AN EMPIRICAL STUDY OF EARLY STOPPING IN GENETIC PROGRAMMING

Nguyen Thi Hien[1], *Nguyen Xuan Hoai*[2], *Nguyen Quang Uy*[1], *Nguyen Thi Quyen*[3]

Abstract

Genetic Programming (GP) is an evolutionary method based on the principles of natural genetic systems. In GP, a population of individuals are randomly generated. This population is evolved through a number of generations by applying some genetic operators such as crossover, mutation and selection. Although GP has been successful applied to many real world problems, there are very few guidelines for determining when to stop GP algorithm. Traditionally, a predefined number of generation is set and GP is stopped when it reaches to the last generation. In this article, we present an empirical study of the impact of early stopping to GP performance. We propose some early stopping criteria for GP. We tested the proposed methods on a number of symbolic regression problems. Our experiment results show that using early stopping helps to maintain the generalisation capacity of GP while significantly reducing its solutions complexity and training time.

Lập trình di truyền (Genetic Programming -GP) là một phương pháp tiến hóa dựa trên các nguyên tắc của di truyền tự nhiên. Trong GP, quần thể của các cá thể được tạo ra ngẫu nhiên, quần thể này được tiến hóa qua một số thế hệ bằng cách áp dụng các toán tử di truyền như lai ghép, đột biến và chọn lọc. Mặc dù GP đã được dùng để giải quyết nhiều bài toán thực tế, song vẫn chỉ có rất ít nghiên cứu xác định thời điểm dừng của thuật toán GP. Chủ yếu theo truyền thống, GP sẽ dừng sau một số thế hệ được xác lập trước. Trong bài viết này, chúng tôi trình bày nghiên cứu thực nghiệm về tác động của việc dừng sớm lên GP. Chúng tôi đề xuất một số tiêu chí dừng sớm cho GP và đã thử nghiệm phương pháp đề xuất trên một số bài toán hồi quy ký hiệu. Kết quả thực nghiệm cho thấy rằng việc sử dụng dừng sớm giúp duy trì khả năng khái quát của GP trong khi giảm đáng kể độ phức tạp của lời giải và thời gian thực hiện.

Index terms

Genetic Programming, Early Stopping, Generalisation, Over-fitting.

1. Introduction

Genetic Programming (GP) is one of the mechanisms in providing solution to non-predefined complexity problems. GP performs a multidimensional search in providing

[1] Le Quy Don Technical University
[2] Hanoi University
[3] Vietnam National University of Forestry

an optimal solution for an user-defined problems. In GP, the population members are represented by trees, corresponding to the solutions of the problems. GP search starts with a population of randomly selected trees, and, the next generation is created by using genetic operators. At each iteration, individuals are evaluated with respect to a performance criteria and assigned a fitness value. The individuals with better fitness values have greater probability to be selected to mate to produce offspring for the next generation.

GP has also 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 [1]. Similar to popular machine learning techniques, one important issue in GP is over-fitting problem: while the errors on the training data may improve over the generations, it may deteriorate on unseen test data. GP over-fitting has been demonstrated in a number of publications [2], [3], [4]. While learning exact solutions may be important in some discovery tasks, machine learning research (ML) [5] has emphasized generalisation over unseen data as the most important aspect. A learning machine should avoid over-fitting the training data to generalize well on unseen data. Recently, GP generalisation has caught more attention, with an increasing number of related publications [6], [7], [8], [9], [10], [11].

In machine learning, there are often two popular strategies to combat over-fitting [12]. The first method uses regularisation techniques to reduce the complexity of the obtained solutions. This is based on Occam's razor principle [5] in which simple hypotheses is preferred. The second method is using early stopping techniques to finish learning process before over-fitting may happen. This method does not eliminate over-fitting but rather tries detect it and stops training once it does so.

Using early stopping technique to avoid over-fitting has been used in machine learning because it is simple, easy to implement, and, in many cases, superior to regularisation [13]. However, there are only two preliminary work on early stopping in GP that have been published [14], [15]. In [14], Tuite et al. adapted three stopping criteria used for training neural network [12] to Grammatical Evolution (GE). These criteria helped GE to detect when to stop during the training process. However their experiments only covered two simple problems, and the impact of early stopping was examined in details.

In this paper we propose and empirically investigate several early stopping criteria for GP. The remainder of the paper is organized as follows. In the next section, we briefly review related work on the role of early stopping in machine learning. Section 3

introduces five early stopping techniques that are examined in this paper. Sections 4 details the experimental settings. The experimental results are presented and discussed in Section 5. The paper concludes and highlights some possible future work in Section 6.

2. Early Stopping in Machine Learning

In machine learning there has been some research on examining the impact of early stopping to the improve the generation of learning system. In an early work early, Prechelt proposed the use of early stopping techniques to improve the generalization of neural network [12]. They introduced three criteria for stopping the training process of neural network when over-fitting is detected. 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].

