Time Control or Size Control? Reducing Complexity and Improving Accuracy of Genetic Programming Models

Aliyu Sani Sambo[1 [0000−0001−7069−1933]], R. Muhammad Atif Azad[1 [0000−0002−4013−5415]], Yevgeniya Kovalchuk[1 [0000−0003−4306−4680]], Vivek Padmanaabhan Indramohan[2], and Hanifa Shah[3]

[1] School of Computing and Digital Technology, Birmingham City University, UK
aliyu.sambo@mail.bcu.ac.uk, {atif.azad, yevgeniya.kovalchuk}@bcu.ac.uk
[2] School of Health Science, Birmingham City University, UK
vivek.indramohan@bcu.ac.uk
[3] Faculty of Computing, Engineering and the Built Environment, Birmingham City University, UK hanifa.shah@bcu.ac.uk

Abstract. Complexity of evolving models in genetic programming (GP) can impact both the quality of the models and the evolutionary search. While previous studies have proposed several notions of GP model complexity, the size of a GP model is by far the most researched measure of model complexity. However, previous studies have also shown that controlling the size does not automatically improve the accuracy of GP models, especially the accuracy on out of sample (test) data. Furthermore, size does not represent the functional composition of a model, which is often related to its accuracy on test data. In this study, we explore the *evaluation time* of GP models as a measure of their complexity; we define the evaluation time as the time taken to evaluate a model over some data. We demonstrate that the evaluation time reflects both a model's size and its composition; also, we show how to measure the evaluation time reliably. To validate our proposal, we leverage four well-known methods to size-control but to control evaluation times instead of the tree sizes; we thus compare size-control with *time-control*. The results show that time-control with a nuanced notion of complexity produces more accurate models on 17 out of 20 problem scenarios. Even when the models have slightly greater times and sizes, time-control counterbalances via superior accuracy on both training and test data. The paper also argues that time-control can differentiate functional complexity even better in an identically-sized population. To facilitate this, the paper proposes Fixed Length Initialisation (FLI) that creates an identically-sized but functionally-diverse population. The results show that while FLI particularly suits time-control, it also generally improves the performance of size-control. Overall, the paper poses evaluation-time as a viable alternative to tree sizes to measure complexity in GP.

Keywords: genetic programming · complexity · evaluation time

1 Introduction

Motivations for controlling the complexity of machine learning (ML) models vary and so does the notion of complexity [3]. One reason for managing the complexity of ML models is to attain models that are only complex enough to explain the phenomenon generating the given data but not too complex to reflect noise in the data. Doing so means that the predictions produced by the models on previously unseen data are accurate [18]; in other words, the model *generalises* well. However, the challenge in this goal is determining when the complexity is just enough. Another incentive for managing complexity is the requirement for models to use computational resources efficiently. For example, some computational environments such as the *Internet of Things* (IoT) devices constrain the evaluation time of an acceptable model even if this compromises its accuracy [13]. In Genetic Programming (GP), preventing the models from growing too complex is also necessary to prevent the evolutionary search from becoming ineffective [11]. A further motivation for managing complexity is the demand for interpretable models: simple models can be more interpretable [14], and the interpretability of ML models is now important. For example, the EU General Data Protection Regulation (GDPR) stipulates a *right to explanation* where ML algorithms are applied to make a decision affecting a person.

The challenge of defining a notion of complexity is compounded in the context of Genetic programming (GP). For example, while ridge regression penalises the growth in the magnitude of numeric coefficients in an otherwise fixed regression model, this penalty does not necessarily work in GP because GP evolves the model itself. Moreover, GP is a versatile tool that can also evolve compilable programs; therefore, minimising the coefficients does not automatically make sense. Also, since during evolution the GP models grow in size, controlling this growth (bloat control) has dominated the landscape of complexity control in GP. However, some previous work [21] shows that controlling the size alone does not automatically produce models that generalise as might have been expected. Moreover, [2] shows that size does not indicate functional composition (or complexity): after all, a very large GP tree may compose a simple linear function; likewise, a small GP tree can compose a highly non-linear function. Together the above challenges show that universally defining complexity is difficult.

This paper uses the *evaluation time* – the computational time required to evaluate a model on the given data – to indicate its complexity. Due to different functional and syntactic compositions of models in the evolving populations, the evaluation time of the models varies. For example, the models made up of computationally expensive functions or exceptionally large syntactic structures take long to evaluate. Unlike size, the evaluation time thus indicates both the syntactic and functional complexity; section 2.2 expands further on that. However, since evaluation times vary from one measurement to another, section 2.3 shows how to measure them reliably.

To control evaluation times, we use four well-known techniques for bloat-control to control evaluation time. However, instead of controlling size, we control evaluation times using the same mechanisms; the techniques thus effect *time-*

control. We then compare the effect of time-control with that of size (or bloat) control on the composition, size and accuracy of the evolving models.

The results of our experiments suggest that time-control with a nuanced notion of complexity outperforms size-control in model-accuracy on 17 out of 20 problem scenarios. Even when time-control produces models with slightly greater times and sizes, it counterbalances via superior accuracy on both training and test data. The paper also shows that time-control can differentiate functional complexity even better in an identically-sized population. To facilitate this, the paper proposes Fixed Length Initialisation (FLI) that creates an identically-sized but functionally-diverse population. The results show that while FLI particularly suits time-control, it also generally improves the performance of size-control.

Following this introductory part, section 2 of this paper provides some background; section 3 details the experiments; section 4 presents the results; and section 5 covers future works and concludes the paper.

2 Background

2.1 Complexity in Genetic Programming

Traditionally, controlling complexity in GP means controlling structural complexity such as the size (bloat control) of the evolved expressions, or the number of encapsulated sub-trees and layers, while ignoring the underlying functional or computational complexity [21, 7, 9, 6, 17]. For example, bloat control penalises a large yet linear expression $4x+8x+2x+x+x$, which is functionally and computationally less complex than a smaller expression $sin(x)$ [2], which is equivalent to its Taylor series expansion $\sum_{n=0}^{\infty}(-1)^n \frac{x^{2n+1}}{(2n+1)!}$. Clearly, the smaller expression $sin(x)$ needs more computational resources than its linear counterpart. Thus, complexity in GP is more than merely the expression size.

