Generalisation and Domain Adaptation in GP with Gradient Descent for Symbolic Regression

Qi Chen
School of Engineering
and Computer Science
Victoria University of Wellington
Wellington, New Zealand
Email: Qi.Chen@ecs.vuw.ac.nz

Bing Xue
School of Engineering
and Computer Science
Victoria University of Wellington
Wellington, New Zealand
Email: Bing.Xue@ecs.vuw.ac.nz

Mengjie Zhang
School of Engineering
and Computer Science
Victoria University of Wellington
Wellington, New Zealand
Email: Mengjie.Zhang@ecs.vuw.ac.nz

***Abstract*—Genetic programming (GP) has been widely applied to symbolic regression problems and achieved good success. Gradient descent has also been used in GP as a complementary search to the genetic beam search to further improve symbolic regression performance. However, most existing GP approaches with gradient descent (GPGD) to symbolic regression have only been tested on the "conventional" symbolic regression problems such as benchmark function approximations and engineering practical problems with a single (training) data set only and the effectiveness on unseen data sets in the same domain and in different domains has not been fully investigated. This paper designs a series of experiment objectives to investigate the effectiveness and efficiency of GPGD with various settings for a set of symbolic regression problems applied to unseen data in the same domain and adapted to other domains. The results suggest that the existing GPGD method applying gradient descent to all evolved program trees three times at every generation can perform very well on the training set itself, but cannot generalise well on the unseen data set in the same domain and cannot be adapted to unseen data in an extended domain. Applying gradient descent to the best program in the final generation of GP can also improve the performance over the standard GP method and can generalise well on unseen data for some of the tasks in the same domain, but perform poorly on the unseen data in an extended domain. Applying gradient descent to the top 20% programs in the population can generalise reasonably well on the unseen data in not only the same domain but also in an extended domain.**

I. Introduction

Symbolic regression is a type of regression analysis that searches the space of mathematical expressions to find the model that best fits a given data set. Different from classical regression, symbolic regression does not have a pre-specified model structure and it attempts to discover both model structures and corresponding parameters/coefficients, which is a difficult task [1]. Genetic programming (GP) [2] is particularly promising for symbolic regression problems due to its flexible representation, which can search for mathematical building blocks, such as mathematical operators, analytic functions, constants, and variables. GP can automatically generate models without pre-defining the structure of the model. Generally, GP is the main approach to solving symbolic regression problems [3], [4], [5], [6], [7].

GP uses a genetic beam search mechanism, which is argued to be powerful for global search. However, GP often has an efficiency problem due to its high computational complexity [8]. Researchers have considered combining GP with fast local search methods, such as hill climbing, simulated annealing, and gradient descent, to improve its effectiveness and efficiency. Gradient descent is a simple yet very efficient local search technique, which has been successfully combined with GP to improve its performance [9], [10], [11], [12]. Performing gradient descent search on the fitness surface in the space of GP individuals' coefficients/constants has been shown to be a promising approach to symbolic regression [10] and classification [9], [13].

Traditionally, most symbolic regression approaches were examined on the given set of data points (i.e. training data) [10]. The performance of the built model on the unseen data, i.e. the generalisation ability of the symbolic regression methods, was typically not evaluated. In real-world applications, whether the built model can perform well on unseen/new data from the same domain is often very important to the users. While the generalisation ability on supervised classification including GP with gradient descent (GPGD) for classification [13], [9] has been widely studied, researchers started to investigate the generalisation abilities of GP for symbolic regression in recent years [14], [15], [16], [17]. Castelli et al. [15] compared the generalisation ability of several GP frameworks for symbolic regression, including some variants of multi-objective genetic programming and operator equalization, a recently defined bloat free genetic programming system, but only one problem was tested in the experiments. There are still many recent works on symbolic regression without having a unseen test set [18], [19], [20], [21], [22], [23].

Recently, domain adaptation and transfer learning become a hot topic in machine learning [24], [25], [26], particularly in neural networks and image recognition and analysis [27], [28], [29], [30]. While the standard supervised learning aims to achieve generalisation on the unseen data points/instances that are of similar distributions (or similar areas/intervals), domain adaptation and transfer learning seek to achieve deeper generalisation on the (unseen) data instances in a different domain, from a different distribution, or at a different area/interval. Compared with standard supervised learning, the problem in domain adaptation and transfer learning is generally much harder. This topic has also attracted good attention in evolutionary computation particularly in memetic computing [31], but no serious research on GP has been reported. Of

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

course, the domain adaptation ability of GPGD for symbolic regression has not been investigated.

A. Goals

The overall goal is to investigate whether GPGD algorithms can generalise well on symbolic regression problems in the same domain and can be effectively adapted to other domains. Specific objectives are:

- whether GPGD algorithms can achieve better performance on the original training data,
- whether GPGD algorithms can generalise well on unseen test data from the same domain,
- whether GPGD algorithms can be adapted well on unseen test data from an extended domain,
- whether GPGD algorithms can be adapted well on unseen test data from a different domain, and
- how the number of data points (training instances) influences the performance on unseen data.

B. Organisation

The remainder of this paper is organised as follows. Section 2 describes the how gradient descent is used in GP for symbolic regression. Section 3 presents the design of the algorithms, the benchmark problems and experiments. Results and discussions are presented in Section 4. Conclusions and future work are given in Section 5.

II. GP with Gradient Descent for Symbolic Regression

In GP with gradient descent (GPGD) for symbolic regression [10], the standard tree based GP was used [1], [2]. Assuming that the audience of this conference know GP well, this section focuses mainly on how gradient descent is applied to GP during the evolutionary process [10].

