Guidelines for defining benchmark problems in Genetic Programming

Miguel Nicolau and Alexandros Agapitos and Michael O'Neill and Anthony Brabazon
Natural Computing Research & Applications Group
Complex & Adaptive Systems Laboratory
University College Dublin, Ireland
Email: {miguel.nicolau, alexandros.agapitos, m.oneill, anthony.brabazon}@ucd.ie

***Abstract*—The field of Genetic Programming has recently seen a surge of attention to the fact that benchmarking and comparison of approaches is often done in non-standard ways, using poorly designed comparison problems. We raise some issues concerning the design of benchmarks, within the domain of symbolic regression, through experimental evidence. A set of guidelines is provided, aiming towards careful definition and use of artificial functions as symbolic regression benchmarks.**

I. Introduction

The Genetic Programming research community has recently recognised the need for defining a suite of benchmark problems. So far, the effort has focussed on surveying the literature and choosing a set of problems depending on their popularity and difficulty [1], [2]. Clearly, the definition of a benchmark suite is an ongoing process – new problems should gain traction and will be added to sets of benchmarks. The aim of the present paper is to suggest guidelines for generating synthetic problems for symbolic regression tasks. Our work therefore has implications for the broader project of defining a detailed framework for benchmark creation.

In a symbolic regression problem, one is given a set of N training examples $\{(x_i, y_i)\}_1^N$, where $y \in \mathbb{R}$ is the response variable and $\mathbf{x} \in \mathbb{R}^d$ is a vector of explanatory variables. The goal is to find a function $F^* : \mathbb{R}^d \rightarrow \mathbb{R}$, such that over the joint distribution $P(\mathbf{x}, y)$ the expected value $\mathbb{E}_{x,y}$ of some specified loss function $L(y, F(\mathbf{x}))$ is minimised:

$$F^*(\mathbf{x}) = \underset{F(\mathbf{x})}{\arg\min} \mathbb{E}_{x,y}[L(y, F(\mathbf{x}))] \tag{1}$$

A typical loss function for such problems is *squared error*, where $L(y, F(\mathbf{x})) = (y - F(\mathbf{x}))^2$.

The motivation for working towards a framework for synthetic problem generation comes from the following observations of typical common practice:

1) The input range of synthetic functions is often arbitrarily chosen.
2) The training sample size N is often arbitrarily defined.
3) Function-set elements are often tailored for specific problems.
4) Lack of predefined and available training/test sets, which often hinders the reproducibility of results.
5) The lack of baselines for contrasting performance of evolved models.
6) The lack of noisy versions of the problems (real-world problems are inherently noisy).
7) In general, the constraint imposed by most learning algorithms can be described as *smoothness* restrictions of one kind or another. This essentially requires a regular behaviour of the underlying target function in small neighbourhoods of the input space. That is, for all input points sufficiently close to each other in some metric, the target function exhibits some behaviour that may be approximated using a constant, linear or low-order polynomial fit.

In this paper, we ran a set of experiments, to empirically test the effect of these observations in the performance of two typical Genetic Programming (GP) systems: Koza-like tree-based GP [3] and Grammatical Evolution (GE) [4]. The objective of these experiments is not to accurately measure and compare both systems, but rather to highlight the effect of neglecting some of the observations made above, when designing benchmark tests.

The next section presents an analysis of steps required when creating benchmarks. Section III presents the experimental setup of the problems and algorithms used, which are analysed in Section IV; finally, Section V draws conclusions and future work directions.

II. Creating Benchmarks

A. Generating Datasets

When designing a benchmark problem using a known function, a dataset must be generated. This usually involves designing a function, choosing the range of the inputs, and generating a dataset.

Not all functions are suitable to be used as benchmarks, however. As an experiment, we generated datasets from the functions shown in Table I. Most of these were extracted from seminal works [5], [6], [7], [8], [9], suggested in the GP benchmarks effort [2]; functions $F14$, $F16$ and $F20$ were developed for this study. All functions were uniformly sampled from the input range $[-5 \dots 5]$ [1].

[1] This was done to deliberately highlight the problem of arbitrarily choosing input ranges; when designing benchmarks, input range sampling techniques (such as Latin Hypercube Sampling [10]) should be used.

978-1-4799-7492-4/15/$31.00

TABLE I. SYMBOLIC REGRESSION PROBLEMS

F_1 :	$f(x_1, x_2) = \frac{e^{-(x_1-1)^2}}{1.2+(x_2-2.5)^2}$
F_2 :	$f(x_1, x_2) =$ $e^{-x_1} x_1^3 cos(x_1) sin(x_1)(cos(x_1) sin^2 x_1 - 1)(x_2 - 5)$
F_3 :	$f(x_1, x_2, x_3, x_4, x_5) = \frac{10}{5+\sum_{i=1}^{5}(x_i-3)^2}$
F_4 :	$f(x_1, x_2, x_3) = 30 \frac{(x_1-1)(x_3-1)}{{x_2}^2(x_1-10)}$
F_5 :	$f(x_1, x_2) = 6 sin(x_1) cos(x_2)$
F_6 :	$f(x_1, x_2) = (x_1 - 3)(x_2 - 3) + 2 sin((x_1 - 4)(x_2 - 4))$
F_7 :	$f(x_1, x_2) = \frac{(x_1-3)^4+(x_2-3)^3-(x_2-3)}{(x_2-2)^4+10}$
F_8 :	$f(x_1, x_2) = \frac{1}{1+{x_1}^{-4}} + \frac{1}{1+{x_2}^{-4}}$
F_9 :	$f(x_1, x_2) = {x_1}^4 - {x_1}^3 + {x_2}^2/2 - x_2$
F_{10} :	$f(x_1, x_2) = \frac{8}{2+{x_1}^2+{x_2}^2}$
F_{11}	$f(x_1, x_2) = {x_1}^3/5 + {x_2}^3/2 - x_2 - x_1$
F_{12} :	$f(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}) =$ $x_1 x_2 + x_3 x_4 + x_5 x_6 + x_1 x_7 x_9 + x_3 x_6 x_{10}$
F_{13} :	$f(x_1, x_2, x_3, x_4, x_5) = -5.41 + 4.9 \frac{x_4-x_1+x_2/x_5}{3x_4}$
F_{14} :	$f(x_1, x_2, x_3, x_4, x_5, x_6) = (x_5 x_6)/(\frac{x_1}{x_2} \frac{x_3}{x_4})$
F_{15} :	$f(x_1, x_2, x_3, x_4, x_5) = 0.81 + 24.3 \frac{2x_2+3x_3^2}{4x_4^3+5x_5^4}$
F_{16} :	$f(x_1, x_2, x_3, x_4, x_5) = 32 - 3 \frac{tan(x_1)}{tan(x_2)} \frac{tan(x_3)}{tan(x_4)}$
F_{17} :	$f(x_1, x_2, x_3, x_4, x_5) = 22 - 4.2(cos(x_1) - tan(x_2))(\frac{tanh(x_3)}{sin(x_4)})$
F_{18} :	$f(x_1, x_2, x_3, x_4, x_5) = x_1 x_2 x_3 x_4 x_5$
F_{19} :	$f(x_1, x_2, x_3, x_4, x_5) = 12 - 6 \frac{tan(x_1)}{e^{x_2}}(x_3 - tan(x_4))$
F_{20} :	$f(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}) = \sum_{i=1}^{5} 1/x_i$
F_{21} :	$f(x_1, x_2, x_3, x_4, x_5) = 2 - 2.1 cos(9.8x_1) sin(1.3x_5)$

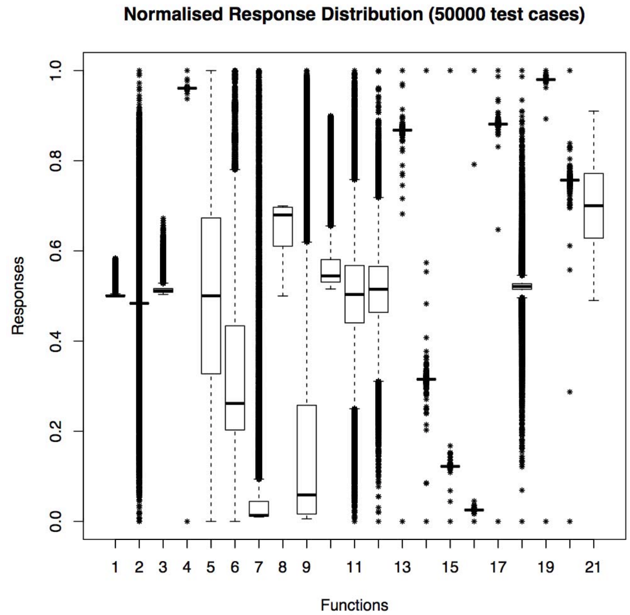

Fig. 1. Normalised response variable distribution over 50000 samples, for each function.