Approaches based on functional complexity recognise that small structures may be more complex than larger ones and hence focus on the functionality of structures. To elicit functional complexity, one approach approximates the evolving expressions by polynomials [23]; complex expressions are approximated by polynomials of a high degree owing to large oscillations in the response of the function. This degree of approximating polynomials is thus minimised in [23]. However, the minimisation requires the evolving expressions to be twice differentiable, a property that is not always guaranteed. To alleviate this constraint, Vanneschi et al. [21] defined a less rigorous measure of functional complexity, whereby the slope of an expression is approximated by a simpler but error prone measure. As such Vanneschi et al. did not control the complexity; instead, they only measured the complexity of evolving expressions. Another approach [1] used the variance of the outputs of the evolving expressions to measure the functional complexity; this approach explicitly minimised the variance and maximised accuracy using a multi-objective optimisation approach. Note, however, that slope of the evolving functions can not indicate complexity when evolving compilable programs for tasks such as robot navigation.

Similarly, statistical learning theory measures the complexity of a space of functions that can be learned using statistical classification. The main techniques include generalisation error bound VC theory and VC dimensions [12] [22].

As the discussion highlights, the above techniques are either specialised to various domains or challenging to implement. In contrast, the present study simply measures the complexity of a model with its evaluation time.

2.2 Evaluating Time is More than Measuring Size

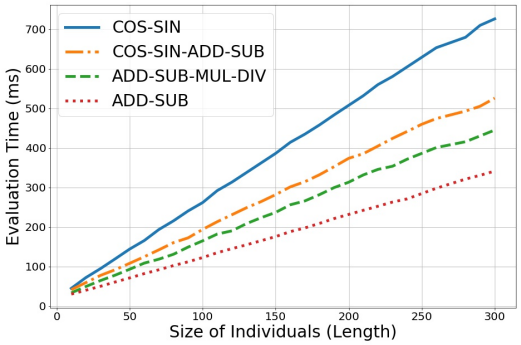


Fig. 1. Relationship between evaluation time, size and the composition of models is shown. Individuals made up of COS and SIN operators have higher average evaluation times than the same-sized individuals from other functions sets. Also, note that size correlates with evaluation time.

While the previous section argues why measuring size is fundamentally different to evaluating time, it is also important to empirically verify that. After all, the evaluation time also increases when the expression size increases; however, we must also ascertain if the evaluation time also practically increases with the functional complexity. Otherwise, measuring time becomes simply a proxy for measuring size. Clearly, that is undesirable.

To this end, we used four different *functions sets* to generate symbolic regression models of different complexities; Fig. 1 details the functions sets. For each functions set, we generated differently sized individuals (10, 20, 30, ..., 300), and in turn for each size we generated 30 random expressions. All the models were then evaluated 50 times, each with the same data. Fig. 1 presents the average evaluation times of individuals according to their size and complexity.

Two trends are clearly visible in Fig. 1: (1) given the same size, the evaluation times of functionally complex individuals are consistently higher than those for their counterparts; and (2) evaluation times are also strongly correlated with

the expression sizes, as expected. Hence, the evaluation times indeed differentiate between functional complexities; however, if a simple function is inefficiently coded as an excessively large expression, it evaluates slower. Therefore, evaluation time control impacts conditionally: it prefers functional simplicity if the sizes of a competing set of individuals are within a certain tolerance (or range); otherwise, it prefers smaller sizes. Note, this tolerance increases as the size of individuals increases. For example, the evaluation time of size 75 with functions set COS-SIN is the same as that for size 175 with the functions set ADD-SUB.

The above findings also predict the limiting behaviour of evaluation time control in GP. In a functionally diverse but a size-converged population – where bloat control is impotent – evaluation times discriminate between functional complexities, whereas in a functionally converged but a size-diverse population, evaluation times discriminate between sizes.

The idea that time control discriminates between functional complexities when sizes have converged prompted us to try a new initialisation scheme. The new scheme starts with a population of identically sized but functionally diverse individuals. We tested the impact of this new initialisation on all methods before applying it to our experiments. Section 3.4 details this scheme and its impact.

2.3 Stabilising Evaluation Time Measurements

A problem with measuring evaluation times is that they vary across multiple executions, and if this variability is high, one cannot reliably estimate the complexity of a given model from a single evaluation. Since this variation results from CPU scheduling that is under the control of the operating system, we can not eliminate this variation totally. However, we found ways to significantly minimise this variation across evaluations.

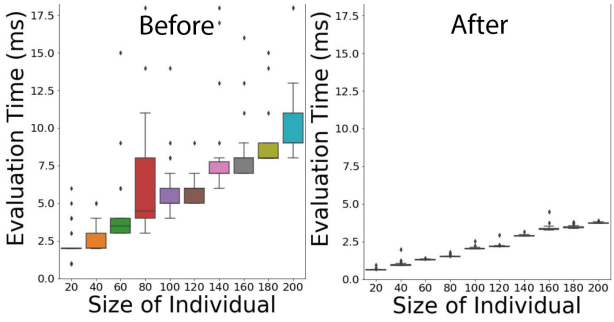
We found that CPU management options can help minimise this variation. These options include: (1) stopping all background services, (2) locking the CPU speed to prevent the operating system power management from interfering, (3) executing the experiments on dedicated processors and (4) assigning the experimental tasks a high priority. Fig. 2 illustrates the impact of these changes. Each box-plot represents multiple evaluation times for an individual of a given size. Clearly the variation decreases significantly after CPU management. Thus, we were able to use a single evaluation to measure the evaluation time.

3 Experiments

We used four existing bloat control techniques to compare size-control with time-control. When controlling time, the evaluation time replaces size in each of the bloat control techniques.