Gradient descent is a local search method, which can guarantee to find a local optimal solution for a particular task. Topchy and Punch [10] applied gradient descent to update the value of constant nodes in a GP tree/program. Mean squared error (MSE) is used to evaluate the fitness of each program during the evolutionary process, which forms a minimisation fitness function, shown by Equation (1).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(t_i - f(X,C))^2 \tag{1}$$

where t_i is the target output, n is the number of training data points/instances, X is a vector of inputs, C is a vector of constants, and $f(X,C)$ is the output of an evolved model, i.e. a GP program.

The gradient of MSE is the vector of partial derivatives with respect to the constant values/nodes in a GP tree. These constant values can be trained by the following learning rule.

$$C_k \leftarrow C_k - \alpha\frac{\partial MSE}{\partial C_k} \tag{2}$$

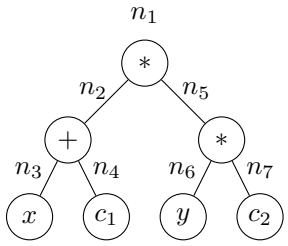

Fig. 1. A program tree.

where α is the learning rate defined by the user, $\frac{\partial MSE}{\partial C_k}$ is the partial derivative of MSE with respect to C_k, and k goes over all the constants in the tree.

To perform gradient descent on a particular GP tree, the original values of the constant nodes are collected, then the algorithm moves to a neighbouring point that is downhill, repeating until it converges to the minimum possible error value or a given amount of time is run out. As soon as the sum of derivatives in all training points is obtained, the value of the constant node can be updated accordingly.

How to work out the partial derivative and the detailed steps of gradient descent are described in the following two subsections.

A. Partial Derivative

Based on Equations (1) and (2), the following Equation (3) can be derived.

$$\frac{\partial MSE}{\partial C_k} = -\frac{2}{n}\sum_{i=1}^{n}(t_i - f(X,C))\frac{\partial f(X,C)}{\partial C_k} \tag{3}$$

Now the task is to work out partial derivative of $f(X,C)$. Suppose n_i is the function of node i in the program tree, according to the chain rule, one can get

$$\frac{\partial f(X,C)}{\partial C_k} = \frac{\partial(n_1(n_2(n_3(...)))}{\partial C_k} = \frac{\partial n_1}{\partial n_2}\frac{\partial n_2}{\partial n_3}...\frac{\partial n_{k-1}}{\partial C_k} \tag{4}$$

In other words, according to chain rule, the gradient of the tree can be broken down into derivatives of nodes on the path from the root of the tree to the given leaf node. For instance, given a program shown in Fig. 1, assume n_i is the function of node i, then $f(X,C) = n_1$ is the final output of the program.

Since n_4 and n_7 are constant nodes, the gradient vector of $f(X,C)$ should contain values from the partial derivatives from n_4 to n_7 . The partial derivatives $f(X,C)$ with respect to n_4 and n_7 are:

$$\frac{\partial f}{\partial n_4} = \frac{\partial n_1}{\partial n_4} = \frac{\partial n_1}{\partial n_2} * \frac{\partial n_2}{\partial n_4} = \frac{\partial(n_2 n_5)}{\partial n_2} * \frac{\partial(n_3 + n_4)}{\partial n_4} = n_5 * 1 = n5 \tag{5}$$

$$\frac{\partial f}{\partial n_7} = \frac{\partial n_1}{\partial n_7} = \frac{\partial n_1}{\partial n_5} * \frac{\partial n_5}{\partial n_7} = \frac{\partial (n_2 n_5)}{\partial n_5} * \frac{\partial (n_6 n_7)}{\partial n_7} = n_2 * n_6 \quad (6)$$

As n_2, n_5, n_6 can be obtained from the program evaluation, the gradient of $f(X, C)$ with respect to different C_k can be calculated accordingly. The derivatives of the functions used in this work are listed in Table I.

TABLE I. PARTIAL DERIVATIVES

Function	$\frac{\partial f}{\partial n_1}$	$\frac{\partial f}{\partial n_2}$
$n_1 + n_2$	1	1
$n_1 - n_2$	1	-1
$n_1 * n_2$	n_2	n_1
n_1 / n_2	n_2^{-1}	$n_1 * n_2^{-2}$

B. Gradient Descent Algorithm

The gradient descent algorithm is performed in the following steps:

1) Get all the constant nodes of a GP tree/program;
2) For constant node C_k, calculate the partial derivative of the program output with respect to C_k, $\frac{\partial f}{\partial C_k}$ according to Equation (4);
3) Calculate the partial derivative of MSE with respect to C_k, $\frac{\partial MSE}{\partial C_k}$ with Equation (3);
4) Calculate the change of C_k with Equation (2) and update C_k;
5) Evaluates the GP tree, if the MSE of the new GP tree is worse than the original one, terminate the training process, make no change on C_k;
6) Obtain the next constant node C_{k+1}, go to step 2 for training C_{k+1};
7) If all the constant nodes of the GP tree have been trained, go on training the next GP tree.

Note that the evolutionary process is still governed by the genetic beam search, and the gradient descent search is only applied to certain individual programs at a generation. The results in [10] show that applying gradient descent search in GP can significantly improve the performance on symbolic regression problems on the training set, but its generalisation and domain adaptation ability on unseen test data have not been investigated.

