Where Should We Stop? An Investigation on Early Stopping for GP Learning

Thi Hien Nguyen[1], Xuan Hoai Nguyen[2], Bob McKay[3], and Quang Uy Nguyen[1]

[1] Le Quy Don University, Vietnam
[2] Hanoi University, Vietnam
[3] Seoul National University, Korea

Abstract. We investigate the impact of early stopping on the speed and accuracy of Genetic Programming (GP) learning from noisy data. Early stopping, using a popular stopping criterion, maintains the generalisation capacity of GP while significantly reducing its training time.

1 Introduction

Genetic Programming (GP) [1] describes a class of evolutionary algorithms that solve problems by finding solutions of non-predefined complexity. GP has often been viewed as a form of machine learning, as it aims to induce relations between input and output data in the form of a functional expression or program. Among the successful real-world applications, learning tasks have been common [2]. Early GP research seldom attended to the generalisation capacity of GP. The focus was on how GP could fit the data by finding an exact solution/relation While learning exact solutions may be important in some discovery tasks, machine learning (ML) [3] has emphasised generalisation over unseen data as the most important aspect. A learning machine should avoid over-fitting the training data. Recently, GP generalisation has caught more attention, with an increasing number of related publications [4, 5, 6, 7, 8, 9, 10, 11]. In particular, GP overfitting has been repeatedly demonstrated: while the errors on the training data may improve over the generations, it may deteriorate on unseen test data.

There are at least two ways to combat over-fitting for learning machines [12] – reducing machine complexity (or regularisation) and early stopping. In the first approach, based on Occam's razor [3], the learning process avoids over-fitting by preferring simple hypotheses. In the second, a learner does not eliminate over-fitting but rather tries detect it and stops training once it does so. Early stopping is widely used for learning processes because it is simple, easy to implement, and, in many cases, superior to regularisation [13]. In GP, there have been a number of attempts to improve GP generalisation by regularisation, through including the complexity of an individual as part of its fitness [14, 15, 16, 17, 6]. However, as reported in [18], reducing complexity of individuals may not lead to better generalisation. To date, the only published work on early stopping for GP has been two preliminary works, [19] and [20]. In [19]. Tuite et al. adapted three stopping criteria adopted from [12] to Grammatical Evolution (GE). These criteria helped GE to detect when to stop during training. However the experiments only covered two simple problems, and no detail of the impact of early stopping

L.T. Bui et al. (Eds.): SEAL 2012, LNCS 7673, pp. 391–399, 2012.

was provided. In [20], we investigated the impact of early stopping on GP learning. Some results suggested that early stopping could trade off generalisation error and run time complexity, but the results were mixed. However we can now explain them as due to ineffective stopping criteria; better criteria lead to better results.

In this paper, we re-investigate the impact of early stopping on GP learning based on Prechelt's stopping criterion [12]. In Section 2, we briefly review related work on GP generalisation, and on the role of early stopping in learning machines. Section 3 introduces our implementation of early stopping. Sections 4 details the experimental settings. The results are presented and discussed in Section 5. The paper concludes in Section 6 by highlighting possible future work.

2 Background

2.1 Over-fitting and Generalisation in GP

Although achieving high generalisation capability is the main objective any learning machine [3], it was neglected in the early work on GP. Before Kushchu published his seminal paper on generalisation in GP [7], there was little in the literature dealing with GP generalisation. In [21], Francone et al. proposed a system called Compiling GP (CGP) and compared its generalisation with that of other machine learning techniques, demonstrating comparable results. Extending the use of the mutation operator was shown to improve generalisation. Zhang [14] proposed avoiding over-fitting through Minimum Description Length (MDL) methods, providing an adaptive mechanism for balancing between accuracy and complexity preferences. He obtained robust results for tasks with noisy or incomplete data. Hooper et al. [15] argued that expression simplification may help GP to avoid over-fitting and obtain better generalisation. In [22], Iba incorporated Bagging and Boosting into GP (BagGP and BoostGP), showing improved generalisation in discovering trigonometric identities, chaotic time series prediction, and 6 bit multiplexer.