After that, Shafi and Abbass used early stopping method to embody a trade-off between training time and generalisation [16]. They investigated the effects of early stopping in learning classifier system (LCS). Their results showed that using early stopping improves the generation ability of LCS as in neural network. A related work in the area is the research conducted by Garvesh Raskutti et al. [17] that applied a data-dependent stopping rule that does not involve hold-out or cross-validation data, and they proved upper bounds on the squared error of the resulting function estimate, measured in either the $L^2(\mathbb{P})$ and $L^2(\mathbb{P}_{\ltimes})$ norm. Their result was showed through simulation that their stopping rule compares favorably to two other stopping rules, one based on hold-out data and the other based on Stein's unbiased risk estimate. In GP, to the best of our knowledge, there are only two preliminary research of using early stopping. The first research is of Tuite et al. [14] and the second research is of ourselves [15]. In this paper, we propose some more early stopping techniques (exactly five criteria versus one in SEAL'paper) and examine their impact to the evolutionary process of GP. We also compare the efficiency of GP with early stopping to GP with Tarpeian which used to combat overfit in GP. In the next section, the proposed techniques will be presented in detailed.

3. Method

This section presents five early stopping techniques used in this paper. The first technique is based on the idea of over-fitting detection that was proposed and preliminarily investigated in [18]. The second technique is a new criterion proposed in this paper. This criteria is based on the correlation between training and validation error to decide stopping time. Three last techniques for early stopping are adapted from on those presented in [12] for training a neural network. In this case, we consider GP as machine learning technique so that using these early stopping criteria is feasible. They have also been used for GP in a preliminary study in [14]. All early stopping techniques use the error on validation set to detect over-fitting of GP systems. Before the training (evolutionary process) commences, the entire data set is divided into three subsets - training (50%), validation (25%), and test sets (25%) in random way. The training data set is used for evaluating the fitness of each individual in GP population. The validation data set is used, as the same as in [19], for detecting over-fitting during GP evolutionary runs. Finally, the test set is used to estimate the generalization capacity of GP over unseen data.

Algorithm 1: Measuring the over-fitting for GP systems [19].

over_fit(0)=0; bvp = Val_Fit(0);
$tbtp$ = Training_Fit(0);
foreach *generation* $g > 0$ **do**
  if *(Training_Fit(g)> Val_Fit(g))* **then**
    over_fit(g) = 0;
  else if *(Val_Fit(g)* $<$ *bvp)* **then**
    over_fit(g) = 0;
    bvp = Val_Fit(g);
    $tbtp$ = Training_Fit(g);
  else
    $over\_fit(g) = |Training\_Fit(g) - Val\_Fit(g)| - |tbtp - bvp|$;

To detect the over-fitting during a run of GP, we used the method proposed by Vanneschi et al. in [19]. The detail of this method is presented in Algorithm 1. In the algorithm, bvp is the "best valid point" and it stands for the best validation fitness (error on the validation set) reached until the current generation of a GP run, excluding the generations (usually at the beginning of the run) where the best individual fitness on the training set has a better validation fitness value than the training one; $tbtp$ stands for

"training at best valid point",i.e. the training fitness of the individual that has validation fitness equal to bvp; Training_Fit(g) is a function that returns the best training fitness in the population at generation g; Val_Fit(g) returns the validation fitness of the best individual on the training set at generation g.

The first technique of stopping criteria for GP aims to stop GP evolutionary process if it detects the over-fitting in m successive generations, where m is a tunable parameter. This stopping criteria is shorted as OF_m and is described in the following formula.

OF_m : stop after generation g when m successive generations where over-fitting happened up to g

$$overfit(g), .., overfit(g-m) > 0$$

This criterion only considers whether the over-fit phenomenon occurs of not. It does not take into account the degree of over-fitting. The next criteria will consider the degree of over-fitting to decide the stopping point for GP.

The second class of stopping criteria will stop when the over-fitting value is increased in m successive generations. This criterion is called VF_m and is detailed below:

$$overfit(g) \geq overfit(g-1) \geq ... \geq overfit(g-m) > 0$$

The termination of GP training process in the second criterion depends on the number of consecutive generations in which the over-fit value increases (more accurately, this value does not decrease). However, one of the possible limitations of this criterion is that it does not consider to the current generation of GP run. In the early GP evolutionary process (in some first generations), the desire for early stopping is not as strong as that in the late generations. In order to rectify this, we propose a variant of the second stopping criterion called the true adaptive stopping criterion and its work as in Algorithm 2.

Algorithm 2: True Adap Stopping

$proAp = \frac{(d-fg)*g}{(lg-fg)*MaxGen}$;
if $proAp \geq random$ **and** $d < m$ **then**
  $Stop = True$

where d is the number of generations where the over-fit value does not decreasing, fg is the generation started the chain of over-fit value not decreasing, lg is the last generation

of the chain of over-fit value not decreasing, $random$ is a random value in $[0, 1]$, g is current generation, $MaxGen$ is maximum number of generation)

Three last classes of early stopping criteria are adapted from those presented in [12] for training a neural network. In [12], Prechelt considered the validation set to predict the trend on test set, so that the validation error is used as an estimate of the generalization error in this paper.

