SAMPLING METHOD FOR EVOLVING MULTIPLE SUBPOPULATIONS IN GENETIC PROGRAMMING

Chu Thi Huong[1], *Nguyen Quang Uy*[1]

Abstract

Sampling techniques are the techniques that use a subset of the training data instead of the full data. These approaches have recently used in Genetic Programing (GP) to speed up the evolving process and to improve its performance. In this paper, we propose a new sampling technique that evolves multiple subpopulations on the sampled datasets. A number of subdatasets are sampled from the training data and a subpopulation is evolved on each of these datasets for a predefined generations. The subpopulations are then combined to form a full population that is evolved on the full training dataset for the rest generations. We tested this sampling technique (shorted as SEMS-GP) on seventeen regression problems and compared its performance to standard GP and two recently proposed sampling techniques (Interleaved Sampling and Random Interleaved). The experimental results show that SEMS-GP achieved better performance compared to other tested methods. Particularly, the training error and the size of the solutions found by SEMS-GP are often significantly better than those found by others.

Index terms

Genetic Programming, Sampling technique, Initialization, Code bloat

1. Introduction

Genetic Programming (GP) is an Evolutionary Algorithm that automatically finds the solutions of unknown structure for a problem [1], [2]. Although, GP has been successfully applied to many real-world problems thanks to its flexible representation, its main drawback is the computational overhead specially on large datasets. To lessen this drawback of GP, researchers have recently studied a number of sampling techniques [3]. The sampling techniques substitute the training data set with a representative subset of smaller size and hence reduce the evaluation cost of the GP algorithm. Depending on whether or not the sampled subsets are modified during the evolutionary process, the sampling techniques in GP can be classified into two main categories: static sampling and dynamic sampling [3], [4]. In static sampling, the sampled subsets are fixed [5], [6] while in dynamic sampling, they are alternated during the evolution [7], [8], [9], [10], [11], [12], [13], [14].

[1] Faculty of Information Technology, Le Quy Don Technical University

Although sampling techniques have been proven to be beneficial for GP performance, especially in difficult real world problems, the previous techniques have often only used one population to evolve on the sampled dataset. In this paper, we proposed a sampling technique that evolves multiple populations on the sampled datasets. Intuitively, evolving multiple populations may help to reduce the training error, preserve better population's diversity and subsequently, improve the generalization ability of the GP solutions. Our method is called *Sampling for Evolving Multiple Subpopulations in Genetic Programming* (SEMS-GP). SEMS-GP divides the whole evolutionary process into two phases. In the first phase, the subdatasets are sampled from the training (full) dataset, and these subdatasets are then used to evolve the subpopulations. The individuals in the subpopulations are trained to fit a part of the training data and then combined to form the full population for the next phase. In the second phase, the combined population is evolved on the full dataset as standard GP until the last generation. Each phase of SEMS-GP uses static data. However, the training data is transferred from the subdatasets to the full dataset at beginning of the second phase. Thus, SEMS-GP can be considered as a hybrid of static and dynamic sampling.

We tested SEMS-GP on seventeen symbolic regression problems and compared its performance with standard GP and two recently proposed dynamic sampling techniques (Interleaved Sampling and Random Interleaved) [12]. The experimental results evidenced the benefit of our sampling technique in improving the training error and reducing the solution's size compared to the other systems. In the next section, we briefly review the related work on the sampling techniques in GP. The proposed sampling technique is presented in Section 3. Section 4 presents the experimental settings adopted in this paper, with the results presented and discussed in Section 5. Finally, Section 6 concludes the paper and highlights some future work.

2. Related Work

Sampling techniques have been investigated by many researchers. Various methods have been proposed with the goal of improving GP generalization, reducing the code bloat and saving computational time. In general, sampling techniques can be grouped into two main categories: static sampling and dynamic sampling [3].