Recently, generalisation in GP has gained more attention. In [9], Panait and Luke investigated the impact of using six common sampling methods on the robustness of GP solutions. None dominated on all problems, showing that the impact of sampling method is problem domain dependent. Paris et al. [23] used GP as the core learning algorithm in a boosting framework to trigger over-fitting on two problems, demonstrating much better performance with boosting. Becker and Seshadri [16] compared two techniques to evolve more comprehensible trading rules. One applied expert-level knowledge of useful arithmetic operators for technical trading, while the other used a complexity penalty in the fitness. The second evolved more comprehensible, better-generalising rules. Mahler et al. [24] tried Tarpeian control on symbolic regression problems and tested for generalisation accuracy, finding mixed results: the effects of Tarpeian control are problem-dependent. In [6], Gagné et al. investigated two methods to improve GP generalisation: the selection of the best-of-run individuals through three separate data sets (training, validation, and test), and the application of parsimony pressure. The validation set results showed somewhat improved stability than parsimony pressure.

More recently, Costelloe and Ryan [4] investigated the role of generalisation in GP. They showed that linear scaling [25] improves GP training performance, but not test performance. They proposed combining Linear Scaling and the No Same Mate strategy [26] for better performance. Vanneschi and Gustafson [11] improved GP generalisation through a crossover based similarity measure. They keep a list of over-fitted individuals, and eliminate individuals that are too similar (based on structural distance or a subtree crossover metric) to individuals in that list. The method was tested on a real-life drug discovery regression problem and showed improvements in GP generalisation. Nguyen Quang Uy et al. [8] showed that semantic information could guide GP crossover to reducing code bloat and improve generalisation on real-valued symbolic regression problems. In [27], Vanneschi at el. proposed a method to quantify/detect over-fitting during GP learning.

2.2 Early Stopping for Learning Machines

The preceding work has focused on avoiding over-fitting to improve GP generalisation, generally through reducing individual complexity. This resembles the common machine learning technique of regularisation. While regularisation often works in GP, recent work has shown that reducing individual complexity does not guarantee better generalisation in GP [18]. Over-fitting is sometimes inevitable. Machine learning also uses another approach: is stopping training when over-fitting is detected [12]. Early stopping has been widely used in neural networks (NN) because of its simplicity and effectiveness. In [12], Prechelt considered three criteria for stopping training. The first criterion stops as soon as the generalisation loss (on an independent validation set) exceeds a predetermined threshold. The second criterion uses the quotient of generalisation loss and progress, while the third stops when generalisation error first increases over s successive training strips. None of the criteria dominated the others in terms of average generalisation performance. However "slower" criteria, stopping later than others, on average improve generalisation, but at the cost of greater training time [12].

In other word, early stopping embodies a trade-off between training time and generalisation. In [28], Shafi and Abbass investigated the effects of early stopping in learning classifier system (LCS); as with NN, they found that early stopping improves generalisation. The preliminary work of Tuite et al. [19] i and of ourselves [20] in applying early stopping to GP was described in the preceding section.

3 Methods

Our method is inspired by Prechelt's work [12] on early stopping criteria for NNs and that of Tuite et al. [19] on early stopping for GE. We use three data sets: training, testing and validation. The validation set is used to estimate the generalisation error of individuals during evolution. Our stopping criterion is the second from [12].[1]

[1] We have investigated 5 different stopping criteria, but only report the best due to space limitations. Tuite et al. [19] reported the third criterion from [12] as best-performing, but we found the second (with a different parameter setting) better.

To specify the stopping criterion, we first define generalisation loss during the evolutionary process in equation 1:

$$GL(g) = 100.(\frac{E_{va}(g)}{E_{opt}(g)} - 1) \tag{1}$$

where $E_{va}(g)$ is the validation error of the best individual at generation g. $E_{opt}(g)$ is the lowest validation error up to generation g:

$$E_{opt}(g) = \min_{g' \leq g} E_{va}(g') \tag{2}$$

Sharp generalisation loss is an indication of ineffective learning. However, if the training error is still decreasing rapidly, generalisation loss might recover, since we assumed that over-fitting only begins when the error decreases slows [12]. Training progress is defined over a training strip of length k: a sequence of k successive generations $n+1, .., n+k$ with $k|n$. It measures by how much the average training error exceeds the minimum training error within the strip:

$$P_k(t) = 1000.(\frac{\sum_{t'=g-k+1}^{g} E_{tr}(g)}{k \min_{t'=g-k+1}^{g} E_{tr}(g)} - 1) \tag{3}$$

where $E_{tr}(g)$ is the training error of the best individual at generation g. We use the stopping criterion from [12] defined as:
PQ_α: stop after the first end-of-strip generation g satisfying $\frac{GL(g)}{P_k(t)} > \alpha$
(In this criterion, α is a tuning parameter, permitting small excursions in the validation error.)