3.1 Bloat Control Techniques

a) Death by Size (DS) [16] is a steady state replacement method that replaces the larger individuals from the present population with a given probability (typ-



(a) (b)

Fig. 2. Using CPU management options decreases variability in evaluation times.

ically 0.7; we use the same). To replace an individual, DS selects two individuals randomly and replaces the larger one probabilistically. By necessity, DS uses steady-state GP.

b) Double Tournament (DT) [16, 15] increases the probability of choosing smaller individuals as parents to encourage the reproduction of similarly small offspring. DT runs two rounds of tournaments. In the first round, it runs n probabilistic tournaments each with a tournament of size 2 to select a set of n individuals. Each of these tournaments selects the smaller individual with a probability of 0.7. Then, in the second round, DT selects the fittest out of the n individuals. We implemented the DT experiments using steady-state GP.

c) Operator Equalisation (OpEq) [4, 20] allows the sizes of individuals to grow only when fitness is improving. It controls the distribution of the population by employing two core functions. The first determines the target distribution (by size) of individuals in the next generation; the second ensures that the next generation matches the target distribution. To define the target distribution, OpEq puts the current individuals into bins according to their sizes and calculates the average fitness score of each bin. This average score is then used to calculate the number of individuals to be allowed in a corresponding bin in the next generation (target distribution). Thus, from one generation to the next the target distribution changes to favour the sizes that produce fit individuals. The width of the bins can vary and thus is a parameter. Bin width of 1 to 10 has been successfully used previously [20]. In our experiments we used the better performing Dynamic OpEq variant [20] and used *bin width* = *5*. Note, OpEq uses generational replacement.

To adapt OpEq to control evaluation time, we had to estimate the time equivalent of the bin width. This value is then used to create bins to classify individuals according their evaluation times in the same way as bin width size is used to create bins to classify individuals by their sizes. To get a reliable estimate we used multiple samples, evaluated multiple times and used the median of several estimates. This is done only once at the beginning of the GP run.

d) The Tarpeian (TP) [19] method controls size-growth by assigning the worst fitness to a fraction W (recommended $W = 0.3$; we use the same) of the individuals that have above-average size. TP uses generational replacement and calculates the average size of the population at every generation.

To adapt this method to control evaluation time we simply replaced the average size with the average evaluation time.

3.2 Test Problems

We use five tough problems to compare the results in this paper. The problems are tough because the results in section 4 show that the accuracy scores are low (less than 41%). Hence, these problems require GP to run long and thus present a good test bed for complexity control because at least the size-complexity in GP grows with long runs. Four of these problems are multi-dimensional (with five or more input variables). The data set for problems 1–4 are available at [5]; Problem 5 is a bi-variate version of the function used in [8]. A summary of the data sets is available in Table 1.

ID	**Problem label**	**No. of Variables**	**No. of instances**
1	Airfoil	5	1503
2	Boston Housing	13	506
3	Concrete Strength	8	1030
4	Energy Efficiency	8	768
5	$y^2x^6 - 2.13y^4x^4 + y^6x^2$	2	250 (x=min:-0.3, step: 0.012; y=x + 0.03)

Table 1. Overview of Test Problems

3.3 Configuration and Parameters

The basic parameters for all the methods are summarised in Table 2. The other key experimental decisions are as follows. First, the individuals with divide-by-zero errors were assigned the worst fitness; as discussed in [10], the protected operators commonly used in GP lead to poor generalisation. Next, the data-sets were randomly split (without replacement) into 80% for training and 20% testing. Finally, the fitness was computed as the normalised root mean squared error (RMSE) and maximised as follows: $\frac{1}{1-\sqrt{\frac{1}{n}\Sigma_{i=1}^{n}(y_i-\hat{y_i})^2}}$.

The experiments were run on Windows 10 (64-bit) with 32GB RAM, and Intel Core i7-6700 CPU @ 3.40GHz (Quad-Core).

3.4 Initialising the Population

Section 2.2 motivated the need for an initialisation scheme that produces functionally diverse but identically sized individuals; such a scheme can increase

Parameter	**Setting**
Number of runs	50
Population size	500
Run terminates	After 35,000 evaluations ($\equiv$ 70 generations)
Random tree/subtree generation	Ramped half-and-half($1 =< depth =< 4$); and Fixed Length Initialisation (see Section 3.4)
Operators & probabilities	One point crossover = 0.9; Point mutation = 0.1
Depth Limit	17
Function set	$+, -, *, /, \sin, \cos$, neg
Constants (ERC)	$\|ERC\| = 100$ (min = 0.05, step: 0.05)
Terminal set	{Input variables} U ERC
Selection	tournament size = 3
Replacement	steady state/generational as per each method

Table 2. Summary of Parameters

the focus of the time-control on differentiating functional complexity. Therefore, we created a *Fixed-Length Initialisation scheme (FLI)* for these experiments. Henceforth, we call the Ramped-Half-and-Half initialisation the Variable Length Initialisation (VLI).

For the present study, we used the FLI to produce an initial population of unique individuals each having the same length (or size) of 10 nodes. Given the functions set size, a fixed length of 10 can easily produce populations of a few hundred unique individuals; we leave studying the impact of varying the lengths to future work. To encourage functional diversity, we do not consider two individuals different if they only differ by numeric constants.

Before applying FLI to our experiments, we examined its impact on all the methods. The charts in Fig. 3 show the mean test fitness accuracy by generation for all the methods and problems. The significance of the differences of the final populations as established by the Mann-Whitney U test are captured in Fig. 4. The figure is colour coded so that green indicates where the accuracy of the final populations produced by FLI are significantly higher, brown where VLI is higher, and yellow where the difference is not significant. FLI produced better results in 16 out of 20 for Time-control and 11 out of 20 for size-control.

We observed that when using OpEq, size-control with VLI was better than size-control with FLI on all the problems. Therefore, for OpEq we compare time-control with the result of size-control with VLI (the better result). For all other methods we used the proposed FLI.