III. METHOD AND EXPERIMENT DESIGN

A. Design of Different GPGD Methods

The previous work [10] has shown that GPGD can perform much better than standard GP for symbolic regression on the (single training) data set. That method called HGP [10] applies gradient descent search three steps to all individual programs at every generation. For presentation convenience, we call it 3HGP in this paper as it applied gradient descent three steps to every individual program. While it performs well on the training data set, we hypothesise that 3HGP cannot generalise well to unseen data sets in the same domain or in a different domain — applying gradient descent too often to the same individual program could make the program go to local optima, and applying gradient descent to all the individual programs in a population might make the population quickly lose the diversity since this could make the programs in a population functionally too similar. Accordingly, applying gradient descent less often to few programs might enhance the generalisation and domain adaptation ability of the GPGD method.

Based on this idea and hypothesis and to investigate the generalisation ability, *seven* GPGD methods at different levels of combining gradient descent with GP are designed and to be examined in this paper, namely GP, 3HGP [10], 1HGP, GGP1, GGP3, GGP5 and GP-L.

- GP denotes the standard GP algorithm without involving any gradient descent search.
- In 3HGP, gradient descent is applied three steps to every individual in every generation during the evolutionary process of GP.
- In 1HGP, gradient descent is applied only one step to every individual in every generation.
- In GGP1, GGP3 and GGP5, the gradient descent search is applied to the best 20% individuals once in every one generation, every three generations and every five generations, respectively.
- GP-L denotes the algorithm that only applies gradient descent search to the best individual only once at the last generation of the GP search.

Obviously, the gradient descent is applied most intensively in 3HGP, and gradually reduces to the least intensive use of gradient descent from 1HGP, GGP1, GGP3, GGP5, GP-L, and GP.

B. Benchmark Problems

To investigate the generalisation and domain adaptation ability of GP with gradient descent for symbolic regression, five benchmark problems are taken from [10], which have been widely used to test GP approaches to symbolic regression. Using these five benchmark problems, we also aim to make fair comparisons of the seven different GP with gradient descent methods.

$$f_1(x, y) = x * y + sin((x - 1)(y + 1)) \quad (7)$$

$$f_2(x, y) = x^4 - x^3 + y^2/2 - y \quad (8)$$

$$f_3(x, y) = 6sin(x)cos(y) \quad (9)$$

$$f_4(x, y) = 8(2 + x^2 + y^3) - 1 \quad (10)$$

$$f_5(x, y) = x^3/5 + y^3/2 - y - x \quad (11)$$

C. Training Data Set and Unseen Test Data Sets

To investigate the generalisation and domain adaption ability of the GPGD methods, four sets of different types of data points/instances are developed for each of the five benchmark problems by providing symbolic regression problems of increasing difficulty. These four sets are designed to mimic/simulate four tasks providing different levels for generalisation and domain adaptation, as stated below.

- The first data set is simply a single training set consisting of a certain number of points/instances within the interval [-3, 3] without any unseen test data. This task here is to use the above seven GPGD methods to evolve a model that approximates a given benchmark function/problem. The performance will be examined on this training set itself and no generalisation evaluation will be done. Most existing research on GP for symbolic regression belongs to this type.
- The second data set also consists of a certain number of points/instances within the interval [-3, 3], but these instances are different from the first data set. This set is used as an *unseen* test set, and the goal is to investigate whether the mathematical model evolved/learned by GPGD from the first (training) set can generalise well on the unseen test instances in the same domain ([-3, 3]). While these instances might or might not follow the same distribution as those in the training set, they are still located in the same interval ([-3, 3]) so we treat them still in the same domain as those in the training set. Compared with the first task above, this task is much harder, and there has not been much work reported to date.
- The third data set is designed to consist of instances within the interval [-6, 6], and all these instances are different from those in the training data set. The goal is to investigate whether the mathematical model evolved/learned by GPGD from the first (training) data set can be adapted (or transferred) to the unseen test instances in an extended domain [-6, 6]. While this interval still includes the previous interval [-3, 3], it also has two additional intervals ([-6,-3] and [3, 6]). So we treat these unseen instances in an *extended* or *overlapped* domain to those in the training set. Compared with the first two tasks above, this task is harder, and no work on GPGD for symbolic regression has been reported to date.
- The fourth data set consists of instances within the interval [3, 6], which we treat as a different domain from those in the training set. Clearly, there are no overlap between the instances in this data set and those in the training set. The task here is to investigate whether the model evolved from the training set can be further adapted (or transferred) to the instances in a different domain. Again, the task is the hardest one among the four, and no work on GP for symbolic regression has ever been reported.

D. Number of Instances in the Data Sets

To test how the number of training instances/data points influences the learning performance, all the experiments are conducted on data sets with two different sizes. The first includes 20 random data points in the training set and every test set, which is intentionally to be consistent with [10]. The second includes 100 random points in the training set and every test set, which is to see whether increasing the number of data points changes the conclusions.

E. Parameter Settings

The parameters settings of GP in all the seven GPGD algorithms can be seen from Table II, which is consistent with that in [10]. The learning rate α in gradient descent is 0.5 as suggested in [10].

TABLE II. PARAMETER SETTINGS

Population Size	200
Function set	{+,-,*,%protected}
Generation	50
Crossover probability	0.6
Mutation probability	0.4
Elitism probability	0.1
Maximum Depth	8

The experiment of every algorithm on each problem with each data set size have been conducted for 100 independent runs. Therefore, 7000 (i.e. 7*5*2*100) experiments have been run for the seven algorithms on five benchmark problems with two different data set sizes. The 7000 runs result in 7000 GP trees (one tree from each run). Each GP tree has a training MSE, and three testing MSEs by applying it to the three test sets. Therefore, 28000 (i.e. 7000*4) results are used here to discuss performance, generalisation and domain adaption ability of GPGD for symbolic regression.