4 Experiments

We conducted experiments on fifteen regression problems, including both synthetic and real-world data sets. The ten synthetic data sets are given in Table 1. These test functions have been extensively used in the GP and Machine Learning literature.

Table 1. The Synthetic Test Functions

1	$F_1(x) = x^4 + x^3 + x^2 + x$
2	$F_2(x) = \cos(3x)$
3	$F_3(x) = \sqrt{x}$
4	$F_4(x) = x_1 x_2 + \sin((x_1 - 1)(x_2 - 1))$
5	$F_5(x) = x_1^4 - x_1^3 + \frac{x_2^2}{2} - x_2$
Friedman1	$F_6(x) = 10 \sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5$
Friedman2	$F_7(x) = \sqrt{x_1^2 + (x_2 x_3 - \frac{1}{x_2 x_4})^2}$
Gabor	$F_8(x) = \frac{\pi}{2} e^{-2(x_1^2 + x_2^2)} \cos[2\pi(x_1 + x_2)]$
Multi	$F_9(x) = 0.79 + 1.27x_1 x_2 + 1.56x_1 x_4 + 3.42x_2 x_5 + 2.06x_3 x_4 x_5$
3-D Mexican Hat	$F_10(x) = \frac{sin(\sqrt{x_1^2 + x_2^2})}{\sqrt{x_1^2 + x_2^2}}$

Table 2. The real-world data sets

Data sets	Abbreviation	Features	Size	Source
Concrete Slump Test	$Slum$	10	103	UCI
Concrete Compressive Strength	$Conc$	9	1030	UCI
Pollen	$Poll$	5	3848	StatLib
Chscase.census6	$Cens$	7	400	StatLib
No2	$No2$	8	500	StatLib

Table 3. Data Sets for Problems. For the synthetic problems, the notation [min, max] defines the range from which the data points are sampled.

Problem	Function	Attribute	Sample size	Training size	Validation size	Test size
1	F_1	$x \in [-1, 1]$	1500	750	375	375
2	F_2	$x \in [0, 2]$	1200	600	300	300
1	F_3	$x \in [0, 4]$	1200	600	300	300
4	F_4	$x_1, x_2 \in [-3, 3]$	1000	500	250	250
5	F_5	$x_1, x_2 \in [-3, 3]$	1000	500	250	250
6	F_6	$x_1, x_2, x_3, x_4, x_5 \in [0, 1]$	1000	500	250	250
7	F_7	$x_1 \in [0, 100]$, $x_2 \in [40\pi, 560\pi]$, $x_3 \in [0, 1]$, $x_4 \in [1, 11]$	1000	500	250	250
8	F_8	$x_1, x_2 \in [0, 1]$	1200	600	300	300
9	F_9	$x_1, x_2, x_3, x_4, x_5 \in [0, 1]$	1000	500	250	250
10	F_{10}	$x_1, x_2 \in [-4\pi, 4\pi]$	1000	500	250	250
11	$Slum$		103	52	25	26
12	$Conc$		1030	515	257	258
13	$Poll$		3848	1924	962	962
14	$Cens$		400	200	100	100
15	$No2$		500	250	125	125

The five real-world data sets were chosen from the UCI machine learning repository [29] and StatLib [30], and are shown in Table 2.

Table 3 shows the ranges from which the inputs for the synthetic problems were drawn, together with (for all problems) the sizes of the data sets into which the training instances were divided.

Since these experiments are about generalisation ability, we corrupted the output of the synthetic test functions by adding Gaussian noise with $\sigma = 0.01$. The parameter settings for the GP systems are given in Table 4. Standard GP is denoted by GPM, while

Table 4. Parameter settings for the GP systems

Population Size	500
Number of generations	150 (for GPM)
Tournament size	3
Crossover probability	0.9
Mutation probability	0.05
Initial Max depth	6
Max depth	15
Non-terminals	+, -, *, / (protected)
	, sin, cos, exp, log (protected)
Standardized fitness	mean absolute error
Number of runs	50

GPV is a variant which checks the stopping criterion at each generation, terminating if it is satisfied. Otherwise GPM and GPV are identical, up to using the same random seed in corresponding runs.