The third class of stopping criteria will stop GP training process as soon as the generalization loss exceeds a certain threshold. The generalization loss at generation g is measured by Equation 1.

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

where$E_{opt}(g)$ is the lowest validation set error up to generation g and is calculated as follows.

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

in which $E_{va}(g)$ is the validation fitness value of the best individual of generation g. Based on the value of generation loss, the third criteria (shorted as GL_α) is defined as:

GL_α: stop after first generation g with $GL(g) > \alpha$

The fourth class of stopping criteria uses the quotient of generalization loss and progress.

In the third criteria, only the generation error is considered to decide whether or not to stop GP training process. However, if the training error still gets decreased rapidly, generalization losses may have a chance to be ”repaired”. The assumption is that over-fitting does not begin until the error on the training data decreases only slowly. The progress training is considered training strip of length k to be a sequence k generations numbered $g + 1, .., g + k$ where g is divisible by k. It measures how much the average training error during the strip larger than the minimum training error during the strip. It is formulated by Equation 3

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

Table 1. *The real-world data sets*

Data sets	Features	Size	Source
Concrete Compressive Strength	9	1030	UCI
Pollen	5	3848	StatLib
Chscase.census6	7	400	StatLib
No2	8	500	StatLib

Based on the calculation of the *progress* training. The four criteria is defined as follows.

PQ_α: stop after the first end of strip at generation g with $\frac{GL(g)}{P_k(t)} > \alpha$

In the experiment in this paper we take strips of length 5 ($k = 5$) as in the previous research [12].

The fifth class of stopping criteria will stop GP when the generalization error gets increased in s successive generations. This is based on the assumption that the increase in generation error indicate the beginning of final over-fitting.

UP_s: stop after generation g iff UP_{s-1} stops after generation $t - k$ and
$E_{va}(g) > E_{va}(g - k)$
UP_1: stop after first end of strip generation g with $E_{va}(g) > E_{va}(g - k)$

4. Experimental Settings

We conducted experiments on fourteen regression problems including both synthetic and real-world data sets. The four real-world data sets were chosen from the UCI machine learning repository [20] and StatLib [21], and are shown in Table *1*. The ten synthetic data sets are given in Table *2*. These functions have been extensively used in the GP and Machine Learning literature. Since these experiments are about generalisation ability of GP systems, we corrupted the output of the synthetic test functions by adding Gaussian noise with the variance $\alpha = 0.01$ to make them harder for GP to predict. 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. The data sets were then divided into training set, validation set and testing set. The number of samples in these three sets on each problems is also presented in Table *3*.

The parameter settings for the GP systems are shown in Table *4*. The number of generations for standard GP (GP) is set at 150. This is the typical values often used by GP researchers and GP practitioners [1]. For GP with stopping criteria, the stopping criterion

Table 2. 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	$F_{10}(x) = \frac{sin(\sqrt{x_1^1 + x_2^2})}{\sqrt{x_1^2 + x_2^2}}$

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	$Conc$		1030	515	257	258
12	$Poll$		3848	1924	962	962
13	$Cens$		400	200	100	100
14	$No2$		500	250	125	125

is checked at each generation. If the criterion is satisfied, then the GP training process is terminated. For each parameter's setting, 100 independent runs were conducted. Since we also examined run times, it is important to note that all runs (100 for each system on each problem) were conducted on a Compaq Presario CQ3414L computer with Intel Core i3-550 Processor (4M Cache, 3.20GHz) running a Ubuntu Linux operating system.

We divided our experiments into two sets. The first set aims to investigate whether the stopping criteria are sensitive to their parameters. To this end, we set the parameters of five stopping criteria as follows. For the first criterion (OF_m) and the second criterion (VF_m), three values of m (3, 9, 18) were tested. For the third criterion (GL_α) and the fourth criterion (PQ_α), α is set at $5, 7, 9$ and $20, 40, 60$ respectively. The last criterion (UP_s) was experienced with three values of parameter s and they are $2, 3, 4$. All parameters were selected from our calibrated experiments to highlight the performance of the stopping criteria.

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	100

The second set of experiments aims to analysed the performance of five stopping criteria and compared them with standard GP and Tarpeian GP of Mahler et at [22]. In this set, we selected the best parameter of each criterion obtained in the first set and compared that criteria with others. The results in two experimental sets will be described in detailed in the following section.

5. Results and Discussions

This section presents the results of two experimental sets in detailed. The results of five stopping criteria with various parameter are presented in the first subsection. The comparison between five stopping criteria, standard GP and Tarpeian GP is presented and discussed after that. In the last subsection, we analysed the stopping points (the generation where each GP system is terminated) of each stopping criteria.