2.1. Static Sampling Techniques

In the early work, Gathercole and Ross proposed a static sampling technique called Historical Subset Selection (HSS) [5] for GP. In each generation, HSS formed the training set by using the best individual of the previous GP run to select the misclassified fitness cases from the full dataset. After that, Hitoshi Iba [6] applied Bagging and Boosting techniques in machine learning to GP. Several subpopulations are evolved on different training sets that are generated by selecting from the base dataset with replacement. The best individuals in each subpopulation are then voted to form the final output for the test data.

2.2. Dynamic Sampling Techniques

Most of the sampling techniques proposed for GP are dynamic techniques. Perhaps, the simplest dynamic sampling was Random Subset Selection (RSS) [7], [8]. RSS randomly selected a subset of the training data in which the probability being selected of each sample is the same in every generation. Another dynamic technique was Dynamic Subset Selection (DSS) [7]. In DSS, the probability being selected of each sample is calculated based on the sample's difficulty (incremented each time the sample is misclassified by one of the GP trees) and the number of generations since the last time the sample was selected. The experimental results showed that RSS and DSS are often less time consuming than GP (about 20%) while their performance is competitive [7].

RSS and DSS were then extended to the hierarchical sampling technique [9], [10] that divided the sampling process into three hierarchical levels. All training data is partitioned into small blocks that match the capacity of the computer memory (RAM) at level 0. Then a block is selected and read into RAM based on RSS method at level 1. Next, the selected block is treated as a full training data set and DSS method is used to select data subset from this set during the evolution. An extension of the hierarchical sampling was the Balanced Block DSS [10] that alters mainly the level 0 block selection to obtain a balanced block in level 1.

Later, Gonccalves et al. proposed two variants of RSS for controlling overfitting [11]. The first variant is similar to the original version in which the size of the random subset is fixed. The second variant allows to define a rate at which new random subset can be selected. Three datasets were used for testing and the results indicated that the second version of RSS reduces overfitting and finds less complex solutions than standard GP.

Recently, Gonccalves et al. proposed an interleaved sampling approach [12]. This approach alternately uses the full dataset and a single sample as the training data in each generation. The experimental results showed that this method improves the generalization ability when compared to standard GP. Interleaved sampling was then used to reduce GP code growth by Azad et al. [13]. In this paper, we propose a new sampling technique for initialising GP population. The detailed description of our method will be presented in the next section.

3. Proposed method

This section presents the proposed sampling technique: *Sampling for Evolving Multiple Subpopulations in Genetic Programming* (SEMS-GP). The structure of the GP system using sampling technique, SEMS-GP, is presented in Figure *1*. SEMS-GP splits the evolution into two phases. In the first phase, k subdatasets are sampled from the original data using sampling without replacement technique. These subdatasets are used to evolve k GP subpopulations. After g' generations, the subpopulations are combined to form an initial population for the second phase. In the second phase, GP evolves the combined population on the full dataset as standard GP.

Specifically, let N, G and $|D|$ be the population size, the number of generations and the size of the training dataset D, respectively, then the size of the subdatasets is $|D|/k$ and the size of the subpopulations in SEMS-GP is $p = N/k$. The subpopulations are evolved on each subdatatset over g' ($g' < G$) generations and the full population is evolved using the full dataset (D) over g ($g = G - g'$) generations. It should be noted that the population size (N) and total number of generations of SEMS-GP (G) are the same as in standard GP.