All runs were conducted on a Compaq Presario CQ3414L computer with Intel Core i3-550 Processor (4M Cache, 3.20 GHz) running Ubuntu Linux operating system.

5 Results and Discussions

For each run, we recorded the generalisation error (GE – measured on the test data) of the best individual of the run, the size of the best individual, the first generation where the best individual of the run was discovered, and the last generation of the run (for GPV). Table 5 presents the results, averaged over 100 runs (with standard deviations). Three different values for α (7, 29, 51) were tested, resulting in three different versions

Table 5. Best Generalisation Errors, Run Times, and p-values and percentage differences, GP with Stopping Criterion (various α) vs GP

	Generalisation Error				p-value of GE			run time				p-value of time		
Functions	PQ_7	PQ_{29}	PQ_{51}	GP	PQ_7	PQ_{29}	PQ_{51}	PQ_7	PQ_{29}	PQ_{51}	GP	PQ_7	PQ_{29}	PQ_{51}
F_1	0.0097±	0.0085±	0.0090±	0.0097±	0.9509	0.2608	0.4975	348.9897±	377.8947±	376.5979±	528.9278±	**0.0000**	**0.0000**	**0.0000**
	0.0080	0.0071	0.0077	0.0081	98.8%	88.4%	92.6%	208.3706	203.1782	201.5762	172.4390	66.6%	73.7%	72.5%
F_2	0.0126±	0.0126±	0.0125±	0.0222±	**0.0005**	**0.0004**	**0.0004**	348.9897±	377.8947±	376.5979±	528.9278±	**0.0000**	**0.0000**	**0.0000**
	0.0138	0.0137	0.0138	0.0223	59.0%	58.4%	58.3%	208.3706	203.1782	201.5762	172.4390	62.7%	66.3%	67.2%
F_3	0.0050±	0.0050±	0.0050±	0.0056±	0.4741	0.3156	0.3113	236.0652±	259.8370±	262.2935±	461.9588±	**0.0000**	**0.0000**	**0.0000**
	0.0044	0.0044	0.0044	0.0046	90.5%	87.1%	90.9%	156.7395	166.9716	166.6070	132.1351	52.1%	56.9%	59.2%
F_4	0.5084±	0.5036±	0.5036±	0.4958±	***0.0368***	0.1222	0.1313	110.4432±	128.4778±	128.5000±	223.3333	**0.0000**	**0.0020**	**0.0020**
	0.0430	0.0485	0.0485	0.0559	100.6%	103.0%	101.9%	162.9480	189.7861	189.5296	221.7522	52.5%	61.7%	60.7%
F_5	1.2100±	1.1854±	1.1688±	1.2342±	0.7429	0.5000	0.3628	400.0000±	421.8750±	429.8105±	563.0632±	**0.0000**	**0.0000**	**0.0000**
	0.5089	0.4907	0.4804	0.5073	99.8%	97.4%	97.1%	202.5774	202.9539	205.2737	166.2303	72.2%	75.6%	76.9%
F_6	1.5875±	1.5708±	1.5381±	1.4677±	***0.0000***	***0.0094***	0.0541	836.1277±	896.7660±	947.1522±	1225.4896±	**0.0000**	**0.0000**	**0.0000**
	0.2981	0.3036	0.2641	0.2317	107.7%	106.6%	103.8%	389.1948	417.8879	417.6109	242.9004	68.6%	73.0%	76.9%
F_7	3.9381±	3.9689±	3.9311±	4.0648±	0.9580	0.7531	0.6594	549.4792±	601.7500±	625.3750±	797.1954±	**0.0000**	**0.0000**	**0.0000**
	2.1808	2.1971	2.1762	1.9215	100%	96.8%	98.0%	322.8258	314.5690	325.4105	237.6541	68.9%	77.4%	81.1%
F_8	0.1456±	0.1414±	0.1412±	0.1312±	0.0656	0.1841	0.1953	529.9000±	564.9300±	568.4600±	747.0100 ±	**0.0000**	**0.0000**	**0.0000**
	0.0603	0.0589	0.0590	0.0483	110.9%	107.7%	107.6%	281.5569	271.7877	270.1209	169.6208	70.9%	75.6%	76.1%
F_9	0.1595±	0.1581±	0.1581±	0.1619±	0.7081	0.5547	0.5547	519.7340±	541.4894±	539.2660±	712.4457±	**0.0000**	**0.0000**	**0.0000**
	0.0470	0.0471	0.0471	0.0399	98.9%	98.4%	99.3%	283.8214	285.9565	285.0949	175.5821	74.7%	76.4%	77.1%
F_{10}	0.0824±	0.0813±	0.0815±	0.0802±	***0.0076***	0.1773	0.1119	188.6186±	261.1667±	271.9896±	565.7113±	**0.0000**	**0.0000**	**0.0000**
	0.0050	0.0057	0.0055	0.0054	102.7%	101.4%	102.4%	208.0677	238.6885	244.9955	212.6491	33.8%	46.6%	49.2%
$Slum$	5.9632±	5.7866±	5.7869±	5.5959±	0.1521	0.4462	0.4489	62.4396±	90.9111±	97.2584±	422.4045±	**0.0000**	**0.0000**	**0.0000**
	1.8028	1.7203	1.7347	1.6211	105.6%	103.1%	102.8%	53.4050	79.0853	80.7470	187.8031	14.7%	21.8%	23.9%
$Conc$	7.8401±	6.9023±	6.9508±	6.8906±	***0.0000***	0.9353	0.6893	1036.8400±	1518.4045±	1541.4286±	1708.7677±	**0.0000**	**0.0443**	0.0735
	1.9861±	0.9700±	1.0593±	1.0114	112.9%	100.4%	101.2%	777.4485	712.0448	708.1432	556.4130	60.8%	85.6%	87.4%
$Poll$	1.7451±	1.7426±	1.7403±	1.7206±	0.5995	0.6366	0.6747	1768.7000±	1995.0500±	2064.0700±	3113.6200±	**0.0000**	**0.0000**	**0.0000**
	0.3320	0.3301	0.3338	0.3275	101.4%	101.3%	101.1%	1105.8256	1199.1555	1190.9865	1090.1410	56.8%	64.1%	66.3%
$Cens$	1.2417±	1.2469±	1.2501±	1.3475±	**0.0000**	**0.0000**	**0.0000**	87.0488±	138.2530±	156.6867±	427.2708±	**0.0000**	**0.0000**	**0.0000**
	0.0509	0.0489	0.0490	0.2042	93.4%	96.0%	97.3%	87.8209	145.4627	167.0125	185.6668	20.8%	33.8%	38.2%
$No2$	0.4846±	0.4822±	0.4813±	0.4740±	0.0663	0.1457	0.1888	164.6364±	235.6566±	254.1100±	567.1600 ±	**0.0000**	**0.0000**	**0.0000**
	0.0421	0.0406	0.0402	0.0387	102.2%	101.7%	101.6%	146.8023	194.6069	201.4310	225.5351	29.0%	41.6%	44.8%