5.1. Parameter Analysis of Stopping Criteria

For each run in five stopping criteria, we recorded the generalization error (GE – measured on the test data set) of the best individual on the training data of the run. This value was then averaged over 100 runs and they are presented in Table *5* and Table *6*. It can be seen from table *5* that both OF_3 and VF_3 perform inefficiently. The reason could be that with this value of m, these two criteria force GP stopping too early. Consequently, GP has not learnt enough to perform well on unseen data. For *True Adap* its performance is roughly equal to the performance of VF_9 and VF_{18}. The benefit of using *True Adap* is that it eliminates the need of selecting the parameter m.

Table 5. Generalization error of VF, OF, True Adap stopping criterion

	$Stopping1$			$Stopping2$			
Problems	OF_3	OF_9	OF_{18}	VF_3	VF_9	VF_{18}	True Adap
F_1	0.025	0.008	0.008	0.013	0.007	0.008	0.008
F_2	0.062	0.023	0.012	0.046	0.013	0.011	0.012
F_3	0.043	0.005	0.003	0.023	0.003	0.003	0.003
F_4	0.541	0.514	0.493	0.526	0.514	0.493	0.498
F_5	3.872	1.305	1.126	3.140	1.308	1.106	1.285
F_6	3.106	1.742	1.556	2.747	1.695	1.454	1.459
F_7	7.223	4.553	3.612	7.082	4.237	3.411	3.260
F_8	0.158	0.132	0.127	0.152	0.132	0.126	0.126
F_9	0.439	0.177	0.149	0.361	0.181	0.147	0.149
F_{10}	0.085	0.083	0.081	0.085	0.082	0.079	0.079
$Conc$	10.420	6.950	6.688	8.910	6.693	6.720	6.704
$Poll$	2.168	1.795	1.835	1.952	1.784	1.727	1.704
$Cens$	1.719	1.427	1.254	1.623	1.421	1.259	1.259
$No2$	0.567	0.503	0.491	0.549	0.500	0.481	0.477

Table 6. Generalization error of GL, PQ and UP stopping criterion

	$Stopping3$			$Stopping4$			$Stopping5$		
Functions	GL_5	GL_7	GL_9	PQ_{20}	PQ_{40}	PQ_{60}	UP_2	UP_3	UP_4
F_1	0.015	0.015	0.012	0.009	0.009	0.009	0.017	0.011	0.011
F_2	0.033	0.030	0.019	0.013	0.013	0.014	0.042	0.023	0.023
F_3	0.008	0.007	0.006	0.005	0.005	0.005	0.010	0.006	0.004
F_4	0.506	0.493	0.484	0.504	0.504	0.504	0.518	0.504	0.496
F_5	2.655	2.037	1.472	1.189	1.184	1.182	1.700	1.344	1.215
F_6	2.406	2.219	1.929	1.571	1.570	1.537	1.978	1.534	1.486
F_7	4.857	4.563	4.386	4.042	3.938	3.893	5.292	4.246	4.036
F_8	0.212	0.189	0.167	0.142	0.141	0.141	0.204	0.153	0.139
F_9	0.348	0.284	0.224	0.158	0.158	0.158	0.231	0.177	0.160
F_{10}	0.081	0.080	0.079	0.081	0.082	0.082	0.084	0.082	0.080
$Conc$	13.149	12.887	12.449	7.541	6.966	6.935	9.686	8.187	7.384
$Poll$	1.714	1.713	1.713	1.743	1.740	1.740	1.881	1.793	1.784
$Cens$	1.485	1.444	1.276	1.247	1.253	1.248	1.542	1.449	1.393
$No2$	0.516	0.527	0.498	0.482	0.482	0.481	0.512	0.487	0.479

Table *6* shows that GL_7 and GL_9 are approximately equal and their generation error is smaller than that of GL_5. Similarly, we can see that the performance of UP_3 and UP_4 is better than that of UP_2 regarding to the generalization error. Two tables also show that only the four criterion, PQ_α, is not very much sensitive to its parameter. It can be seen from Table *6* the generation error of PQ_{20}, PQ_{40} and PQ_{60} are mostly equal.

Overall, the results in this subsection show that the performance of all stopping criteria excepts the four criterion (PQ_α) is rather sensitive to their parameter. However, we can experimentally determine the good values of the parameter in each criterion and these values are relevant for all tested problems in this paper.