Although using inputs sampled from the same range, the response variable for each function can have a completely different distribution. Fig. 1 shows an histogram of the response variable for each function defined in Table I, with 50000 samples drawn randomly from the input range specified. The responses were normalised to the range $[0 \ldots 1]$, using the formula:

$$r_i = \frac{v_i - v_{\text{min}}}{v_{\text{max}} - v_{\text{min}}}$$

where v_i is a response value, v_{min} is the minimum response observed, v_{max} is the maximum response observed, and r_i is the normalised response value.

The figure shows how varied the distribution of response values can be. Some functions are evenly distributed over the response domain, such as $F5$ or $F21$, while others present more challenging skewed distributions, such as $F8$ or $F9$. Other functions present very challenging distributions, such as $F18$, with a tail of outlier values spread upwards and downwards from the median, or even functions such as $F16$, with only a handful of outliers, very distant and unevenly distributed from the median.

The latter two types of response distributions are very challenging as benchmarks, as they are very erratic, and unlikely to be found in real-world problems. To model such responses, a much larger sample set would be required (50000 samples were used to plot each function). Furthermore, the use of MSE as an error measure (as in the majority of evolutionary systems for symbolic regression) will result in very large errors, due to the extreme values found in the response variable. For example, the 50000 samples drawn from function $F16$ have a median of 32.0, with an inter-quartile range of 6.07, but minimum and maximum observed samples of $-6.527053E6$ and $2.50911394E8$ respectively.

B. Sample Size

The distributions shown in Fig. 1 highlight how unevenly distributed a response variable can be, over the chosen input range. The size of the training and test samples must reflect this, and be adjusted accordingly.

Fig. 2 shows data generated in the same way as in Fig. 1, but with sample sizes of 50 (top) and 1000 (bottom), for training purposes. These figures show how the sample size can drastically affect the distribution of responses. Some functions like $F5$ and $F21$ present essentially the same distribution across the response range. Functions such as $F11$ and $F12$ show essentially the same distribution with different sample sizes, but with a much higher density of outliers, making them very challenging. Finally, functions such as $F13$ to $F20$ show very radical changes in the distribution of responses, an indication both of ruggedness and insuitable small sample sizes.

C. Different Sample Sizes

The previous sections also highlight how the size of a drawn sample can paint a different picture of the behaviour of the underlying function. As such, when designing benchmarks, different sample sizes can be used, to control the difficulty of the problem. In this study, training sets of 50, 100, 500 and 1000 samples were used.

D. Using the same samples

The previous section also shows how different input sample sets can have very different response distributions. Depending on the ruggedness of the function, this can have the effect of substantially altering the performance of the learning method applied to a benchmark.

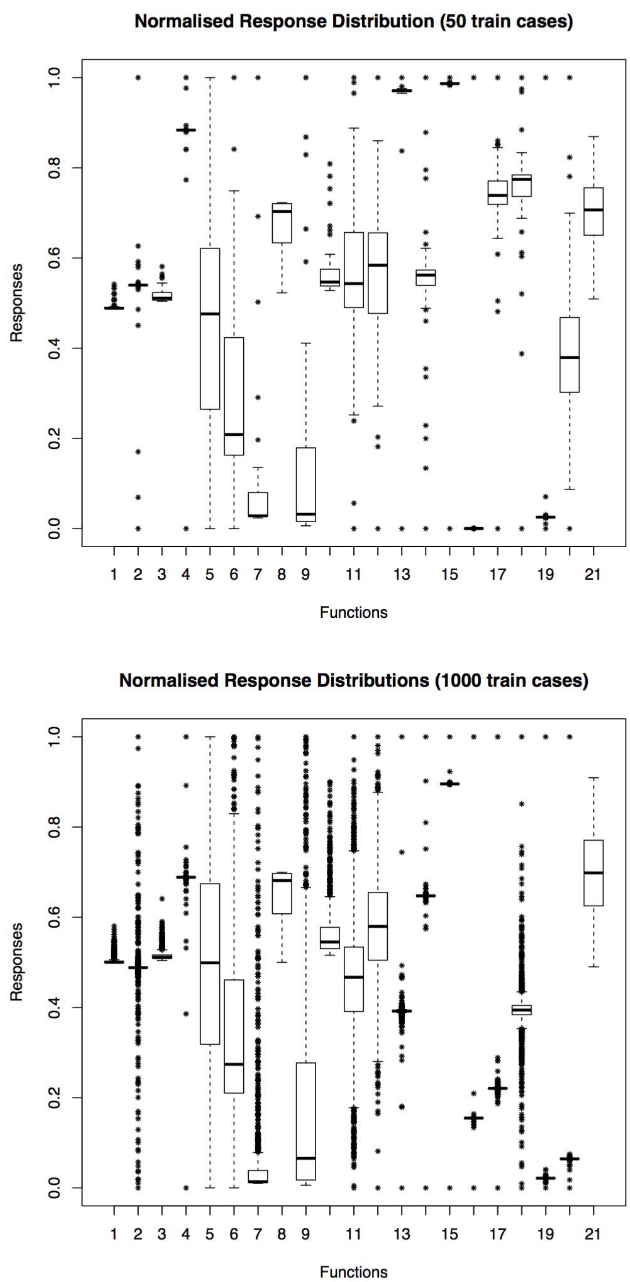

Fig. 2. Normalised response variable distribution over 50 (top) and 1000 (bottom) samples, for each function.

To exemplify this, 1000 sets of 50 samples each were created for both $F5$ and $F16$. Fig. 3 shows boxplots of the standard deviations of all 1000 sets, for both functions. This figure highlights the range of standard deviations that different samples can have; while most samples drawn from $F5$ exhibit a similar range, samples for $F16$ show an extreme variance in their range.

Optimally large sample sizes for each function can reduce this effect, but can also be computationally infeasible. Therefore different algorithms should be compared using the exact same samples, both for training and for test performance measurement (data can be easily made available online).

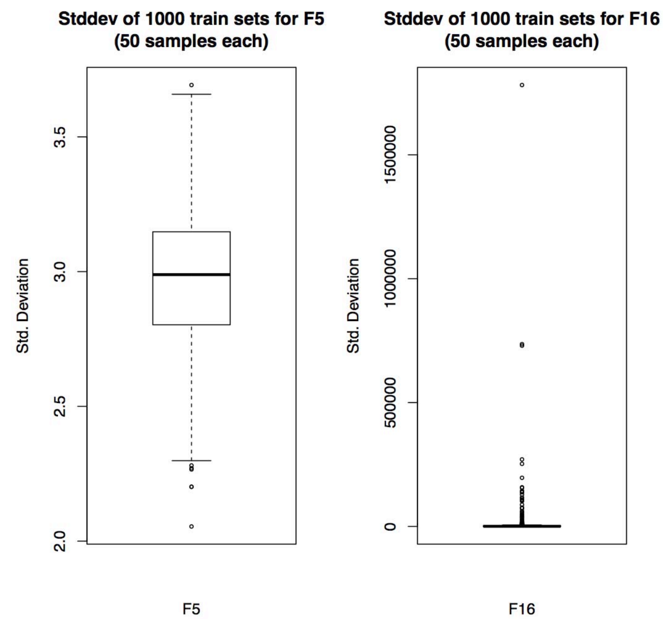

Fig. 3. Boxplots of distribution of standard deviation of response variable, for 1000 sets of 50 random samples each.

E. Large Test Set

This analysis of the variability of the response within small samples also raises questions about how to consistently measure the performance of an algorithm on unseen data. Based both on the range of the inputs and the variability of the response, a suitably large set of unseen samples should be used; in this study, a test set of 50000 unseen samples was used. As mentioned in the previous section, this sample should also always be the same, when comparing different algorithms.

F. Artificial Noise

The objective behind benchmarking a modelling algorithm is to simulate its application to real world data, which is inherently noisy. As such, benchmarks should provide datasets with noise added.

Given a synthetic function F(x), noisy datasets are generated according to $y_i = F(x_i) + \epsilon_i$. We propose the addition of normally-distributed errors. In this case, the error ϵ_i is generated from a normal distribution with zero mean and variance adjusted [11] so that

$$\mathbb{E}|\epsilon| = \mathbb{E}_x|F(x) - median_x F(x)| \qquad (2)$$

giving a 1/1 signal-to-noise ratio. For that, we set the standard deviation of our Gaussian noise σ as follows:

$$\sigma = \mathbb{E}_x|F(x) - median_x F(x)|/\sqrt{2.0/\pi} \qquad (3)$$

Other types of error can be used, such as slash-distributed errors [11]. In this case, $\epsilon_i = s \cdot (u/v)$, where $u \sim N(0,1)$ (normal distribution with zero mean and unit standard deviation) and $v \sim U[0,1]$ (uniform distribution in [0.0, 1.0]). The scale factor is adjusted to give a 1/1 signal-to-noise ratio. This is performed as follows. We first generate a number of N ratios $\{r_1, \ldots, r_N\}$, where $r_i = u/v$. We then calculate the standard deviation σ_r of the sample. Each ϵ_i is then set as:

$$\epsilon_i = r_i \cdot (\mathbb{E}_x|F(x) - median_x F(x)|/\sigma_r) \cdot \sqrt{\pi/2} \qquad (4)$$

The slash distribution has very thick tails and is often used as extreme to test robustness.

G. Baselines

It is common practice in Evolutionary Computation to compare results using a baseline system. In many publications, this is usually one of the earliest systems developed: a simple GA [12] when using Genetic Algorithms, or Koza's original system [3] when using Genetic Programming.

This however is not enough. GP is known to sometimes severely underperform, even in symbolic regression problems, and baselines such as a constant can sometimes outperform it [13]. Therefore a significant gain in precision when comparing to the original GA or GP systems is not necessarily evidence of good performance.

In this study, two baselines were used. One is a constant, which is just the average response observed in the training set; another is a linear regression model, applied to the training set. The results section shows how these two baselines can sometimes provide as good or better performance as GP.

III. Experiments

A. Benchmark setup

Each of the functions listed in Table I was used, with eight versions: 4 training set sizes (50, 100, 500, 1000), and their equivalent with added noise, as discussed previously.

B. Run Parameters

We applied standard GP and GE systems to all benchmarks, along with constant and linear regression baselines. The experimental setup for GP and GE was very similar; both are shown in Table II. These are typical setups as seen in literature, consisting of the four arithmetic operators and five unary functions. Sub-tree crossover was used with GP, along with sub-tree mutation (sub-trees of depth d are replaced with a random sub-tree of depth between 1 and d); with GE, linear variable 1-point crossover was used, along with integer mutation (with l being the length of the individual). The constants used with GP were $[-0.9..-0.1] \cup [0.1..0.9]$, in steps of 0.1 (18 constants total); in the case of GE, digit concatenation [14] was used, such as in the following grammar (for a three input function):

```
<e>   ::=   + <e> <e> | - <e> <e>
          | * <e> <e> | / <e> <e>
          | exp <e>   | ln <e>    | sqrt <e>
          | sin <e>   | tanh <e>
          | x\[0\]    | x\[1\]    | x\[2\]
          | <nums>.<nums>
<nums>::= <nums>0 | <nums>1
          | <nums>2 | <nums>3
          | <nums>4 | <nums>5
          | <nums>6 | <nums>7
          | <nums>8 | <nums>9
          | 0 | 1 | 2 | 3 | 4
          | 5 | 6 | 7 | 8 | 9
```

TABLE II. Experimental Setup

Parameter	GP	GE
Number of independent runs	100	
Total number of generations	50	
Population size	500	
Initialisation tree depth	5	
Maximum tree depth	10	unlimited
Tournament Size	4	5
Crossover rate	.9	.5
Mutation rate	.1	$1/l$
Number of elites	1	50
Function set	+,-,*,/,e^x,$ln(x)$,$\sqrt{x}$,$sin(x)$,$tanh(x)$	

The initial populations were initialised using well-known techniques: ramped half-and-half for GP [3], and sensible initialisation for GE [15]. No wrapping operator was used with GE.

Some operators used protected versions. Division returned 1 for x/y if $y < 1e-5$; natural logarithm returned x if $x \leq 0$; and squared root returned x if $x < 0$.

IV. Results and Analysis

A. Measuring test performance

GP systems tend to model input data quite well. This sometimes leads to overfitting of the training data. For this reason, training performance should never be reported as a means to measure system performance [2].

In order to avoid overfitting the training data, some approaches use various approaches, such as validation sets. These should be drawn from the training data provided for the benchmark, and never from the test data.

B. Expected performance

Measuring the test performance of stochastic algorithms requires specific measurements. The GP community has moved on from reporting the performance of a single run of the system, and nowadays the mean best performance of a minimum of 30 independent runs is usually reported. This is done in order to measure the expected performance of an algorithm, and to test its statistical significance with other approaches.

Many publications report the performance of the model achieved by the "best run", however (e.g. to compare its performance against other reported results). This is a mistake, as to choose a best run requires a comparison between all runs, making them no longer independent. If running GP to

[2] It can sometimes still be useful to report, however, to highlight the degree of overfitting (or lack of) in the system.

achieve a good model requires n runs, then a minimum of $n \times 30$ runs should be done, for statistical purposes.

For this reason, in this study, the median performance of all independent runs is reported, for GP and GE, in the (previously unseen) test set. This performance is then compared against the constant and linear regression models (which, being deterministic, require only one run). As a measure of variability of test performance between runs, the inter-quartile range (iqr) of the set of 30 runs is also reported, for GP and GE.

C. Results

Tables III to V show the results obtained. For each benchmark setup, the performance of GP, GE, a constant and Linear Regression are reported. The performance of GP and GE is reported in several ways: the median test performance of the models from all 100 runs, along with the iqr; the percentage of such models returning an infinity value in at least one test case; the median number of descents per run (i.e. number of times the training performance of the best model improved), along with the iqr; and the amount of training error decrease by the best model of a run, when compared to its initial population.

These results correlate quite well to the difficulty of the benchmarks, as seen in Fig. 1. Functions $F1$, $F3$, and $F13$ to $F20$ are extremely hard to model, with the given input range and training set sizes, and the test performance of all approaches varies from $1.0E4$ all the way to over $2.0E13$.

For the other functions, the GP approach generally provides a better performing median model when compared to the other methods, along with a smaller iqr than GE.

An interesting exception is $F21$ (also known as korns-12 [9]), which although well behaved in the response range, proved to be a very hard function to model. The best performance was achieved by both a constant and GE; an analysis of the GE model revealed that a constant had been evolved. Another challenging function was $F3$, for which GP and GE achieved a relatively low test error, but a linear regression model was a better choice [3].

Linear regression and constants are in fact occasionally as good or better than the evolutionary models. Fig. 4 plots the performance improvement of the median GP model over the constant predictor, versus twice the standard deviation of the response variable in the test set, for all 21 functions. There is a clear relationship between variance of response variable and GP performance improvement; a notable exception is $F2$, a notoriously hard to model function.

In terms of training sample size, a larger sample almost always leads to better test performance of the median model. Training with noisy data also tends to produce less precise models, as expected. Notable exceptions are the performance of GP in $F1$, $F5$ and $F8$, where training with noisy data and a large sample size produces better performing median models; a probable explanation is that noise helps in preventing overfitting the training data.