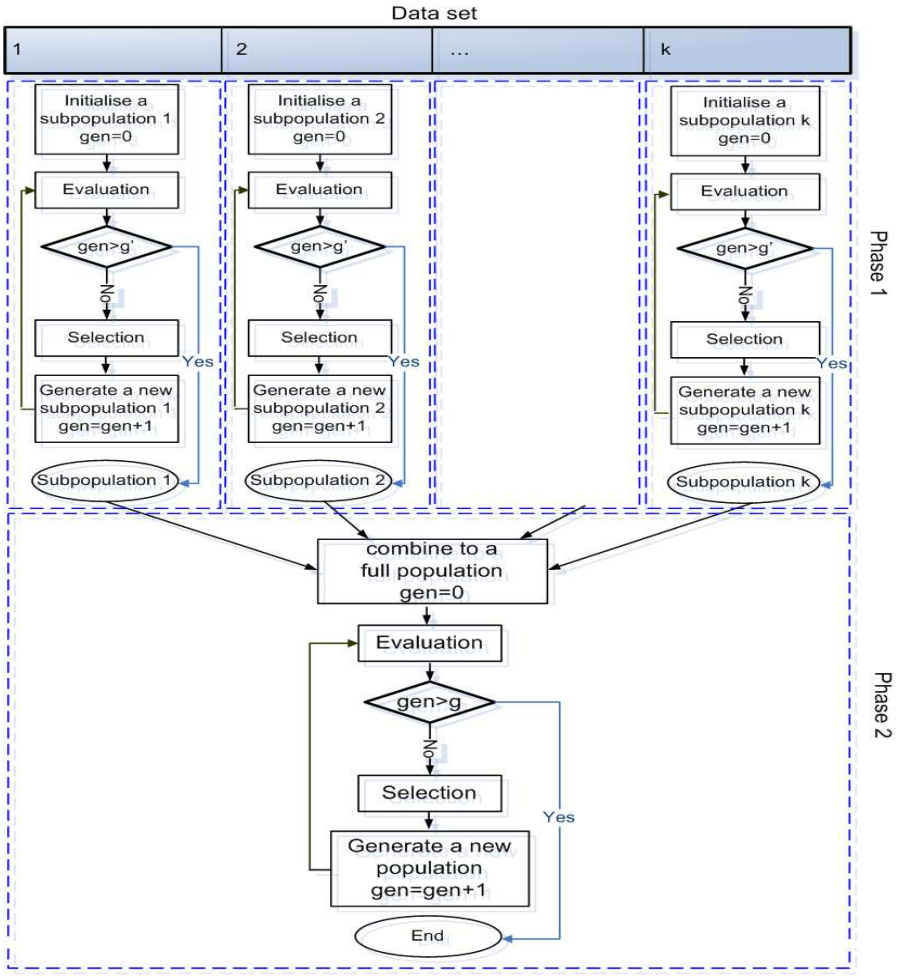

Fig. 1. *Structure of GP system using the sampling technique (SEMS-GP).*

4. Experimental Settings

We compared SEMS-GP with standard GP (referred to as GP) and two most recently proposed sampling techniques [12]: *Interleaved sampling* (referred to as Interleaved) and *Random Interleaved sampling* (shorted as RI 50%). Interleaved sampling alternatively used the full dataset and a single sample as the training data in each generation. In

RI 50%, a random number in [0,1] is generated in each generation and this number is used to decide whether a single sample or the full dataset is selected to assess individual's fitness. We tested these methods on 17 regression problems including eleven GP benchmark problems recommended in the literature [15] and six problems taken from UCI machine learning dataset [16]. The detailed descriptions of the tested problems including its name, its abbreviation, number of features, number of training and testing samples are shown in Table *1*.

Table 1. *Problems for testing SEMS-GP*

Shorthanded	Name	Features	Training	Testing
A. **Benchmarking Problems**				
F1	vladislavleva-4	5	500	500
F2	vladislavleva-5	3	300	2640
F3	vladislavleva-7	2	300	1000
F4	korns-1	5	1000	1000
F5	korns-2	5	1000	1000
F6	korns-3	5	1000	1000
F7	korns-4	5	1000	1000
F8	korns-11	5	1000	1000
F9	korns-12	5	1000	1000
F10	korns-14	5	1000	1000
F11	korns-15	5	1000	1000
B. **UCI Problems**				
F12	airfoil_self_noise	5	800	703
F13	ccpp	4	1000	1000
F14	3D_spatial_network	3	750	750
F15	protein_Tertiary_Structure	9	1000	1000
F16	concrete	8	500	530
F17	Appliances_energy_prediction	26	10500	9235

The experimental GP parameters are shown in Table *2*. These are the typical settings often used by GP researchers. The raw fitness is the absolute error on all fitness cases. Therefore, smaller values are better. The number of subdatatsets (k) was set at 5 in this paper [1]. The number of generations for evolving subpopulation (g') was set at $t\%$ of G. