Variance based Selection to Improve Test Set Performance in Genetic Programming

R. Muhammad Atif Azad
BDS Group
CSIS Department
University of Limerick, Ireland.
atif.azad@ul.ie

Conor Ryan
BDS Group
CSIS Department
University of Limerick, Ireland.
conor.ryan@ul.ie

ABSTRACT

This paper proposes to improve the performance of Genetic Programming (GP) over unseen data by minimizing the *variance* of the output values of evolving models alongwith reducing error on the training data. Variance is a well understood, simple and inexpensive statistical measure; it is easy to integrate into a GP implementation and can be computed over arbitrary input values even when the target output is not known.

Moreover, we propose a simple *variance based selection* scheme to decide between two models (individuals). The scheme is simple because, although it uses bi-objective criteria to differentiate between two competing models, it does not rely on a multi-objective optimisation algorithm. In fact, standard multi-objective algorithms can also employ this scheme to identify good trade-offs such as those located around the *knee* of the *Pareto Front*.

The results indicate that, despite some limitations, these proposals significantly improve the performance of GP over a selection of high dimensional (multi-variate) problems from the domain of symbolic regression. This improvement is manifested by superior results over test sets in three out of four problems, and by the fact that performance over the test sets does not degrade as often witnessed with standard GP; neither is this performance ever inferior to that on the training set. As with some earlier studies, these results do not find a link between expressions of small sizes and their ability to generalise to unseen data.

Categories and Subject Descriptors

I.2.2 [**Artificial Intelligence**]: Automatic Programming—*Program Synthesis, Program Modification*; G.1.6 [**Numerical Analysis**]: Unconstrained Optimization; D.1.2 [**Programming Techniques**]: Automatic Programming; I.2.6 [**Artificial Intelligence**]: Learning—*Induction*; I.5.1 [**Pattern Recognition**]: Models

General Terms

Algorithms, Experimentation, Performance

Keywords

Genetic Programming, Variance, Over-fitting, Symbolic Regression, Regularization

GECCO'11, July 12–16, 2011, Dublin, Ireland.

1. INTRODUCTION

Typically, the goal when using *Machine Learning* [4] is to infer a phenomenon from a finite and short set of samples. This set of samples is called the *training set*. However, these training sets often have limitations: they are only a snap shot of the overall phenomenon, so the *true* overall picture can be somewhat different; moreover, due to lack of accuracy in measuring the data samples, they may have some noise, that is, they may contain errors.

The challenge, then, is to infer the *general* underlying pattern from this finite training set. Thus, as we get closer to explaining (modelling accurately) the training data, we should also explain the *out of sample* (or *test*) data to an acceptable degree of accuracy. However, typically as we reduce the error on the training data, the error on the test data increases. This disparity in errors is often viewed as *over-fitting*, although formal definitions of over-fitting exist [15, pp-67].

The Minimum Description Length (MDL) [15, pp171-174] principle dictates that we should look for *simple* and accurate models of a phenomenon. The MDL also relates to the popular yet debatable argument that since there are fewer short hypotheses than long ones, it is less likely to find a short one that fits the data only coincidentally; thus, an acceptably fit short hypothesis is likely to be a truer explanation of the phenomenon under investigation than an equally good longer one. A word of caution is needed though: the *representation* used by the competing models should be consistent; otherwise, the comparison of sizes becomes meaningless. For example, the size of an expression representing a transcendental function increases manifold if the same function is represented by its Taylor Series. In other words, *compactness* is not always the same as simplicity.

In Genetic Programming (GP), a large body of work looks at reducing the size of the representations. Often termed as *bloat control* [5, 11, 19, 20], this control is necessary to keep the expressions small or compact for a variety of reasons, not least of which is the limited availability of computer memory [14]. However, the question is: can we equate small (possibly compact) models to *simple* models? Although, some

evidence [25] exists, suggesting that controlling the size can promote simplicity and reduce over-fitting, other studies disagree with it [2, 22, 21, 12].

This paper looks at the *smoothness* of *response surfaces* of evolving models in GP to improve performance over the test sets. To estimate the smoothness, we note the *variance* of the output of evolving expressions over the training data set. Variance is a well known statistical measure; it is easy to implement and integrate into a standard GP framework and is computationally inexpensive. In this study, we use GP to evolve models to minimize both error and variance over the training set.

We also introduce a variance based selection scheme so that we can work in a single-objective framework despite having two objectives (namely, error minimization, as in standard GP, but also now variance) . This scheme first decides between two candidate solutions by establishing *Pareto Dominance*. If that is not possible, then it looks for the better trade-off between them. As with [6], we look for a trade-off so that the gain in one objective more than compensates for the loss in the other objective; however, our approach is simpler as we do not have to rank the population according to *dominance scores*, neither do we need to establish the neighbourhood of the solutions in question to make this trade-off.

To ascertain the efficacy of our scheme we compare its performance with that of standard GP on a selection of real world, multi-variate problems from the symbolic regression domain. Results indicate that in three out of four problems our scheme improves performance over the test sets while maintaining significantly lower variance. Even in the fourth problem, the test set performance continues to improve over time. Moreover, when the performance of standard GP on test sets begins to degrade, it remains stable for variance based GP. The results also show that, despite the difference in performance, the sizes of evolving expressions do not differ significantly across the two GP setups.

