signal oscillations, a function $f(x)$ is Hölderian with exponent $\alpha_p \in [0, 1]$ at x_0 if $\exists c \; \forall \tau$ such that $osc_\tau(t) \leq c\tau^{\alpha_p}$, with

$$osc_\tau(x_0) = \sup_{x',x'' \in [x_0-\tau, x_0+\tau]} |f(x') - f(x'')| \; . \tag{3}$$

Now, since $x' = x_0 + h$ in Eq. 2, we can also write that

$$\alpha_p(x_0) = \liminf_{h \rightarrow 0} \frac{\log |f(x_0 + h) - f(x_0)|}{\log |h|} \; . \tag{4}$$

Therefore, the problem is that of finding an α_p that satisfies 2 and 3, and in order to simplify this process we can set $\tau = \beta^r$. Then, we can write $osc_\tau \approx c\tau_p^\alpha = \beta^{(\alpha_p r + b)}$, which is equivalent to $log_\beta(osc_\tau) \approx \alpha_p r + b$.

Therefore, an estimation of the regularity can be built at each point by computing the slope of the regression between the logarithm of the oscillations osc_τ

Table 1. GP parameters used for real world experiments.

Parameter	Description
Population size	500 individuals
Iterations	100 generations
Initialization	Ramped Half-and-Half
Crossover probability	$p_c = 0.5$
Mutation probability	$p_\mu = 0.5$
Initial max. depth	Six levels
Selection	Lexicographic Parsimony Tournament
Survival	Elitism
Runs	30

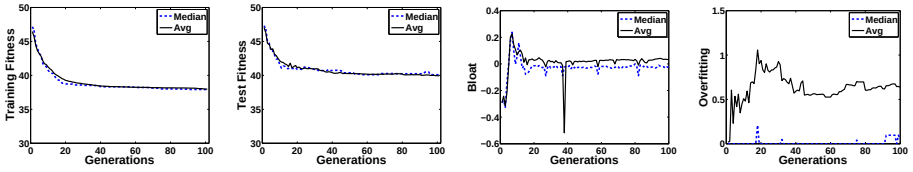
and the logarithm of the dimension of the neighborhood at which the oscillations τ are computed. Here, we use least squares regression to compute the slope, with $\beta = 2.1$ and $r = 0.1, 0.2, \ldots, 9$. Also, it is preferable not to use all sizes of neighborhoods between two values τ_{min} and τ_{max}. Hence, we calculate the oscillation at point x_0 only on intervals of the form $[x_0 - \tau_r : x_0 + \tau_r]$. For a function in $\mathbb{R}^D$, τ_r is defines a hyper-volume around x_0, such that $d(x', x_0) \leq \tau_r$ and $d(x'', x_0) \leq \tau_r$, where $d(a, b)$ is the Euclidean distance between a and b.

The implicit difficulties of applying SFC are not an issue for RFC, since it is not necessary to consider each problem dimension separately. Moreover, unlike the concept of curvature, estimating signal regularity does not require derivatives in a multidimensional space. Therefore, the RFC provides a straightforward measure of the intuitive functional behavior that SFC attempts to capture.

4 Experiments

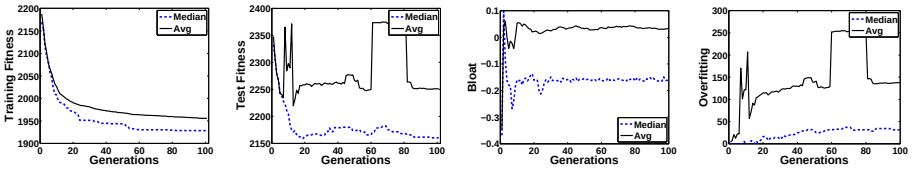
Test problems and experimental setup In order to test both SFC and RFC we use the same two real world problems as [14]. They consist on predicting the value of two pharmacokinetic parameters for a set of candidate drug compounds based on their molecular structure. The first data set relates to the parameter of human oral bioavailability, and contains 260 samples in a 241 dimensional space. The second data set relates to the median lethal those of a drug, or toxicity, and contains 234 samples from a 626 dimensional space. The experimental setup for our GP is summarized in Table 1. Most parameters are the same as those in [14], except for the probabilities of each genetic operator. The reason for this is that we want a GP system that does not bloat, however the Operator Equalisation technique used in [14] was considered to be too computationally costly. Therefore, we combine simple one-point crossover with standard subtree mutation in order to avoid code bloat and still achieve good fitness scores. The choice of a GP without bloat instead of a typical GP was made because we want to confirm that a causal relationship between bloat and overfitting is not a general one.

