Noname manuscript No.
(will be inserted by the editor)

? Title of paper

Thuong Pham Thi · Hai Nam Le · Xuan Hoai Nguyen · Anthony Brabazon

Received: date / Accepted: date

Abstract In this paper, we investigating the effects of noise to generalization and over-fitting in GP and answer the question: how to built a suit of benchmarks whose difficult can be controlled by over-fitting for researching the relationship between the generalization and the over-fitting in GP. We run experiments on 14 Symbolic Regression benchmark problems. These problems are added noise with three types of noise, each type of noise corresponds to three levels of noise. The experimental results give us the conclusions that GP is a very sensitive method to noise; and that each type of noise and level of noise will different effect to over-fitting ability of GP. Moreover, it also gives us a suit of 140 generated data sets added noise. To group these data sets to clusters that can control degrees of difficult by over-fitting, we propose a new measure to quantify the over-fitting of problems. So, next results of this paper are a suit of 140 Symbolic Regression benchmark problems grouped to four clusters by increase over-fitting and the new measure to quantify the over-fitting of problems in GP.

Keywords Genetic Programming · noise model · Benchmark Problems · overfitting

Thuong Pham Thi
Thai Nguyen University, Viet Nam
Tel.: ++84-0912-838-646
E-mail: ptthuong@ictu.edu.vn

Hai Nam Le
Hanoi Universuty, Viet Nam
E-mail: : namlehai90@gmail.com

Xuan Hoai Nguyen
Hanoi Universuty, Viet Nam
E-mail: : nxhoai@hanu.edu.vn

Anthony Brabazon
University College Dublin, Ireland
E-mail: : anthony.brabazon@ucd.ie

1 Introduction

Genetic Programming (GP) was developed by Koza [19] in 1992. Based on observations of biological systems, it uses an abstraction of Darwins natural selection mechanisms to evolve population of solutions to problems [29]. GP can solve many tasks in various areas, in which one of main tasks is Machine Learning.

In Machine Learning, generalization and over-fitting are the two most important issues. They can be seen as two sides of the same coin should be addressed simultaneously. There are many researches in GP on these issues as [15][12][8][24][26][33][25]. However, in most of these researches authors used testing problems chosen unsystematically. In particular, so far there is any benchmark for researching generalization related to over-fitting issue in GP. Moreover, no free lunch (NFL) theorems give us that no algorithm that performs well on all classes of problems [37]. Therefore, it is very necessary to have benchmark problems with various difficulty levels can be controlled by over-fitting.They will make it easier for researchers test generalization and scalable abilities of algorithms over a wide range of problems with different difficulties and complexities [13] . So, one of the goals in this paper is to build a such suite of benchmarks, within the domain of symbolic regression, especially they are grouped into different clusters whose difficult levels controlled by over-fitting.

One of causes of over-fitting is noise in real-world training data [39]. In Machine Learning, researchers proposed many noise models with various noise types and noise levels to simulate actual noise in data and to make over-fitting challenges for learners. They have shown that the different noise types, noise levels will make the various effects on over-fitted or generalization ability of learners [28][11][18][2][6][5][3][21][7]. However, in GP, to the best of our knowledge, there has not been any thorough research on this issue before. Therefore, the second goal of this paper is studying effects of noise to over-fitted and generalization ability of leaner in GP fully and systematically. The following are the main contributors in this paper:

- Propose a new method to quantify the over-fitted in GP. This is the first work need to be done. It is the base for controlling the difficult of benchmark by over-fitting. Because the measure "the amount of over fit"proposed by Vanneschi usually use in GP has a few limits (see the section 2.2). So, we propose the new method to improve these limits.
- Survey some noise models in Machine Learning and investigate the impacts of noise to over-fitted and generalization ability of solutions in GP.
- Provide a suite of 140 instances of symbolic regression benchmarks with various types of noise, levels of noise grouped into clusters by increasing difficult levels.

The remainder of this paper is organized as follows. In the section 2, we briefly present the background and related works include: some brief surveys of GP benchmarks, the noise models in Machine Learning; some issues about how to quantify the amount of over-fitting of solution in GP. Next, noise

configurations used for experiments in this study; a new method to quantify the over-fitting of benchmark is presented in the section 3.2. The experimental settings are given in the section 4. The section 5, we show results and the discussions. Finally, the paper is closed by summary achieved research results and some future works.

2 Background and related works

2.1 GP benchmarks

Building a set of benchmarks for GP along with a measure of the difficulties has been recently received much attention from researchers. Devising a good benchmark suite would be recognized as an important issue for the next ten years of GP [22].

In [14] authors presented a list of symbolic regression functions with the aim at collecting the opinions of GP community about it. Next, in [36] authors presented the results of a community survey regarding this list. In [13] authors introduced benchmark criteria and listed a suit of instances satisfies these criteria. They also shown how to determine the difficulty of some specific problems, but not clear (Ex: Royal Tree problems, Order and Majority, ... their difficulty can be changed by modifying some values of parameters while Boolean problems are scalable in difficulty because they are self-contained, no requiring external data-set or domain knowledge). In [17] John and colleagues suggested that benchmarks should not be selected arbitrarily but instead should be drawn from an underlying probability distribution that must reflect the problem instances which the algorithm is likely to be applied to in the real-world. However, authors did not show how to quantify the difficulty level of problem. Michael ONeill and colleagues in [22] identified the problem difficulty relates not only to the issue of representation, but also to the choice of genetic operators and fitness functions. A fitness landscape metaphor can be helpful to understand the difficulty of a problem [30][38]. So, in [34][32] authors introduced two measures of problem hardness for GP, based on the concept of fitness landscape: fitness-distance correlation (FDC) and negative slope coefficient (NSC). The NSC succeeded in correctly quantifying the difficulty of some GP benchmarks and real-life problems [35]. The dissimilarity measure defined in [9] worked by calculating the probability of correctly applying the operator once. Clodhna Tuite and and colleagues explore for a design of a dynamic benchmark for symbolic regression, based on semantic distance between evaluated functions [4].

Most recent, Miguel Nicolau and colleagues shown seven issues concerning the design of artificial functions as symbolic regression benchmarks and provided a set of guidelines for careful definition of these functions. Besides, he also pointed out that the size of the data set can affect the distribution of response variable and it also can be used to control the difficulty of the problem. A larger sample almost always leads to better test performance of

the learned model. In other words, the smaller size of the training set, more difficult problem to learn [27]. Thomas Helmuth and Lee Spector presented a suite of 29 benchmarks selected from sources of introductory computer science programming problems for GP. These benchmarks vary in difficulty and can be useful for assessing the capabilities of a program synthesis system [31].

Researchers mainly focus on the questions: how to develop a set of benchmarks for GP and how to determine the difficulty of the problem. So far, most of these benchmarks only use for learning exact solutions. There is not any a suit of benchmarks whose difficult controlled by over-fitting for aim at researching generalization ability of learner. So, in this paper we build a such suite by adding noise into benchmarks systematically.

2.2 Quantifying the amount of over-fitting in GP

The over-fitting issue is open encountered in all supervised Machine Learning schemes. This is one of the main causes to loss of generalization power of solutions. To study the generalization ability of solutions relate to over-fitting of benchmark problems explicitly, first, we need to built a measure to quantify over-fitting of problems. In Machine Learning, there are some researches introduced the formula to quantify the amount of over-fitted of solutions as [10][1]. Especially in GP, researches on this issue is lacking. In [20], Silva and colleagues proposed a method to calculate "the amount of over-fitting"of solutions in GP (details see in [20]). However, this method failed to capture the best test point (btp) in the case as shown in the Figure 1.a). In this case, actual best test point (btp_a) is found to be different from expected point (btp_e) that need to be captured. Therefore, at generations on the red graph (from btp_a to btp_e), despite to be over-fitted, the amount of over-fitting is not calculated. This is the context has not been covered in the formula of Silva. So, we propose the new method to solve this issue.

2.3 Noise models in ML and GP

In this section, we introduce a number of noise models used in Machine Learning and in GP. Then, we choose one from them for our experiments and explain why we choose it.

As shown in [5], two main sources of noise: (1) error in measurement of sensors; (2) the random error is created when batch processes or experts gather data. Authors also described three characteristics of the process generating noise: (1) the distribution of noise: noise is often drawn in an random manner and assumed under the Gaussian distribution or the Normal Distribution corresponding to numerical attributes or categorical attributes; (2) where the noise is introduced: noise can be found in four situations as shown in the Table 1; (3) the magnitude of the generated noise values: the magnitude of noise is often relative to each variable and within the limits of this variable data domain.