The paper is organised as follows: section 2 gives the background on theoretical treatment of over-fitting and different notions of variance used in the GP literature to improve test set performance; section 3 introduces variance as a measure of smoothness, describes its limitations and then details the variance based selection scheme; section 4 discusses the experimental setup used in this study, describes the problem suite, presents the results and discusses them; finally, section 5 concludes the paper.

2. VARIANCE AND OVER-FITTING

Perhaps the most popular characterisation of over-fitting in machine learning is the so called *bias-variance trade-off* [4, pp147-152]. Given a modelling method like GP that produces a model $y(\mathbf{x})$ to approximate some target function $\langle t|\mathbf{x}\rangle$, the estimate $y(\mathbf{x})$ usually depends on the training data D sampled from $\langle t|\mathbf{x}\rangle$. Ideally, the models produced in different trials of the modelling method with different instances of D would *consistently* produce an acceptably similar output and low error when evaluated on the same $\mathbf{x}$. Thus, the overall evaluation of a modelling technique amounts to *averaging* the error produced by differently evolved models at point $\mathbf{x}$:

$$E_D[\{y(\mathbf{x}) - \langle t|\mathbf{x}\rangle\}^2] = \underbrace{\{E_D[y(\mathbf{x})] - \langle t|\mathbf{x}\rangle\}^2}_{(bias)^2} + \underbrace{E_D[\{y(\mathbf{x}) - E_D[y(\mathbf{x})]\}^2]}_{variance}$$

where the *bias* measures the extent to which average prediction over all the instances of D differs from the target function; bias is usually approximated with the squared error function in GP literature. The *variance* of output values determines how sensitive the modelling method is to a particular instance of D: the higher the variance, the lower the consistency of output and vice versa. Generally, in GP literature, variance is estimated by evaluating error on a *test* data set not used during model training. Typically, as training performance improves, testing performance deteriorates to implicitly indicate increasing variance. However, some studies mention variance explicitly. A brief review of such studies now follows.

Keijzer and Babovic [10] eliminate variance over a set of instances of D by using ensemble models. Ensemble modelling combines multiple models into a single one. Keijzer and Babovic combine several models, trained over disjoint data sets, by *averaging* their output. Since this average model is the *resulting model*, the variance term is effectively eliminated over the instances of D considered during individual evolutionary runs. While this decreases sensitivity to a particular data set, the authors concede that there is no free lunch: variance is still non-zero over the entire distribution of D.

Moore [16] reduced variance over the test set errors by randomly initialising the training set before evaluating every new generation of individuals. He showed that by periodically changing the training set, the variance across different runs was significantly lower than with having a fixed training set for the entire evolutionary run. Other examples of varying training set to achieve better generalisation include [9] and [3].

2.1 Behavioural Complexity and Over-fitting

As mentioned earlier, *simpler* models can better explain the underlying phenomenon; therefore, they are likely to be of a more consistent quality across different data sets. While a lot of work in GP deals with reducing the size of evolving expressions, far fewer studies address *behavioural* complexity. For example, $sin(x)$ has fewer nodes than $x + x + x + x$, but is behaviourally more complex [2]. In other words, $x + x + x + x$ has a *smoother response surface* [22].

To evolve smoother models with STROGANOFF (a system for evolving tree-like polynomials) Nikolaev et al. [17] use *ridge regression*, minimising the magnitude of coefficients alongside enhancing model accuracy. Since, large coefficients suggest more variability in the response surface, ridge regression penalizes the corresponding individual through a *regularization parameter* in the fitness function. Then, in [18] they directly measure the curvature of evolving polynomials with the *variance functional* $V[f]$ such that

$$V[f] = \int \left| \frac{\partial^2 f(\mathbf{x})}{\partial x^2} \right|^2 d\mathbf{x}.$$

The polynomial $f(\mathbf{x})$ should be twice differentiable; Nikolaev et al. satisfied such a hard constraint with a specialised functions set of *basis* polynomials.

Vladislavleva et al. [22] estimate the smoothness by estimating the *non-linearity* of evolving expressions. Instead of computing derivatives, they approximate a GP evolved expression with a *Chebyshev* polynomial. The non-linearity of a GP-function occupying a node in a GP tree is a function of the degree of the Chebyshev polynomial that approximates that GP-function given the range of input values feeding into its node from the child nodes and the non-linearity of the child nodes. The ranges of values input to the GP node in question and the corresponding Chebyshev polynomial are determined during fitness calculation. Vladislavleva et al. used a multi-objective approach to minimise non-linearity and approximation error; they switched non-linearity with expression sizes in alternate generations to evolve compact models with smooth surfaces.

Castelli et al. [7] use a multi-objective algorithm to minimise the training error and the variance of *errors on the training set*. Reducing the variance of errors means that the evolving models should consistently fit all the training points regardless of the smoothness of associated response surface. This improved performance on a particular test set; however, they also found that if they replaced the variance of errors with the number of nodes as an objective, the multi-objective set-up performed just as well. Moreover, countering bloat with a single objective algorithm did not improve the performance over the test set.