IV. RESULTS AND DISCUSSIONS

Tables III and IV show the results of the seven methods on the training data [-3, 3], the unseen test data from the same domain [-3, 3], the unseen test data from an extended domain [-6, 6], and the unseen test data from a different domain [3, 6]. Tables V and VI show the results of significance tests on the experiments with 20 data points and 100 data points, respectively. Table VII presents the computational time. In the tables, "Mean" and "Std" show the average and standard deviation of the 100 results that were obtained from 100 independent runs in each algorithm.

A. Results on the Training Set

As can be seen from Table III, 3HGP generally achieved the best training performance, which is much better than all the other six algorithms on all the five benchmark problems. This is consistent with that in [10], where only a training set was used. Generally, from 3HGP, 1HGP, GGP1, GGP3, GGP5, GP-L to GP, where the gradient descent search was used from the most intensively to the least, the performance is gradually reduced. The number of data points as 20 or 100 does not change the pattern of the results.

The results suggest that introducing gradient descent as a complementary search to the genetic beam search of GP can increase the search ability of the algorithm to obtain better performance on the training set. The more use of the gradient descent search can generally result in better performance.

TABLE III. RESULTS ON THE TRAINING SET [-3, 3] AND TEST SET FROM THE SAME DOMAIN [-3, 3]

Function	Method	Training: [-3,3]		Test 1: [-3,3]	
		20 points Mean±Std	100 points Mean±Std	20 points Mean±Std	100 points Mean±Std
f_1	GP	0.31±0.04	0.46±0.03	0.69±0.74	0.44±0.08
	3HGP	0.21±0.05	0.4±0.03	4.25±5.55	0.9±0.98
	1HGP	0.23±0.05	0.41±0.03	4.67±10.69	0.67±0.37
	GGP1	0.3±0.05	0.45±0.03	0.69±0.62	0.45±0.08
	GGP3	0.31±0.04	0.46±0.03	0.74±0.86	0.45±0.07
	GGP5	0.31±0.04	0.46±0.03	0.59±0.46	0.44±0.06
	GP-L	0.31±0.04	0.46±0.03	0.70±0.76	0.44±0.08
f_2	GP	22.48±18.43	13.12±16.71	64.31±61.55	17.93±25.85
	3HGP	15.73±12.79	11.32±14.58	66.05±81.84	19.72±28.34
	1HGP	19.5±11.23	12.95±15.58	103.78±193.93	20.87±27.78
	GGP1	25.22±17.15	25.81±18.21	62.93±47.89	41.3±30.91
	GGP3	21.3±18.21	16.2±17.55	55.98±54.97	24.79±29.5
	GGP5	21.68±17.08	14.11±17.28	55.66±50.28	20.44±28.18
	GP-L	22.41±18.39	13.07±16.68	63.96±61.26	17.90±25.88
f_3	GP	4.94±1.28	4.86±1.83	521.88±1024.94	6.44±7.37
	3HGP	2.65±0.96	2.61±1.6	272.82±573.9	3.01±1.61
	1HGP	3.2±0.94	2.78±1.45	340.5±708.52	3.36±1.63
	GGP1	4.79±1.6	4.63±2.28	257.52±525.63	6.25±6.41
	GGP3	4.94±1.65	4.49±2.08	397.16±760.98	5.71±4.46
	GGP5	4.95±1.55	4.71±1.91	297.35±603.3	6.86±8.33
	GP-L	4.91±1.27	4.83±1.83	522.09±1029.7	6.43±7.38
f_4	GP	0.46±0.21	0.61±0.1	2.17±1.82	0.53±0.08
	3HGP	0.16±0.1	0.26±0.13	7.97±14.2	0.32±0.27
	1HGP	0.25±0.18	0.34±0.15	6.97±14.32	0.34±0.14
	GGP1	0.52±0.21	0.63±0.06	2.14±2.11	0.54±0.04
	GGP3	0.43±0.2	0.62±0.1	2.31±1.67	0.53±0.08
	GGP5	0.46±0.2	0.63±0.08	2.45±1.97	0.54±0.06
	GP-L	0.46±0.21	0.61±0.1	2.16±1.80	0.52±0.08
f_5	GP	3.63±1.75	3.0±1.73	6.5±5.81	3.11±1.71
	3HGP	1.62±1.18	1.14±1.27	14.44±25.65	1.41±1.27
	1HGP	1.98±1.29	1.59±1.46	22.06±63.7	1.94±1.57
	GGP1	3.29±1.68	2.79±1.77	12.27±28.46	2.94±1.73
	GGP3	3.48±1.68	2.81±1.82	17.87±49.47	2.92±1.74
	GGP5	3.59±1.68	2.9±1.8	10.53±17.54	3.07±1.75
	GP-L	3.62±1.75	2.99±1.73	7.93±12.31	3.11±1.72