[3]Note that both GP and GE achieved better performance than linear regression for this problem in several runs, but not in their median run.

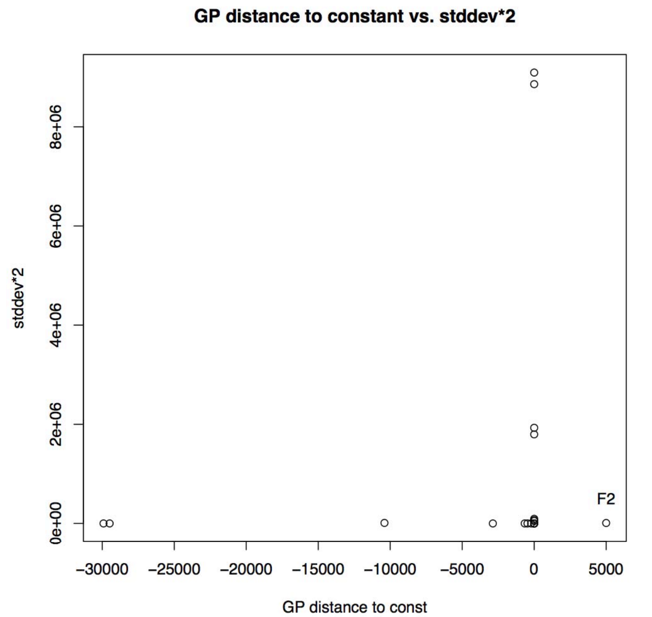

Fig. 4. GP performance difference to constant predictor versus double standard deviation of response variable in test set, for all 21 functions.

V. Conclusions & Future Work

This paper presented an experimental analysis of the difficulties in designing good benchmarks for Genetic Programming and similar systems. Several points were identified as crucial in designing artificial datasets from known functions, in the domain of symbolic regression. Crucially, the “smoothness” of synthetic regression problems is seldom studied in GP benchmarks, and the results obtained highlight this, with some results worse than those obtained by simple constant or linear scaling models. We do not imply that only smooth functions should be used, but if a synthetic problem is known to be highly non-smooth, it should be avoided.

This study is preliminary work, highlighting many of the issues in designing regression benchmarks, adding to those already raised previously [1]. Future work will continue this effort, addressing the raised issues and designing benchmark datasets, which will be available to the community. The issue of which metrics to report in the context of benchmarking will also be further investigated.

References

[1] J. McDermott, D. R. White, S. Luke, L. Manzoni, M. Castelli, L. Vanneschi, W. Jaskowski, K. Krawiec, R. Harper, K. D. Jong, and U.-M. O'Reilly, “Genetic programming needs better benchmarks,” in *Genetic and Evolutionary Computation - GECCO 2012, Genetic and Evolutionary Computation Conference, Philadelphia, USA, July 7-11, 2012, Proceedings*, T. S. et al., Ed. ACM, 2012, pp. 791–798.

[2] D. R. White, J. McDermott, M. Castelli, L. Manzoni, B. W. Goldman, G. Kronberger, W. Jakowski, U.-M. OReilly, and S. Luke, “Better gp benchmarks: community survey results and proposals,” *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 14, no. 1, pp. 3–29, 2013.

[3] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: MIT Press, 1992.

TABLE III. RESULTS - PART 1

	GENETIC PROGRAMMING				GRAMMATICAL EVOLUTION				CONST.	L.R
Data setup	**Test MSE**	∞	**Descents per run**	**Decrease per run**	**Test MSE**	∞	**Descents per run**	**Decrease per run**		
Function 1										
T1000	0.006 (0.003)	1%	20 (7)	59.18% (15.81)	0.009 (0.004)	0%	23 (6)	36.16% (27.05)	0.013	0.012
T500	0.006 (0.003)	1%	20 (5)	59.73% (16.42)	0.009 (0.005)	4%	24 (7)	33.41% (34.48)	0.013	0.012
T100	0.007 (0.030)	1%	19 (6)	69.02% (13.73)	0.009 (0.005)	4%	24 (7)	45.18% (33.10)	0.013	0.012
T50	0.023 (19.679)	2%	18 (5)	80.79% (12.29)	0.012 (0.007)	1%	24 (8)	45.86% (32.88)	0.013	0.012
T1000-N	0.005 (0.002)	0%	20 (6)	51.94% (15.48)	0.009 (0.005)	0%	23 (7)	28.53% (30.55)	0.013	0.012
T500-N	0.006 (0.003)	1%	18 (6)	49.53% (14.74)	0.009 (0.004)	1%	23 (7)	32.37% (25.99)	0.013	0.012
T100-N	0.007 (0.008)	2%	17 (6)	61.24% (11.28)	0.009 (0.006)	1%	24 (6)	41.60% (28.06)	0.013	0.012
T50-N	0.016 (5.124)	3%	18 (6)	68.93% (12.60)	0.013 (0.008)	0%	24 (7)	37.27% (27.88)	0.013	0.012
Function 2										
T1000	2.1256E7 (1.02E4)	0%	17 (7)	7.87% (2.97)	2.1250E7 (2.85E3)	0%	19 (6)	3.70% (4.96)	2.13E7	2.13E7
T500	2.1267E7 (5.25E4)	4%	19 (7)	18.67% (6.82)	2.1262E7 (3.03E4)	0%	21 (6)	17.11% (7.50)	2.13E7	2.13E7
T100	2.1594E7 (4.80E5)	2%	19 (5)	67.78% (6.88)	2.1598E7 (4.90E5)	0%	19 (6)	66.20% (3.30)	2.12E7	2.13E7
T50	2.1726E7 (3.47E5)	1%	17 (7)	83.99% (7.94)	2.1721E7 (2.17E5)	0%	21 (5)	81.85% (4.41)	2.13E7	2.13E7
T1000-N	2.1257E7 (1.01E4)	0%	19 (6)	7.26% (2.56)	2.1251E7 (7.68E3)	0%	20 (5)	6.59% (6.44)	2.13E7	2.13E7
T500-N	2.1267E7 (6.49E4)	1%	19 (6)	14.29% (6.26)	2.1264E7 (2.96E4)	0%	23 (5)	13.89% (6.66)	2.13E7	2.13E7
T100-N	2.1594E7 (7.26E4)	1%	21 (6)	49.40% (5.32)	2.1594E7 (1.43E4)	0%	20 (6)	46.75% (7.36)	2.12E7	2.13E7
T50-N	2.1598E7 (3.15E4)	2%	18 (8)	66.18% (5.85)	2.1598E7 (3.51E4)	0%	14 (10)	62.35% (5.67)	2.12E7	2.13E7
Function 3										
T1000	0.011 (0.004)	0%	16 (8)	44.91% (23.16)	0.013 (0.006)	0%	22 (6)	25.83% (34.88)	0.016	0.008
T500	0.012 (0.003)	0%	17 (7)	42.97% (30.90)	0.015 (0.005)	0%	20 (11)	18.53% (38.32)	0.016	0.008
T100	0.012 (0.006)	2%	17 (6)	50.75% (23.04)	0.016 (0.017)	0%	21 (8)	42.31% (38.19)	0.016	0.009
T50	0.014 (0.003)	1%	14 (7)	56.52% (36.79)	0.016 (0.003)	0%	19 (10)	25.68% (40.76)	0.016	0.009
T1000-N	0.012 (0.004)	0%	16 (7)	32.89% (25.04)	0.013 (0.006)	0%	21 (7)	18.61% (27.40)	0.016	0.008
T500-N	0.012 (0.003)	1%	16 (8)	30.31% (24.97)	0.014 (0.005)	0%	21 (8)	15.71% (25.18)	0.016	0.008
T100-N	0.013 (0.005)	2%	17 (8)	39.81% (23.02)	0.016 (0.002)	2%	23 (8)	26.05% (28.18)	0.016	0.009
T50-N	0.016 (0.002)	4%	12 (7)	34.54% (23.97)	0.016 (0.001)	0%	21 (9)	15.84% (21.25)	0.016	0.010
Function 4										
T1000	2.0679E13 (7.87E8)	2%	21 (9)	26.92% (9.18)	2.0679E13 (1.83E5)	0%	18 (8)	24.51% (8.29)	2.07E13	2.07E13
T500	2.0679E13 (8.91E8)	3%	21 (7)	36.97% (17.06)	2.0679E13 (2.04E5)	0%	19 (7)	31.23% (8.26)	2.07E13	2.07E13
T100	2.0679E13 (3.84E8)	4%	21 (6)	49.97% (17.30)	2.0679E13 (4.43E5)	3%	19 (7)	31.66% (12.95)	2.07E13	2.07E13
T50	2.0679E13 (5.22E11)	7%	22 (7)	74.27% (12.50)	2.0679E13 (8.30E7)	5%	18 (7)	61.38% (16.46)	2.07E13	2.07E13
T1000-N	2.0679E13 (1.17E8)	4%	22 (6)	19.76% (8.76)	2.0679E13 (2.64E5)	0%	19 (7)	17.70% (6.21)	2.07E13	2.07E13
T500-N	2.0679E13 (5.70E8)	2%	20 (8)	27.23% (14.16)	2.0679E13 (6.70E8)	0%	18 (7)	26.19% (9.01)	2.07E13	2.07E13
T100-N	2.0679E13 (3.11E8)	4%	22 (6)	36.45% (12.11)	2.0679E13 (2.40E6)	0%	20 (7)	27.06% (9.60)	2.07E13	2.07E13
T50-N	2.0679E13 (2.31E12)	8%	23 (9)	54.94% (8.76)	2.0679E13 (1.42E9)	8%	18 (7)	39.70% (11.53)	2.07E13	2.07E13
Function 5										
T1000	0.520 (3.053)	0%	15 (6)	92.51% (39.82)	3.490 (5.966)	0%	16 (7)	49.20% (71.67)	9.025	9.053
T500	0.544 (2.475)	0%	16 (6)	92.97% (34.73)	4.422 (5.317)	0%	18 (6)	48.67% (66.43)	9.026	9.094
T100	0.493 (2.382)	0%	17 (6)	94.04% (33.58)	3.559 (5.819)	0%	18 (5)	53.34% (73.19)	9.025	9.358
T50	0.284 (1.307)	0%	21 (5)	95.60% (14.95)	1.833 (5.766)	0%	19 (7)	74.83% (55.83)	9.202	9.859
T1000-N	0.386 (2.710)	0%	17 (6)	41.53% (15.81)	4.512 (6.090)	0%	14 (10)	19.88% (28.61)	9.026	9.021
T500-N	0.489 (2.046)	0%	17 (6)	43.45% (12.28)	5.719 (6.002)	0%	13 (13)	16.11% (32.07)	9.030	9.033
T100-N	2.496 (4.051)	2%	20 (6)	42.16% (14.54)	5.046 (5.185)	0%	20 (6)	25.46% (36.01)	9.027	9.206
T50-N	3.770 (5.562)	4%	19 (7)	43.79% (16.17)	6.457 (4.071)	0%	20 (7)	29.90% (34.81)	9.035	9.545
Function 6										
T1000	8.122 (8.409)	0%	22 (7)	95.08% (5.46)	13.047 (19.963)	0%	23 (6)	92.43% (11.59)	2.19E2	71.170
T500	7.050 (11.760)	0%	22 (6)	96.16% (6.48)	13.528 (19.140)	0%	23 (6)	92.99% (10.18)	2.19E2	70.994
T100	7.843 (14.573)	2%	22 (6)	95.75% (6.90)	12.903 (16.934)	0%	24 (6)	92.76% (9.77)	2.19E2	71.989
T50	15.600 (19.304)	1%	21 (7)	94.15% (6.26)	21.291 (21.139)	0%	23 (5)	91.76% (7.81)	2.19E2	70.971
T1000-N	9.738 (11.531)	1%	22 (6)	47.57% (5.29)	5.437 (15.704)	0%	24 (5)	47.60% (5.60)	2.21E2	73.018
T500-N	10.957 (10.868)	2%	22 (6)	50.54% (6.13)	13.837 (17.580)	0%	24 (5)	49.71% (5.47)	2.20E2	73.367
T100-N	21.978 (21.309)	2%	23 (8)	52.51% (8.73)	16.865 (19.174)	0%	23 (5)	48.17% (6.00)	2.19E2	73.130
T50-N	3226.268 (532.102)	8%	24 (7)	58.09% (7.06)	37.411 (1.57E2)	0%	24 (6)	54.24% (6.32)	2.19E2	82.268
Function 7										
T1000	1.220E3 (1.04E3)	1%	23 (8)	82.37% (10.79)	2.0080E3 (6.90E2)	0%	24 (4)	67.62% (17.09)	4.09E3	3.11E3
T500	1.249E3 (8.28E2)	0%	23 (7)	84.60% (8.68)	1.9719E3 (6.59E2)	0%	25 (4)	70.42% (12.09)	4.05E3	2.99E3
T100	2.204E3 (1.97E4)	0%	22 (8)	84.77% (9.67)	2.2378E3 (8.85E2)	0%	24 (6)	67.66% (15.87)	4.13E3	3.16E3
T50	2.811E4 (7.79E7)	7%	22 (8)	85.73% (9.09)	1.2862E4 (1.33E5)	1%	25 (6)	67.78% (19.13)	4.20E3	3.41E3
T1000-N	1.440E3 (9.12E2)	3%	23 (6)	59.21% (9.39)	1.9784E3 (6.15E2)	0%	24 (6)	49.59% (8.25)	4.08E3	3.10E3
T500-N	1.443E3 (7.73E2)	0%	22 (7)	64.43% (7.42)	1.8810E3 (4.81E2)	0%	24 (4)	55.49% (8.90)	4.04E3	2.98E3
T100-N	5.815E3 (3.47E4)	5%	22 (7)	61.04% (7.16)	2.1036E3 (9.53E2)	0%	23 (4)	47.15% (13.34)	4.16E3	3.19E3
T50-N	2.846E4 (1.22E6)	6%	21 (8)	64.13% (7.66)	3.1148E3 (2.89E4)	0%	24 (5)	48.20% (17.15)	4.15E3	3.24E3
Function 8										
T1000	0.076 (0.049)	0%	14 (5)	71.89% (20.24)	0.099 (0.188)	0%	20 (7)	52.35% (81.37)	0.232	0.233
T500	0.069 (0.058)	0%	14 (7)	76.07% (22.86)	0.127 (0.188)	0%	18 (8)	48.63% (78.83)	0.232	0.234
T100	0.078 (0.087)	0%	15 (5)	78.28% (23.34)	0.139 (0.190)	0%	19 (8)	59.05% (81.38)	0.232	0.234
T50	0.063 (0.110)	1%	13 (6)	83.56% (21.60)	0.177 (0.117)	0%	19 (9)	44.93% (65.34)	0.255	0.255
T1000-N	0.049 (0.072)	0%	14 (4)	40.98% (11.76)	0.130 (0.158)	0%	18 (9)	25.87% (32.62)	0.233	0.233
T500-N	0.080 (0.063)	0%	14 (6)	34.77% (11.78)	0.137 (0.172)	0%	19 (8)	24.41% (32.49)	0.233	0.234
T100-N	0.103 (0.097)	2%	14 (7)	36.88% (11.62)	0.159 (0.132)	0%	18 (8)	23.76% (32.95)	0.236	0.239
T50-N	0.204 (0.121)	2%	14 (6)	37.88% (6.41)	0.263 (0.120)	0%	17 (7)	26.53% (32.14)	0.297	0.300