3. OUR APPROACH: VARIANCE BASED SELECTION

We aim for a simple measure of smoothness of a model: the measure should be cheap to compute, easy to understand and easy to plug into a GP implementation. Furthermore, to use this measure we should not necessarily *require* a multi-objective algorithm for optimisation. To fulfil these aspirations, we estimate smoothness by simply measuring the variance of output values of an expression over the *training* data. To use it within a single-objective optimisation framework, we later describe a modified tournament selection scheme.

We must note that the variance of output of a function is not the same as the variance of its errors with respect to a target data distribution. The former is a measure of smoothness *independent of target output*, whereas the latter is a measure of consistent approximation. While the latter can still improve generalisation by preferring the models that fit the training data overall, it can not be used to ascertain the smoothness of the model in question beyond the *known* data. Instead, we can still measure the variance of output over any arbitrary input values (without knowing the corresponding target output) to estimate how the model behaves beyond the training points. This can be particularly useful because in real life problems data can be in short supply. Also, reducing variance directly over the training data may counteract minimisation of error. However, in this study we measure the variance on just the training points.

Variance over training data should provide a good measure of smoothness as a smoother response surface should have less variance than that of an over-fitting and noise hugging model. However, we do recognise that it is not a strictly monotonic measure of smoothness. To illustrate this point, consider $y_1 = x$ and $y_2 = sin(x)$ over a range $[0 : 1]$, then $var(y_2) < var(y_1)$; $var(y_i)$ is the variance of y_i. However, with a data set representative of a linear function, the error for a linear approximation should outweigh the lesser variance associated with the non-linear function; otherwise, a linear function of such a high *slope* would seriously over-fit if the target model is significantly non-linear. Moreover, we are more concerned with close competitions such as, for example, between $y_2 = sin(x)$ and $y_3 = sin(2\pi x)$ when GP attempts to fine tune the evolving models: $var(y_2) < var(y_3)$.

Another issue with variance is that, unlike derivatives, it is oblivious to the change over the input axis: a high variance over a large Δx may still represent a smooth surface. As this is a preliminary investigation we do not expect to find a fool proof measure, neither do we know if the associated complexity of such a measure would translate into significantly better results. However, we highlight the *possible* limitations so future investigations can be mindful of them and may even address them.

We set the variance of the output values of training data as the maximum allowable variance for any model during evolution: any model with higher variance is deemed over-complex and is assigned the worst possible fitness value. For example, a model that outputs values that oscillate *about* the target values may have the same mean squared error as the one that consistently outputs either higher or lower values. In this case the first model is over-complex and highly likely to over-fit, and is duly penalized.

3.1 Modified Tournament Selection

Initially, we linearly added variance and training error as a fitness measure; however, the evolution almost ignored the variance term and solely targeted error reduction. Therefore, instead of working out an optimum *weight* for the variance term to suitably calibrate the fitness function, we opted for a tournament selection scheme that considers both training error and variance to discriminate between two individuals. Although, we could use a multi-objective GA, we kept to a single-objective GA in this study.

In this scheme we compare two candidate models **A** and **B** in a step wise fashion: if we can not decide in one step then we move to the next step. First, we check if **A** *dominates* **B**, i.e., if **A** is at least equal to **B** in one objective and superior in the other objective then **A** wins. If neither model dominates the other then we ascertain if **A** improves over **B** in one objective without giving away as much in the other objective [1]; such an attempt can take us closer to the *knee* of the *Pareto Front* [6]. To decide this we determine if the *rectilinear distance* of **A** ($|error + variance|$) is smaller than the *euclidean distance* ($\sqrt{error^2 + variance^2}$) of **B**. This situation is exemplified in Figure 1: if **A** falls in the shaded regions then it is selected. However, if we are still undecided, then we pick the model with smaller variance. If **A** is such a model, it should fall in the *Tie Breaking Region* in Figure 1. Finally, if all options are exhausted then the model with smaller number of nodes is preferred. Pseudocode for this selection scheme is given in Algorithm 1.

As a housekeeping measure, we penalize the individuals

[1]It is easy to show that by selecting a point in the shaded regions, gain in one dimension significantly outdoes the loss in the other. Suppose point **A** lies in the upper shaded region, then $\Delta x/\Delta y = cot(\theta)$, where θ is the angle between $\overline{\mathbf{AB}}$ and the *normal* drawn from **B** on the $y-$axis. Since $\theta < \pi/4$ (unless **B** lies on the $x-$axis), $cot(\theta) > 1$. A similar reasoning applies to the lower shaded region.