TABLE IV. RESULTS ON TEST SETS FROM AN EXTENDED DOMAIN AND A DIFFERENT DOMAIN

Function	Method	Test 2: [-6,6]		Test 3: [3,6]	
		20 points Mean±Std	100 points Mean±Std	20 points Mean±Std	100 points Mean±Std
f_1	GP	1.07±0.59	1.01±0.91	1.63±1.36	1.69±2.68
	3HGP	10.49±17.7	11.92±21.23	2.25±1.47	18.26±34.38
	1HGP	10.3±18.83	5.07±8.16	2.42±1.44	5.91±10.67
	GGP1	1.36±1.18	1.21±1.28	2.02±1.75	1.82±3.01
	GGP3	1.31±0.92	1.09±0.98	1.83±1.47	1.46±2.27
	GGP5	1.08±0.63	0.89±0.72	1.72±1.44	1.19±1.58
	GP-L	1.08±0.59	1.02±0.94	1.65±1.36	1.71±2.70
f_2	GP	1.35E5±1.20E5	2.01E4±3.81E4	1.00E5±8.88E4	3.34E4±6.34E4
	3HGP	1.53E5±1.12E5	4.39E4±1.07E5	1.13E5±8.46E4	5.85E4±8.32E4
	1HGP	1.76E5±1.04E5	3.36E4±4.67E4	1.26E5±7.59E4	5.55E4±7.80E4
	GGP1	1.60E5±1.07E5	5.83E4±4.64E4	1.20E5±8.09E4	1.03E5±8.57E4
	GGP3	1.27E5±1.20E5	3.39E4±4.60E4	9.40E4±8.79E4	6.00E4±8.40E4
	GGP5	1.37E5±1.17E5	2.55E4±4.17E4	1.05E5±8.85E4	4.47E4±7.49E4
	GP-L	1.35E5±1.20E5	2.01E4±3.82E4	1.00E5±8.88E4	3.36E4±6.34E4
f_3	GP	246.38±739.7	356.52±331.85	66.49±185.66	1106.98±1158.17
	3HGP	1491.35±6468.82	1137.23±2066.24	1083.57±2436.27	4211.07±1.02E4
	1HGP	624.56±1846.55	867.99±1573.61	446.82±669.87	2779.51±4305.0
	GGP1	111.63±167.87	404.82±338.83	64.84±173.01	1334.18±1314.26
	GGP3	73.41±92.12	429.53±341.66	72.97±197.79	1305.62±1281.04
	GGP5	117.52±165.18	379.49±307.54	65.18±154.46	1217.42±1237.69
	GP-L	230.23±711.17	353.34±331.08	66.86±188.11	1094.97±1147.64
f_4	GP	7.24±44.3	1.18±0.17	0.95±0.4	1.49±0.25
	3HGP	174.59±312.37	9.54±8.85	2.04±3.15	26.69±32.0
	1HGP	133.81±283.75	7.31±11.17	1.08±1.22	19.23±34.01
	GGP1	1.5±2.12	1.21±0.08	1.04±0.38	1.54±0.14
	GGP3	30.13±143.67	1.18±0.19	0.96±0.34	1.49±0.27
	GGP5	1.26±2.07	1.2±0.1	0.95±0.34	1.52±0.17
	GP-L	9.04±53.76	1.18±0.18	0.95±0.39	1.48±0.27
f_5	GP	999.09±400.83	1376.84±1199.46	2500.18±1042.42	4083.98±2727.85
	3HGP	1043.22±897.57	1419.74±2941.67	1837.17±1481.48	2496.72±3616.92
	1HGP	994.36±657.99	1229.34±1891.59	2331.78±1653.21	3004.41±3850.45
	GGP1	966.0±428.78	1005.44±990.95	2236.13±1088.61	3021.77±2375.16
	GGP3	998.29±422.46	1142.02±1158.56	2391.7±1038.66	3414.67±2759.59
	GGP5	1020.18±423.63	929.45±858.0	2535.38±1058.36	3002.95±2216.49
	GP-L	998.73±400.05	1374.59±1194.54	2502.66±1041.29	4074.88±2714.17

B. Results on the Test Set from the Same Domain

Table III shows that when applying the obtained GP trees (i.e. models) to unseen test set from the same domain as the training set [-3,3], the overall pattern of the results is different from that in the training set. A clear pattern is that the number of data points (20 or 100) considerably influences the generalisation of the algorithms.

When using 100 random points, 3HGP achieved the best performance in three of the five benchmark problems and second best in one benchmark problem. However, when using 20 random points, where the task is harder than using 100 points, 3HGP did not achieve the best results in any of the five benchmark problems, but obtained the worst result in one benchmark problem. Meanwhile, 1HGP obtained the worst results in three out of the five problems. In contrast, in all the five problems, the best performance was achieved by one of the three algorithms that only applied gradient descent on the best 20% individuals, i.e. GGP1, GGP3 and GGP5.

The results show that the algorithm that performed well on the training data can generalise well on unseen test data from the same domain when the number of data points is relatively large, but not when the number of data points is small. The reason is that this is a standard supervised learning process and overfitting occurred. Although this has not been seriously investigated for symbolic regression, this is consistent with results from classification [32], [33]. When using a small number of data points, applying gradient descent to every individual in every generation in 3HGP and 1HGP may lead to a poor generalisation, but applying only on the best individuals can result in better generalisation.

C. Results on the Test Set from an Extended Domain

The results of applying the obtained GP trees to unseen test data from an extended domain [-6, 6] are shown in Table IV by Test 2. It can be observed that all the algorithms have much larger MSEs than that in Table III, but it is not unexpected because the tasks from the extended domain are much harder than from the same domain [-3, 3].

From Table IV, it can be observed that regardless of the number of data points, 3HGP obtained the worst results in four of the five problems, which is an opposite pattern from that on the training set. The best performance was achieved by one of the algorithms where gradient descent was applied but not on all individuals. This pattern is similar to the first test set [-3, 3] but clearer, which further indicates that GP with gradient descent can have good generalisation to a similar/extended domain, but applying gradient descent to all individuals is not useful.