4 Results

We compare the accuracy, complexity and compositions of the models produced by each method to controlling size and time. For accuracy, although our key measure is test fitness (accuracy on out-of-sample data), we also report training fitness; the higher the value the better. For complexity we report both the size

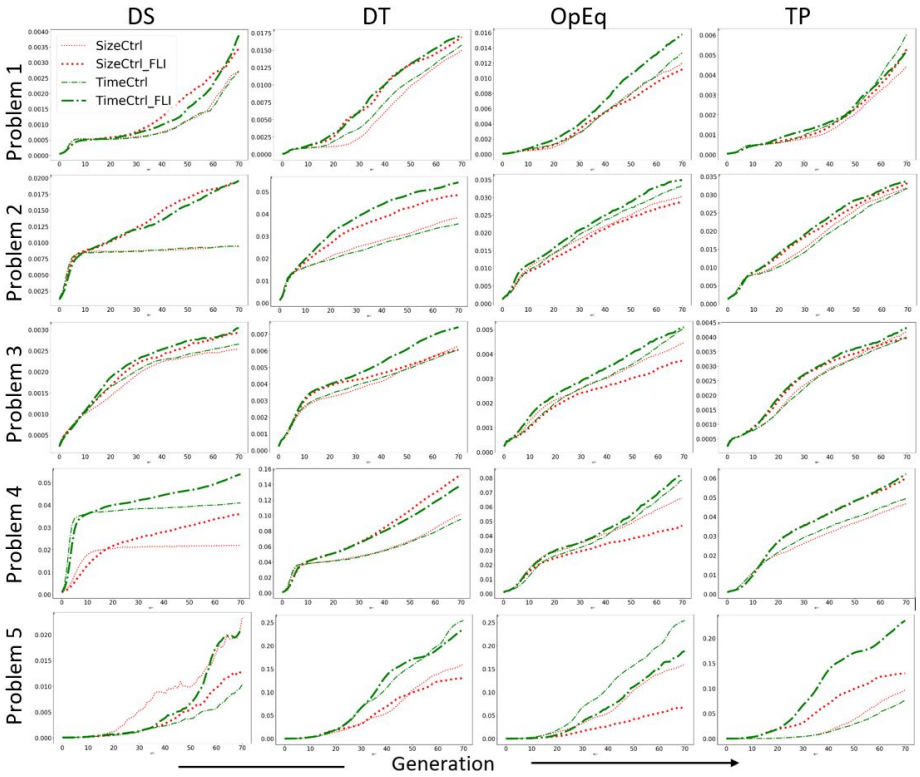


Fig. 3. Comparing the test fitness of initialisation schemes, VLI and FLI. The mean test fitness values are plotted by generation. The thick lines represent FLI and the thin VLI; the green and red lines represent time-control and size-control respectively.

and evaluation times of the models; the lower the values the better. Finally, to give further insight into the complexity of the evolved models, we report the composition of final populations as to what percentage of the genetic material comprised of more or less complex mathematical functions.

Figures 6, 7, 8 and 9 show how the test set accuracy, size and evaluation times of both time-control and size-control evolve with each of DS, DT, OpEq and TP. The figures show that for all the methods the values of all the measures increase continuously through to the final generations. Therefore, we evaluate the statistical significance of the differences in the performances in the final generations and report it in Fig. 5. Also, unless stated otherwise, henceforth, the discussion of results concerns Fig. 5.

Statistical Significance: Fig. 5 shows the colour-coded results of the Mann-Whitney U statistical test comparing the final populations of time-control and size-control. The table contains results for all the test problems and techniques. The attributes tested include the evaluation time, size, training and test fitness (accuracy on out-of-sample data). The p-values included in Fig. 5 statistically

	DEATH BY SIZE			DOUBLE TOURNAMENT			OPERATOR EQUALISATION			TARPEIAN		
	VLI Fitness (mean)	FLI Fitness (mean)	p-values	VLI Fitness (mean)	FLI Fitness (mean)	p-values	VLI Fitness (mean)	FLI Fitness (mean)	p-values	VLI Fitness (mean)	FLI Fitness (mean)	p-values
	Problem 1			Problem 1			Problem 1			Problem 1		
SizeCtrl	0.00278	0.00349	2.96E-66	0.01501	0.01691	3.60E-40	0.01206	0.01124	1.25E-16	0.00449	0.00534	4.27E-61
TimeCtrl	0.00278	0.00397	1.57E-267	0.01576	0.01718	6.21E-122	0.01336	0.01578	1.64E-91	0.00609	0.00529	1.04E-29
	Problem 2			Problem 2			Problem 2			Problem 2		
SizeCtrl	0.00958	0.01957	0	0.03836	0.04855	0	0.03032	0.02887	1.59E-11	0.03203	0.03336	1.69E-35
TimeCtrl	0.00961	0.01966	0	0.03765	0.05424	0	0.03347	0.035	1.19E-20	0.03176	0.03416	6.85E-59
	Problem 3			Problem 3			Problem 3			Problem 3		
SizeCtrl	0.00266	0.00306	1.70E-168	0.00628	0.00606	3.99E-07	0.00469	0.00392	4.50E-186	0.00438	0.00418	9.43E-23
TimeCtrl	0.00282	0.00322	4.53E-200	0.00607	0.00743	0	0.00515	0.00515	0.29144419	0.00417	0.0045	1.31E-77
	Problem 4			Problem 4			Problem 4			Problem 4		
SizeCtrl	0.02207	0.03611	0	0.10185	0.15296	0	0.06602	0.04706	0	0.0468	0.05983	0
TimeCtrl	0.04099	0.05381	0	0.09512	0.13909	0	0.07845	0.08261	6.99E-08	0.0495	0.0621	0
	Problem 5			Problem 5			Problem 5			Problem 5		
SizeCtrl	0.02323	0.01316	5.60E-10	0.16036	0.13006	6.13E-47	0.16036	0.06566	0	0.09681	0.13006	0
TimeCtrl	0.01032	0.0204	0.400547	0.25417	0.23543	1.34E-08	0.25417	0.18866	0	0.07608	0.23543	0

= Difference not significant = Significant and in favour of FLI = Significant and in favour of VLI

Fig. 4. Testing the significance of the impact of the new FLI initialisation scheme. In the final populations, FLI test fitness accuracy improved 16 of 20 for time-control and 11 of 20 for size-control.

compare the metrics of time-control against those of size-control. The rows are green when time-control is significantly better (more for accuracy, and less for both size and evaluation time), brown when it is significantly worse, and yellow when the difference is not significant.