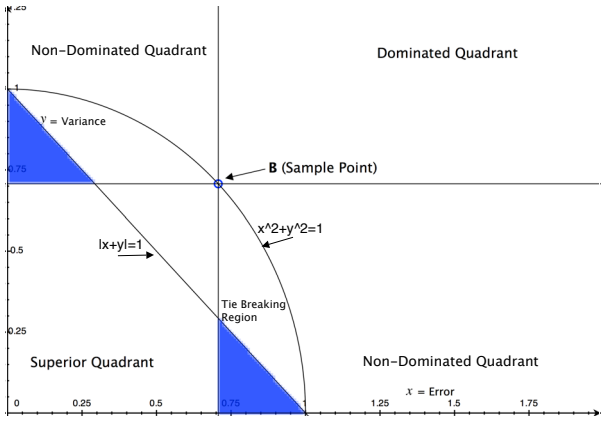

Figure 1: The variance based selection scheme is depicted. A sample point (B) is selected from an arc in the error-variance space such that the radius of this arc = 1; it is not the Pareto Front. The straight line shows the points with rectilinear distance = 1. Shaded regions are the set of points from the non-dominated regions such that their rectilinear distance distance is less than the euclidean distance of point B. Tie breaking region has the points from the non-dominated region that have a lower variance than B and do not fall in the shaded region.

generating NaN values by assigning them the worst possible values for training score and variance.

Algorithm 1 Variance based selection is outlined.

```
a ← false
if A ≻ B then {A ≻ B ⇒ A dominates B}
  a ← true
else
  if (B ⊁ A) ∧ (rect(A) < eucl(B)) then
    a ← true
  else
    if (rect(B) ≥ eucl(A)) ∧ (var(B) > var(A)) then
      a ← true
    else
      if (var(B) = var(A)) ∧ (nodes(A) < nodes(B))
      then
        a ← true
      end if
    end if
  end if
end if
```

4. EXPERIMENTS

To estimate the efficacy of the proposed method, we compare its performance with standard GP on four multi-variate problems from the domain of symbolic regression. Also, as an additional benchmark, we replace variance with the number of nodes in the selection scheme. For each of these problems Table 1 lists the corresponding labels used henceforth to refer to them, as well as the input dimensionality and the number of data points available. As a policy uniform across all the problems, 70% of the data points were used for training during evolutionary runs; the remaining 30% were reserved for testing. A brief description of each problem now follows.

Table 1: Problem suite used in this study.

Problem Label	|{Input Variables}|	|{Data Points}|
`Dow Chemical`	57	1066
`Toxicity`	626	234
`Bioavailability`	241	359
`Concrete Strength`	8	1030

Table 2: Configuration parameters for the runs.

Population Size	500.
Run Terminates at	exhausting 1.2×10^6 tree nodes.
Operator Probabilities	Xover: 0.9; Point mutation: 0.1.
Tournament Size	2
Lexicographic [13] Parsimony Pressure	Yes.
Replacement	Steady state, inverse tournament
Functions set	$\{+, -, \times, /\}$
Terminal set	{Input variables} $\cup$ ERC.
ERC	ERC $= \{c \mid c \in \Re \wedge -5 \leq c < 5\}$. $\lvert\text{ERC}\rvert = 50$.
Normalised Fitness	$\frac{1}{1+MSE}$
Initialisation	Ramped half & half (max. initial depth = 4)

The first problem, `Dow Chemical`, presents data from a real industrial application at Dow Chemical[2]. The objective is to map 57 process measurements such as temperature, pressures and flows to a chemical composition.

The second problem, `Toxicity`, is another real life problem; it involves mapping 626 factors representing the molecular structures of a set of candidate drug compounds to a *pharmacokinetic* parameter. The parameter of choice in this case is median lethal dose, also informally called toxicity.

As with `Toxicity`, in the third problem (`Bioavailability`), we predict another parameter of drug compounds: 241 input variables are used to predict the percentage of orally submitted dose of a drug that effectively reaches the systemic blood circulation. Further details of these two problems can be obtained from [1].

In the last problem, `Concrete Strength`, we predict a quantitative value of compressive strength of concrete. This output is a highly non-linear function of eight input variables; these variables characterise the composition of concrete. The data source is UCI Machine Learning repository [24]; the problem itself is detailed in [23].

4.1 GP Parameters

Configuration of GP parameters for this study is shown in Table 2. Since, we only want to ascertain improvement in performance with variance based selection, we do not attempt to tailor the experimental set-ups to each problem; instead, we uniformly employ a standard set-up.

During population initialisation and mutation, we select ephemeral random constants (ERCs) as often as the problem

[2]This problem was presented as a challenge organised by Arthur Kordon at EvoStar 2010. For details see: http://casnew.iti.upv.es/index.php/evocompetitions/105-symregcompetition