Table 1 Possible situations of noise in data sets

Input data	Output
Noise in train	Noise in test
Noise in test	
Noise in train and test	

In [28] Mark introduced a noise model to add noise into each state variable in Reinforcement Learning as following: assume the noise level was set to k% then the state value at the next time step would be $x + random(-x * k, x * k) + random(-0.0001, 0.0001)$, where x is value at current time step. Adding noise is done easily with this model. However, it is naive and do not reflects the characteristics of the noise fully. So, Mark introduced the second noise model, also used in [18], with two assumptions are considered to be true for all data sets: (1) the variables of the data set (both the independent and the dependent variables) are normally distributed; (2) noise is randomly distributed and independent from the data. Noise is generated as follows:

For every case (y_i, x_i) in the data set L: The pair (y_i, x_i) of the dependent variable Y and the matrix of independent variables X is substituted by another (y_i, x_i) by a probability of n, where n is the noise level. The new pair is calculated by the following formula:

$$x_{ij}^{'} = \begin{cases} x_{ij} + \sigma_{x_j} z_j & if\ p_{ij} \geq n; \\ x_{ij} & if\ p_{ij} \leq n. \end{cases} \quad y_i^{'} = \begin{cases} y_i + \sigma_y z_j & if\ p_{ij} \geq n; \\ y_i & if\ p_{ij} \leq n. \end{cases}$$

$z_{ij} = norminv(p_{ij}), j \in [1, .., k]$

where: σ_{x_j} is the standard deviation of x_j; z_j is a normally distributed random variable and is calculated by the inverse function of density-probability of the normal distribution for a value of p_{ij}, having a mean value of zero and a standard deviation equal to unity; $p_{ij} \in (0, 1)$ is a probability variable produced by a random value generator following a uniform distribution.

However, in many cases, the assumption of underlying normal distribution of variables in the data set is not satisfied. So, in [5] David and colleagues introduced how to generate artificial noise (without these assumptions) as following: for each variable x, repeat $N * k$ times four steps include: (1) Randomly generate a value R_x with a Gaussian distribution and lying in the range of values of x; (2). Choose a instance R_i between 1 and N, with a uniform distribution randomly; (3) Make sure R_i has not been chosen previously selected. If it has return to Step 2; (4) Assign R_x to R_i. Where, the magnitude of noise is within a range between min and max of the corresponding variable, k is a noise level, n is the size of data set for the data set. This model assumed that noise is under the Gaussian distribution which is the common assume used in noise models. It is the noise model referenced widely in Machine Learning algorithms [5]. So in this paper, we use this noise model for our experiments.

3 Methods

3.1 Generating noise

It is interesting to study different perspectives of noise presence, but in this paper, we only focus on the problem of learning from noisy training instances. In general, noise in the training data set is found to give most of difficulty to learners [3]. As introduced in the section 2.3, we use the noise model introduced in [5] and just add noise into the original training set while the testing set is remained the same as the original. Noise is added to independent variables x (noise type x), response variable y (noise type y), and both x & y (noise type xy) with the noise levels: 0% (free Noise), 10% (low noise), 30% (medium noise) and 50% (high Noise). Some symbols with noise configurations as following:

- *F: training set with 0 % noise (the original data set)*
- *Lx: 10% noise x in the training data*
- *Mx: 30% noise x in the training data*
- *Hx: 50% noise x in the training data*
- *Ly: 10% noise y in the training data*
- *My: 30% noise y in the training data*
- *Hy: 50% noise y in the training data*
- *Lxy: 10% noise xy in the training data*
- *Mxy: 30% noise xy in the training data*
- *Hxy: 50% noise xy in the training data*

A point worth noting here is distributions of the independent variables x and response variable y are not significantly changed after adding noise. The Table 2 shows p values obtained by comparing the difference in x, y to x, y added noise using Mann Whitney U Test use, with 95% confidence. All $pvalue \geq 0.05$ for all variables of all problems)

3.2 A new method to quantify the over-fitting

In this section we propose a new method to measure the over-fitting ability (OV) of benchmarks based on the measure "the amount of over-fitting"introduced by Silva and colleagues in [20]. The reasons and basis for proposing the new method are following:

First, As analyzed in the section 2.2, the measure of amount of over-fitting proposed by Silva failed to capture the best test point (btp) in some cases, which leads to errors in the assigning to the amount of over-fitting by 0 at some generations over-fitted.

Second, Silva has assumed using elitism in our GP runs, so $training\_fit(g) \leq tbtp$. the amount of over-fitting at generation g was calculated as follows (see more the Figure 1.b):

$$overfit(g) = |training\_fit(g) - test\_fit(g)| - |tbtp - btp|$$

Table 2 p values obtained by comparing the difference in x, y to x, y added noise using Mann Whitney U Test use, with 95% confidence

Problem	Var	Lx	Mx	Hx	Ly	My	Hy	Lxy	Mxy	Hxy
Kei10	x1	0.951	0.80	0.48				0.79	0.96	0.48
	x2	0.92	0.66	0.37				0.65	0.53	0.34
	y				0.65	0.72	0.57	0.99	0.69	0.75
Kei11	x1	0.97	0.35	0.47				0.70	0.79	0.47
	x2	0.78	0.89	0.19				0.95	0.98	0.19
	y				0.65	0.70	0.45	0.95	0.81	0.62
Kei12	x1	0.98	0.82	0.73				0.82	0.70	0.85
	x2	0.82	0.96	0.47				0.83	0.55	0.47
	y				0.86	0.85	0.92	0.69	0.96	0.28
Kei13	x1	0.67	0.49	0.62				0.78	0.63	0.61
	x2	0.62	0.79	0.24				0.90	0.75	0.24
	y				0.47	0.60	0.88	0.91	0.68	0.82
Kei14	x1	0.83	0.42	0.86				0.76	0.94	0.86
	x2	0.50	0.68	0.25				0.98	0.55	0.25
	y				0.92	0.77	0.84	0.85	0.39	0.55
Kei15	x1	0.89	0.70	0.99				0.66	0.76	0.99
	x2	0.65	0.80	0.47				0.72	0.69	0.47
	y				0.86	0.61	0.61	0.85	0.59	0.15
Vla1	x1	0.86	0.76	0.77				0.84	0.93	0.77
	x2	0.53	0.95	0.34				0.53	0.95	0.34
	y				0.73	0.68	0.93	0.75	0.23	0.89
Vla5	x1	0.95	0.43	0.80				0.75	0.95	0.80
	x2	0.94	0.63	0.39				0.98	0.89	0.39
	x3	0.95	0.96	0.59				0.92	0.91	0.59
	y				0.83	0.49	0.63	0.78	0.56	0.68
Vla6	x1	0.80	0.35	0.98				0.95	0.52	0.982
	x2	0.86	0.71	0.22				0.87	0.52	0.22
	y				0.59	0.82	0.80	0.83	0.69	0.98
Vla8	x1	0.98	0.89	0.74				0.86	0.86	0.739
	x2	0.68	0.94	0.48				0.57	0.72	0.48
	y				0.60	0.94	0.48	0.87	0.26	0.74
Nguyen_1	x	0.89	0.98	0.64				0.85	0.81	0.64
	y				0.82	0.85	0.83	0.76	0.96	0.78
Nguyen_2	x	0.72	0.98	0.37				0.95	0.87	0.375
	y				0.75	0.63	0.80	0.51	0.52	0.79
Nguyen_3	x	0.91	0.57	0.30				0.90	0.38	0.305
	y				0.82	0.70	0.88	0.47	0.77	0.60
Nguyen_4	x	0.68	0.56	0.72					0.93	0.72
	y				0.80	0.74	0.52	0.64	0.89	0.36

$$
\begin{aligned}
&= A'B' - AB \\
&= A'B' - MN \\
&= |test\_fit(g) - btp| + |training\_fit(g) - tbtp| \\
&= A'M + B'N \\
&= a_g + b_g
\end{aligned}
\tag{1}
$$

where: *btp* represents the best fittest error reached from when beginning of the run until to the current generation, where the best individual on the training

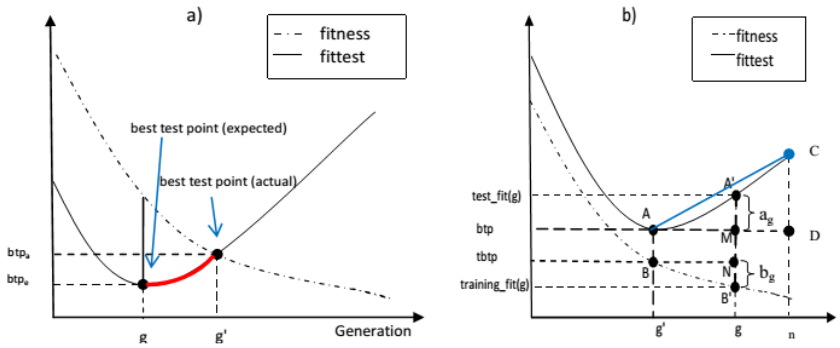

Fig. 1 a) issue in the algorithm calculates amount of over-fitting proposed by Silva; b) the measure of the amount of over-fitting

set has a better test than fitness error; *tbtp*: the fitness error of the individual that has fittest error equal to *btp*; $trainingfit(g)$ is a function that returns the best fitness error in the population at generation g; $testfit(g)$: returns the fittest error of the best individual on the training set at generation g.

