A comparison of predictive measures of problem difficulty for classification with Genetic Programming

Leonardo Trujillo[1], Yuliana Martínez[1], Edgar Galván-López[2], and Pierrick Legrand[3]

[1] Doctorado en Ciencias de la Ingeniería, Departamento de Ingeniería Eléctrica y Electrónica, Instituto Tecnológico de Tijuana, Tijuana BC, México
[2] Distributed Systems Group, School of Computer Science and Statistics, Trinity College Dublin, Ireland
[3] Université Victor Segalen, Bordeaux, France
IMB, Institut de Mathématiques de Bordeaux, UMR CNRS 5251, France
ALEA Team, INRIA Bordeaux Sud-Ouest, France
{leonardo.trujillo.ttl,ysaraimr}@gmail.com,
edgar.galvan@scss.tcd.ie,
pierrick.legrand@u-bordeaux2.fr

Abstract. In the field of Genetic Programming (GP) a question exists that is difficult to solve; how can problem difficulty be determined? In this paper the overall goal is to develop predictive tools that estimate how difficult a problem is for GP to solve. Here we analyse two groups of methods. We call the first group Evolvability Indicators (EI), measures that capture how amendable the fitness landscape is to a GP search. The second are Predictors of Expected Performance (PEP), models that take as input a set of descriptive attributes of a problem and predict the expected performance of a GP system. These predictive variables are domain specific thus problems are described in the context of the problem domain. This paper compares an EI, the Negative Slope Coefficient, and a PEP model for a GP classifier. Results suggest that the EI does not correlate with the performance of GP classifiers. Conversely, the PEP models show a high correlation with GP performance. It appears that while an EI estimates the difficulty of a search, it does not necessarily capture the difficulty of the underlying problem. However, while PEP models treat GP as a computational black-box, they can produce accurate performance predictions.

Keywords: Genetic Programming, Performance prediction, Classification

1 Introduction

Genetic Programming (GP) deals with the development of evolutionary algorithms (EA's) for automatic program induction [10], However, as for every EA,

GP systems are stochastic search process, with many degrees of freedom and heuristic components. Therefore, as of yet, it is not possible to derive, from first principles, weather GP can solve a particular problem or task. A current goal within the GP community is to estimate how *hard* a problem instance might be for a specific GP. Such a measure could allow researchers to correctly choose and tune a GP search without the need of actually executing the code [24], which usually is computationally expensive.

If we want to measure the difficulty of a problem in GP, we can consider at least two different frames of reference. The first is the problem domain, which is independent of the method used to solve the problem [7]. The second frame of reference is to consider a perspective directly related to the process used to find a solution; in the case of GP this frame of reference corresponds with the search space and fitness landscape [9]. Let us first describe the latter.

The concept of a fitness landscape has dominated the way geneticists think about biological evolution and has been adopted by the EA community as a way to visualize evolution dynamics. Formally, a fitness landscape, as specified in [19], can be defined as a triplet (x, χ, f): (a) a set x of configurations, (b) a notion χ of neighbourhood, distance or accessibility on x, and finally, (c) a fitness function f. The local and global structure of the fitness landscape describes the underlying difficulty of a search. In general, most meta-heuristics work under the assumption that the fitness of a candidate solution, a point on the fitness landscape, is positively correlated with the fitness of (some) of its neighbours. Such a property can be defined as the *evolvability* of a landscape [1, 16]. Hence, some researchers have proposed measures that characterize the evolvability of a fitness landscape, what are here referred to as evolvability indicators (EI).

Another approach is to use the problem domain as the frame of reference, and characterize the difficulty of a problem based on the expected performance the GP search, a quantity that is derived from domain specific features of each problem instance [6, 21, 22]. This is a more pragmatic approach, the evolutionary search is taken as a black-box process and the performance of GP on a set of training problems is used to build predictors of the expected performance on unseen problems, following a machine learning methodology. In what follows, we refer to such measures of problem difficulty as Predictors of Expected Performance (PEPs).

The remainder of this paper proceeds as follows. Section 2 gives an overview of related work. Then, Section 3 describes how GP can be used for classification and presents the GP classifier used to perform the comparative analysis. The experimental results and analysis are given in Section 4. Finally, Section 5 contains a summary and conclusions.

2 Background

Landscapes and problem difficulty have been the subject of a good deal of research regarding EA's. For instance, researchers have developed work on landscape correlation [25], autocorrelation [13], epistasis [4] and monotonicity [15].