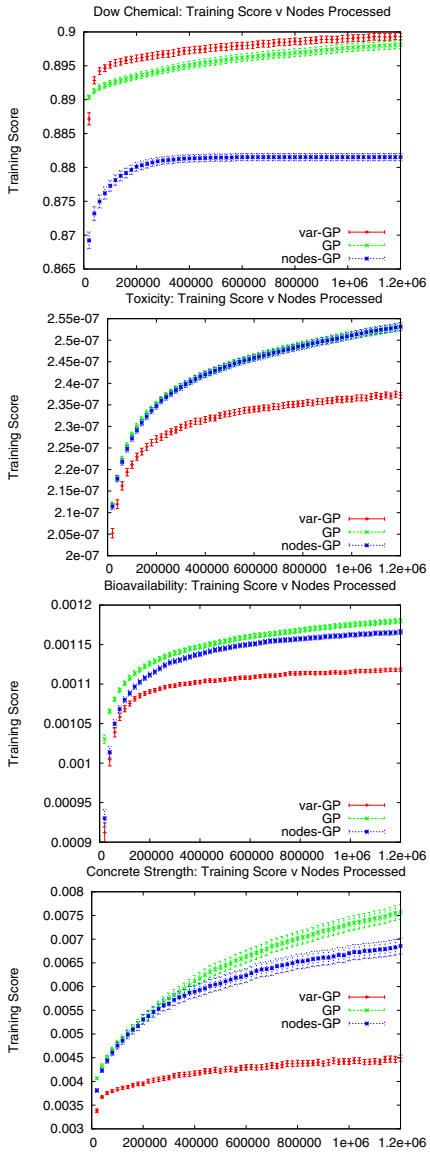

Figure 2: For the best fit individual, mean score on the training data is plotted for each problem.

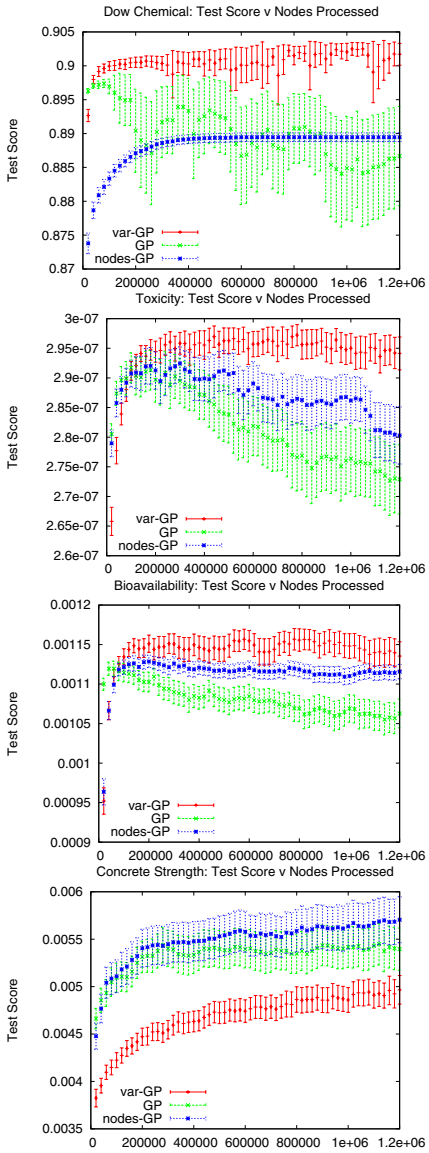

Figure 3: For the best fit individual, mean score on the unseen data is plotted for each problem.

specific variables. To do this, first we randomly decide between a constant and a variable, before choosing uniformly from within their sets.

4.2 Performance Measures

Our primary measure of performance difference is normalised error on unseen data (Test Score); however, we also note normalised error on training set (Training Score), variance on training set and size of the evolving expressions. While we measure training and test set performances for obvious reasons, we also look for a consistently lower variance to correlate it with any performance differences. Similarly, expression sizes are plotted to verify if any difference in test set performance can be linked to reduction in sizes of evolving expressions: some qualitative [2, 22] and quantitative arguments [21, 12] go against a strong or causal link.

We note all these statistics for the best *fit* individual. The best fit individual in regular GP runs has the best training score in the population; correspondingly, the best-fit individual in variance based GP is the best as per criteria outlined in section 3.

We ascertain the significance of performance differences between the two GP setups. To facilitate this, each sampled point in the performance plots depicts an average over 500 runs. Then, as in [8], the 95% confidence limits of the error bars at each point are computed as follows:

$$\overline{X} \pm 1.96 \frac{\sigma}{\sqrt{n}}$$

where $\overline{X}$ and σ are the mean and standard deviation of n observations; $n = 500$ represents the number of runs in this case. We can be 95% confident that the statistical population lies within these limits, and that a lack of overlap with

another error bar means that the corresponding populations are different.

Ideally, the results with variance based selection should be superior to those with the corresponding benchmarks on all counts; however, some trade-off is expected between training and test set performances. Moreover, a lesser training performance should coincide with a stable or an improving performance over the test set.