of GPV (PQ_7, PQ_{29}, PQ_{51} in the table).[2]. We tested the significance of the differences in generalisation error between GPM and GPV, using a two-tailed pairwise t-test with confidence level 0.95 ($\alpha = 0.05$). The p-values are shown.[3] Our null and alternative hypotheses were:

- H_0 = "the average GE of GPM and GPV are the same".
- H_1 = "GPM and GPV have different average GE".

In Table 5, if H_0 is rejected the printed p-value is bolded (if GPV is better than GP) or italicised and bolded (if GPV is worse than GPM). When α is small, early stopping can degrade the generalisation capacity of GP. For $\alpha = 7$ (column PQ7), four functions (F_4, F_6, F_{10}, $Conc$) show worse generalisation from GPV solutions than from GPM (H_0 is rejected). This reduces to one function for $\alpha = 29$ (column PQ29) and none for $\alpha = 51$ (i.e. H_0 is accepted in most cases); on the contrary, in two cases for the latter, GPB has strictly better generalisation. Table 5 also shows the percentage ratio between GE of solutions found by each GPV system with that found by GP (averaged over all runs). The average run time (with standard deviations) of each system is given in the right haft of Table 5, with p-values for a two-tailed t-test with null and alternative hypotheses as follows:

- H_0 = "the average run times of GPM and GPV are the same".
- H_1 = "GPM and GPV have different average runtime".

For all settings of $alpha$, GPV has significantly better run time than GPM, on all problems. As α increases, so does the relative runtime of GPV – but generally rather slowly. Overall, the better overall performance of larger α values is probably worth the increased runtime.