For GP, locality has been used to measure problem difficulty based on genotype to phenotype mappings [3, 12, 11]. However, this paper focuses on one of the most successful EI in GP literature, the Negative Slope Coefficient (NSC). The following discussion presents the NSC and reviews its predecessor, Fitness Distance Correlation (FDC).

2.1 Evolvability Indicators

FDC is a measure for problem difficulty originally proposed for genetic algorithms (GA's) [8] and later extended to GP [20]. The logic behind FDC proceeds as follows. Assume that we can compute the genotypic distance between each valid individual and the (global) optimum to a problem. If this distance is negatively correlated with the fitness of each individual then the search problem should be characterized as *easy*, and it should be characterized as difficult if no correlation is detected. Moreover, a problem should be considered to be deceptive if the correlation is positive. While FDC has shown to be reliable in many test cases, its more glaring weakness is that the optimal solution must be known *a priori*, somewhat not realistic for real-world problems.

Following the same general assumptions of FDC, Vanneschi et al [23] propose the NSC. In the case of NSC, knowledge about the global optimum is not required. Instead, NSC relies on the concept of *fitness clouds*, a scatter plot where for each genotype x a point is plotted on a 2-D plane, where the horizontal axis corresponds with the fitness of x given by $f(x)$, and the vertical axis represents the fitness $f(y)$ of a neighbouring genotype y. The hypothesis behind NSC is that the *fitness cloud shape* provides a meaningful description of the evolvability of a problem for GP-based search. The NSC is computed by assuming a piecewise linear relationship between $f(x)$ and $f(y)$ for a sample of M individual genotypes and computing the slope of the scatter points within a set of equally spaced segments of the $f(x)$ axis. In the original implementation, individuals are sampled using the Metropolis-Hastings algorithm, neighbours are generated using standard sub-tree mutation, and the representative neighbour y for each genotype x is chosen using tournament selection. The NSC is given in the range of $(-\infty, 0]$, where a value of 0 represents a highly evolvable (assumed to be easy) problem, and a negative NSC indicates a less evolvable (more difficult) problem.

2.2 Predictors of Expected Performance

Another way to characterize problem difficulty is to attempt to predict the expected performance that a GP search will achieve on a given problem instance, a more direct approach. Following this line of thought, two approaches have been proposed in GP literature. First, consider the work of [5], where the frame of reference of the problem and of the search method are combined to derive Predictors of Expected Performance (PEP's) for GP. In [5], the authors propose linear predictive models based on a sampling of the fitness landscape, given by

$$P(\mathbf{t}) \approx a_0 + \sum_{\mathbf{p} \in \mathcal{S}} a_{\mathbf{p}} \cdot d(\mathbf{p}, \mathbf{t}) \ , \tag{1}$$

where $P(\mathbf{t})$ is the predicted performance, $\mathbf{t}$ is the target functionality, $d(\mathbf{p}, \mathbf{t})$ is a distance measure (such a distance measure is a common fitness function for many application domains of GP), $\mathcal{S}$ is the set of all possible program behaviours, and where each behaviour $\mathbf{p}$ represents the set of program outputs obtained from the set of fitness cases for a particular problem. Hence, PEP's are derived by sampling the space $\mathcal{S}$ of possible program behaviours, which can also be seen as the space of possible programs. These models were tested on symbolic regression problems and 4-input Boolean problems for GP, achieving good results.

The second, more recent work, is more tightly centered within the problem frame of reference [6, 21, 22], and proceeds as follows. Given a problem p, for which we want to compute a performance prediction, extract a feature vector $\boldsymbol{\beta} = (\beta_1, \beta_2, ...\beta_N)$ of N distinct features that *describe the properties of* p. Then, a PEP P is given by a kernel function K,

$$P(\boldsymbol{\beta}) \approx K(\boldsymbol{\beta}) \ . \tag{2}$$

Notice that the form of K is not restricted in any way. For instance, [6] uses a linear function similar to the one proposed in [5]. However, [21] tests more complex linear models and also non-linear models. Moreover, using this approach the feature vector $\boldsymbol{\beta}$ should be designed specifically for the domain of p. For example, [6] propose problem features for symbolic regression and Boolean problems, and their results show that the predictive accuracy surpasses that of the fitness-based models of [5]. In the case of [21], the authors predict the performance of a GP-classifier and use descriptive features that describe the geometry of the class samples. In that work, a quadratic linear model and symbolic regression models achieve the best performance.