Accuracy of Models: Time-control produced significantly more accurate models (on both training and test data) across all problems and all control techniques except on three occasions. The exceptions are problem 1 on TP (the difference is not significant), and problem 2 on DS and problem 5 on TP where size-control outperformed time-control. Overall, time-control outperformed size-control on training and test accuracy on 17 of the 20 occasions and matched size-control on one occasion.

Complexity of Models: Time-control produced less complex (evaluation time and size) models with 2 out of the 4 control techniques; the techniques are DS and DT. As seen in Fig. 5, DS produced simpler models on all the problems except on problem 2 where the difference in evaluation times is not significant. Likewise, DT produced simpler models on 4 out of 5 problems, the exception being problem 3.

Composition of Models: Table 3 counts and differentiates the nature of nodes constituting the GP trees in the final populations to understand the composition of the genetic material therein. Consistent with the results on evaluation times and sizes, time-control with DS and DT used smaller percentages of complex mathematical functions: the percentages of tree nodes containing SIN and COS with time-control are smaller than the respective figures for size-control. Likewise, OpEq and TP – much like their results on evaluation times and sizes – use greater percentages of SIN and COS.

	DEATH BY SIZE			DOUBLE TOURNAMENT			OPERATOR EQUALIZATION			TARPEIAN		
	Size Ctrl (Mean)	Time Ctrl (Mean)	p-values	Size Ctrl (Mean)	Time Ctrl (Mean)	p-values	Size Ctrl (Mean)	Time Ctrl (Mean)	p-values	Size Ctrl (Mean)	Time Ctrl (Mean)	p-values
Problem 1	**AIRFOIL**											
Evln_time	0.00619	0.005	4.7E-300	0.00943	0.00864	2.67E-245	0.01629	0.02737	0	0.00796	0.00745	2.7E-102
Length	91.53	80.97	2.26E-96	143	135.25	1.04E-97	80.03	120.75	0.00E+00	116.42	107.01	2.05E-128
Train_Fitness	0.00319	0.00359	2.03E-04	0.0148	0.01488	7.81E-23	0.01051	0.0137	8.21E-205	0.00476	0.00478	2.16E-01
Test_Fitness	0.00349	0.00397	5.85E-05	0.01691	0.01718	8.413E-38	0.01206	0.01578	2.84E-209	0.00534	0.00529	1.03E-01
Problem 2	**BOSTON**											
Evln_time	0.00154	0.00154	1.13E-106	0.00425	0.00343	0.00E+00	0.00839	0.0145	0.00E+00	0.00778	0.00787	2.52E-09
Length	9.44	9.41	7.30E-12	95.94	83.9	1.99E-147	57.28	91.46	0.00E+00	90.13	91.49	1.26E-12
Train_Fitness	0.00678	0.00674	0.00157	0.03419	0.03769	0.00E+00	0.02213	0.02558	9.7E-230	0.02346	0.02417	2E-32
Test_Fitness	0.00958	0.00944	2.65E-05	0.04855	0.05424	2.16E-294	0.03032	0.035	8.72E-139	0.03336	0.03416	1.71E-02
Problem 3	**CONCRETE**											
Evln_time	0.00397	0.00366	2.95E-59	0.00619	0.0068	1.994E-67	0.01629	0.02666	0.00E+00	0.00782	0.00956	3.56E-188
Length	44.06	41.98	1.18E-36	74.77	89.1	2.81E-163	60.51	87.03	0.00E+00	86.19	88.13	6.74E-11
Train_Fitness	0.00344	0.00357	3.18E-07	0.007	0.00841	0	0.00527	0.00593	2.99E-95	0.0047	0.005	2.08E-46
Test_Fitness	0.00306	0.00322	7.91E-11	0.00606	0.00743	0	0.00469	0.00515	1.10E-58	0.00418	0.0045	9.19E-54
Problem 4	**ENERGY**											
Evln_time	0.00443	0.00193	0.00E+00	0.00763	0.0063	2.84E-221	0.01806	0.02931	0.00E+00	0.009	0.00956	5.06E-21
Length	40.98	13.69	0	78.39	69.19	8.254E-87	60.79	89.22	0	93.28	97.27	2.91E-08
Train_Fitness	0.038	0.05655	0	0.15619	0.14312	2.086E-51	0.06819	0.08433	4.8E-149	0.06214	0.06431	1.91E-31
Test_Fitness	0.03611	0.05381	0	0.15296	0.13909	4.375E-80	0.06602	0.08261	3.4E-161	0.05983	0.0621	4.79E-34
Problem 5	**X2Y6..**											
Evln_time	0.00153	0.00148	8.54E-53	0.00164	0.00153	1.115E-21	0.00149	0.00954	0.00E+00	0.00308	0.00298	3.29E-26
Length	28.54	27.86	3.28E-28	32.94	30.21	1.41E-119	28.16	77.04	0.00E+00	87.72	81.39	7.07E-79
Train_Fitness	0.01765	0.02694	2.76E-05	0.18468	0.28112	0	0.21448	0.22432	1.53E-295	0.11117	0.11028	3.61E-02
Test_Fitness	0.01316	0.0204	5.15E-09	0.13006	0.23543	0	0.16036	0.18866	5.36E-274	0.08636	0.08446	2.67E-02

=Difference Not Significant = Significant and Favourable to Time-Ctrl. = Significant and Favourable to Size-Ctrl.

Fig. 5. Results of Mann-WhitneyU test for significance in the differences between the final populations of time-control and size-control. Time-control produced more accurate training and test scores in 17 out of 20 tests. While time-control with the steady-state methods (DS and DT) produced simpler (smaller sizes and evaluation times) models than size-control in 9 out of 10 tests, time-control with the generational methods (OpEq and TP) produced more complex models in 8 out of 10 tests.