4.3 Results

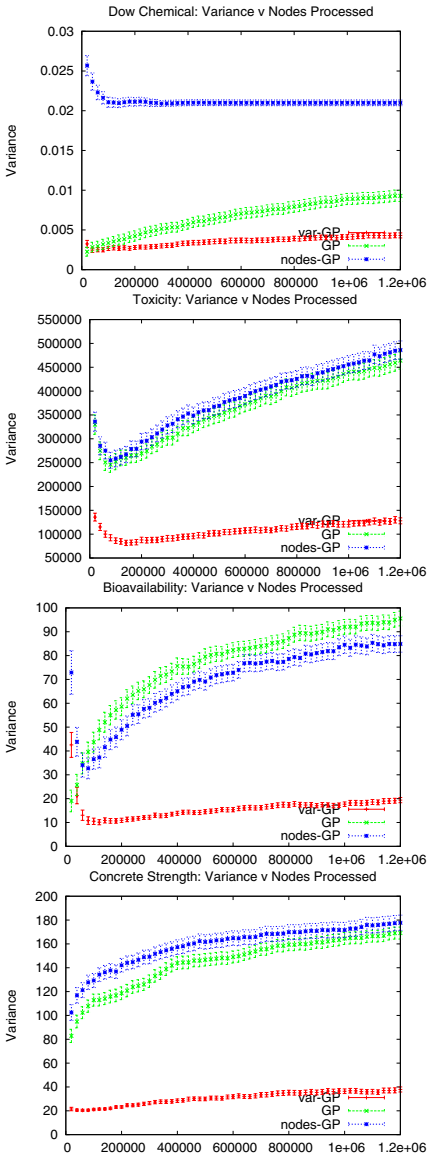

Figure 4: For the best fit individual, variance on training data is plotted for each problem.

Figures 2-5 show results of the experiments. The figures refer to standard GP as `GP`, to size minimising GP as `nodes-GP`, and to variance based GP as `var-GP`.

First, we compare `var-GP` with `GP`. Figures 2 and 3 show results on training and test sets respectively. Clearly, `var-GP` performs better than `GP` on both training and test sets for the first problem (`Dow Chemical`). For `Toxicity` and

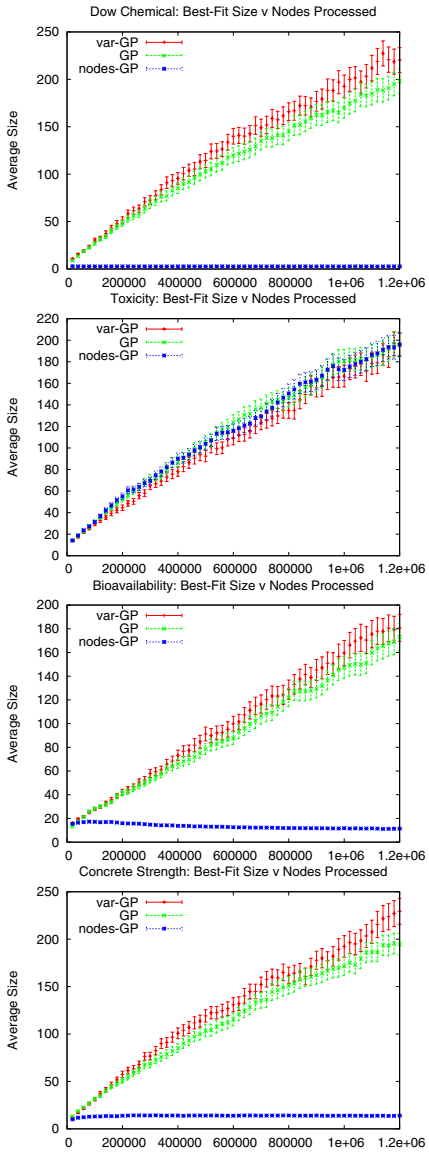

Figure 5: For the best fit individual, tree size is plotted for each problem.

`Bioavailability`, `var-GP` sacrifices some training performance for a relatively stable and superior test set performance. However, in the last problem, `Concrete Strength`, `var-GP` is inferior both in training and test set performances. Still, in all the cases with `var-GP` score on test set never degrades in the manner shown by `GP` in Figure 3 for the first three problems.

Also, from the scales of the Figures 2 and 3, we notice that when evolution stops, `var-GP` performs at least as well on the test sets as it does on the training sets. The same is not true for `GP` in the last two problems.

Results in Figures 4 and 5 are consistent across all the problems. Figure 4 shows that variance with `var-GP` on training sets is significantly lower than that with `GP`. Figure 5 shows that the expressions sizes are very similar across the two setups despite considerable differences observed otherwise. Hence, this result supports the previously cited ar-

gument that differing test set performance can not be correlated with a change in tree sizes.

For `nodes-GP`, the training and test results are closer to those with `GP` than those with `var-GP`. When compared with `var-GP`, the test results of `nodes-GP` on the first two problems are consistently inferior; on `Bioavailability` the results are generally inferior but converge towards the end of the runs. On the last problem, as with `GP`, `nodes-GP` performs better than `var-GP`.

The results for variance and expression sizes show that despite evolving much smaller expressions (except on `toxicity`) than with the other two setups, `nodes-GP` has a much higher variance than with `var-GP`.

4.4 Discussion

The scales of training results on the four problems show that their difficulty for Genetic Programming varies. For the first problem, GP attains a mean training score of 0.89 whereas for `Toxicity`, the second problem, the scores are of the order of 10^{-7}; the other two problems fall somewhere in the middle. It may be possible to improve performance by using some non-linear functions (for example, transcendental functions); however, that is not the objective here. What is important is that despite so much variation on the difficulty scale, `var-GP` performs better than `GP` on test sets; otherwise, it consistently improves over time (as happens in the last problem). Also, unlike the two benchmarks, the performance of `var-GP` on the test sets at the end of the runs is *never* worse than the corresponding training performance.