In both approaches described above, the task of deriving the predictive model K is solved using a machine learning strategy that proceeds as follows, First, generate a set $\mathcal{Q}$ of problems (for instance, symbolic regression or classification problems), such that each sample $p \in \mathcal{Q}$ represents a distinct problem instance. The problems in $\mathcal{Q}$ can be real world problems, however its more practical to generate synthetic problems using a predefined probabilistic model. Second, from each $p \in \mathcal{Q}$ extract a vector of descriptive features $\boldsymbol{\beta}$. Third, solve each problem in $\mathcal{Q}$ using a specific GP system, this generates a performance estimate ϵ for each problem. In this formulation, ϵ could be the performance of a single GP execution or a statistic computed from m independent runs. Then, the problem is to find an optimal predictive model K^o of GP performance, such that

$$K^o = \underset{K}{argmin} \{Err[K(\beta_i), \epsilon_i]\} \, \forall p \in Q, \tag{3}$$

where $Err[,]$ represents an error measure, such as the root-mean-square error (RMSE). In practice, in order to derive K, the set $\mathcal{Q}$ is divided into a training set $\mathcal{T}$, a validation set $\mathcal{V}$ and a testing set $\mathcal{U}$. The learning problem can then be solved with standard regression techniques [6, 21, 22] or with non-linear symbolic regression [21, 22]. An important final observation regarding PEP's is that they can generalize quite easily, and can used to predict the performance of other stochastic or black-box algorithms such as neural networks [5, 22].

2.3 Discussion and Limitations

After reviewing the basic methodology of EI's and PEP's, one practical and computational difference stands out. On the one hand, EI's use a very large sampling of the search space to derive an accurate measure. In effect, this means that for every new problem instance it is necessary to perform this costly computational step. However, executing the actual GP search might in fact be faster. Therefore, the best use of EI's would be to characterize a whole class of problems, where the estimate of search difficulty provided by the EI could generalize to the entire class of problems.

On the other hand, a PEP model is used quite easily and directly for each new problem instance. Practically, the only possible bottleneck would be the computational cost of calculating the set of descriptive features for each problem. However, in order to learn a new PEP a very large number of experimental runs must be carried to derive the training data. Moreover, each PEP is strongly linked to a specific GP system and implementation, and even small deviations from the configuration of the GP system might cause the predictive model to break-down.

3 Classification with GP

In supervised classification a pattern $\mathbf{x} \in \mathbb{R}^P$ has to be classified as belonging to one of M distinct classes $\omega_1, ..., \omega_M$ using a training set $\mathcal{T}$ of P-dimensional patterns with a known classification. The idea is to build a mapping $g(\mathbf{t}) : \mathbb{R}^P \rightarrow M$, that assigns each pattern $\mathbf{t}$ to a corresponding class ω_i, where g is derived based on evidence provided by $\mathcal{T}$. GP can be used in different ways to solve such classification tasks [10, 2]. However, this work uses the approach proposed in [26], denoted as the Probabilistic GP Classifier (PGPC).

3.1 PGPC Classifier

In PGPC, GP is used to evolve a mapping $h(\mathbf{x}) : \mathbb{R}^P \rightarrow \mathbb{R}$ that transforms each input pattern $\mathbf{x}$ into a point on the real line. Moreover, it is assumed that the behaviour of h can be modeled using multiple Gaussian distributions, each corresponding to a single class [26]. The distribution of each class $\mathcal{N}(\mu, \sigma)$ is derived from the examples provided for it in set $\mathcal{T}$, by computing the mean μ and standard deviation σ of the outputs obtained from h on these patterns. Then, from the distribution $\mathcal{N}$ of each class a fitness measure can be derived using Fisher's linear discriminant; for a two class problem it proceeds as follows. After the Gaussian distribution $\mathcal{N}$ for each class are derived, a distance is required. In [26], Zhang and Smart propose a distance measure between both classes as

$$d = \frac{|\mu_1 - \mu_2|}{\sigma_1 + \sigma_2} \ , \tag{4}$$

where μ_1 and μ_2 are the means of the Gaussian distribution of each class, and σ_1 and σ_2 their standard deviations. When this measure tends to 0, it is the

Table 1. Parameters for the PGPC system used in the experimental tests.