Furthermore, [23], the over-fitting be defined as follows: *"Given a hypothesis space H, a hypothesis h H is said to overfit the training data if there exists some alternative hypothesis h H, such that h has smaller error than h over the training samples, but h has smaller error than h over the entire distribution of instances"*. It helps us locate over-fitting point and not recommended that the amount of over-fit should be a combination of training error and testing error. So,to exactly reflect the amount of over-fitting on the graph of fittest error, we propose to eliminate the quantity b_g in the formula (1), then, a new measure for calculating the amount of over-fitting at generation g will be:

$$overfit(g) = a_g \tag{2}$$

Third, we see that over-fitting can be can be characterized by early (or later) over-fitting and the amount of over-fitting. In other words, the these factors will make the characteristics of over-fitting. Over-fitting will increase if amount of the over-fitting increases and the over-fitting occur earlier.

From three above discussions, we proposed the new method that using the formula (3) as new measure of OV. Here, OV is considered as an square root of a half of area of the triangle ACD as shown in the Figure 1. b). The algorithm 1 presents details of this new method.

$$OV = \sqrt{(a_n * (n - btp))/2} \tag{3}$$

where : n is the number of generations; *btp* is best test point be found after scanning from generation 1 to generation n; a_i is the mount over-fitting at generation i; Avg_{AOO} is mean of ai with $i : btp, .., n$

Algorithm 1: Quantify the over-fitting

$overfit(0) = 0$
$btp = test\_fit(0)$
$tbtp = training\_fit(0)$
for $generationi = 1$ **to** $n - 1$
  if $(test\_fit(i) < tbp)$
    $overfit(i) = 0$
    $btp = test\_fit(i)$
    $tbtp = training\_fit(i)$
  else $overfit(i) = test\_fit(i) - btp$
$OV = \sqrt{overfit(n-1) * (n - btp)/2}$
return OV

4 Experimental setting

4.1 Benchmark problems

As shown in [27] not all functions are suitable to be used as benchmarks. Here, we choose Symbolic Regression functions from [14] as shown in the Table 3. Two reasons for selecting these functions are: (1) they did not receive any response from Genetic Programming community [36]; (2) they have various response variable distributions to ensure the best coverage. In the Figure 2), shows response variable distributions of these functions: Kei12, Val5, Vla6, Nguyen_1 are quite like Normal Distribution; functions like Kei10, Kei11, Vla1 with a few outliers in one tailed; Kei 15, Vla5 with more evenly outlier in both 2-tailed; functions as Kei14, Vla8, Nguyen_2, Nguyen_3, Nguyen_4 with much higher density of outliers and very skew making them very challenging.

Towards a framework for synthetic problem generation shown in [27], all data sets generated from chosen functions have common features: input range, how to sample for the training and testing set, the number of samples in the testing set are the same in [14]. However, because the size of training set could be used to analyze the difficulty of the problem [27]. So to control the difficulty of the problem only through the measurement OV, we generate the training set with the size of 200 samples for all functions. This size is recommended by Thomas and Lee Spector in [31]. They recommended that for most problems, training cases should be between 100 and 200, depending on the difficulty of the problem as well as the dimensionality of the input space.

4.2 GP Parameters Setting

The GP parameters used for our experiments are shown in the Table 4. These are typical settings often used by GP researchers and practitioners [19]. However, to stabilize the variance of the intermediate quantities in the individual tree we use an analytic quotient (AQ) operator [16] instead of the protected division (PD).

Table 3 GP Benchmark Regression Problems

Name	Definition	Training Data	Testing Data
Kei10	$x_1^{x_2}$	U[0, 1, 200]	E[0, 1, 0.01]
Kei11	$x_1 x_2 + \sin((x_1 - 10)(x_2 - 1))$	U[-3, 3, 200]	E[-3, 3, 0.01]
Kei12	$x_1^4 - x_1^3 + \frac{x_2^2}{2} - x_2$	U[-3, 3, 200]	E[-3, 3, 0.01]
Kei13	$\sin(x_1)\cos(x_2)$	U[-3, 3, 200]	E[-3, 3, 0.01]
Kei14	$\frac{8}{2+x_1^2+x_2^2}$	U[-3, 3, 200]	E[-3, 3, 0.01]
Kei15	$\frac{x_1^3}{5} + \frac{x_2^3}{2} - x_2 - x_1$	U[-3, 3, 200]	E[-3, 3, 0.01]
Vla1	$\frac{e^{(x_1-1)^2}}{1.2+(x_2-2.5)^2}$	U[0.3, 4, 200]	E[-0.2, 4.2, 0.01]
Vla5	$30\frac{(x_1-1)(x_3-1)}{x_2^2(x_1-10)}$	x_1 : U[0.05, 2, 200] x_2 : U[1, 2, 200] x_3 : U[0.05, 2, 200]	x_1 : E[−0.05, 2.1, 0.15] x_2 : E[0.95, 2.05, 0.1] x_3 : E[−0.05, 2.1, 0.15]
Vla6	$6\sin(x_1)\cos(x_2)$	U[0.1, 5.9, 200]	E[-0.05, 6.05, 0.02]
Val8	$\frac{(x_1-3)^4+(x_2-3)^3-(x_2-3)}{(x_2-2)^4+10}$	U[0.05, 6.05, 200]	E[-0.25, 6.35, 0.2]
Nguyen_1	$x^3 + x^2 + x$	U[-1,1,200]	U[-1,1,100]
Nguyen_2	$x^4 + x^3 + x^2 + x$	U[-1,1,200]	U[-1,1,100]
Nguyen_3	$x^5 + x^4 + x^3 + x^2 + x$	U[-1,1,200]	U[-1,1,100]
Nguyen_4	$x^6 + x^5 + x^4 + x^3 + x^2 + x$	U[-1,1,200]	U[-1,1,100]

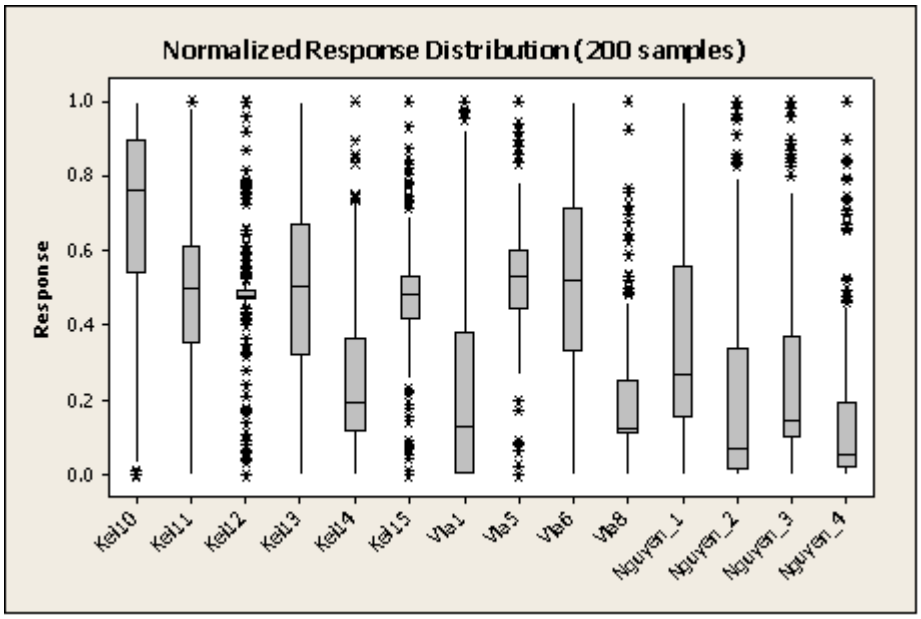

Fig. 2 Normalized response variable distribution over 200 samples for each function (response variable normalized by the formula in [27])