In Section 3 we presented SFC and RFC, however there is one more important consideration that must be accounted for before computing these scores. Since we are to compute a complexity score for any program tree, it will sometimes be the case that a particular program will use only a fraction of the total



(a) Training (b) Testing (c) Bloat (d) Overfitting

Fig. 2. Evolution of the bioavailability problem.



(a) Training (b) Testing (c) Bloat (d) Overfitting

Fig. 3. Evolution of the toxicity problem.

dimensions (expressed as terminals) of the problem. Therefore, a decision must be made regarding which dimensions to include when either SFC or RFC are computed. In this case we only use those dimensions that are part of a programs computations. Finally, we point out that all of our code was developed using Matlab 2009a, using the GPLAB toolbox for our GP system [9], and the FracLab toolbox to compute the Hölder exponents of the RFC measure [4].

Results We begin by presenting the results obtained for training fitness, test fitness, bloat and overfitting for the bioavailability and toxicity problems, shown in figures 2 and 3 respectively. In each case we present the median and average performance over all runs. There are several noticeable patterns in these plots. First, in both cases we can see that our GP systems converges to good fitness scores [14]. Second, the GP system does not incur bloat, in fact the median values in both problems fall below zero, which means that the average population size actually decreased with respect to the initial population. This is an interesting result, where bloat is eliminated from our runs using a simple strategy. Third, we see that in both problems there is only a small amount of overfitting based on the median scores, with values much lower than those published in [14]. However, the average overfitting is substantially larger than the median value, this indicates that our runs produced outliers.

The evolution of the complexity measures are presented in figures 4 and 5 for the bioavailability and toxicity problems respectively. Each plot considers a different representative statistics for each measure, as stated in Section 3, and shows the median value of this statistics over all the runs. However, it is not possible to draw a clear conclusion regarding the relationship, or correlation, between complexity and the overfitting curves shown in figures 2(d) and 3(d).

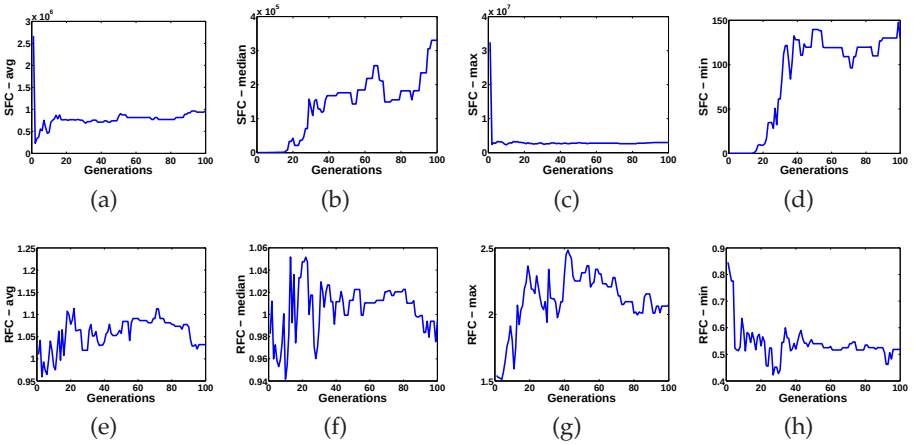


Fig. 4. Evolution of both complexity measures for the bioavailabitly problem. (a)-(d) correspond to the SFC measure and show the average, median, max and min of the SFC values of all problem dimensions. (a)-(d) correspond to the RFC measure and show the average, median, max and min of all the Hölder exponents computed for each fitness case. All plots show the median value obtained over all 30 runs.