Parameter	**Description**
Population size	200 individuals.
Generations	200 generations.
Initialization	*Ramped Half-and-Half*, with 6 levels of maximum depth.
Operator probabilities	Crossover $p_c = 0.8$; Mutation $p_\mu = 0.2$.
Function set	$\left\{+, -, *, /, \sqrt{}, sin, cos, log, x^y, \lvert\cdot\rvert, if\right\}$
Terminal set	$\{x_1, ..., x_i, ..., x_P\}$ Where each x_i is a dimension of the data patterns $\mathbf{x} \in \mathbb{R}^P$
Bloat control	Dynamic depth control.
Initial dynamic depth	6 levels.
Hard maximum depth	20 levels.
Selection	Lexicographic parsimony tournament
Survival	Keep best elitism

worst case scenario because the mapping of both classes overlap completely, and when it tends to ∞ it represents the optimal separation. To normalize the above measure, the fitness for an individual mapping h is given by

$$f_d = \frac{1}{1+d} \ . \tag{5}$$

After executing the GP, the best individual found determines the parameters for the Gaussian distribution $\mathcal{N}_i$ associated to each class. Then, a new test pattern $\mathbf{x}$ is assigned to class i when $\mathcal{N}_i$ gives the maximum probability.

4 Comparative Analysis

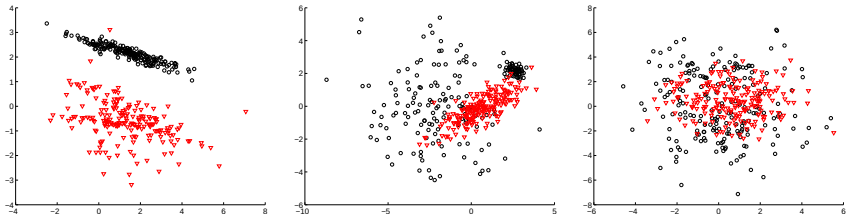
The main goal of the experimental work is to evaluate and compare the predictive accuracy of a state-of-the-art EI (NSC) and a PEP model for a GP-based classifier PGPC. To this end, a large set of synthetic classification problems are generated and solved with PGPC, executing 30 independent runs on each problem and computing the average classification error as the estimate of the expected performance of PGPC.

4.1 Classification Performance

Table 1 presents the setup for the PGPC system. A GP with Koza style crossover and mutation and dynamic depth control to minimize bloat [18] was used. The PGPC classifier is implemented using Matlab 2011a and the GPLAB toolbox [17].

To evaluate the performance of PGPC, 300 two-class classification problems are randomly generated using Gaussian mixture models (GMM's), these conform

set $\mathcal{Q}$. Examples of the classification problems generated are shown in Figure 1, depicting sample points of two different classes(circles and triangles), scattered over the $\mathbb{R}^2$ plane. The use of randomly generated GMM's allows us to generate either unimodal or multimodal classes, with different amounts of class overlap. All class samples lie within the closed 2-D interval $x, y \in [-10, 10]$, and 200 sample points were randomly generated for each class.



(a) $\epsilon = 0.01$, $\sigma = 0.01$ (b) $\epsilon = 0.15$, $\sigma = 0.03$ (c) $\epsilon = 0.33$, $\sigma = 0.03$

Fig. 1. Three classification problems and the average classification error ϵ achieved by PGPC and standard deviation σ of 30 independent runs.

Then, for every problem $p \in \mathcal{Q}$ the average test error of PGPC is computed from 30 independent runs, where the training (70% of the 200 samples) and testing (30%) sets where randomly determined at the start of each run. Figure 1 also specifies the average classification error ϵ achieved by PGPC on each problem and the corresponding standard deviation σ. In all three cases, and for all problems, the average provides a useful performance estimate given the small standard deviation.

4.2 Evolvability for Classification Problems

This section presents the results of computing the NSC on each of the classification problems in set $\mathcal{Q}$. The approach is quite straightforward, since it is possible to directly apply the NSC algorithm to every problem. The algorithm described in [23] is used her with the same parameters except for the total amount of sampled individuals M. Whereas in [23] $M = 40,000$, here $M = 10,000$, a practical choice to reduce computation time; however, some informal tests showed that the results are consistently similar for both values in the group of experiments reported here. Figure 2 presents a scatter plot where the horizontal axis is the average classification error and the vertical axis is the NSC.