We also find that, although `nodes-GP` can significantly contain code-growth, only once does it perform better than `GP` on the test sets. Thus, unlike Castelli et al. [7], we can not conclude yet that bi-objective optimisation, regardless of the objectives, consistently improves test set performance. Moreover, since neither this study nor that of Castelli et al. involved compact functions (for example, transcendental functions), some relationship between size and simplicity might exist. However, we do agree that there can be merit in further investigating the effect of multi-objective optimisation on test set performance.

Given the approach taken in this paper, reducing variance on the *training* data may hinder decreasing the error as much as it could be. Perhaps this happens in the last problem where an inferior training performance could not translate into a superior test set performance in the available time frame. However, this is a preliminary investigation. Further work can look into utilising data sets separate from training set to avoid a direct conflict with error reduction. However, additional data can be expensive or even unavailable for many real world applications: for example, the data sets for the second and third problem are already very sparse. Therefore, the new techniques should not demand too much data.

5. CONCLUSIONS

In this paper, we have proposed to use variance of output values over the training data to measure *smoothness* of response surfaces that we evolve with Genetic Programming. We propose this measure because it is widely familiar, easy to implement and integrate into a typical GP implementation, and does not significantly add to the computational expense of GP runs. We highlight that variance is not a strictly monotonic measure of smoothness but we also discuss the mitigating circumstances.

We also propose a simple tournament scheme that considers both training error and variance in discriminating between two candidate solutions. While, the foremost criterion used to discriminate is Pareto Dominance, we also use a number of secondary criteria. We do so to avoid using a standard multi-objective algorithm that ranks the population based on a variety of factors. Among the secondary criteria, first we look for a good trade-off: the gain in one dimension should more than offset the loss in the other. As we understand, the approach used to find this trade-off is novel. If we can not get a suitable trade-off, then we select the solution with lower variance or with fewer number of nodes in that order of priority.

We test our proposals on four high dimensional real life problems and find that in three cases we get test set results better than those with standard GP. Moreover, the test set performance never degrades in the manner associated with standard GP - a manner also visible in three of the four problems. Also, this performance is never worse than that on the training set. Finally, we observe that the difference in the test set performances does not correspond to a change in the sizes of evolving expressions.

The study also opens up further research avenues. These include finding a better yet simple measure of function variability for GP, using a standard multi-objective GA with the same or new measures, using standard approaches to find the *knee-solutions*, and using a data set other than training set to estimate variance without compromising on the learning efficiency particularly when training data is scarce.

6. REFERENCES

[1] F. Archetti, S. Lanzeni, E. Messina, and L. Vanneschi. Genetic programming for human oral bioavailability of drugs. In M. Keijzer, M. Cattolico, D. Arnold, V. Babovic, C. Blum, P. Bosman, M. V. Butz, C. Coello Coello, D. Dasgupta, S. G. Ficici, J. Foster, A. Hernandez-Aguirre, G. Hornby, H. Lipson, P. McMinn, J. Moore, G. Raidl, F. Rothlauf, C. Ryan, and D. Thierens, editors, *GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, volume 1, pages 255–262, Seattle, Washington, USA, 8-12 July 2006. ACM Press.

[2] R. M. A. Azad and C. Ryan. Abstract functions and lifetime learning in genetic programming for symbolic regression. In J. Branke, M. Pelikan, E. Alba, D. V. Arnold, J. Bongard, A. Brabazon, J. Branke, M. V. Butz, J. Clune, M. Cohen, K. Deb, A. P. Engelbrecht, N. Krasnogor, J. F. Miller, M. O'Neill, K. Sastry, D. Thierens, J. van Hemert, L. Vanneschi, and C. Witt, editors, *GECCO '10: Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 893–900, Portland, Oregon, USA, 7-11 July 2010. ACM.

[3] T. F. Bersano-Begey and J. M. Daida. A discussion on generality and robustness and a framework for fitness set construction in genetic programming to promote robustness. In J. R. Koza, editor, *Late Breaking Papers at the 1997 Genetic Programming Conference*, pages 11–18, Stanford University, CA, USA, 13–16 July 1997. Stanford Bookstore.

[4] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[5] T. Blickle and L. Thiele. Genetic programming and redundancy. In J. Hopf, editor, *Genetic Algorithms within the Framework of Evolutionary Computation (Workshop at KI-94, Saarbrücken)*, pages 33–38, Im Stadtwald, Building 44, D-66123 Saarbrücken, Germany, 1994. Max-Planck-Institut für Informatik (MPI-I-94-241).

[6] J. Branke, K. Deb, H. Dierolf, and M. Osswald. Finding Knees in Multi-Objective Optimization. In *Parallel Problem Solving from Nature - PPSN VIII*, pages 722–731, Birmingham, UK, Sept. 2004. Springer-Verlag. Lecture Notes in Computer Science Vol. 3242.