6 Conclusions

We have presented a study of the impact of early stopping on GP learning, focusing on its learning efficiency (generalisation error and runtime complexity). The results from 10 synthetic regression and 5 real-world problems show that early stopping improves GP learning efficiency by significantly reducing training time while retaining, or even slightly improving, the quality of the solutions it learns. It also confirms the value of Prechelt's second stopping criterion [12] with different settings of parameter $alpha$ than were used by Prechelt. The results somewhat contradict those reported in [19], where this stopping criterion is found to be less effective for GE. We conjecture that this results from the the different settings of α, and that Tuite et al. might see better results from the second criterion with increased values of α.

In future, we plan to test early stopping criteria on more problems, and to compare with regularisation via Tarpeian Control [24] and similar methods. Since early stopping

[2] [12] and [19] tested α values of 2.5, 5, and 7.5. For our problem domains, these values were too small, and we recalibrated α for the best performance of GPV.

[3] Values shown as 0.0000 were truncated from the actual value so as to fit the table on the page.

uses only the validation and training errors of individuals, and does not interfere with the fitness function or individual complexity metrics as does regularisation, the two (regularisation and early stopping) could potentially be used in combination. Studying such a combination is in our intermediate future plans.

Acknowledgments. The ICT at Seoul National University provided research facilities for this study. This work was funded by The Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.01-2011.08. The first author would like to thank the R@FIT funding of school of Information Technology, Le Quy Don University for providing financial support for her to present this paper.

References

[1] Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. The MIT Press, Cambridge (1992)
[2] Poli, R., McPhee, W.L.N.: A Field Guide to Genetic Programming (2008), `http://lulu.com`
[3] Mitchell, T.M.: Machine Learning. McGraw Hill (1997)
[4] Costelloe, D., Ryan, C.: On Improving Generalisation in Genetic Programming. In: Vanneschi, L., Gustafson, S., Moraglio, A., De Falco, I., Ebner, M. (eds.) EuroGP 2009. LNCS, vol. 5481, pp. 61–72. Springer, Heidelberg (2009)
[5] Foreman, N., Evett, M.: Preventing overfitting in GP with canary functions. In: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation (GECCO 2005), pp. 1779–1780. ACM (2005)
[6] Gagné, C., Schoenauer, M., Parizeau, M., Tomassini, M.: Genetic Programming, Validation Sets, and Parsimony Pressure. In: Collet, P., Tomassini, M., Ebner, M., Gustafson, S., Ekárt, A. (eds.) EuroGP 2006. LNCS, vol. 3905, pp. 109–120. Springer, Heidelberg (2006)
[7] Kushchu, I.: Genetic programming and evolutionary generalization. IEEE Transactions on Evolutionary Computation 6, 431–442 (2002)
[8] Uy, N.Q., Hien, N.T., Hoai, N.X., O'Neill, M.: Improving the Generalisation Ability of Genetic Programming with Semantic Similarity based Crossover. In: Esparcia-Alcázar, A.I., Ekárt, A., Silva, S., Dignum, S., Uyar, A.Ş. (eds.) EuroGP 2010. LNCS, vol. 6021, pp. 184–195. Springer, Heidelberg (2010)
[9] Panait, L., Luke, S.: Methods for Evolving Robust Programs. In: Cantú-Paz, E., Foster, J.A., Deb, K., Davis, L., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Kendall, G., Wilson, S.W., Harman, M., Wegener, J., Dasgupta, D., Potter, M.A., Schultz, A., Dowsland, K.A., Jonoska, N., Miller, J., Standish, R.K. (eds.) GECCO 2003. LNCS, vol. 2724, pp. 1740–1751. Springer, Heidelberg (2003)
[10] Paris, G., Robilliard, D., Fonlupt, C.: Exploring Overfitting in Genetic Programming. In: Liardet, P., Collet, P., Fonlupt, C., Lutton, E., Schoenauer, M. (eds.) EA 2003. LNCS, vol. 2936, pp. 267–277. Springer, Heidelberg (2004)
[11] Vanneschi, L., Gustafson, S.: Using crossover based similarity measure to improve genetic programming generalization ability. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation (GECCO 2009), pp. 1139–1146. ACM (2009)
[12] Prechelt, L.: Early Stopping - But When? In: Orr, G.B., Müller, K.-R. (eds.) NIPS-WS 1996. LNCS, vol. 1524, pp. 55–69. Springer, Heidelberg (1998)