The results clearly suggest that the NSC does not correlate with PGPC performance, in particular we can see how many problems are characterized as easy (with NSC equal or close to zero) even when the performance achieved by PGPC is quite poor. This suggests that an EI such as the NSC is limited as a predictor of GP performance since it only considers the frame of reference of the search process; i.e., it can only provide an approximate measure of the difficulty

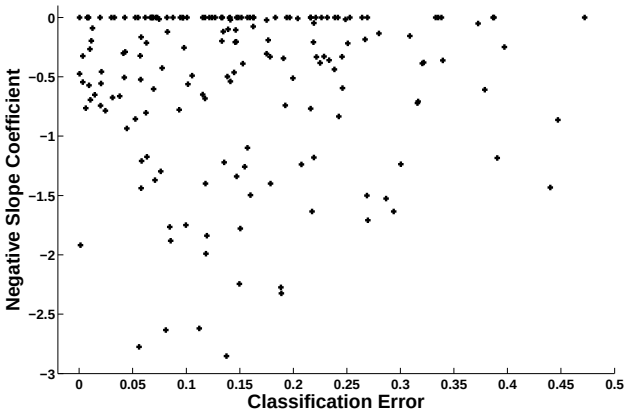


Fig. 2. Scatter plot of the average classification error achieved by PGPC on each problem and the corresponding NSC value. Pearson's correlation coefficient $\rho = 0.02$.

of the search but tells us very little regarding the difficulty of the underlying problem that the GP is intended to solve.

4.3 Prediction of Classification Performance

This section, we show how a PEP estimates the performance of PGPC on the set of classification problems presented above. The PEP is derived following the approach described in [21]. Therefore, the feature vector for each problem $\boldsymbol{\beta}$ is composed of the following problem descriptors.

Problem Descriptors

- The geometric mean ratio of the pooled standard deviations to standard deviations of the individual populations (SD), often used as part of a homogeneity test [14].
- Volume of Overlap Region (VOR) provides an estimate of the amount of overlap between both classes in feature space [7]. This measure is computed by finding, for each feature, the maximum and minimum value of each class and then calculating the length of the overlap region. The length obtained from each feature can then be multiplied in order to obtain a measure of volume overlap. VOR is zero when there is at least one dimension in which the two classes do not overlap.
- Feature efficiency (FE), measures the amount by which each feature dimension contributes to the separation of both classes. When there is a region of overlap between two classes on a feature dimension, then data is considered ambiguous over that region along that dimension. However, it is possible to progressively remove the ambiguity between both classes by separating those

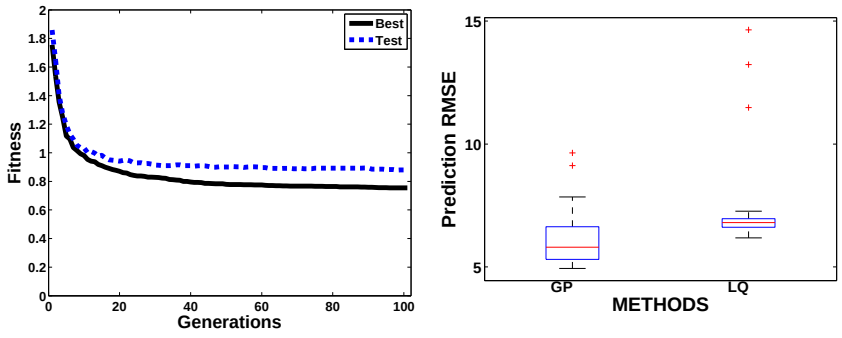
points that lie inside the overlapping region. The efficiency of each feature is defined as the fraction of all remaining points separable by that feature, and the maximum feature efficiency (FE) is taken as the representative value for a two-class problem.

– The Class Distance Ratio (CDR) compares the dispersion within the classes to the gap between the classes [7]. It is computed as follows: for each data sample the Euclidean distance to its nearest-neighbour is computed, within and outside its own class. Then, the CDR is the ratio of the averages of all intraclass and interclass nearest-neighbour distances.