D. Results on the Test Set from a Different Domain

The results of applying the obtained GP trees to unseen test data from a different domain [3, 6] are shown in Table IV by Test 3. This task is much harder than all the other three tasks. The results show that all the algorithms have much higher MSEs than in the other three tasks.

The pattern here is not clear enough, but it can be seen that 3HGP performed worse than the other algorithms in general. The algorithms where gradient descent was not applied to all individuals achieved the best performance in most cases. This suggests that when applying to a different domain, the problem becomes much harder and it is difficult to develop a good algorithm to perform well. New approaches are needed to solve such problems.

TABLE V. SIGNIFICANCE TEST ON EXPERIMENTS WITH 20 POINTS

		Training		Test 1		Test 2		Test 3	
		T1 vs GP	T2 vs GGP5	T1 vs GP	T2 vs GGP5	T1 vs GP	T2 vs GGP5	T1 vs GP	T2 vs GGP5
f1	GP	=	=	=	=	=	=	=	=
	3HGP	-	-	+	+	+	+	+	+
	HGP	-	-	+	+	+	+	+	+
	GGP1	=	=	=	+	+	=	=	=
	GGP3	=	=	=	=	+	=	=	=
	GGP5	=	=	=	=	=	=	=	=
	GP-L	-	=	=	=	=	=	=	=
f2	GP	=	=	=	=	=	=	=	=
	3HGP	-	-	=	=	=	=	=	=
	HGP	=	-	=	+	+	+	=	=
	GGP1	=	=	=	=	=	=	=	=
	GGP3	=	=	=	=	=	=	=	=
	GGP5	=	=	=	=	=	=	=	=
	GP-L	-	=	-	=	=	=	+	=
f3	GP	=	=	=	=	=	=	=	=
	3HGP	-	-	=	=	+	+	+	+
	HGP	-	-	=	=	+	+	+	+
	GGP1	=	=	=	=	=	=	=	=
	GGP3	=	=	=	=	=	=	=	=
	GGP5	=	=	=	=	=	=	=	=
	GP-L	-	=	=	=	=	=	=	=
f4	GP	=	=	=	=	=	=	=	=
	3HGP	-	-	+	+	+	+	=	=
	HGP	-	-	+	+	+	+	=	=
	GGP1	=	+	=	=	=	=	=	+
	GGP3	=	-	=	=	+	+	=	=
	GGP5	=	=	=	=	=	=	=	=
	GP-L	-	=	=	=	=	=	=	=
f5	GP	=	=	=	=	=	=	=	=
	3HGP	-	-	=	=	=	=	-	-
	HGP	-	-	=	=	=	=	=	=
	GGP1	=	=	=	=	=	=	-	-
	GGP3	=	=	=	=	=	=	=	=
	GGP5	=	=	=	=	=	=	=	=
	GP-L	-	=	=	=	=	=	=	=

E. Results of Significance Tests

In order to further test and compare the generalisation ability of the seven algorithms, the on-parametric statistical significance test, Wilcoxon test, was used in the experiments, where the significance level is 0.05. The results of the significance tests for the experiments with 20 data points are shown in Table V and for the experiments with 100 data points are shown in Table VI. In the tables, T1 shows the results of the six GP with gradient decent algorithms against standard GP without using the gradient descent, and T2 shows the results of GGP5 against the other six algorithms. The "+" ("–") shows the MSE of the corresponding algorithm (i.e. in the corresponding row) is significantly larger (smaller) than that of standard GP (or GGP5). "=" means they are similar.

According to Tables V and VI, it can be seen that in the training set, HGP3 and HGP achieved significantly smaller MSE than GP and GGP5 in almost all cases. For Test 1, when having 20 data points, HGP3 and HGP did not achieve significantly better (i.e. smaller) results than GP and GGP5 on any function, but worse (i.e. larger) than GP and GGP5 in

TABLE VII. COMPUTATIONAL TIME (IN MILLISECOND)

20 data points					
Method	f_1	f_2	f_3	f_4	f_5
GP	103.32±59.9	142.54±73.87	132.91±71.49	126.53±96.83	116.29±79.15
HGP3	3587.28±1976.84	3046.05±1413.7	4244.9±1759.09	3915.93±2100.3	3766.34±1502.35
HGP	2081.34±943.8	2026.42±750.16	2436.51±1031.03	1862.91±731.23	2188.55±620.63
GGP1	533.73±333.92	817.19±390.44	889.21±437.44	432.46±213.97	799.3±334.04
GGP3	263.28±165.61	398.86±168.17	365.56±207.24	280.38±170.92	382.85±196.51
GGP5	194.05±124.72	306.42±143.56	309.9±165.34	239.63±150.53	303.66±179.4
GP-L	99.98±47.07	147.76±72.46	134.23±70.05	122.33±90.62	121.52±88.95
100 data points					
Method	f_1	f_2	f_3	f_4	f_5
GP	360.12±133.94	397.59±90.0	379.71±106.58	263.62±60.05	328.81±85.57
HGP3	3.09E4±1.56E4	2.17E4±1.36E4	4.14E4±2.00E4	2.56E4±1.54E4	4.02E4±1.96E4
HGP	1.28E4±6194.15	1.34E4±7409.13	2.11E4±9481.55	9890.35±5524.53	1.62E4±6843.35
GGP1	2489.25±2019.59	4438.2±2674.3	6246.13±3263.4	874.18±833.27	4166.12±2183.24
GGP3	1064.9±731.18	1757.02±1032.35	2479.83±1245.54	591.06±390.34	1630.09±831.99
GGP5	717.13±480.33	1107.43±548.18	1447.06±657.79	571.86±448.25	1195.19±673.08
GP-L	355.64±118.13	436.91±223.39	462.19±197.12	278.82±50.64	389.09±219.44