[7] M. Castelli, L. Manzoni, S. Silva, and L. Vanneschi. A comparison of the generalization ability of different genetic programming frameworks. In *IEEE Congress on Evolutionary Computation (CEC 2010)*, Barcelona, Spain, 18-23 July 2010. IEEE Press.

[8] D. Costelloe and C. Ryan. On improving generalisation in genetic programming. In L. Vanneschi, S. Gustafson, A. Moraglio, I. De Falco, and M. Ebner, editors, *Proceedings of the 12th European Conference on Genetic Programming, EuroGP 2009*, volume 5481 of *LNCS*, pages 61–72, Tuebingen, Apr. 15-17 2009. Springer.

[9] J. M. Daida, T. F. Bersano-Begey, S. J. Ross, and J. F. Vesecky. Computer-assisted design of image classification algorithms: Dynamic and static fitness evaluations in a scaffolded genetic programming environment. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 279–284, Stanford University, CA, USA, 28–31 July 1996. MIT Press.

[10] M. Keijzer and V. Babovic. Genetic programming, ensemble methods and the bias/variance tradeoff - introductory investigations. In R. Poli, W. Banzhaf, W. B. Langdon, J. F. Miller, P. Nordin, and T. C. Fogarty, editors, *Genetic Programming, Proceedings of EuroGP'2000*, volume 1802 of *LNCS*, pages 76–90, Edinburgh, 15-16 Apr. 2000. Springer-Verlag.

[11] W. B. Langdon. Quadratic bloat in genetic programming. In D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, and H.-G. Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, pages 451–458, Las Vegas, Nevada, USA, 10-12 July 2000. Morgan Kaufmann.

[12] F. Larkin. *Artificial Evolution Approaches to Address the Data Challenges Encountered During Financial Forecasting*. PhD thesis, University of Limerick, May 2010.

[13] S. Luke and L. Panait. Lexicographic parsimony pressure. In W. B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke, and N. Jonoska, editors, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 829–836, New York, 9-13 July 2002. Morgan Kaufmann Publishers.

[14] S. Luke and L. Panait. A comparison of bloat control methods for genetic programming. *Evolutionary Computation*, 14(3):309–344, Fall 2006.

[15] T. M. Mitchell. *Machine learning*. McGraw Hill, New York, US, 1996.

[16] F. W. Moore. Improving means and variances of best-of-run programs in genetic programming. In M. W. Evens, editor, *Proceedings of the Ninth Midwest Artificial Intelligence and Cognitive Science Conference (MAICS-98)*, pages 95–101, Russ Engineering Center, Wright State University, Dayton, Ohio, USA, 20-22 Mar. 1998. AAAI Press.

[17] N. Nikolaev, L. M. de Menezes, and H. Iba. Overfitting avoidance in genetic programming of polynomials. In D. B. Fogel, M. A. El-Sharkawi, X. Yao, G. Greenwood, H. Iba, P. Marrow, and M. Shackleton, editors, *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*, pages 1209–1214. IEEE Press, 12-17 May 2002.

[18] N. Y. Nikolaev and H. Iba. Regularization approach to inductive genetic programming. *IEEE Transactions on Evolutionary Computing*, 54(4):359–375, Aug. 2001.

[19] R. Poli and N. McPhee. Parsimony pressure made easy. In M. Keijzer, G. Antoniol, C. B. Congdon, K. Deb, B. Doerr, N. Hansen, J. H. Holmes, G. S. Hornby, D. Howard, J. Kennedy, S. Kumar, F. G. Lobo, J. F. Miller, J. Moore, F. Neumann, M. Pelikan, J. Pollack, K. Sastry, K. Stanley, A. Stoica, E.-G. Talbi, and I. Wegener, editors, *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 1267–1274, Atlanta, GA, USA, 12-16 July 2008. ACM.

[20] T. Soule and J. A. Foster. Effects of code growth and parsimony pressure on populations in genetic programming. *Evolutionary Computation*, 6(4):293–309, Winter 1998.

[21] L. Vanneschi, M. Castelli, and S. Silva. Measuring bloat, overfitting and functional complexity in genetic programming. In J. Branke, M. Pelikan, E. Alba, D. V. Arnold, J. Bongard, A. Brabazon, J. Branke, M. V. Butz, J. Clune, M. Cohen, K. Deb, A. P. Engelbrecht, N. Krasnogor, J. F. Miller, M. O'Neill, K. Sastry, D. Thierens, J. van Hemert, L. Vanneschi, and C. Witt, editors, *GECCO '10: Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 877–884, Portland, Oregon, USA, 7-11 July 2010. ACM.

[22] 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 Transactions on Evolutionary Computation*, 13(2):333–349, Apr. 2009.

[23] I. C. Yeh. Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete Research*, 28(12):1797 – 1808, 1998.

[24] I. C. Yeh. UCI machine learning repository, 2007.

[25] B.-T. Zhang and H. Mühlenbein. Balancing accuracy and parsimony in genetic programming. *Evolutionary Computation*, 3(1):17–38, 1995.