Predictors of Expected Performance: Two different PEP models are tested, a linear model with quadratic terms (LQ-PEP) and a symbolic regression models derived with GP (GP-PEP), reported to achieve the best results in [21]. In the case of the GP-PEP models, the problem descriptors are used as terminal elements $T = \{SD, VOR, FE, CDR\}$, while the function set is defined as $F = \left\{+, -, *, /, \sqrt{}, sin, cos, log, x^y, |\cdot|, if\right\}$. Moreover, fitness is computed by the RMSE calculated on a set of n training problems, given by

$$f(K) = \sqrt{\frac{\sum_{i=1}^{n}(K(\beta_i) - \epsilon_i)^2}{n}} \; . \tag{6}$$

where β_i is the vector of descriptive features and ϵ_i is the performance estimate on a training problem i. Finally, the GP is executed with the following parameters: 200 individuals, 100 generations, ramped half-and-half initialization, 0.8 crossover probability, 0.2 mutation probability, dynamic depth bloat control with a maximum depth of 12 levels, and lexicographic tournament selection.



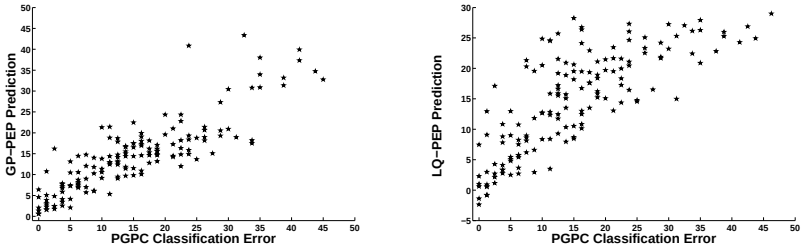
(a) (b)

Fig. 3. Evolution of best fitness and fitness computed on the testing set, where both plots show the median over 30 independent runs and Box plot comparison of both PEP models.

For both types of PEP models the set $\mathcal{Q}$ of classification problems is divided into a training set and a testing set, each with 50% of the problems, and 30 runs are executed with different random partitions.

For the GP-PEP models the evolution of best fitness and test fitness is presented in Figure 3(a), which shows that the learning process is not over-fitted based on the similarity, with only a marginal difference, of both curves, which represent the median value over all runs.

Figure 3(b) presents a box plot comparison of the 30 executions of the GP-PEP and LQ-PEP models, based on the predictive accuracy achieved on the test set of each run. In general, both PEP models exhibit a very good predictions, with the median error around 5 and 7 percentage points of classification accuracy. However, the figure also shows that the symbolic regression models achieve a better performance, and a statistical t-test confirms this at the 1% significance level. Another look at the predictive accuracy of the PEP models is shown in the scatter plots of Figure 4. In these plots the predictive classification error is plotted against the average error of PGPC on each problem, using the best LQ-PEP and GP-PEP models. It is clear that the prediction of the PEP models is strongly correlated with the average performance of PGPC, and the Pearson's correlation coefficient presented with each plot confirms this observation.



(a) GP-PEP $\rho = 0.86$ (b) LQ-PEP $\rho = 0.77$

Fig. 4. Scatter plots that show the average performance of PGPC (x-axis) and the predicted performance of each PEP model (y-axis). The legend specifies Pearson's correlation coefficient ρ.

5 Concluding Remarks

Is it possible to predict how *hard* a problem is for a Genetic Programming system without actually running the search process? This has been questioned at considerable length within the GP community over the last twenty years, where some good work has been done, with the development of promising proposals and perspectives.

In this paper, two groups of problem difficulty prediction tools are analyzed, named Evolvability Indicators and Predictors of Expected Performance. The

former group of measures attempt to capture how amenable the fitness landscape is to a GP, whereas the latter groups takes as input a set of descriptive features of a problem and produces as output an estimate of the expected performance of the GP search.

The key lessons of this study are the following. Firstly, while EIs (in this work the Negative Slope Coefficient is considered) can give a good estimation of the difficulty of the search problem, they are not necessarily correlated with expected performance; Secondly, the results suggest that PEPs achieve a highly accurate prediction of GP performance.

References

1. L. Altenberg. *The evolution of evolvability in genetic programming*, pages 47–74. MIT Press, Cambridge, MA, USA, 1994.
2. J. Eggermont, J. N. Kok, and W. A. Kosters. Genetic programming for data classification: partitioning the search space. In *Proceedings of the 2004 ACM symposium on Applied computing*, SAC '04, pages 1001–1005, New York, NY, USA, 2004. ACM.
3. E. Galván-López, J. McDermott, M. O'Neill, and A. Brabazon. Defining locality as a problem difficulty measure in genetic programming. *Genetic Programming and Evolvable Machines (accepted)*, 12(4):365–401, 2011.
4. D. E. Goldberg, K. Deb, and J. Horn. Massive Multimodality, Deception, and Genetic Algorithms. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, Amsterdam, 1992. Elsevier Science Publishers, B. V.
5. M. Graff and R. Poli. Practical performance models of algorithms in evolutionary program induction and other domains. *Artif. Intell.*, 174(15):1254–1276, October 2010.
6. M. Graff and R. Poli. Performance models for evolutionary program induction algorithms based on problem difficulty indicators. In *Proceedings of the 14th European conference on Genetic programming*, EuroGP'11, pages 118–129, Berlin, Heidelberg, 2011. Springer-Verlag.
7. T. K. Ho and M. Basu. Complexity measures of supervised classification problems. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(3):289–300, March 2002.
8. T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 184–192, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.
9. K. E. Kinnear. Fitness landscapes and difficulty in genetic programming. In *Proceedings of the First IEEE Conference on Evolutionary Computing*, pages 142–147, Piscataway, NY, 1994. IEEE Press.
10. J. R. Koza. *Genetic programming II: automatic discovery of reusable programs*. MIT Press, Cambridge, MA, USA, 1994.
11. E. G. López, J. McDermott, M. O'Neill, and A. Brabazon. Defining locality in genetic programming to predict performance. In *IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2010.
12. E. G. López, J. McDermott, M. O'Neill, and A. Brabazon. Towards an understanding of locality in genetic programming. In M. Pelikan and J. Branke, editors, *GECCO*, pages 901–908. ACM, 2010.

13. B. Manderick, M. K. de Weger, and P. Spiessens. The genetic algorithm and the structure of the fitness landscape. In R. K. Belew and L. B. Booker, editors, *ICGA*, pages 143–150. Morgan Kaufmann, 1991.
14. D. Michie, D. J. Spiegelhalter, C. C. Taylor, and J. Campbell, editors. *Machine learning, neural and statistical classification*. Ellis Horwood, Upper Saddle River, NJ, USA, 1994.
15. B. Naudts and L. Kallel. A comparison of predictive measures of problem difficulty in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 4(1):1–15, 2000.
16. M. O'Neill, L. Vanneschi, S. Gustafson, and W. Banzhaf. Open issues in genetic programming. *Genetic Programming and Evolvable Machines*, 11(3-4):339–363, September 2010.
17. S. Silva and J. Almeida. Gplab–a genetic programming toolbox for matlab. In L. Gregersen, editor, *Proceedings of the Nordic MATLAB conference*, pages 273–278, 2003.
18. 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.
19. P. F. Stadler and C. R. Stephens. Landscapes and Effective Fitness. *Comments on Theoretical Biology*, 8(4):389–431, 2003.
20. M. Tomassini, L. Vanneschi, P. Collard, and M. Clergue. A study of fitness distance correlation as a difficulty measure in genetic programming. *Evol. Comput.*, 13(2):213–239, June 2005.
21. L. Trujillo, Y. Martínez, E. Galván-López, and P. Legrand. Predicting problem difficulty for genetic programming applied to data classification. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, GECCO '11, pages 1355–1362, New York, NY, USA, 2011. ACM.
22. L. Trujillo, Y. Martínez, and P. Melin. Estimating classifier performance with genetic programming. In *Proceedings of the 14th European conference on Genetic programming*, EuroGP'11, pages 274–285, Berlin, Heidelberg, 2011. Springer-Verlag.
23. L. Vanneschi, M. Clergue, P. Collard, M. Tomassini, and S. Vérel. Fitness clouds and problem hardness in genetic programming. In *GECCO (2)*, pages 690–701, 2004.
24. D. C. Wedge and D. B. Kell. Rapid prediction of optimum population size in genetic programming using a novel genotype -: fitness correlation. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, GECCO '08, pages 1315–1322, New York, NY, USA, 2008. ACM.
25. E. Weinberger. Correlated and uncorrelated fitness landscapes and how to tell the difference. *Biological Cybernetics*, 63(5):325–336, 1990.
26. M. Zhang and W. Smart. Using gaussian distribution to construct fitness functions in genetic programming for multiclass object classification. *Pattern Recogn. Lett.*, 27(11):1266–1274, August 2006.