4.1 Discussion

Section 1 argued that sensible management of complexity should produce models that are only complex enough to explain the phenomenon generating the given data but not too complex. The results show that time control almost consistently delivers superior accuracy despite splitting results on complexity measures. Even so, the increased complexity with time-control with OpEq and TP is not off the scale as is typically the case with the standard, unrestrained GP.

As to why time-control with OpEq and TP produces greater complexity is not exactly clear at present; however, it is worth noting that these two methods require generational replacement where the size (or time) distributions of the entire generations must be computed before allowing new individuals in. In contrast, DT and DS are steady state methods where a new individual replaces the loser of a tournament.

Interestingly, Fixed Length Initialisation (FLI) improved the results with not only time control but more often than not even with size control. The results encourage further investigation into this initialisation technique. FLI is designed to promote compositional (functional) diversity and thus allow time-control to distinguish complexity based more on composition than on size. However, FLI

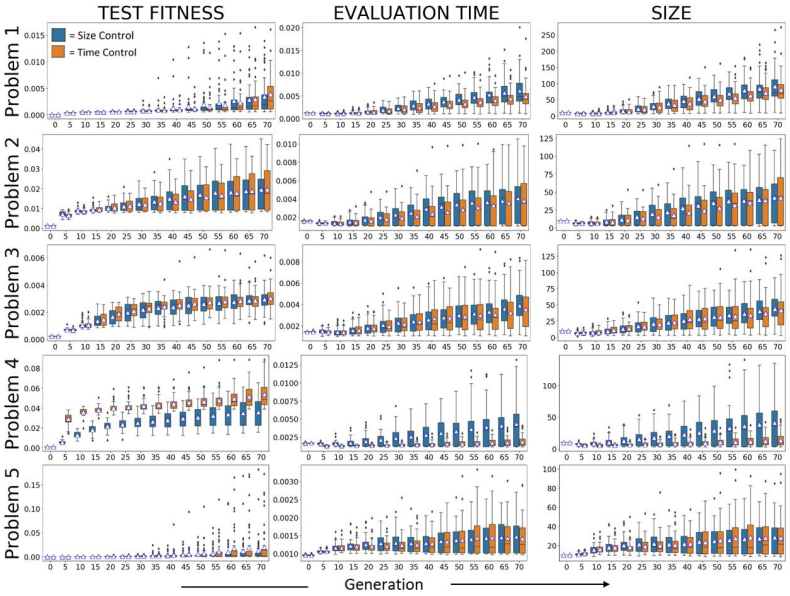


Fig. 6. Death By Size: Comparing changes in metrics by generation between time-control and size-control using DS.

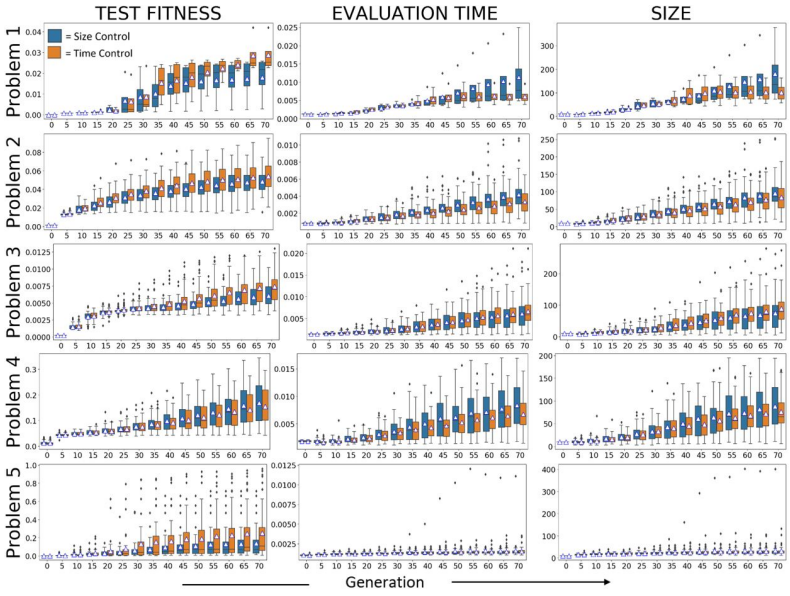


Fig. 7. Double Tournament: Comparing changes in metrics by generation between time-control and size-control using DT.

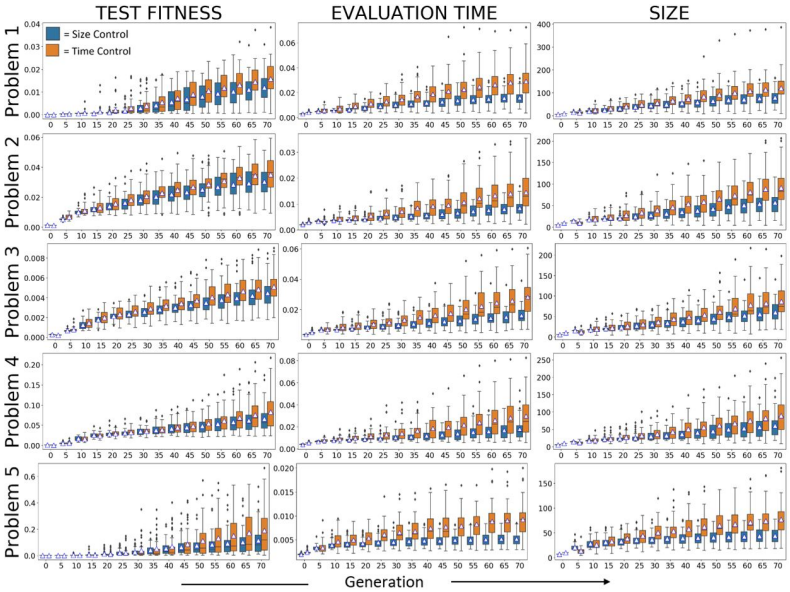


Fig. 8. Operator Equalisation: Comparing changes in metrics by generation between time-control and size-control using OpEq.

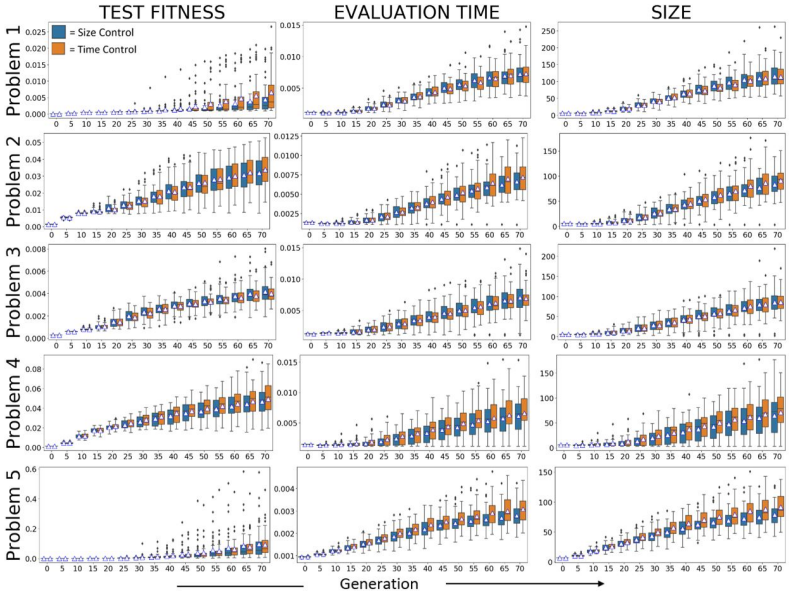


Fig. 9. Tarpeian: Comparing changes in metrics by generation between time-control and size-control using TP.

can not enforce size similarity beyond the initial generation; therefore, further work must investigate the effects of promoting size similarity in the remaining evolution and see if that further intensifies the effect of time-control.

		Problem 1		Problem 2		Problem 3		Problem 4		Problem 5	
DT	Component Type:	Size Ctrl	Time Ctrl	Size Ctrl	Time Ctrl	Size Ctrl	Time Ctrl	Size Ctrl	Time Ctrl	Size Ctrl	Time Ctrl
	SIN & COS Operators:	19.15%	17.55%	17.67%	9.15%	10.38%	6.82%	18.90%	13.14%	8.02%	5.23%
	ADD & SUB Operators:	33.28%	34.27%	27.80%	30.17%	27.24%	29.72%	24.40%	27.94%	14.84%	12.11%
DS	Component Type:	Size Ctrl	Time Ctrl	Size Ctrl	Time Ctrl	Size Ctrl	Time Ctrl	Size Ctrl	Time Ctrl	Size Ctrl	Time Ctrl
	SIN & COS Operators:	15.29%	8.89%	11.25%	10.42%	9.78%	7.69%	16.88%	5.16%	6.66%	4.42%
	ADD & SUB Operators:	36.31%	37.95%	30.80%	28.77%	32.82%	36.03%	26.17%	30.77%	14.45%	14.55%
OpEq	Component Type:	Size Ctrl	Time Ctrl	Size Ctrl	Time Ctrl	Size Ctrl	Time Ctrl	Size Ctrl	Time Ctrl	Size Ctrl	Time Ctrl
	SIN & COS Operators:	11.95%	15.85%	12.06%	16.64%	9.42%	13.28%	18.26%	20.73%	9.38%	12.11%
	ADD & SUB Operators:	27.18%	24.98%	24.82%	23.23%	23.53%	21.71%	22.76%	21.46%	19.14%	18.04%
TP	Component Type:	Size Ctrl	Time Ctrl	Size Ctrl	Time Ctrl	Size Ctrl	Time Ctrl	Size Ctrl	Time Ctrl	Size Ctrl	Time Ctrl
	SIN & COS Operators:	14.05%	15.70%	17.29%	17.34%	15.81%	15.21%	18.07%	20.31%	8.47%	9.90%
	ADD & SUB Operators:	29.86%	30.46%	26.49%	25.01%	25.02%	24.03%	23.16%	23.29%	20.04%	19.26%

Table 3. Composition of the final populations.

5 Conclusions and Future Work

This paper asks the question - why not use time instead of size to measure complexity in GP? Unlike model size, evaluation time is a function of both syntactic and computational characteristics of a model. This measure is broadly applicable, and although this paper studies regression problems, in principle, evaluation time can represent complexity in other domains as well.

A criticism of evaluation time is the variability in its repeated measurements; therefore, this paper shows how to minimise this variability.

The results indicate that the nuanced notion of complexity in time-control almost consistently produces superior accuracy on both training and test data. Even when time-control produces slightly greater sizes or times, the correspondingly superior accuracy counter-weighs these increases. After all, the complexity-control is not the end-goal alone; instead, it should also accompany better accuracy. Even so, the increase in complexity is not off the scale as is typically the case with unrestrained GP.

The paper also shows that time-control can differentiate functional complexity especially when the population has identically-sized individuals. To facilitate this, the paper proposes Fixed Length Initialisation (FLI) that creates an identically-sized but functionally-diverse population. The results show that while FLI particularly suits time-control, it also generally improves the performance of size-control.

Overall, the paper poses evaluation time as a promising alternative to counting nodes in GP.