TABLE IV. RESULTS - PART 2

	GENETIC PROGRAMMING				GRAMMATICAL EVOLUTION				CONST.	L.R
Data setup	**Test MSE**	∞	**Descents per run**	**Decrease per run**	**Test MSE**	∞	**Descents per run**	**Decrease per run**		
Function 9										
T1000	3.0582E2 (3.54E2)	1%	24 (7)	99.30% (0.86)	4.2984E2 (4.34E3)	0%	21 (6)	98.47% (14.05)	2.98E4	2.80E4
T500	2.7631E2 (5.33E2)	1%	25 (7)	99.18% (0.98)	2.3656E2 (2.09E3)	0%	21 (8)	98.74% (7.14)	2.98E4	2.81E4
T100	4.2077E2 (9.76E2)	4%	23 (7)	99.16% (1.29)	6.6300E2 (3.13E3)	0%	21 (9)	97.92% (8.54)	2.99E4	2.87E4
T50	6.9419E2 (8.67E3)	2%	25 (8)	99.33% (0.84)	1.0495E3 (9.06E3)	0%	22 (8)	97.84% (15.64)	2.98E4	3.09E4
T1000-N	5.1948E2 (2.09E3)	2%	24 (7)	62.18% (6.58)	1.6804E3 (6.90E3)	0%	21 (7)	56.31% (11.50)	2.98E4	2.80E4
T500-N	7.7476E2 (1.11E4)	0%	23 (8)	58.76% (6.27)	1.1827E3 (7.63E3)	0%	22 (7)	53.17% (10.97)	2.98E4	2.83E4
T100-N	6.3628E4 (1.63E6)	7%	24 (6)	61.31% (9.00)	3.0825E3 (4.23E4)	1%	22 (7)	55.29% (9.25)	3.00E4	2.89E4
T50-N	6.7189E5 (1.62E7)	9%	26 (7)	61.83% (8.22)	6.0710E3 (3.59E4)	0%	22 (5)	53.48% (8.78)	2.98E4	2.91E4
Function 10										
T1000	0.279 (0.108)	0%	18 (5)	39.81% (27.14)	0.300 (0.035)	0%	20 (6)	31.56% (9.74)	0.447	0.447
T500	0.288 (0.076)	0%	17 (6)	35.70% (24.32)	0.301 (0.038)	0%	21 (7)	32.02% (12.25)	0.446	0.446
T100	2.372 (80.917)	2%	17 (7)	43.78% (34.05)	4.860 (65.325)	1%	19 (7)	35.59% (24.23)	0.449	0.454
T50	5.980 (95.050)	1%	15 (6)	53.14% (35.05)	8.567 (1.02E2)	0%	19 (6)	61.73% (24.36)	0.449	0.456
T1000-N	0.285 (0.078)	0%	17 (5)	24.76% (16.88)	0.297 (0.038)	0%	20 (6)	21.38% (5.28)	0.446	0.447
T500-N	0.290 (0.077)	1%	17 (5)	24.28% (13.98)	0.306 (0.033)	0%	20 (6)	19.58% (6.63)	0.446	0.447
T100-N	88.186 (312.320)	4%	16 (6)	27.91% (15.66)	89.678 (1.53E2)	3%	19 (5)	26.56% (13.14)	0.446	0.455
T50-N	130.474 (2372.416)	4%	16 (5)	50.26% (13.21)	1.0428E2 (4.42E2)	0%	20 (6)	48.03% (16.84)	0.470	0.477
Function 11										
T1000	31.436 (19.868)	0%	25 (5)	88.20% (9.71)	60.983 (65.461)	0%	20 (7)	72.97% (21.66)	4.97E2	1.05E2
T500	28.217 (22.092)	0%	24 (6)	87.72% (9.29)	48.948 (43.434)	0%	20 (6)	78.40% (19.12)	4.97E2	1.05E2
T100	37.296 (40.815)	1%	26 (6)	88.03% (9.93)	51.005 (59.904)	0%	22 (6)	79.67% (19.41)	4.99E2	1.07E2
T50	39.613 (287.120)	1%	24 (6)	89.70% (5.77)	54.051 (55.370)	0%	21 (6)	78.84% (15.13)	4.99E2	1.09E2
T1000-N	34.865 (20.293)	1%	25 (7)	36.93% (8.84)	64.616 (48.583)	0%	20 (6)	28.11% (17.10)	4.98E2	1.07E2
T500-N	36.887 (28.726)	0%	25 (6)	32.93% (11.00)	50.474 (35.974)	0%	22 (6)	27.46% (14.71)	4.97E2	1.07E2
T100-N	132.679 (646.587)	3%	24 (6)	44.73% (7.94)	81.858 (82.513)	0%	21 (7)	34.85% (19.36)	4.97E2	1.11E2
T50-N	2643.213 (2064.343)	1%	24 (6)	37.93% (6.94)	97.679 (1.04E2)	0%	20 (8)	31.70% (11.32)	4.97E2	1.68E2
Function 12										
T1000	6.753E2 (1.54E2)	1%	22 (7)	44.92% (12.37)	1.2088E3 (5.36E2)	0%	16 (8)	10.26% (33.59)	1.33E3	1.33E3
T500	6.965E2 (3.72E2)	1%	23 (7)	39.11% (16.78)	1.2647E3 (5.64E2)	0%	15 (6)	12.65% (26.55)	1.33E3	1.34E3
T100	4.470E3 (1.17E5)	4%	26 (7)	48.19% (21.46)	4.4063E3 (8.29E4)	0%	21 (10)	25.21% (15.45)	1.34E3	1.43E3
T50	1.317E4 (2.20E5)	6%	26 (5)	62.73% (10.38)	2.3144E4 (1.85E5)	0%	19 (8)	51.32% (13.55)	1.33E3	1.67E3
T1000-N	6.859E2 (3.25E2)	0%	22 (7)	23.36% (8.09)	7.5194E2 (5.75E2)	0%	16 (8)	18.22% (16.50)	1.33E3	1.33E3
T500-N	6.916E2 (5.71E2)	3%	23 (8)	19.91% (7.79)	7.8305E2 (6.21E2)	0%	16 (8)	14.86% (11.47)	1.33E3	1.34E3
T100-N	1.135E4 (3.53E5)	3%	27 (8)	32.95% (11.77)	9.7407E3 (1.24E5)	0%	19 (7)	21.57% (6.83)	1.33E3	1.47E3
T50-N	6.403E5 (3.32E6)	7%	28 (7)	52.09% (8.46)	1.0243E5 (1.93E6)	0%	22 (9)	40.79% (14.68)	1.37E3	1.65E3
Function 13										
T1000	1.962763E13 (2.58E7)	0%	22 (6)	34.38% (9.11)	1.9628E13 (2.70E3)	0%	21 (6)	12.04% (8.05)	1.96E13	1.96E13
T500	1.962761E13 (7.34E7)	4%	22 (6)	39.16% (10.25)	1.9628E13 (5.15E4)	0%	20 (7)	11.23% (7.60)	1.96E13	1.96E13
T100	1.962761E13 (2.55E9)	13%	23 (6)	56.42% (10.29)	1.9628E13 (2.12E7)	0%	21 (6)	25.26% (19.97)	1.96E13	1.96E13
T50	1.962757E13 (6.60E8)	9%	24 (6)	70.10% (20.52)	1.9628E13 (7.70E7)	5%	23 (5)	42.67% (22.47)	1.96E13	1.96E13
T1000-N	1.962764E13 (1.43E7)	6%	22 (6)	23.57% (5.82)	1.9628E13 (5.25E4)	0%	19 (10)	8.46% (6.47)	1.96E13	1.96E13
T500-N	1.962763E13 (3.79E7)	11%	21 (7)	26.14% (7.95)	1.9628E13 (9.86E4)	0%	21 (7)	9.47% (6.07)	1.96E13	1.96E13
T100-N	1.962762E13 (2.58E9)	15%	22 (4)	40.40% (9.19)	1.9628E13 (1.97E7)	1%	23 (5)	18.70% (15.27)	1.96E13	1.96E13
T50-N	1.962757E13 (5.18E10)	14%	24 (4)	58.82% (9.87)	1.9628E13 (9.14E7)	0%	23 (5)	39.53% (15.80)	1.96E13	1.96E13
Function 14										
T1000	1.163947E9 (2.54E4)	4%	16 (6)	3.74% (1.61)	1.1639E9 (7.49E4)	0%	17 (7)	2.84% (1.56)	1.16E9	1.16E9
T500	1.163954E9 (5.53E4)	8%	20 (6)	6.41% (2.24)	1.1640E9 (5.88E4)	0%	18 (6)	5.91% (1.94)	1.16E9	1.16E9
T100	1.164027E9 (8.45E5)	7%	23 (6)	27.94% (7.43)	1.1640E9 (7.96E5)	0%	19 (7)	21.35% (12.44)	1.16E9	1.16E9
T50	1.164175E9 (1.44E6)	3%	23 (6)	62.98% (7.29)	1.1639E9 (1.43E6)	4%	21 (8)	49.35% (11.82)	1.16E9	1.16E9
T1000-N	1.163953E9 (2.46E4)	5%	18 (6)	2.71% (1.23)	1.1639E9 (1.19E4)	0%	18 (7)	1.76% (1.27)	1.16E9	1.16E9
T500-N	1.163943E9 (4.89E4)	5%	18 (7)	4.63% (2.34)	1.1639E9 (4.93E4)	0%	18 (8)	4.03% (1.91)	1.16E9	1.16E9
T100-N	1.164657E9 (2.98E6)	3%	23 (6)	26.96% (6.79)	1.1640E9 (4.48E5)	0%	17 (7)	18.00% (9.75)	1.16E9	1.16E9
T50-N	1.164330E9 (1.82E6)	3%	24 (5)	57.96% (7.13)	1.1639E9 (1.43E6)	4%	21 (9)	49.43% (9.56)	1.16E9	1.16E9
Function 15										
T1000	2.216200E9 (6.39E2)	1%	19 (5)	10.30% (5.45)	2.2162E9 (3.79E2)	3%	24 (6)	9.38% (5.10)	2.22E9	2.22E9
T500	2.216200E9 (5.87E2)	2%	19 (6)	12.31% (4.87)	2.2162E9 (1.86E2)	2%	24 (5)	10.58% (3.49)	2.22E9	2.22E9
T100	2.216201E9 (6.61E3)	8%	17 (8)	33.06% (17.37)	2.2162E9 (2.56E3)	5%	22 (6)	19.61% (21.08)	2.22E9	2.22E9
T50	2.216220E9 (3.79E4)	18%	16 (7)	51.57% (12.74)	2.2162E9 (1.21E5)	9%	21 (5)	44.81% (24.38)	2.22E9	2.22E9
T1000-N	2.216200E9 (6.08E2)	0%	20 (7)	8.15% (3.20)	2.2162E9 (4.32E2)	0%	23 (6)	6.42% (3.23)	2.22E9	2.22E9
T500-N	2.216199E9 (5.73E2)	2%	20 (6)	8.84% (3.94)	2.2162E9 (3.52E2)	1%	22 (5)	7.94% (3.65)	2.22E9	2.22E9
T100-N	2.216202E9 (4.95E5)	16%	19 (7)	30.85% (8.04)	2.2162E9 (1.77E3)	3%	23 (6)	19.90% (11.89)	2.22E9	2.22E9
T50-N	2.216250E9 (1.20E5)	16%	18 (7)	42.54% (9.87)	2.2162E9 (1.43E4)	11%	20 (7)	38.95% (12.18)	2.22E9	2.22E9
Function 16										
T1000	9.308650E11 (1.37E4)	4%	15 (9)	69.77% (0.79)	9.3086E11 (0.000)	0%	20 (13)	0.28% (0.96)	9.31E11	9.31E11
T500	9.308650E11 (1.79E4)	2%	17 (8)	71.11% (1.49)	9.3086E11 (5.38E3)	0%	23 (7)	0.93% (3.23)	9.31E11	9.31E11
T100	9.308650E11 (4.15E4)	6%	17 (7)	86.09% (2.73)	9.3086E11 (1.41E3)	0%	18 (16)	5.15% (12.94)	9.31E11	9.31E11
T50	9.308650E11 (3.67E5)	15%	17 (9)	92.47% (1.87)	9.3086E11 (3.90E3)	1%	21 (7)	5.25% (21.15)	9.31E11	9.31E11
T1000-N	9.308650E11 (1.54E4)	1%	17 (7)	61.34% (1.51)	9.3086E11 (0.000)	0%	13 (13)	0.25% (1.06)	9.31E11	9.31E11
T500-N	9.308650E11 (2.79E4)	6%	16 (6)	62.71% (3.40)	9.3086E11 (1.67E4)	0%	20 (8)	0.70% (2.57)	9.31E11	9.31E11
T100-N	9.308650E11 (4.93E4)	8%	18 (7)	76.35% (3.48)	9.3086E11 (2.02E4)	0%	21 (10)	9.87% (13.71)	9.31E11	9.31E11
T50-N	9.308650E11 (1.34E5)	10%	18 (7)	79.69% (4.27)	9.3086E11 (3.65E4)	0%	22 (6)	18.47% (26.88)	9.31E11	9.31E11

TABLE V. RESULTS - PART 3

	GENETIC PROGRAMMING				GRAMMATICAL EVOLUTION				CONST.	L.R.
Data setup	**Test MSE**	∞	**Descents per run**	**Decrease per run**	**Test MSE**	∞	**Descents per run**	**Decrease per run**		
Function 17										
T1000	5.808386E8 (1.31E3)	4%	17 (8)	19.94% (2.62)	5.8084E8 (4.51E2)	0%	22 (9)	1.70% (2.66)	5.81E8	5.81E8
T500	5.808392E8 (7.10E3)	7%	17 (10)	23.06% (3.74)	5.8084E8 (9.55E2)	0%	22 (5)	1.20% (5.36)	5.81E8	5.81E8
T100	5.808426E8 (2.34E6)	20%	17 (9)	34.12% (7.74)	5.8084E8 (1.19E4)	2%	22 (8)	10.54% (14.31)	5.81E8	5.81E8
T50	5.809396E8 (1.80E8)	48%	20 (9)	55.71% (13.24)	5.8086E8 (5.94E5)	5%	25 (7)	32.20% (23.02)	5.81E8	5.81E8
T1000-N	5.808385E8 (1.30E3)	3%	16 (7)	15.24% (2.89)	5.8084E8 (8.20E2)	0%	22 (6)	1.95% (1.47)	5.81E8	5.81E8
T500-N	5.808393E8 (4.56E3)	5%	17 (8)	18.49% (2.86)	5.8084E8 (2.55E3)	0%	23 (7)	3.26% (3.97)	5.81E8	5.81E8
T100-N	5.808417E8 (1.26E5)	8%	19 (8)	27.82% (5.68)	5.8084E8 (6.11E4)	0%	22 (6)	9.30% (13.93)	5.81E8	5.81E8
T50-N	5.849402E8 (7.45E5)	25%	22 (7)	45.35% (13.08)	5.8093E8 (1.37E6)	6%	25 (6)	28.39% (21.57)	5.81E8	5.81E8
Function 18										
T1000	1.086989E4 (3.69E4)	1%	24 (8)	78.80% (89.13)	4.1258E4 (4.62E4)	0%	20 (8)	3.99% (82.03)	4.08E4	4.08E4
T500	2.459615E4 (5.93E4)	5%	26 (9)	56.89% (86.52)	4.2307E4 (3.49E4)	0%	20 (7)	5.40% (4.15)	4.08E4	4.10E4
T100	6.713410E4 (4.55E5)	3%	26 (8)	43.30% (49.03)	4.8814E4 (1.65E5)	1%	18 (8)	27.90% (15.63)	4.08E4	4.25E4
T50	5.094047E4 (1.62E6)	3%	24 (8)	55.28% (48.85)	4.9665E4 (3.76E5)	0%	20 (8)	42.94% (23.00)	4.09E4	4.13E4
T1000-N	1.373869E4 (4.01E4)	1%	25 (7)	46.65% (56.20)	4.1610E4 (5.55E4)	0%	18 (9)	3.28% (62.06)	4.08E4	4.08E4
T500-N	1.225567E4 (4.08E4)	2%	27 (8)	57.70% (50.54)	4.3329E4 (6.26E4)	0%	23 (6)	4.69% (2.52)	4.09E4	4.11E4
T100-N	6.779343E4 (4.16E5)	4%	26 (8)	32.99% (13.18)	5.1310E4 (8.09E4)	0%	19 (7)	23.76% (10.07)	4.08E4	4.30E4
T50-N	2.539613E5 (5.84E6)	7%	24 (8)	43.21% (12.03)	4.7245E4 (3.21E5)	1%	19 (7)	33.42% (17.93)	4.08E4	4.21E4
Function 19										
T1000	8.072103E11 (3.25E5)	5%	22 (7)	4.46% (2.16)	8.0721E11 (1.62E5)	0%	24 (6)	3.04% (2.26)	8.07E11	8.07E11
T500	8.072103E11 (4.49E5)	5%	23 (7)	6.97% (1.85)	8.0721E11 (2.66E5)	0%	21 (7)	5.27% (2.06)	8.07E11	8.07E11
T100	8.072270E11 (2.43E8)	8%	23 (7)	23.64% (7.40)	8.0721E11 (1.24E8)	2%	19 (7)	15.75% (9.42)	8.07E11	8.07E11
T50	8.072174E11 (1.34E8)	11%	23 (6)	54.89% (9.31)	8.0721E11 (8.59E6)	4%	20 (8)	37.11% (22.43)	8.07E11	8.07E11
T1000-N	8.072102E11 (2.15E5)	4%	23 (6)	4.24% (1.64)	8.0721E11 (1.82E5)	0%	24 (5)	2.22% (1.45)	8.07E11	8.07E11
T500-N	8.072102E11 (3.72E5)	3%	23 (7)	5.48% (1.53)	8.0721E11 (1.36E5)	0%	22 (7)	3.69% (1.51)	8.07E11	8.07E11
T100-N	8.072162E11 (2.31E9)	9%	26 (7)	20.36% (5.84)	8.0721E11 (1.25E7)	7%	21 (10)	10.80% (7.64)	8.07E11	8.07E11
T50-N	8.072163E11 (2.71E7)	9%	25 (7)	44.05% (9.32)	8.0721E11 (6.99E6)	3%	20 (7)	26.73% (16.54)	8.07E11	8.07E11
Function 20										
T1000	2.680063E7 (3.13E4)	5%	19 (11)	16.85% (20.19)	2.6810E7 (8.30E3)	0%	19 (13)	17.31% (12.98)	2.68E7	2.68E7
T500	2.680764E7 (5.65E3)	2%	18 (9)	23.91% (13.47)	2.6807E7 (3.11E3)	0%	18 (7)	16.70% (10.12)	2.68E7	2.68E7
T100	2.681777E7 (1.93E5)	7%	19 (8)	28.78% (12.93)	2.6814E7 (2.50E4)	0%	17 (8)	29.41% (13.02)	2.68E7	2.68E7
T50	2.682065E7 (8.78E4)	6%	19 (10)	47.87% (16.37)	2.6813E7 (1.08E5)	0%	14 (8)	44.38% (16.41)	2.68E7	2.68E7
T1000-N	2.680719E7 (2.66E4)	5%	19 (10)	17.58% (13.21)	2.6807E7 (3.81E3)	0%	20 (10)	10.96% (7.64)	2.68E7	2.68E7
T500-N	2.680746E7 (1.73E3)	2%	18 (8)	18.90% (7.76)	2.6807E7 (1.47E3)	0%	19 (7)	13.13% (7.69)	2.68E7	2.68E7
T100-N	2.682727E7 (8.35E5)	8%	19 (7)	26.73% (13.07)	2.6815E7 (4.69E4)	0%	17 (6)	26.83% (12.05)	2.68E7	2.68E7
T50-N	2.683119E7 (7.36E4)	7%	19 (5)	40.66% (18.47)	2.6843E7 (1.32E5)	0%	14 (7)	39.72% (23.96)	2.68E7	2.68E7
Function 21										
T1000	1.074 (0.007)	0%	10 (5)	5.45% (13.34)	1.064 (0.000)	0%	14 (12)	0.06% (0.05)	1.064	1.071
T500	1.088 (0.024)	0%	11 (5)	4.63% (7.78)	1.066 (0.000)	0%	21 (7)	0.14% (0.34)	1.066	1.085
T100	1.204 (0.082)	1%	14 (5)	22.44% (10.15)	1.065 (0.000)	0%	15 (13)	0.08% (0.45)	1.065	1.099
T50	1.364 (0.155)	1%	16 (5)	39.92% (11.74)	1.110 (0.404)	4%	16 (10)	7.50% (22.33)	1.071	1.273
T1000-N	1.078 (0.012)	1%	11 (6)	2.86% (4.10)	1.064 (0.000)	0%	18 (12)	0.02% (0.03)	1.064	1.074
T500-N	1.111 (0.036)	1%	11 (6)	4.43% (4.12)	1.067 (0.000)	0%	20 (10)	0.03% (0.10)	1.067	1.084
T100-N	1.305 (0.192)	4%	14 (5)	17.64% (6.89)	1.064 (0.054)	0%	14 (12)	0.02% (3.89)	1.064	1.080
T50-N	1.509 (0.317)	8%	14 (5)	29.66% (5.82)	1.064 (0.513)	2%	20 (7)	0.36% (18.24)	1.064	1.320

[4] M. O'Neill and C. Ryan, *Grammatical Evolution - Evolutionary Automatic Programming in an Arbitrary Language*, ser. Genetic Programming. Kluwer Academic, 2003, vol. 4.

[5] 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*, vol. 13, no. 2, pp. 333–349, 2009.

[6] L. Pagie and P. Hogeweg, "Evolutionary consequences of coevolving targets," *Evolutionary Computation*, vol. 5, no. 4, pp. 401–418, 1997.

[7] M. Keijzer, "Improving symbolic regression with interval arithmetic and linear scaling," in *Genetic Programming, 6th European Conference, EuroGP 2003, Essex, UK, April 14-16, 2003, Proceedings*, ser. Lecture Notes in Computer Science, C. Ryan, T. Soule, M. Keijzer, E. Tsang, R. Poli, and E. Costa, Eds., vol. 2610. Springer, 2003, pp. 70–82.

[8] R. Poli, "A simple but theoretically-motivated method to control bloat in genetic programming," in *Genetic Programming, 6th European Conference, EuroGP 2003, Essex, UK, April 14-16, 2003, Proceedings*, ser. Lecture Notes in Computer Science, C. Ryan, T. Soule, M. Keijzer, E. Tsang, R. Poli, and E. Costa, Eds., vol. 2610. Springer, 2003, pp. 204–217.

[9] M. F. Korns, "Accuracy in symbolic regression," in *Genetic Programming Theory and Practice IX*, ser. Genetic and Evolutionary Computation, R. Riolo, E. Vladislavleva, and J. H. Moore, Eds. New York: Springer, 2011, pp. 129–151.

[10] M. D. McKay, "Latin hypercube sampling as a tool in uncertainty analysis of computer models," in *Winter Simulation, 24th Conference, WSC 1992, Arlington, VA, USA, Proceedings*, J. J. Swain, D. Goldsman, R. C. Crain, and J. R. Wilson, Eds. ACM, 1992, pp. 557–564.

[11] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2000.

[12] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, 1989.

[13] J. McDermott. (2014, March) Genetic programming needs better baselines. [Online]. Available: http://jmmcd.net/2013/12/19/gp-needs-better-baselines.html

[14] M. Nicolau and I. Dempsey, "Introducing grammar based extensions for grammatical evolution," in *IEEE Congress on Evolutionary Computation, CEC 2006, Vancouver, BC, Canada, July 16-21, 2006, Proceedings*. IEEE Press, 2006, pp. 2663–2670.

[15] C. Ryan and A. Azad, "Sensible initialisation in grammatical evolution," in *Genetic and Evolutionary Computation - GECCO 2003, Genetic and Evolutionary Computation Conference, Chicago, IL, USA, July 12-16, 2004, Workshops, Proceedings*, E. Cantú-Paz, J. A. Foster, K. Deb, L. Davis, R. Roy, U.-M. O'Reilly, H.-G. Beyer, R. K. Standish, G. Kendall, S. W. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. A. Potter, A. C. Schultz, K. A. Dowsland, N. Jonoska, and J. F. Miller, Eds. AAAI, 2003.