Table 4 Run and Evolutionary Parameters Values

Parameter	Value
Problems	Shown in the Table 3
EA used in GP systems	Elitist, generational, expression tree representation
Function set	+, -, *, / (AQ)
Terminal set	Regression variables and one random constant in [0.0, 1.0]
No. of generations	151
Population size	500
Tournament size	4
Tree creation	Ramped half-and-half (depths of 2 to 6)
Max. tree depth	15
Sub tree crossover rate	0.9
Sub tree mutation rate	0.1
No. of Runs	51
Fitness function	RMSE

Table 5 Mean of OVs of the best model trained on generated data sets using the formula (3)

	F	Lx	Mx	Hx	Ly	My	Hy	Lxy	Mxy	Hxy
Kei10	0	0.01	0.01	0.30	0.01	0.26	0.14	0.00	0.09	0.37
Kei11	0	1.89	2.44	0.91	4.40	4.82	2.88	3.99	0.71	1.82
Kei12	0	0.18	4.96	9.31	18.70	20.34	15.10	0.21	2.03	11.13
Kei13	0	0.00	0.09	0.31	0.04	0.09	0.13	0.24	0.06	1.04
Kei14	0	0.10	0.43	0.51	0.65	0.82	0.68	0.01	0.12	0.36
Kei15	0	0.00	0.02	0.71	0.28	1.21	1.96	0.33	1.74	1.00
Vla1	0	0.04	0.09	3.70	0.04	0.43	0.16	0.19	0.24	0.30
Vla5	0	0.26	0.25	0.30	0.00	0.05	0.81	0.13	0.41	0.59
Vla6	0	0.00	0.10	0.34	0.04	0.08	0.06	0.00	0.00	0.52
Vla8	0	0.01	0.10	0.44	0.00	0.17	0.11	0.00	1.10	0.11
Nguyen_1	0	1.12	0.40	0.43	1.28	0.84	1.10	0.00	0.00	0.37
Nguyen_2	0	0.65	0.05	1.03	0.05	1.40	2.29	0.28	1.75	0.39
Nguyen_3	0	0.00	0.38	1.30	1.37	1.15	2.79	0.21	0.14	1.02
Nguyen_4	0	0.15	0.08	3.19	4.95	0.19	4.12	0.78	2.63	2.72

5 Results and discussion

In this section we present the experimental results of running GP with all noise configurations for all chosen benchmarks. Experimental results include: OV of the best learned model as shown in the Table 5; generalization errors; complexities of the best model, all is averaged over 51 runs, as shown in the Table 6, the Table 7.

5.1 Grouping generated data sets by over-fitting (OV)

Table 5 shows 140 data sets grouped to four clusters by their OVs. Each cluster includes benchmarks with OVs are increasing. The clustering results are

Table 6 Fittest (mean) of the best learned model by GP with noise configurations

Problem	F	Lx	Mx	Hx	Ly	My	Hy	Lxy	Mxy	Hxy
Kei-10	0.05	0.08	0.12	0.17	0.07	0.10	0.16	0.07	0.16	0.20
Kei-11	0.56	0.91	1.66	2.33	0.96	1.18	1.56	1.03	2.47	2.70
Kei-12	1.83	24.0	31.2	45.8	18.8	39.2	40.4	14.7	45.2	56.0
Kei-13	0.67	1.41	2.05	2.34	0.96	1.68	2.19	1.35	2.60	2.70
Kei-14	0.10	0.22	0.32	0.59	0.19	0.27	0.42	0.35	0.56	0.67
Kei-15	0.73	0.97	1.68	2.44	1.43	1.46	2.55	1.39	2.56	3.19
Vla-1	0.07	0.07	0.10	1.86	0.09	0.11	0.12	0.08	0.13	0.19
Vla-5	0.34	0.44	0.66	0.62	0.36	0.47	0.65	0.53	0.75	0.74
Vla-6	1.99	2.10	2.54	2.66	2.09	2.22	2.39	2.19	2.45	2.80
Vla-8	1.12	1.59	1.89	2.17	1.33	1.51	2.13	1.61	2.10	2.26
Nguyen-1	0.00	0.12	0.28	0.32	0.13	0.30	0.49	0.12	0.45	0.61
Nguyen-2	0.01	0.06	0.17	0.47	0.11	0.29	0.72	0.17	0.42	0.50
Nguyen-3	0.01	0.05	0.29	0.47	0.13	0.44	0.80	0.24	0.64	0.68
Nguyen-4	0.02	0.14	0.65	0.93	0.82	0.58	1.09	0.54	0.71	1.49
Mean	0.53	2.30	3.11	4.51	1.96	3.56	3.97	1.74	4.37	5.34

Table 7 Model complexity (mean) of the best learned model by GP with noise configurations

Problem	F	Lx	Mx	Hx	Ly	My	Hy	Lxy	Mxy	Hxy
Kei-10	270	311	338	348	288	315	338	331	309	288
Kei-11	348	349	360	387	399	352	377	364	405	420
Kei-12	291	408	473	442	381	414	428	376	447	531
Kei-13	355	377	440	422	378	383	401	356	447	420
Kei-14	205	261	276	284	259	257	270	258	262	271
Kei-15	311	338	328	394	346	355	389	332	391	396
Vla-1	328	312	341	324	339	337	359	329	324	342
Vla-5	278	361	362	400	324	346	380	357	442	425
Vla-6	399	410	410	389	391	414	428	415	415	407
Vla-8	381	391	358	338	373	372	334	371	337	315
Nguyen-1	159	331	406	402	354	380	380	396	411	453
Nguyen-2	220	331	481	420	351	385	476	405	410	408
Nguyen-3	245	288	382	406	381	405	390	365	423	435
Nguyen-4	254	304	423	442	472	404	425	432	432	482
Kei-10	426	411	420	449	405	427	361	403	411	476

shown in the Table 8, the Table 9. To cluster, we used the k-means clustering algorithm with the Euclidean distance, repeat 100 times and use of median rather than mean centering, because it is more robust against outliers. This clustering algorithm is available in the open source cluster software: Cluster 3.0 (you can download at: *http://bonsai.hgc.jp/˜mdehoon/software/cluster/*). You also can download data sets and the reference table providing detailed information for each data set in folder "data Sets"at address: *https://www.dropbox.com/s/udek9c9ww44cjbu/Data%20sets.rar?dl=0*