References

1. Azad, R.M.A., Ryan, C.: Variance based selection to improve test set performance in genetic programming. In: Proceedings of the 13th annual conference on Genetic and evolutionary computation. pp. 1315–1322. ACM, Dublin, Ireland (2011), http://dl.acm.org/citation.cfm?id=2001754
2. Azad, R.M.A., Ryan, C.: A simple approach to lifetime learning in genetic programming based symbolic regression. Evolutionary Computation **22**(2), 287–317 (jul 2014). https://doi.org/doi:10.1162/EVCO_a_00111, http://www.mitpressjournals.org/doi/abs/10.1162/EVCO_a_00111
3. Couture, M.: Complexity and chaos-state-of-the-art; formulations and measures of complexity. Tech. rep., DEFENCE RESEARCH AND DEVELOPMENT CANADA VALCARTIER (QUEBEC) (2007)
4. Dignum, S., Poli, R.: Operator equalisation and bloat free GP. In: et al, M.O. (ed.) Proceedings of the 11th European Conference on Genetic Programming, EuroGP 2008. Lecture Notes in Computer Science, vol. 4971, pp. 110–121. Springer, Naples (26-28 Mar 2008). https://doi.org/10.1007/978-3-540-78671-9_10
5. Dua, D., Karra Taniskidou, E.: UCI machine learning repository (2017), http://archive.ics.uci.edu/ml
6. Falco, I.D., Iazzetta, A., Tarantino, E., Cioppa, A.D., Trautteur, G.: A kolmogorov complexity-based genetic programming tool for string compression. In: Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation. pp. 427–434. Morgan Kaufmann Publishers Inc., Morgan Kaufmann, Las Vegas, Nevada, USA (2000)
7. Griinwald, P.: Introducing the minimum description length principle. Advances in minimum description length: Theory and applications **3**, 3–22 (2005)
8. Gustafson, S., Burke, E.K., Krasnogor, N.: On improving genetic programming for symbolic regression. In: Corne, D., Michalewicz, Z., McKay, B., Eiben, G., Fogel, D., Fonseca, C., Greenwood, G., Raidl, G., Tan, K.C., Zalzala, A. (eds.) Proceedings of the 2005 IEEE Congress on Evolutionary Computation. vol. 1, pp. 912–919. IEEE Press, Edinburgh, Scotland, UK (2-5 Sep 2005), http://ieeexplore.ieee.org/servlet/opac?punumber=10417&isvol=1
9. Iba, H., de Garis, H., Sato, T.: Genetic programming using a minimum description length principle. In: Kinnear, Jr., K.E. (ed.) Advances in Genetic Programming, chap. 12, pp. 265–284. MIT Press, Cambridge, MA, USA (1994), http://cognet.mit.edu/sites/default/files/books/9780262277181/pdfs/9780262277181_chap12.pdf
10. Keijzer, M.: Improving symbolic regression with interval arithmetic and linear scaling. In: European Conference on Genetic Programming. pp. 70–82. EuroGP, Springer, Essex, UK (2003)
11. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA, USA (1992), http://mitpress.mit.edu/books/genetic-programming
12. Kulkarni, S.R., Harman, G.: Statistical learning theory: a tutorial. Wiley Interdisciplinary Reviews: Computational Statistics **3**(6), 543–556 (2011)
13. Kumar, A., Goyal, S., Varma, M.: Resource-efficient machine learning in 2 KB RAM for the internet of things. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 70, pp. 1935–1944. PMLR, International Convention Centre, Sydney, Australia (06–11 Aug 2017)

14. Lipton, Z.C.: The mythos of model interpretability. Commun. ACM **61**(10), 36–43 (Sep 2018). https://doi.org/10.1145/3233231, http://doi.acm.org/10.1145/3233231
15. Luke, S., Panait, L.: Fighting bloat with nonparametric parsimony pressure. In: Merelo-Guervos, J.J., Adamidis, P., Beyer, H.G., Fernandez-Villacanas, J.L., Schwefel, H.P. (eds.) Parallel Problem Solving from Nature - PPSN VII. pp. 411–421. No. 2439 in Lecture Notes in Computer Science, LNCS, Springer-Verlag, Granada, Spain (7-11 Sep 2002). https://doi.org/10.1007/3-540-45712-7_40, http://www.springerlink.com/openurl.asp?genre=article&issn=0302-9743&volume=2439&spage=411
16. Luke, S., Panait, L.: A comparison of bloat control methods for genetic programming. Evolutionary Computation **14**(3), 309–344 (Fall 2006). https://doi.org/10.1162/evco.2006.14.3.309, http://cognet.mit.edu/system/cogfiles/journalpdfs/evco.2006.14.3.309.pdf
17. Mei, Y., Nguyen, S., Zhang, M.: Evolving time-invariant dispatching rules in job shop scheduling with genetic programming. In: Castelli, M., McDermott, J., Sekanina, L. (eds.) EuroGP 2017: Proceedings of the 20th European Conference on Genetic Programming. LNCS, vol. 10196, pp. 147–163. Springer Verlag, Amsterdam (19-21 Apr 2017). https://doi.org/10.1007/978-3-319-55696-3_10
18. Paris, G., Robilliard, D., Fonlupt, C.: Exploring overfitting in genetic programming. In: Liardet, P., Collet, P., Fonlupt, C., Lutton, E., Schoenauer, M. (eds.) Evolution Artificielle, 6th International Conference. Lecture Notes in Computer Science, vol. 2936, pp. 267–277. Springer, Marseilles, France (27-30 Oct 2003). https://doi.org/10.1007/b96080, revised Selected Papers
19. Poli, R.: A simple but theoretically-motivated method to control bloat in genetic programming. In: Ryan, C., Soule, T., Keijzer, M., Tsang, E., Poli, R., Costa, E. (eds.) Genetic Programming, Proceedings of EuroGP'2003. LNCS, vol. 2610, pp. 204–217. Springer-Verlag, Essex (14-16 Apr 2003). https://doi.org/doi:10.1007/3-540-36599-0_19, http://www.springerlink.com/openurl.asp?genre=article&issn=0302-9743&volume=2610&spage=204
20. Silva, S., Dignum, S., Vanneschi, L.: Operator equalisation for bloat free genetic programming and a survey of bloat control methods. Genetic Programming and Evolvable Machines **13**(2), 197–238 (2012)
21. Vanneschi, L., Castelli, M., Silva, S.: Measuring bloat, overfitting and functional complexity in genetic programming. In: et al, J.B. (ed.) GECCO '10: Proceedings of the 12th annual conference on Genetic and evolutionary computation. pp. 877–884. ACM, Portland, Oregon, USA (7-11 Jul 2010). https://doi.org/10.1145/1830483.1830643
22. Vapnik, V.N.: Statistical learning theory. Adaptive and learning systems for signal processing, communications, and control, Wiley, New York, NY (1998), OCLC: 845016043
23. Vladislavleva, E.J., Smits, G.F., Den Hertog, D.: 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 (2009). https://doi.org/10.1109/TEVC.2008.926486, http://ieeexplore.ieee.org/document/4632147/