[13] Finno, W., Hergert, F., Zimmermann, H.: Improving model selection by nonconvergent methods. Neural Networks 6, 771–783 (1993)
[14] Zhang, B.T., Muhlenbein, H.: Balancing accuracy and parsimony in genetic programming. Evolutionary Computation 3, 17–38 (1995)
[15] Hooper, D., Flann, N.: Improving the accuracy and robustness of genetic programming through expression simplification. In: Proceedings of the First Annual Conference on Genetic Programming 1996, vol. 428. MIT Press (1996)
[16] Becker, L., Seshadri, M.: Comprehensibility and overfitting avoidance in genetic programming for technical trading rules. Technical report, Worcester Polytechnic Institute (2003)
[17] Liu, Y., Khoshgoftaar, T.: Reducing overfitting in genetic programming models for software quality classification. In: Proceedings of the Eighth IEEE Symposium on International High Assurance Systems Engineering, pp. 56–65 (2004)
[18] Silva, S., Vanneschi, L.: Operator equalisation, bloat and overfitting: a study on human oral bioavailability prediction. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation (GECCO 2009), pp. 1115–1122 (2009)
[19] Tuite, C., Agapitos, A., O'Neill, M., Brabazon, A.: Early stopping criteria to counteract overfitting in genetic programming. In: Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation, GECCO 2011, pp. 203–204. ACM, New York (2011)
[20] Hien, N.T., Hoai, N.X., Uy, N.Q., McKay, R.: Where should we stop? - an investigation on early stopping for gp learning. Technical Report TRSNUSC:2011:001, Strutural Complexity Laboratory, Seoul National University, Seoul, Korea (February 2011)
[21] Francone, F., Nordin, P., Banzhaf, W.: Benchmarking the generalization capabilities of a compiling genetic programming system using sparse data sets. In: Proceedings of the First Annual Conference on Genetic Programming 1996, pp. 72–80. MIT Press (1996)
[22] Iba, H.: Bagging, boosting, and bloating in genetic programming. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1053–1060. Morgan Kaufmann (1999)
[23] Paris, G., Robilliard, D., Fonlupt, C.: Exploring Overfitting in Genetic Programming. In: Liardet, P., Collet, P., Fonlupt, C., Lutton, E., Schoenauer, M. (eds.) EA 2003. LNCS, vol. 2936, pp. 267–277. Springer, Heidelberg (2004)
[24] Mahler, S., Robilliard, D., Fonlupt, C.: Tarpeian Bloat Control and Generalization Accuracy. In: Keijzer, M., Tettamanzi, A.G.B., Collet, P., van Hemert, J., Tomassini, M. (eds.) EuroGP 2005. LNCS, vol. 3447, pp. 203–214. Springer, Heidelberg (2005)
[25] Keijzer, M.: Improving Symbolic Regression with Interval Arithmetic and Linear Scaling. In: Ryan, C., Soule, T., Keijzer, M., Tsang, E.P.K., Poli, R., Costa, E. (eds.) EuroGP 2003. LNCS, vol. 2610, pp. 70–82. Springer, Heidelberg (2003)
[26] Gustafson, S., Burke, E.K., Krasnogor, N.: On improving genetic programming for symbolic regression. In: Proceedings of the 2005 IEEE Congress on Evolutionary Computation, vol. 1, pp. 912–919. IEEE Press, Edinburgh (2005)
[27] Vanneschi, L., Castelli, M., Silva, S.: Measuring bloat, overfitting and functional complexity in genetic programming. In: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation (GECCO 2010), pp. 877–884. ACM (2010)
[28] Shafi, K., Abbass, H.A., Zhu, W.: The Role of Early Stopping and Population Size in XCS for Intrusion Detection. In: Wang, T.-D., Li, X., Chen, S.-H., Wang, X., Abbass, H.A., Iba, H., Chen, G.-L., Yao, X. (eds.) SEAL 2006. LNCS, vol. 4247, pp. 50–57. Springer, Heidelberg (2006)
[29] Blake, C., Keogh, E., Merz, C.J.: UCI machine learning repository (1998)
[30] Vlachos, P.: Statlib project repository (2000)