TABLE VI. SIGNIFICANCE TEST ON EXPERIMENTS WITH 100 POINTS

		Training		Test 1		Test 2		Test 3	
		T1 vs GP	T2 vs GGP5	T1 vs GP	T2 vs GGP5	T1 vs GP	T2 vs GGP5	T1 vs GP	T2 vs GGP5
f1	GP	=	=	=	=	=	=	=	=
	3HGP	-	-	+	+	+	+	+	+
	HGP	-	-	+	+	+	+	+	+
	GGP1	-	-	=	=	=	=	=	=
	GGP3	=	=	=	=	=	+	=	=
	GGP5	=	=	=	=	=	=	=	=
	GP-L	-	=	=	=	=	=	=	=
f2	GP	=	=	=	=	=	=	=	=
	3HGP	=	-	=	=	=	=	+	=
	HGP	=	=	=	=	=	=	=	=
	GGP1	+	+	+	+	+	+	+	+
	GGP3	=	=	=	=	=	=	=	=
	GGP5	=	=	=	=	=	=	=	=
	GP-L	-	=	=	=	=	=	=	=
f3	GP	=	=	=	=	=	=	=	=
	3HGP	-	-	-	-	+	+	+	+
	HGP	-	-	-	-	+	+	+	+
	GGP1	=	=	=	=	=	=	=	=
	GGP3	=	=	=	=	=	=	=	=
	GGP5	=	=	=	=	=	=	=	=
	GP-L	-	=	=	=	=	=	=	=
f4	GP	=	+	=	=	=	=	=	=
	3HGP	-	-	-	-	+	+	+	+
	HGP	-	-	-	-	+	+	+	+
	GGP1	-	=	=	=	=	=	=	=
	GGP3	-	=	=	=	=	=	=	=
	GGP5	-	=	=	=	=	=	=	=
	GP-L	-	=	=	=	=	=	=	=
f5	GP	=	=	=	=	=	+	=	+
	3HGP	-	-	-	-	-	=	-	-
	HGP	-	-	-	-	-	=	-	=
	GGP1	=	=	=	=	=	=	-	=
	GGP3	=	=	=	=	-	=	=	=
	GGP5	=	=	=	=	-	=	-	=
	GP-L	-	=	=	=	=	+	=	+

some cases, which may indicate the appearance of overfitting. When having 100 data points, where overfitting is less likely than when having only 20 data points, HGP3 and HGP achieved significantly better results than GP and GGP5. This is consistent with our hypothesis that applying too many gradient decent steps may lead to relatively poor performance on unseen data. The results in Test 2 and Test 3 show that HGP3 and HGP in general cannot be well adopted to data from the an extended domain or from a different domain.

F. Computational Time

Table VII shows the average and standard deviation of the computational times used by each algorithm in the 100 independent runs, where the values are expressed in milliseconds.

According to Table VII, all the algorithms are reasonably fast and the largest average time is still less than 1 minute. Comparing different algorithms, 3HGP used the longest time in all cases, which is around 30 times longer than GP. The computational time gradually reduces from 3HGP, 1HGP, GGP1, GGP3, GGP5, GP-L, to GP, which is consistent with the number of applications of the gradient descent search.

V. CONCLUSIONS AND FUTURE WORK

The overall goal was to investigate whether GPGD for symbolic regression can generalise well on unseen problems in the same domain and can be effectively adapted to other domains. To achieve this, a number of experiments have been conducted on seven GPGD algorithms on five commonly used benchmark functions of varying difficulty. The algorithms were examined on training data, unseen test data from the same domain/interval, from an extended domain, and from a different domain with two different numbers of data points/instances. The experimental results show that on the training set, GP with more gradient descent performed better. The best algorithm on the training set generalised well on unseen data from the same domain when the number of data points was relatively large, but not when the number was small. For unseen data from an extended domain, applying gradient descent to only a small proportion of individuals in GP generalised better than applying to all individuals. All the algorithms could not be adapted well on unseen data from a different domain, since the task is very challenging. The results also showed that performing more gradient descent search always increased the computational cost, but not always better for generalisation or adaptation. Finding an appropriate way to apply gradient descent can not only improve its performance on training data, but also increase its generalisation ability, which is key for real-world tasks.

This paper provides a simple way to investigate the generalisation and domain adaptation ability of symbolic regression algorithms, but discovers very interesting findings, which opens the door for future research in this direction. In future, we will further investigate the key factors influencing

the generalisation and domain adaptation ability of symbolic regression algorithms. We also intend to propose a novel GP approach to symbolic regression, which can generalise well on unseen data/problems from the same domain or even adapted well on unseen data from different domains.

REFERENCES

[1] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone, *Genetic Programming – An Introduction; On the Automatic Evolution of Computer Programs and its Applications*. Morgan Kaufmann, 1998.

[2] J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*. Cambridge, MA, USA: MIT Press, 1992.

[3] I. Icke and J. Bongard, "Improving genetic programming based symbolic regression using deterministic machine learning," in *IEEE Congress on Evolutionary Computation*, 2013, pp. 1763–1770.