Table 8 Four groups of generated data sets were clustered by criteria OV

	Name of data set (Index of data set)			
Cluster 0 (C0)	Kei2.My	Kei12.Ly	Kei12.Hy	Kei12.Hxy
	Kei12.Hx	Kei12.Mx	Nguyen_4.Ly	Kei11.My
	Kei11.Ly	Nguyen_4.Hy	Kei11.Lxy	Vla1.Hx
	Nguyen_4.Hx			
Cluster 1 (C1)	Vla1.Lxy	Nguyen_4.My	Kei12.Lx	Vla8.My
	Vla1.Hy	Nguyen_4.Lx	Kei10.Hy	Nguyen_3.Mxy
	Vla5.Lxy	Kei13.Hy	Kei14.Mxy	Vla8.Hxy
	Vla8.Hy	Vla6.Mx	Kei14.Lx	Vla8.Mx
	Kei10.Mxy	Kei13.My	Vla1.Mx	Kei13.My
	Vla6.My	Nguyen_4.Mx	Vla6.Hy	Kei13.Mxy
	Nguyen_2.Mx	Vla5.My	Nguyen_2.Ly	Kei13.Ly
	Vla1.Lx	Vla1.Ly	Vla6.Ly	Kei15.Mx
	Vla8.Lx	Kei10.Mx	Kei10.Ly	Kei10.Ly
	Kei14.Lxy	Kei10.Lxy	Kei10.F	Kei11.F
	Kei12.F	Kei13.F	Kei14.F	Kei15.F
	Vla1.F	Vla5.F	Vla6.F	Vla8.F
	Nguyen_1.tr10	Nguyen_2.F	Nguyen_3.F	Nguyen_4. F
	Kei13.Lx	Kei15.Lx	Vla6.Lx	Nguyen_3.Ly
	Vla5.Ly	Vla8.Ly	Vla6.Lxy	Vla8.Lxy
	Nguyen_1.Lxy	Vla6.Mxy	Nguyen_1.Mxy	
Cluster 2 (C2)	Nguyen_1.My	Kei14.My	Vla5.Hy	Nguyen_4.Lxy
	Kei15.Hx	Kei11.Mxy	Kei14.Hy	Kei14.Ly
	Nguyen_2.Lx	Vla5.Hxy	Vla6.Hxy	Kei14.Hx
	Vla8.Hx	Kei14.Mx	Nguyen_1.Hx	Vla1.My
	Vla5.Mxy	Nguyen_1.Mx	Nguyen_2.Hxy	Nguyen_3.Mx
	Kei10.Hxy	Nguyen_1.Hxy	Kei14.Hxy	Vla6.Hx
	Kei15.Lxy	Kei13.Hx	Kei10.Hx	Vla5.Hx
	Vla1.Hxy	Kei15.Ly	Nguyen_2.Lxy	Vla5.Lx
	Kei10.My	Vla5.Mx	Vla1.Mxy	Kei13.Lxy
	Nguyen_3.Lxy	Kei12.Lxy		
Cluster 3 (C3)	Kei11.Hy	Nguyen_3.Hy	Nguyen_4.Hxy	Nguyen_4.Mxy
	Kei11.Mx	Nguyen_2.Hy	Kei12.Mxy	Kei15.Hy
	Kei11.Lx	Kei11.Hxy	Nguyen_2.Mxy	Kei15.Mxy
	Nguyen_2.My	Nguyen_3.Ly	Nguyen_3.Hx	Nguyen_1.Ly
	Kei15.My	Nguyen_3.My	Nguyen_1.Lx	Nguyen_1.Hy
	Vla8.Mxy	Kei13.Hxy	Nguyen_2.Hx	Nguyen_3.Hxy
	Kei15.Hxy	Kei11.Hx		

5.2 Analysis of the robustness of GP to noise

Experimental results show that for all three noise types: noise x, noise y, noise x & y: the more noise, the greater fittest error) of the best solution (see the cases in bold in the Table 6) with most of benchmark functions. To be more clear, we can observe fittest error of the best model over the generations with the noise configurations in the Figure 4. The cause can be the higher noise levels and would make the problem more difficult to learn exact models. For example, fitness errors of some functions shown in the Figure 3 (graph of all functions can see in https://www.dropbox.com/s/gnuinkzicoq9jdr/Graph.rar?dl=0) clearly state that: fitness error of the best model with noise levels of 0% noise is smaller

Table 9 Information of clusters

	Cluster 0	Cluster 1	Cluster 2	Cluster 3
No. of observations	13	63	38	26
Within cluster sum of squares	592.7936	0.28177	1.447554	11.98758
Average distance from centroid	4.256253	0.049527	0.146812	0.527065
Max distance from centroid	15.390471	0.168109	0.453639	1.491932
Cluster centroids	4.949519	0.021502	0.386764	1.385605
Distances between centroids	$C0 \rightarrow C1$: 4.928017 $C0 \rightarrow C2$: 4.562755 $C0 \rightarrow C3$: 3.563914	$C1 \rightarrow C2$: 0.365262 $C1 \rightarrow C3$: 1.364103	$C2 \rightarrow C3$: 0.998841	
OV range: [min, max]	[3.186061, 20.33999]	[0, 0.189611]	[0.208218, 0.841303]	[0.913296, 2.877537]
OV level	High	Free	Low	Medium

than with 10% noise ($finess(0\%) < fitness(10\%)$); similarly, $finess(10\%) < fitness(30\%)$; $fitness(30\%) < fitness(50\%)$.

The greater noise level (with all three types of noise) not only make the generalization ability reduced but also increase the complexity of learned models with most of problems (see the Table 7). Thus, we can conclude GP is not powerful Machine Learning technique with noise.

5.3 Analysis of the differences between the noise types, noise levels

Beside some common features as shown in the section 5.2, each noise type, each noise level has has a few distinct characteristics from: (1) the magnitude of fittest error; (2) the convergence rate; (3) the over-fitted ability of solutions. To show the difference we use a function: sign(x): returns (+) if $x > 0$ else returns (-).

5.3.1 The differences in the magnitude of fittest error

Most of the problems with the same noise level, the same function, the fittest error of solutions that training on data set added noise x,y is often less than on data set added noise xy:

- Consider the cases of noise x and noise xy: with most of benchmark functions, fittest error of the best model on original data with noise x is often less than with noise xy. In the Table 10, the negative (-) cases show the fittest error with noise xy is less than with. At the level of noise by 10%, there are 11 from 14 functions include: Kei11, Kei14, Kei15, Vla1, Vla5, Vla6, Vla8, Nguyen_1, Nguyen_2, Nguyen_3 and Nguyen-4 (11/14 functions). There are 13/14; 13/14 functions corresponding to levels of noise 30%; 50%.

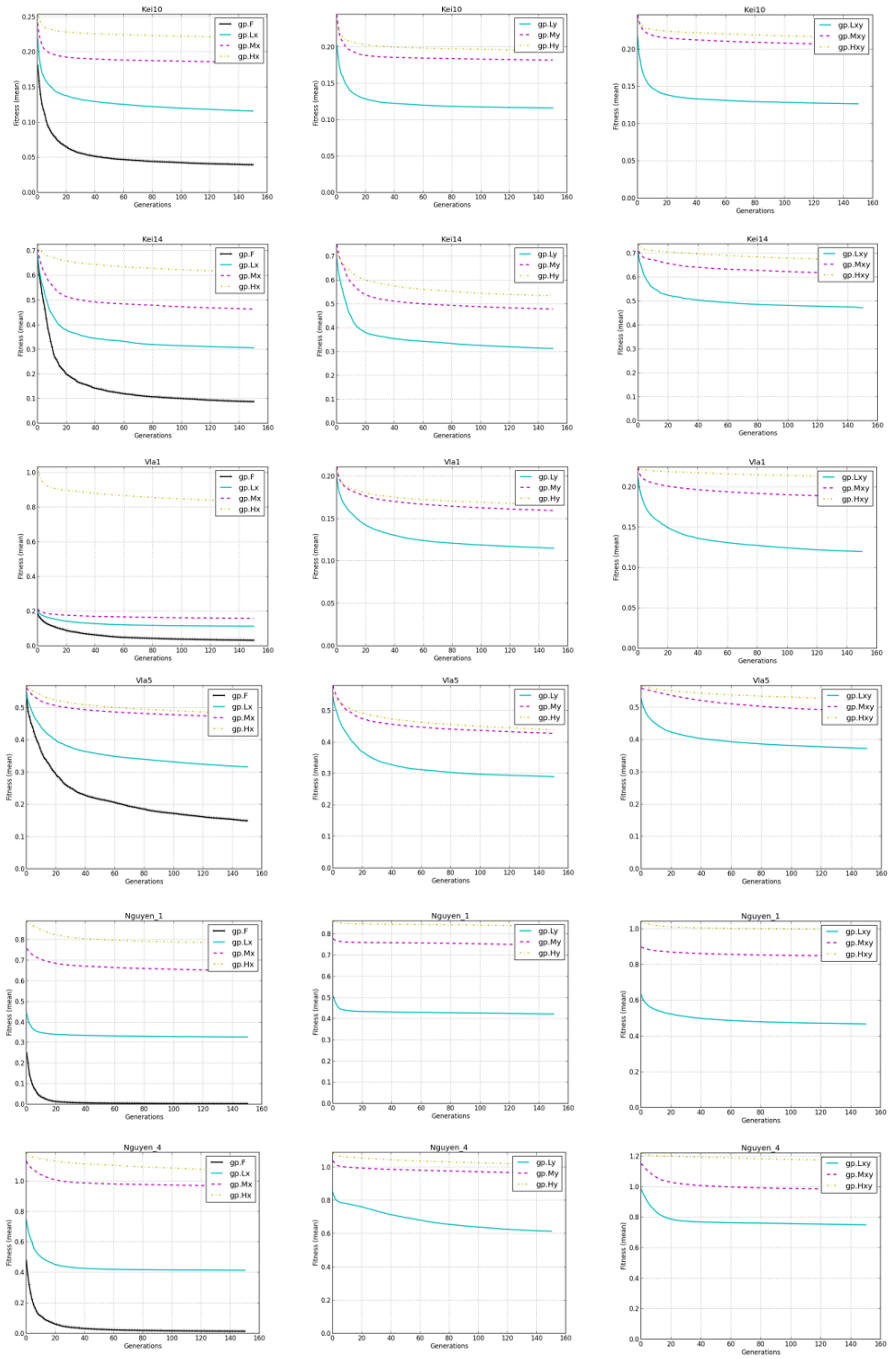

Fig. 3 Fitness (mean) of the best learned model by GP with noise configurations

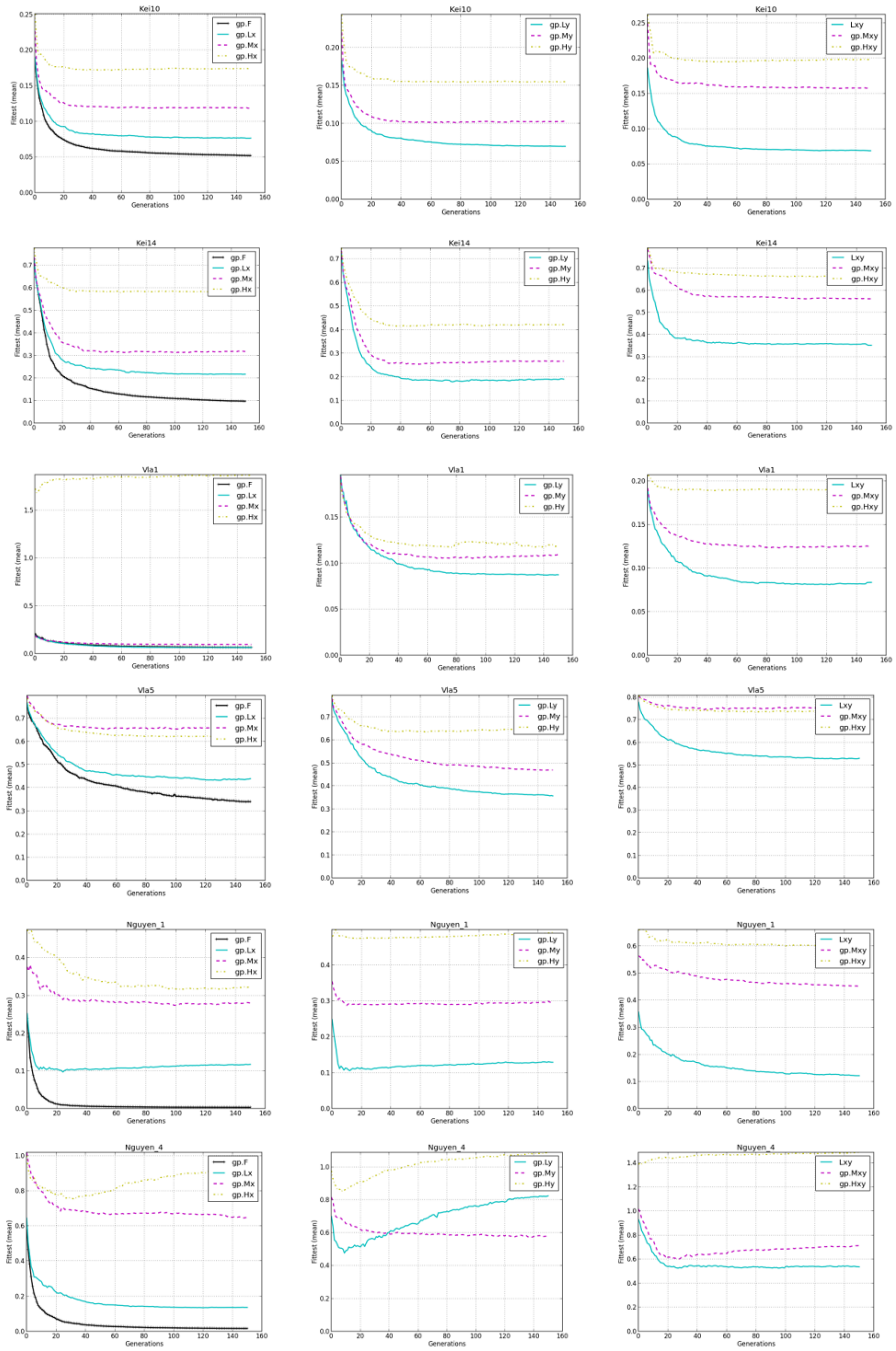

Fig. 4 Fittest (mean) of the best learned model by GP with noise configurations

Table 10 The difference (result of subtracting fittest errors) in the fittest error of the best model with various noise types: (a) sign(Fittest(Noise x) - Fittest(Noise y)); (b) sign(Fittest(Noise x) - Fittest(Noise xy)); c) sign(Fittest(noise y)-Fittest(noise xy)

Problem	(a)			(b)			(c)		
	10%	*30%*	*50%*	*10%*	*30%*	*50%*	*10%*	*30%*	*50%*
Kei10	+	+	+	+	-	-	+	-	-
Kei11	-	+	+	-	-	-	-	-	-
Kei12	+	-	+	+	-	-	+	-	-
Kei13	+	+	+	+	-	-	-	-	-
Kei14	+	+	+	-	-	-	-	-	-
Kei15	-	+	-	-	-	-	+	-	-
Vla1	-	-	+	-	-	+	+	-	-
Vla5	+	+	-	-	-	-	-	-	-
Vla6	+	+	+	-	+	-	-	-	-
Vla8	+	+	+	-	-	-	-	-	-
Nguyen_1	-	-	-	-	-	-	+	-	-
Nguyen_2	-	-	-	-	-	-	-	-	+
Nguyen_3	-	-	-	-	-	-	-	-	+
Nguyen_4	-	+	-	-	-	-	+	-	-

– Consider the case of noise y and noise xy: there are 8/14, 14/14, 12/14 functions at the noise level of 10%, 30%, 12%,respectively. Their fittest errors with noise x are less than with noise xy.
– Consider the case of noise x and noise y. The difference is not clear. The number of problems that noise x give fittest error greater than noise y are 7/14; 5/14; 6/14 corresponding to the levels of noise by 10%; 30%; 50%.

5.3.2 The differences in the convergence rate

To explore different effects of noise to the speed of convergence of the best solution, we calculated the mean of convergence speed over generations by taking fitness error of previous generation minus this error of later generation and cumulative, then averaged. The results shown in the Table 11.

Results in the Table 11 give us comments: (1) With the same noise type, the converging rate decreased when the noise increased on most of benchmark functions. More clearly, we can see the Table 12 is built from the Table 11 in which cells with sign (+) indicate $CR(0\%noise) > CR(10\%noise)$;$CR(10\%) > CR(30\%)$ and $CR(30\%) > CR(50\%)$, it is true for all noise types; (2) see the Table 13 most of functions, with the same level of noise, the convergence rate of the best model on the data set with noise x, y is larger than on the data set with noise xy, especially when the noise increased. However, this distinction is not clear between noise x and noise y:

1. Considering the case of noise x and noise xy:
   (a) At the level of noise by 10%, there are 9 functions from 14 benchmark functions: Kei10, Kei11, Kei14, Kei15, Vla1, Vla5, Vla6 Nguyen-3, Nguyen-4 have $CR(noisex) > CR(noisexy)$.
   (b) At the level of noise by 30%, all benchmark functions (except Kei11, Vla6, Nguyen-4) have $CR(noisex) > CR(noisexy)$ (11/14 functions).

Table 11 Mean of the convergence rate across generations (measured through the fitness error of the best model)

Problem	F	Lx	Mx	Hx	Ly	My	Hy	Lxy	Mxy	Hxy
Kei10	9.E-04	7.E-04	4.E-04	2.E-04	6.E-04	4.E-04	3.E-04	6.E-04	3.E-04	2.E-04
Kei11	1.E-03	4.E-03	2.E-03	2.E-03	2.E-03	2.E-03	2.E-03	2.E-03	2.E-03	1.E-03
Kei12	3.E-01	2.E-01	1.E-01	7.E-02	2.E-01	8.E-02	6.E-02	2.E-01	6.E-02	4.E-02
Kei13	1.E-02	6.E-03	3.E-03	2.E-03	9.E-03	5.E-03	2.E-03	7.E-03	2.E-03	2.E-03
Kei14	4.E-03	3.E-03	2.E-03	8.E-04	3.E-03	2.E-03	1.E-03	2.E-03	7.E-04	5.E-04
Kei15	9.E-03	8.E-03	3.E-03	2.E-03	6.E-03	4.E-03	3.E-03	5.E-03	2.E-03	2.E-03
Vla1	1.E-03	6.E-04	4.E-04	1.E-03	6.E-04	4.E-04	3.E-04	6.E-04	3.E-04	9.E-05
Vla5	3.E-03	2.E-03	6.E-04	6.E-04	2.E-03	1.E-03	8.E-04	1.E-03	5.E-04	3.E-04
Vla6	8.E-03	5.E-03	2.E-03	2.E-03	5.E-03	4.E-03	3.E-03	4.E-03	2.E-03	1.E-03
Vla8	6.E-03	3.E-03	2.E-03	1.E-03	4.E-03	3.E-03	1.E-03	4.E-03	1.E-03	4.E-04
Nguyen_1	2.E-03	8.E-04	7.E-04	7.E-04	6.E-04	2.E-04	2.E-04	1.E-03	4.E-04	3.E-04
Nguyen_2	2.E-03	1.E-03	2.E-03	4.E-04	8.E-04	4.E-04	2.E-04	1.E-03	8.E-04	4.E-04
Nguyen_3	3.E-03	2.E-03	1.E-03	1.E-03	1.E-03	7.E-04	7.E-04	2.E-03	6.E-04	5.E-04
Nguyen_4	3.E-03	2.E-03	1.E-03	8.E-04	2.E-03	5.E-04	5.E-04	2.E-03	1.E-03	3.E-04

Table 12 The difference in the convergence rate (CR) of the best model on the data sets have the same type of noise but various noise levels: (a) sign(CR(0%)- CR(10%)); (b) sign(CR(10%)- CR(30%)); (c) sign(CR(30%)- CR(50%))

Problem	Noise x			Noise y			Noise xy		
	(a)	(b)	(c)	(a)	(b)	(c)	(a)	(b)	(c)
Kei10	+	+	+	+	+	+	+	+	+
Kei11	-	+	-	-	-	-	-	-	+
Kei12	+	+	+	+	+	+	+	+	+
Kei13	+	+	+	+	+	+	+	+	+
Kei14	+	+	+	+	+	+	+	+	+
Kei15	+	+	+	+	+	+	+	+	-
Vla1	+	+	-	+	+	+	+	+	+
Vla5	+	+	+	+	+	+	+	+	+
Vla6	+	+	+	+	+	+	+	+	+
Vla8	+	+	+	+	+	+	+	+	+
Nguyen_1	+	+	+	+	+	+	+	+	+
Nguyen_2	+	-	+	+	+	+	+	+	+
Nguyen_3	+	+	-	+	+	+	+	+	+
Nguyen_4	+	+	+	+	+	+	+	+	+

(c) At the level of noise by 50%, all benchmark functions have $CR(noisex) > CR(noisexy)$ (14/14 functions).

2. Considering the case of noise y and noise xy: At the levels of noise by 10%, 30%, 50%, respectively, there are 8/14, 10/14, 13/14 functions which the convergence rate of the best learned model on data set with noise y is greater than on data set with noise xy.
3. Considering the case of noise x and noise y, the difference is not clear. At the levels of noise by 10%, 30%, 50%, respectively, there are 8/14, 7/14, 7/14 functions which the converging rate of the best model on data set added noise x is greater than on data set added noise y.

Table 13 The difference in the convergence rate (CR) of the best model on the data sets have the same noise level but various noise types: (a) sign(CR (noise x) - CR (noise y)); (b) sign(CR (noise x) - CR (noise xy)); (c) sign(CR (noise y) - CR (noise xy))

Problem	(a)			(b)			(c)		
	10%	30%	50%	10%	30%	50%	10%	30%	50%
Kei10	+	-	-	+	+	+	-	+	+
Kei11	+	+	+	+	-	+	+	-	+
Kei12	-	+	+	-	+	+	+	+	+
Kei13	-	-	-	-	+	+	+	+	+
Kei14	-	-	-	+	+	+	+	+	+
Kei15	+	-	-	+	+	+	+	+	+
Vla1	+	+	+	+	+	+	-	+	+
Vla5	-	-	-	+	+	+	+	+	+
Vla6	-	-	-	+	-	+	+	+	+
Vla8	-	-	-	-	+	+	+	+	+
Nguyen_1	+	+	+	-	+	+	-	-	-
Nguyen_2	+	+	+	-	+	+	-	-	-
Nguyen_3	+	+	+	+	+	+	-	+	+
Nguyen_4	+	+	+	+	-	+	-	-	+

5.3.3 The differences in the OV

The over-fitting ability may be caused by training models on limited samples (eg, BEN-7-8 BEN, BEN-9 where the range of variables on the training set is smaller than on the testing set) or samples contain noise. To analyze the impact of noise types and noise levels to the OV, we see results shown in Table 5, Table 14, 15.

Noise types and noise levels have some common features: (1) The greater the noise level, the greater the OV. It is true with most benchmark functions (see cells(+) in the Table 14). For example, considering noise x, there are 14/14, 10/14, 13/14 functions that $OV(10\%noise) \geq OV(0\%noise)$; $OV(30\%) > OV(10\%)$; $OV(50\%) > OV(30\%)$. (2) with noise level by 50, all functions are over-fitted. Besides the above common features, they also have different characteristics, especially at the level of 30%, most of functions OV (noise y) are usually greater than OV (noise x, noise xy) (see cells (-) in 15). This give us that noise y usually creates more over-fitting challenges than other types of noise.

6 Conclusions and future works

In this paper, we focus on three purposes: (1) proposing a method to quantify the over-fitting of benchmarks in GP; (2) developing a suit of symbolic regression benchmarks includes four groups with increasing (OV) difficulty levels; (3) investigating the impact of the different types of noise with various noise levels on the effectiveness of GP and answering the question if GP is a method which strong to noise? The achieved research results include:

Table 14 The difference in OV of the best model trained on data sets with the same noise type but various noise levels: (a) sign(OV(10% noise)-OV(0% noise)); (b) sign(OV(30% noise)-OV(10% noise)); (c) sign(OV(50% noise)-OV(30% noise))

Problem	Noise x			Noise y			Noise xy		
(a)	(b)	(c)	(a)	(b)	(c)	(a)	(b)	(c)	
Kei10	+	+	+	+	+	-	+	+	+
Kei11	+	+	-	+	+	-	+	-	+
Kei12	+	+	+	+	+	-	+	+	+
Kei13	+	+	+	+	+	+	+	-	+
Kei14	+	+	+	+	+	-	+	+	+
Kei15	+	+	+	+	+	+	+	+	-
Vla1	+	+	+	+	+	-	+	+	+
Vla5	+	-	+	+	+	+	+	+	+
Vla6	+	+	+	+	+	-	+	+	+
Vla8	+	+	+	+	+	-	+	+	-
Nguyen_1	+	-	+	+	-	+	+	+	+
Nguyen_2	+	-	+	+	+	+	+	+	-
Nguyen_3	+	+	+	+	-	+	+	-	+
Nguyen_4	+	-	+	+	-	+	+	+	+

Table 15 The difference in OV of the best model trained on data sets with the same noise level but various noise types: (a) sign(OV (noise x) - OV (noise y)); (b) sign(OV (noise x) - OV (noise xy)); (c) sign(OV (noise y) - OV (noise xy))

Problem	(a)			(b)			(c)		
	10%	30%	50%	10%	30%	50%	10%	30%	50%
Kei10	+	-	+	+	-	-	-	-	+
Kei11	-	-	-	-	+	-	-	-	-
Kei12	-	-	-	-	+	-	-	-	-
Kei13	-	+	+	-	+	-	+	-	+
Kei14	-	-	-	+	+	+	-	-	-
Kei15	-	-	-	-	-	-	+	+	-
Vla1	+	-	+	-	-	+	+	-	+
Vla5	+	+	-	+	-	-	+	+	-
Vla6	-	+	+	+	+	-	-	-	+
Vla8	+	-	+	+	-	+	+	+	+
Nguyen_1	-	-	-	+	+	+	-	-	-
Nguyen_2	+	-	-	+	-	+	+	+	-
Nguyen_3	-	-	-	-	+	+	-	-	-
Nguyen_4	-	-	-	-	-	+	-	+	-

– The new measure to quantify the OV using the formula (3). It is a combination of two factors: the amount of over-fitting using the formula (2); over-fitting happens sooner or later when evolution best model over generations.
– A suite of symbolic regression benchmark data sets. These data sets are grouped to four clusters based on their OVs. Levels of OV of Cluster 0, Cluster 3, Cluster 2 and Cluster 1 are assigned to “high”, “medium”, “low”and “free”by us. Data sets in each cluster are sorted in order of descending OV.