5.2. Comparing Stopping Criteria with Standard GP and Tarpeian GP

This section presents a comparison between the best configuration of the five stopping criteria. For each stopping criteria we selected the best configuration found in the previous subsection. Precisely, OF_{18}, VF_{18}, GL_9, PQ_{60}, and UP_4 are used for comparison in this subsection. The best configuration of each stopping criteria is compared with

Table 7. Generalization error of stopping criteria, GP and Tarpeian GP. If a method is better than standard GP, its result is printed bold faced

Functions	OF_{18}	VF_{18}	GL_9	PQ_{60}	UP_4	Tar	GP
F_1	0.008	0.008	0.012	0.009	0.011	0.008	0.010
F_2	**0.012**	**0.011**	0.019	**0.014**	0.023	**0.014**	0.022
F_3	**0.003**	**0.003**	0.006	0.005	**0.004**	0.006	0.006
F_4	0.493	0.493	0.484	0.504	0.496	0.488	0.496
F_5	1.126	1.106	1.472	1.182	1.215	1.179	1.234
F_6	1.556	1.454	1.929	1.537	1.486	1.613	1.468
F_7	3.612	**3.411**	4.386	3.893	4.036	**3.311**	4.065
F_8	0.127	0.126	0.167	0.141	0.139	0.138	0.131
F_9	0.149	**0.147**	0.224	0.158	0.160	0.153	0.163
F_{10}	0.081	0.079	0.079	0.082	0.080	**0.082**	0.080
$Conc$	6.688	6.720	12.449	6.935	7.384	7.343	6.891
$Poll$	1.835	1.727	1.713	1.740	1.784	1.655	1.721
$Cens$	**1.254**	**1.259**	**1.276**	**1.248**	1.393	1.376	1.348
$No2$	0.491	0.481	0.498	0.481	0.479	0.480	0.474

standard GP and Tarpeian GP. Tarpeian GP is an approach to reduce the code growth in GP [22]. The detailed description of Tarpeian GP is presented in Algorithm 3. The $target\_ratio$ is set at 0.3 in this paper.

Algorithm 3: A Tarpeian Wrapper

$Function : Evaluation(program)$
begin
  if $size(program) > average\_pop\_size$ **and** $random < target\_ratio$ **then**
    return very_low_fitness;
  else
    return fitness(program);

In order to compare the efficiency of the above GP systems, we used three metrics. The first metric is the generation error of each GP system. The second metric is the size of the best individual found by each method and the last metric is their computational time. The generation error of five stopping criteria, standard GP (GP) and Tarpeian GP (Tar) is presented in Table *7* In this table, if the performance of a method is significantly better than standard GP (using Student t-test with a confident level of 95%), its result is printed bold faced.

It can be seen from Table *7* that most stopping criteria performs better than standard GP. Apparently, the generation error of stopping criteria is often smaller than that of standard GP. On some problems, stopping GP system can significantly improve the generation error. The statistical test results show that all stopping criteria are significantly better than standard GP on some problems. Among five criteria, the table shows that FV_{18} is the best criterion and it is significantly better than standard GP on five out of fourteen problems. Comparing between stopping GP systems and Tarpeian GP, the table

Table 8. *Size of the best individual of stopping criteria*

Functions	OF_{18}	VF_{18}	GL_9	PQ_{60}	UP_4	Tar	GP
F_1	128.7	128.7	101.3	109.66	138.65	101.1	143.8
F_2	135.6	136.1	109.1	115.4	149.8	119.9	157.1
F_3	134.6	134.9	105.4	111.6	**142.51**	100.6	145.1
F_4	68.7	72.1	85.8	60.5	79.4	61.3	80.2
F_5	147.6	150.4	134.6	138.76	**158.8**	113.5	158.7
F_6	163.1	174.9	148.7	163.5	**183.4**	134.9	185.5
F_7	162.7	**171.4**	136.1	156.8	**176.6**	152.9	172.3
F_8	188.1	**187.9**	151.2	171.1	**198.4**	146.4	183.8
F_9	127.7	130.1	95.4	115.6	**140.4**	102.2	138.0
F_{10}	111.1	134.6	136.6	77.0	143.4	114.8	163.8
$Conc$	215.0	216.1	49.3	205.7	16.6	161.3	225.5
$Poll$	120.0	136.1	151.5	130.4	167.2	131.1	174.4
$Cens$	108.9	125.2	99.6	88.3	123.72	104.3	151.7
$No2$	66.2	115.0	83.6	98.0	124.76	111.9	154.0

shows that stopping GP systems are often slightly better than Tarpeian GP. However, this table also shows that the difference between the generation error of all GP systems is only marginal.

The second metric is the size of the best solution found by each GP system. In each run, the best individual on the training data is selected. The size of this individual (the number of tree nodes) was recored. This values was averaged over 100 runs and they are presented in Table *8*. This table that all stopping GP systems except UP_4 help to find simpler solutions compared standard GP. For UP_4, the size of its solution is roughly equal to the size of the solution found by GP on most problems. Conversely, the size of the best solution found by other applying stopping criteria to GP is often much smaller than that of GP and it is approximately equal to the size of the solutions achieved by Tarpeian GP.