[4] M. Keijzer, "Improving symbolic regression with interval arithmetic and linear scaling," in *Genetic programming*. Springer, 2003, pp. 70–82.

[5] M. Kommenda, G. Kronberger, S. Winkler, M. Affenzeller, and S. Wagner, "Effects of constant optimization by nonlinear least squares minimization in symbolic regression," in *Proceeding of the fifteenth annual conference companion on Genetic and evolutionary computation conference companion (GECCO'13)*. ACM, 2013, pp. 1121–1128.

[6] W. B. Langdon, R. Poli, N. F. McPhee, and J. R. Koza, "Genetic programming: An introduction and tutorial, with a survey of techniques and applications," in *Computational Intelligence: A Compendium*. Springer, 2008, pp. 927–1028.

[7] I. Arnaldo, K. Krawiec, and U.-M. O'Reilly, "Multiple regression genetic programming," in *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation*, ser. GECCO '14, 2014, pp. 879–886.

[8] V. Dabhi and S. Chaudhary, "Empirical modeling using genetic programming: a survey of issues and approaches," *Natural Computing*, pp. 1–28, 2014.

[9] M. Zhang and W. Smart, "Learning weights in genetic programs using gradient descent for object recognition," in *Applications of Evolutionary Computing*. Springer, 2005, pp. 417–427.

[10] A. Topchy and W. F. Punch, "Faster genetic programming based on local gradient search of numeric leaf values," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'01)*. Morgan Kaufmann, 2001, pp. 155–162.

[11] M. Graff, R. Pena, and A. Medina, "Wind speed forecasting using genetic programming," in *IEEE Congress on Evolutionary Computation (CEC)*, 2013.

[12] M. Graff, A. Graff-Guerrero, and J. Cerda-Jacobo, "Semantic crossover based on the partial derivative error," in *Genetic Programming*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2014, vol. 8599, pp. 37–47.

[13] M. Zhang and W. Smart, "Genetic programming with gradient descent search for multiclass object classification," in *Genetic Programming*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, vol. 3003, pp. 399–408.

[14] L. Vanneschi, D. Rochat, and M. Tomassini, "Multi-optimization improves genetic programming generalization ability," in *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '07, 2007, pp. 1759–1759.

[15] M. Castelli, L. Manzoni, S. Silva, and L. Vanneschi, "A comparison of the generalization ability of different genetic programming frameworks," in *IEEE Congress on Evolutionary Computation (CEC)*, 2010, pp. 1–8.

[16] 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*. ACM, 2009, pp. 1139–1146.

[17] S. Mahler, D. Robilliard, and C. Fonlupt, "Tarpeian bloat control and generalization accuracy," in *Genetic Programming*. Springer, 2005, pp. 203–214.

[18] S. Gustafson, E. K. Burke, and N. Krasnogor, "On improving genetic programming for symbolic regression," in *IEEE Congress on Evolutionary Computation*, vol. 1, 2005, pp. 912–919.

[19] E. Bautu, A. Bautu, and H. Luchian, "Symbolic regression on noisy data with genetic and gene expression programming," in *7th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, 2005, pp. 1–4.

[20] N. Q. Uy, N. X. Hoai, M. ONeill, R. I. McKay, and E. Galván-López, "Semantically-based crossover in genetic programming: application to real-valued symbolic regression," *Genetic Programming and Evolvable Machines*, vol. 12, no. 2, pp. 91–119, 2011.

[21] M. Schmidt and H. Lipson, "Symbolic regression of implicit equations," in *Genetic Programming Theory and Practice VII*, 2010, pp. 73–85.

[22] G. F. Smits, E. Vladislavleva, and M. E. Kotanchek, "Scalable symbolic regression by continuous evolution with very small populations," in *Genetic Programming Theory and Practice VIII*, 2011, pp. 147–160.

[23] A. W. Ragalo and N. Pillay, "A building block conservation and extension mechanism for improved performance in polynomial symbolic regression tree-based genetic programming," in *4th World Congress on Nature and Biologically Inspired Computing (NaBIC)*, 2012, pp. 123–129.

[24] S. J. Pan, I. Tsang, J. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Transactions on Neural Networks*, vol. 22, no. 2, pp. 199–210, 2011.

[25] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[26] R. Xia, C. Zong, X. Hu, and E. Cambria, "Feature ensemble plus sample selection: Domain adaptation for sentiment classification," *IEEE Intelligent Systems*, vol. 28, no. 3, pp. 10–18, 2013.

[27] M. Ghifary, W. B. Kleijn, and M. Zhang, "Domain adaptive neural networks for object recognition," in *13th Pacific Rim International Conference on Artificial Intelligence*, 2014, pp. 898–904.

[28] M. Long, J. Wang, G. Ding, J. Sun, and P. Yu, "Transfer feature learning with joint distribution adaptation," in *IEEE International Conference on Computer Vision (ICCV)*, 2013, pp. 2200–2207.

[29] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014, conference version of the paper. [Online]. Available: https://hal.inria.fr/hal-00911179

[30] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *Computer Vision ECCV 2010*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, vol. 6314, pp. 213–226.

[31] L. Feng, Y. Ong, M. Lim, and I. Tsang, "Memetic search with inter-domain learning: A realization between cvrp and carp," *IEEE Transactions on Evolutionary Computation*, no. 99, p. Online:10.1109/TEVC.2014.2362558., 2014.

[32] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. John Wiley & Sons, 2012.

[33] P. Espejo, S. Ventura, and F. Herrera, "A survey on the application of genetic programming to classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 40, no. 2, pp. 121–144, 2010.