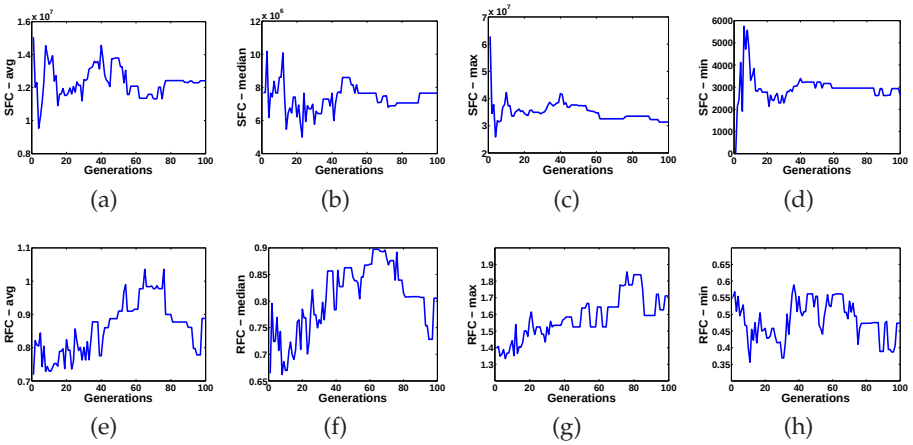
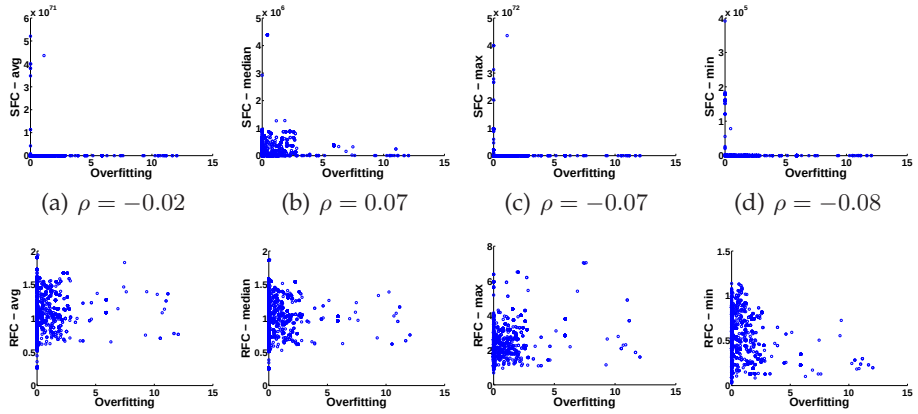


Fig. 5. Evolution of both complexity measures for the toxicity problem. (a)-(d) correspond to the SFC measure and show the average, median, max and min of the SFC values of all problem dimensions. (a)-(d) correspond to the RFC measure and show the average, median, max and min of all the Hölder exponents computed for each fitness case. All plots show the median value obtained over all 30 runs.

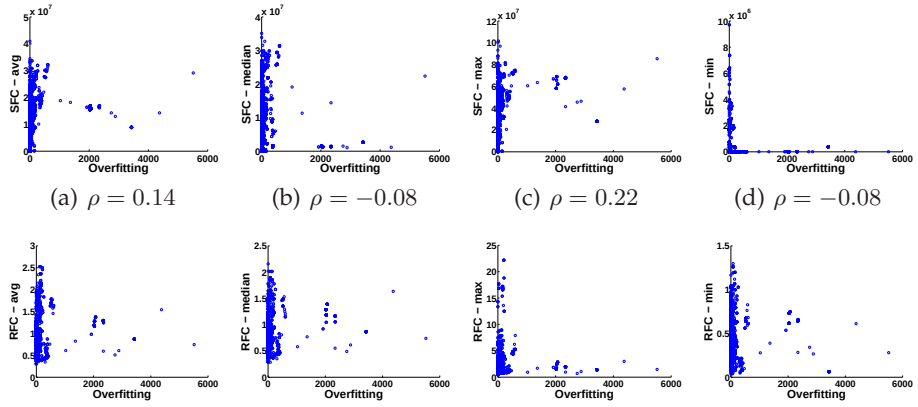
Therefore, scatter plots of complexity and overfitting are shown in figures 6 and 7 for the bioavailability and toxicity problems respectively. These plots take the best individual from each generation and each run, and their overfit-



(a) $\rho = -0.02$ (b) $\rho = 0.07$ (c) $\rho = -0.07$ (d) $\rho = -0.08$

(e) $\rho = 0.15$ (f) $\rho = 0.10$ (g) $\rho = 0.35$ (h) $\rho = -0.22$

Fig. 6. Scatter plots of both complexity measures for the bioavailabitly problem. (a)-(d) correspond to the SFC measure and show the average, median, max and min. (a)-(d) correspond to the RFC measure and show the average, median, max and min. All plots also show Pearson's correlation coefficient ρ.



(a) $\rho = 0.14$ (b) $\rho = -0.08$ (c) $\rho = 0.22$ (d) $\rho = -0.08$

(e) $\rho = 0.12$ (f) $\rho = 0.15$ (g) $\rho = 0.00$ (h) $\rho = -0.04$

Fig. 7. Scatter plots of both complexity measures for the toxicity problem. (a)-(d) correspond to the SFC measure and show the average, median, max and min. (a)-(d) correspond to the RFC measure and show the average, median, max and min. All plots also show Pearson's correlation coefficient ρ.

ting and complexity scores as ordered pairs. Additionally, each plot shows the Pearson's correlation coefficient ρ between each complexity score and overfitting. For analysis, we follow the empirical estimates of [2] and consider values of $\rho > 0.15$ as an indication of positive correlation, $\rho < -0.15$ as negative correlation, and $-0.15 \leq \rho \leq 0.15$ as an indication of no correlation. It is important

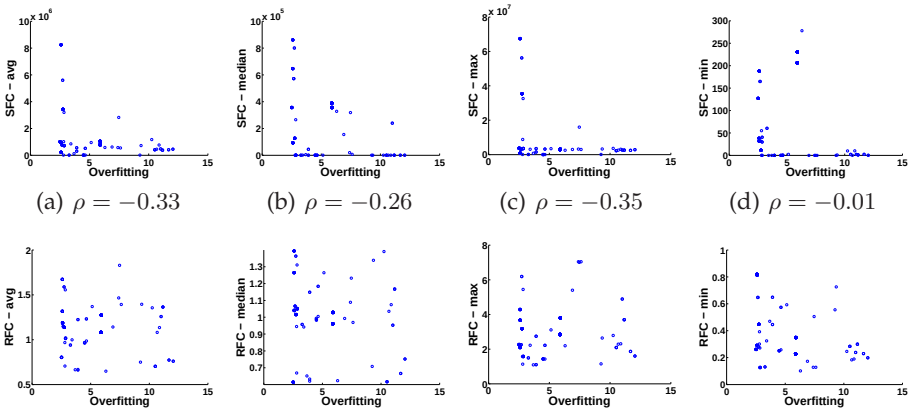
to point out that SFC is expected to have a positive correlation with overfitting while RFC should have a negative one given the definition of each. Most tests show and absence of correlation, with only SFC-max achieving the expected result on toxicity, and RFC-min on bioavailability.

Since the best case scenario would be to have a complexity measure that predicts when a program is overfitted, then it is more informative to discard those individuals with low overfitting, and focus on individuals with a large amount of overfitting. In other words, if a complexity measure cannot be used as an indicator of extreme cases of overfitting, then its usefulness would be quite low. Therefore, in figures 8 and 9 we show the same scatter plots as before, however in this case we only present the 5% of individuals with the highest amount of overfitting. First, consider the results for the SFC measure. In most cases, SFC shows a negative correlation with overfitting on both problems, the exact opposite of what we expected. It is only SFC-min that shows the expected correlation on the toxicity problem. Second, for RFC on the bioavailability problem, only RFC-median and RFC-min show the expected negative correlation. However, given the low overfitting on this problem, we should only expect to detect a small amount of correlation in the best case scenario. On the other hand, for the toxicity problem all RFC variants show the expected correlation. Hence, in this case for the higher overfitting values (the top 5%), the RFC measure could provide an indication of program overfitting.

It appears that the above results are not very strong, given low correlation values achieved even in the best cases. Nonetheless, we would like to stress the difficulty of the problem at hand. In both test cases we have a small sample of data that lies within a highly multidimensional space. Therefore, extracting a proper description of program behavior, and thus functional complexity, is not trivial. For instance, the RFC measures relies on the pointwise Hölder exponent, and in order to estimate it at any point it is necessary to consider how the function oscillates within a series of progressively larger and concentric local neighborhoods. However, in both of our test cases it becomes extremely difficult to correctly estimate the oscillations of the function, and sometimes it cannot be done, because of the sparseness of the data. One possible way to solve this problem is to generate or interpolate the missing data in a way that preserves the regularity of the function [3], however this is left as future work. In summary, if any measure can produce, even a slight indication of overfitting, this should suggest that we are proceeding in a promising direction.

5 Summary and concluding remarks

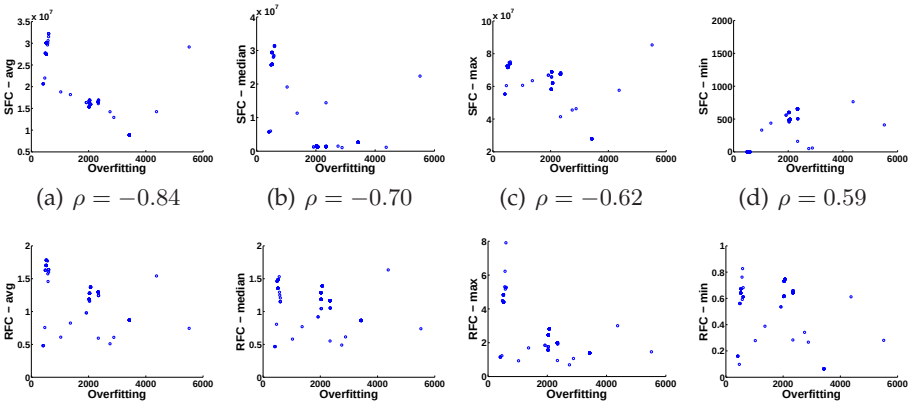
This work presents an empirical study of functional complexity in GP. It is commonly stated that simpler solutions to a problem should be preferred because they are less likely to be overfitted to the training examples. In this work, we study two complexity measures, Slope-based Functional Complexity [14], and Regularity-based Functional Complexity derived from the concept of Hölderian regularity. We measure the evolution of SFC and RFC on a bloat-free GP system using two real-world problems, in order to evaluate if they can serve as



(a) $\rho = -0.33$ (b) $\rho = -0.26$ (c) $\rho = -0.35$ (d) $\rho = -0.01$

(e) $\rho = -0.12$ (f) $\rho = -0.20$ (g) $\rho = -0.05$ (h) $\rho = -0.41$

Fig. 8. Scatter plots of both complexity measures for the bioavailabitly problem with the 5% of individuals that have the highest overfitting scores. (a)-(d) SFC measure, showing the average, median, max and min. (a)-(d) RFC measure, showing the average, median, max and min. All plots also show Pearson's correlation coefficient ρ.



(a) $\rho = -0.84$ (b) $\rho = -0.70$ (c) $\rho = -0.62$ (d) $\rho = 0.59$

(e) $\rho = -0.58$ (f) $\rho = -0.40$ (g) $\rho = -0.71$ (h) $\rho = -0.44$

Fig. 9. Scatter plots of both complexity measures for the toxicity problem with the 5% of individuals that have the highest overfitting scores. (a)-(d) SFC measure, showing the average, median, max and min. (a)-(d) RFC measure, showing the average, median, max and min. All plots also show Pearson's correlation coefficient ρ.

indicators of overfitting during a GP run. Overall, the results show almost no correlation between both complexity measures and program overfitting. However, when we consider only highly overfitted solutions, then the RFC measures does appear to provide a somewhat useful indicator of overfitting. On the other hand, the SFC measure fails to achieve any useful correlation with

program overfitting, and in some cases it produces the opposite of what is expected. Nonetheless, it is important to stress that this should be taken as an initial, and partial, empirical study of functional complexity in GP. Therefore, further research should focus on a more comprehensive evaluation of these, and possibly other, measures of complexity as indicators of GP overfitting.

Acknowledgements This work was partially supported by FCT (INESC-ID multiannual funding) through the PIDDAC Program funds. Sara Silva and Leonardo Vanneschi acknowledge project PTDC/EIA-CCO/103363/2008 from FCT, Portugal.

References

1. S. Jaffard. Wavelet techniques in multifractal analysis. In *Fractal Geometry and Applications: A Jubilee of Benoit Mandelbrot, Proceedings of Symposium in Pure Mathematics*, volume 72, pages 91–151, 2004.
2. T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 184–192. Morgan Kaufmann, 1995.
3. P. Legrand and J. Lévy-Véhel. Local regularity-based interpolation. In *WAVELET X, Part of SPIE's Symposium on Optical Science and Technology*, volume 5207, 2003.
4. J. Lévy-Véhel and P. Legrand. *Thinking in Patterns*, chapter Signal and Image Processing with FRACLAB, pages 321–322. 2004. http://fraclab.saclay.inria.fr/homepage.html.
5. J. McDermott, E. Galvan-Lopez, and M. O'Neill. A fine-grained view of GP locality with binary decision diagrams as ant phenotypes. In R. Schaefer et al., editors, *Proceedings of PPSN 2010*, volume 6238 of *LNCS*, pages 164–173, Krakow, Poland, 2010. Springer.
6. T. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
7. J.-M. Morvan. *Generalized Curvatures*. Springer, 1st edition, 2008.
8. J. Rosca. Generality versus size in genetic programming. In *Proceedings of Genetic Programming 1996*, pages 381–387. MIT Press, 1996.
9. S. Silva and J. Almeida. Gplab: A genetic programming toolbox for matlab. In *Proceedings of the Nordic MATLAB conference*, pages 273–278, 2003.
10. S. Silva and E. Costa. Dynamic limits for bloat control in genetic programming and a review of past and current bloat theories. *Genetic Programming and Evolvable Machines*, 10(2):141–179, 2009.
11. C. Tricot. *Curves and Fractal Dimension*. Springer-Verlag, 1995.
12. L. Trujillo, P. Legrand, and J. Lévy-Véhel. The estimation of hölderian regularity using genetic programming. In *Proceedings of GECCO 2010*, pages 861–868, New York, NY, USA, 2010. ACM.
13. L. Trujillo, P. Legrand, G. Olague, and C. Pérez. Optimization of the hölder image descriptor using a genetic algorithm. In *Proceedings of GECCO 2010*, pages 1147–1154, New York, NY, USA, 2010. ACM.
14. L. Vanneschi, M. Castelli, and S. Silva. Measuring bloat, overfitting and functional complexity in genetic programming. In *Proceedings of GECCO 2010*, pages 877–884, New York, NY, USA, 2010. ACM.
15. E. J. Vladislavleva, G. F. Smits, and D. Den Hertog. Order of nonlinearity as a complexity measure for models generated by symbolic regression via pareto genetic programming. *IEEE Trans. Evol. Comp*, 13:333–349, April 2009.