We also statistical tested the difference between the results of stopping GP systems, Tarpeian GP and the results of standard GP in Table *8* using a Student t-test. The statistical test showed that most of the different is significant at the confident level of 95%. Only on some configuration of UP_4 and VF_{18}, the size of the best solution found these stopping criteria is equal to that of standard GP. In Table *8*, when the size of the best solution of early stopping GP and Tarpeian GP is not significantly different from the size of the solution found by standard GP, this result is printed bold faced.

The last metric used for comparing between stopping GP system and standard GP is their computational time. All systems were conducted on the same computer system of Intel Core i3-550 Processor (4M Cache, 3.20 GHz) and running Ubuntu Linux operating system. For each run, we recorded the execution time in seconds. This value is averaged over 100 runs and presented in Table *9*. It can be seen from this table that applying stopping criteria often lead to reducing execution time of GP system. The table shows that on many problems, the computational time of stopping GP system is smaller than

Table 9. Run time of stopping criteria

Functions	OF_{18}	VF_{18}	GL_9	PQ_{60}	UP_4	Tar	GP
F_1	486.5	486.7	325.2	377.2	500.1	363.0	528.9
F_2	454.6	459.1	308.8	319.2	453.8	380.1	488.2
F_3	427.2	428.0	226.1	264.5	422.6	316.4	462.0
F_4	166.5	184.9	231.0	176.3	217.9	176.3	212.4
F_5	514.6	527.1	408.1	426.9	556.8	462.3	563.1
F_6	520.0	1128.5	799.0	950.6	1173.6	897.1	1225.5
F_7	660.2	736.3	486.4	630.9	793.3	651.9	797.2
F_8	700.6	705.1	447.0	573.2	733.3	747.0	573.8
F_9	626.0	665.1	379.3	482.3	698.6	482.3	712.5
F_{10}	220.6	403.3	434.4	272.0	454.9	394.7	565.7
$Conc$	1728.4	1750.0	120.3	1549.1	1531.6	1736.4	1708.8
$Poll$	1995.8	1749.9	2879.6	2059.5	2895.7	2474.0	3113.6
$Cens$	206.0	329.1	201.1	164.1	283.9	308.0	427.3
$No2$	175.0	393.7	198.1	260.5	379.7	404.1	567.2

that of standard GP. In general, the results in this subsection show that, most early stopping criteria examined in this paper help GP to find simpler solutions and also slightly better generation error. They also aid to reduce execution time of GP systems. The results also show that the new techniques proposed in this paper, VF_{18}, is the best techniques among five tested stopping criteria.

5.3. Analysis of Stopping Point of Criteria

Since the objective of applying early stopping criteria is to early terminate GP training process, it is important to examine if they achieve this objective during the evolutionary process of GP. In order to investigate this, we selected the best configuration of each criteria and recorded the generation in each GP system where the training process is finished. The generation in which each GP system is terminated on two problems (Pollen and Cons) of five criteria is presented in Figure *1* [1].

It can be seen from Figure *1* that all stopping GP systems were stopped earlier than standard GP. It should be noted that standard GP is always finished after 150 generation and this is presented by the black dots at the top of the figure. For five stopping criteria, it can be seen that UP_4 usually stops later than four other criteria. Apparently, the stopping generations of UP_4 are very close to the last generation of standard GP. This explain UP_4 why the solutions found by UP_4 are often as large as those found by standard GP. Four other stopping criteria often help GP to finish much earlier than standard GP. Particularly, OF_{18} is the criteria that finished earliest. In most run, OF_{18} often finished before generation 80. Overall, the results in this section show that most stopping criteria achieved their objective in terminating GP systems earlier. Consequently, they could find simpler solutions and also reduce execution time of GP systems.

[1]We also drawn the figure for other problems and their results are similar to the results in this figure.

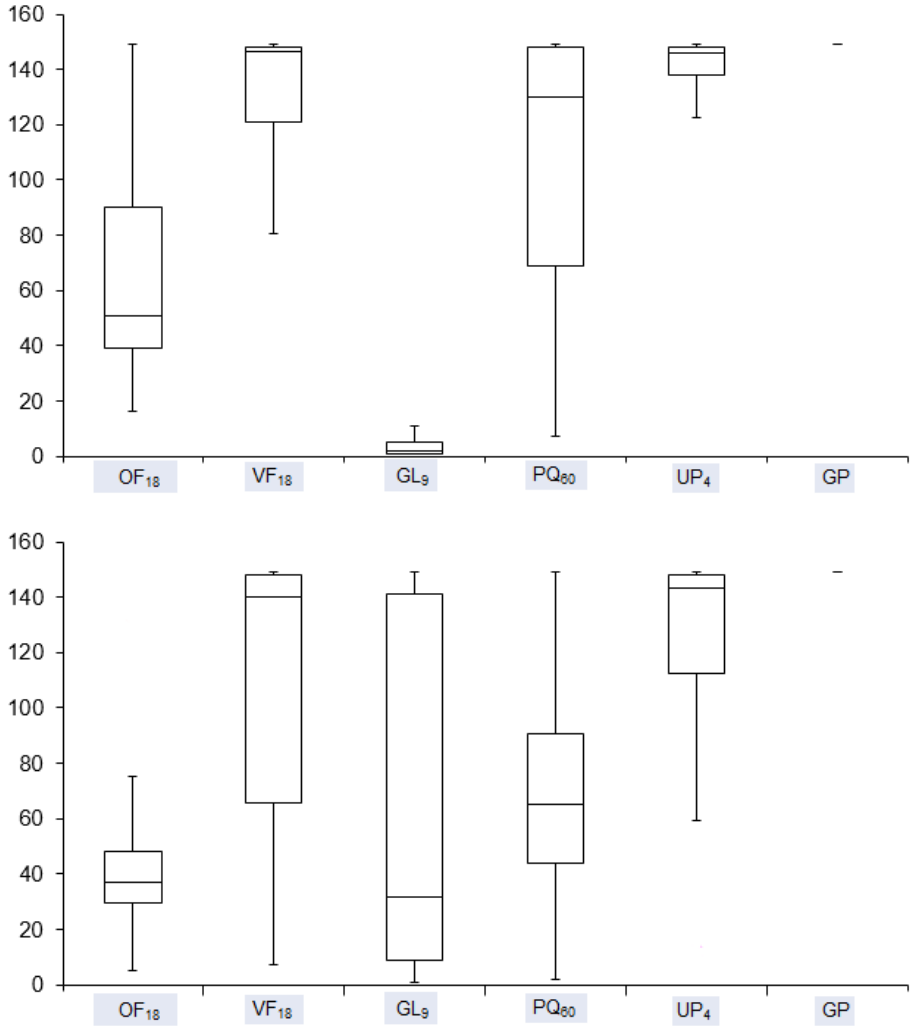

Fig. 1. *Distribution of stop generation of the stopping criteria and GP on Pollen and Conc problems*

6. Conclusion and Future Work

This paper presented an empirical study of early stopping in Genetic Programming (GP). We introduced and tested five stopping criteria on a number of symbolic regression problems. among five stopping criteria, two were proposed in this paper and three were adapted from those for using in neural network. The results of early stopping techniques were compared with standard GP and Tarpeian GP and showed that most early stopping criteria helped GP to find simpler solutions, reduce computational time and subsequently improve the generalization ability of GP system. Moreover the results also showed that one of the new stopping techniques proposed in this paper (VF_{18}) was often the best

techniques among all tested criteria.

There are some research areas for future work which arise from this paper. First, we would like to test the early stopping criteria in this paper on more difficult problems. The early stopping techniques examined in this paper helped GP to find simpler solutions and reduce execution time.However, their improvement of generalization error is not remarkable. One of the reason could be that the problems tested in this paper are still early for standard GP to learn and the over-fitting of standard GP is not significant. On more difficult problems, standard GP may accumulate more over-fitting and early stopping may be more important on those difficult problems. In the future, we aim to examine early stopping techniques on more difficult problems particularly on the real world problems. Second, at the theoretical level, we would like to study and apply optimal stopping theory in mathematics to GP [23].

Acknowledgment

This work was funded by The Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.01-2014.09.

References

[1] R. Poli and W. L. N. McPhee, *A Field Guide to Genetic Programming*. http://lulu.com, 2008.

[2] G. Paris, D. Robilliard, and C. Fonlupt, "Exploring overfitting in genetic programming," in *Evolution Artificielle, 6th International Conference*, ser. Lecture Notes in Computer Science, vol. 2936. Springer, 2003, pp. 267–277.

[3] U. N. Quang, T. H. Nguyen, X. H. Nguyen, and M. O'Neill, "Improving the generalisation ability of genetic programming with semantic similarity based crossover," in *Proceedings of the 13th European Conference on Genetic Programming, EuroGP 2010*, ser. LNCS, vol. 6021. Istanbul: Springer, 7-9 Apr. 2010, pp. 184–195.

[4] I. Kushchu, "Genetic programming and evolutionary generalization," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 5, pp. 431–442, Oct. 2002.

[5] T. M. Mitchell, *Machine Learning*. McGraw Hill, 1997.

[6] M. Amir Haeri, M. M. Ebadzadeh, and G. Folino, "Improving GP generalization: a variance-based layered learning approach," *Genetic Programming and Evolvable Machines*, vol. 16, no. 1, pp. 27–55, Mar. 2015.

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

[8] N. Foreman and M. Evett, "Preventing overfitting in GP with canary functions," in *Proceedings of the 2005 conference on Genetic and Evolutionary Computation (GECCO'05)*. ACM, 2005, pp. 1779–1780.

[9] C. Gagné, M. Schoenauer, M. Parizeau, and M. Tomassini, "Genetic programming, validation sets, and parsimony pressure," in *Proceedings of the 9th European Conference on Genetic Programming*, ser. Lecture Notes in Computer Science, vol. 3905. Springer, 2006, pp. 109–120.