– An analysis of the robustness of the GP to noise. Experimental results show that GP is a very sensitive method to noise. The higher noise level, the lower generalization ability of GP. This is general feature of all noise type and noise level. However, each type of noise and each level of noise has its own characteristics that impact on the evolution of GP solutions, as analyzed in the section 5.3. These characteristics include: (1) the magnitude of fittest error, 2) the convergence rate and 3) the OV of the best leaned model by GP. Another conclusion is that, most of problems, noise x and noise y have many similar points about (1) and (2). Noise xy is usually has (1) greater than noise x, y while (2) is lower. All noise types have (3) many quite points, the higher noise level, the greater OV. So to study the generalization ability and over-fitted level of solutions when they trained on the data set contains noise, we can just interested in one of these noise types. But, for the study of the affect of noise to the magnitude of the fittest error and convergence rate of solutions we need to study a couple of noise types as $(noisexy; noisex)$ or $(noisey; noisexy)$. These are also our recommendations for researchers who aspire to yourself design benchmarks for these study goals.

There are several future researches arisen from this paper. First we will propose a more general method to quantify the OV for Machine Learning methods. Second we will try to conduct more experiments other noise models to have more information about them. Last, we will propose a new method to solve the over-fitting issue in GP using a suit of benchmarks introduced in this paper.

Acknowledgements

References

1. Marcel Bilger, Willard G Manning, et al. Measuring overfitting and mispecification in nonlinear models. Technical report, HEDG, c/o Department of Economics, University of York, 2011.
2. Mark A. Friedl Carla E. Bro dley. Identifying mislab eled training data. *Journal of Artifcial Intelligence Research*, 11:pp.131–167 (1999).
3. Nicolo Cesa-Bianchi, Shai Shalev-Shwartz, and Ohad Shamir. Online learning of noisy data. *Information Theory, IEEE Transactions on*, 57:pp. 7907–7931 (2011).
4. Michael ONeill Clodhna Tuite and Anthony Brabazon. Towards a dynamic benchmark for genetic programming. In *In Proc. GECCO13 (2013)*.
5. Albert Fornells David, Albert. A study of the effect of different types of noise on the precision of supervised learning techniques. *Artif Intell Rev*, 33:pp. 275306 (2010).
6. Acuna Daza. An algorithm for detecting noise on supervised classification. In *In Proc. WCECS'07, (2007)*.
7. Benot Frnay. Classification in the presence of label noise: a survey. *IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS*, 15:pp.845869 (2014).
8. Ivo Goncaves and Sara Silva. Balancing learning and overfitting in genetic programming with interleaved sampling of training data. In *In Proc. EuroGP'13 (2013)*.
9. S. Gustafson and L Vanneschi. Operator-based distance for genetic programming: Subtree crossover distance. In *In Proc. EuroGP'05 (2005)*.

10. Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.
11. Ray J. Hickey. Noise modelling and evaluating learning from examples. *Artificial Intelligence*, 8:pp.157–179 (1996).
12. Joana B.Melo et al Ivo Goncalves, Sara Silva. Random sampling technique for overfitting control in genetic programming. In *In Proc. EuroGP'12 (2012)*.
13. Sean Luke James McDermott, David R. White. Genetic programming needs better benchmarks. In *In Proc. GECCO'12(2012)*.
14. Sean Luke et al James McDermott, David R.White. Genetic programming needs better benchmarks. In *In Proc. GECCO12 (2012)*.
15. R.Muhammad Atif Azad Jeannie Fitzgerald and Conor Ryan. A bootstrapping approach to reduce over-fitting in genetic programming. In *In Proc. GECCO13 (2013)*.
16. Russ H. Drieberg Ji Ni and Peter I. Rockett. The use of an analytic quotient operator in genetic programming. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, 17:pp. 146152 (2013).
17. Jerry Swan John Woodward, Simon Martin. Benchmarks that matter for genetic programming. In *In Proc. GECCO14(2014)*.
18. Elias Kalapanidas, Nikolaos Avouris, Marian Craciun, and Daniel Neagu. Machine learning algorithms: A study on noise sensitivity. In *In Proc. 1st Balcan Conference in Informatics, (2003)*.
19. John R Koza. *Genetic programming: on the programming of computers by means of natural selection*, volume 1. MIT press, 1992.
20. Mauro Castelli Leonardo Vanneschi and Sara Silva. Measuring bloat, overfitting and functional complexity in genetic programming. In *In Proc. GECCO'10 (2010)*.
21. Po-Ling Loh and Martin J Wainwright. High-dimensional regression with noisy and missing data: Provable guarantees with non-convexity. In *In Proc: Advances in Neural Information Processing Systems, (2011)*.
22. S. Gustafson M. ONeill, L. Vanneschi and W. Banzhaf. Open issues in genetic programming. In *In Proc. GPEM'10(2010)*.
23. T Mitchell. *Machine Learning*. McGraw Hill, 1996.
24. Hien-Thi Nguyen. A study of generalization in genetic programming, specialized in applied mathematics and computer science. military technical academy, vietnam (2014).
25. Nguyen Xuan Hoai J.M. Dermott Nguyen quang Uy, Pham Tuan Anh. Subtree semantic geometric crossover for genetic programming, accepted for publication. *Journal of Genetic Programming and Evolvable Machines*, pages pp.x–xx? (2014).
26. Nguyen Xuan Hoai Bob MacKay Nguyen Thi Hien and Nguyen Quang Uy. Where should we stop - an investigation on early stopping for gp learning. In *In Proc. SEAL'12 (2012)*.
27. Miguel Nicolau, Alexandros Agapitos, Michael ONeill, and Anthony Brabazon. Guidelines for defining benchmark problems in genetic programming. In *In Proc. CEC 15 (2015)*.
28. Mark D Pendrith and C Sammut. *On reinforcement learning of control actions in noisy and non-Markovian domains*. Citeseer, 1994.
29. Riccardo Poli, William B Langdon, Nicholas F McPhee, and John R Koza. *A field guide to genetic programming*. Lulu. com, 2008.
30. P. F Stadler. Fitness landscapes. In *In Proc. Biological Evolution and Statistical Physics (2014)*.
31. Lee Spector Thomas Helmuth. General program synthesis benchmark suite. In *In Proc. GECCO 15 (2015)*.
32. Marco Tomassini, Leonardo Vanneschi, Philippe Collard, and Manuel Clergue. A study of fitness distance correlation as a difficulty measure in genetic programming. *Evolutionary Computation*, 13:pp. 213–239(2005).
33. Hoai et al Uy, Hien. Improving the generalization ability of genentic programming with semantic similarity based crossover. In *In Proc. EuroGP'10 (2010)*.
34. Vanneschi. L. theory and practice for efficient genetic programming. phd thesis, faculty of sciences, university of lausanne, switzerland (2004).

35. Tomassini M. Collard P. Vanneschi, L. and S V erel. Negative slope coefficient. a measure to characterize genetic programming. In *In Proc. the 9th European Conference on Genetic Programming (2006)*.
36. David R White, James McDermott, Mauro Castelli, Luca Manzoni, Brian W Goldman, Gabriel Kronberger, Wojciech Jaśkowski, Una-May OReilly, and Sean Luke. Better gp benchmarks: community survey results and proposals. *Genetic Programming and Evolvable Machines*, 14:pp. 3–29 (2013).
37. David H. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8:pp. 1341–1390 (1996).
38. Sewall Wright. *The roles of mutation, inbreeding, crossbreeding, and selection in evolution*, volume 1. na, 1932.
39. J. P. Castagna Z. P. Liu. Avoiding overfitting caused by noise using a uniform training mode. In *In Proc. on Neural Networks (1999)*.