[10] L. Panait and S. Luke, "Methods for evolving robust programs," in *Genetic and Evolutionary Computation – GECCO-2003*, ser. LNCS, vol. 2724. Chicago: Springer-Verlag, 12-16 Jul. 2003, pp. 1740–1751.

[11] L. Vanneschi and S. Gustafson, "Using crossover based similarity measure to improve genetic programming generalization ability," in *Proceedings of the 11th Annual conference on Genetic and evolutionary computation (GECCO'09)*. ACM, 2009, pp. 1139–1146.

[12] L. Prechelt, "Early stopping - but when?" in *Neural Networks: Tricks of the Trade, volume 1524 of LNCS, chapter 2*. Springer-Verlag, 1997, pp. 55–69.

[13] W. Finno, F. Hergert, and H. Zimmermann, "Improving model selection by nonconvergent methods," *Neural Networks*, vol. 6, pp. 771–783, 1993.

[14] C. Tuite, A. Agapitos, M. O'Neill, and A. Brabazon, "Early stopping criteria to counteract overfitting in genetic programming," in *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation*, ser. GECCO '11. New York, NY, USA: ACM, 2011, pp. 203–204.

[15] N. T. Hien, N. X. Hoai, R. McKay, and N. Q. Uy, "Where should we stop? - an investigation on early stopping for gp learning," in *Proceedings of the 9th Conference on Simulated Evolution And Learning (SEAL-2012)*. Springer-Verlag, 2012, pp. 391–399.

[16] K. Shafi and H. Abbass, "The role of early stopping and population size in xcs for intrusion detection," in *Proceedings of the 6th International Conference on Simulated Evolution and Learning*. Springer, 2006, pp. 50–57.

[17] G. Raskutti, M. J. Wainwright, and B. Yu, "Early stopping and non-parametric regression: An optimal data-dependent stopping rule," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 335–366, Jan. 2014. [Online]. Available: http://dl.acm.org/citation.cfm?id=2627435.2627446

[18] N. T. Hien, N. X. Hoai, N. Q. Uy, and R. McKay, "Where should we stop? - an investigation on early stopping for gp learning," Strutural Complexity Laboratory, Seoul National University, Seoul, Korea, Tech. Rep. TRSNUSC:2011:001, February 2011.

[19] L. Vanneschi, M. Castelli, and S. Silva, "Measuring bloat, overfitting and functional complexity in genetic programming," in *GECCO '10: Proceedings of the 12th annual conference on Genetic and evolutionary computation*. Portland, Oregon, USA: ACM, 7-11 Jul. 2010, pp. 877–884.

[20] C. Blake, E. Keogh, and C. J. Merz, "Uci repository of machine learning databases," Department of Information and Computer Science, University of California, Irvine, CA, 1998. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

[21] P. Vlachos, "Statlib project repository, http://lib.stat.cmu.edu," 2000. [Online]. Available: http://lib.stat.cmu.edu

[22] S. Mahler, D. Robilliard, and C. Fonlupt, "Tarpeian bloat control and generalization accuracy," in *Proceedings of the 8th European Conference on Genetic Programming, EuroGP'2005*, ser. LNCS, vol. 3447. Springer, 2005, pp. 203–214.

[23] G. Peskir and A. Shiryaev, *Optimal Stopping Theory and Free-Boundary Problems*, ser. Lectures in Maths. (ETH). Birkhauser, 2006.

Manuscript received 11-04-2016; accepted 14-12-2016.

■

Nguyen Thi Hien received her BSc Degree in Computer Science from Hanoi University of Science, MSc Degree in IT from School of Information Technology, Hanoi National University of Vietnam. She received the Ph.D. degree in Fundamentals of Mathematics for Informatics from the Le Quy Don Technical University, in 2014. Since 2002, she is lecturer at Faculty of IT, Le Quy Don Technical University. Her research interest are in the domain of Evolutionary Algorithms , Genetic Programming and Machine Learning.

Nguyen Quang Uy received his PhD degree in computer science from University College Dublin, Ireland. He is currently a senior lecturer at the Faculty of IT, Le Quy Don Technical University. His research interests include evolutionary algorithms, machine learning, natural language processing, computer vision and network analysis.

Nguyen Xuan Hoai is currently the director of IT R&D center. His research interests include Evolutionary Learning, Statistical Machine Learning. He is an editorial board member of Genetic Programming and Evolvable Machines journal.

Nguyen Thi Quyen received her BSc Degree in Mathematics from Hanoi National University of Education, MSc Degree in Mathematics from Hanoi University of Science, Hanoi National University of Vietnam. Since 2005, she is lecturer at Vietnam National University of Forestry. Her research interest are in the domain of Difference Equations, Numerical Analysis and Machine Learning.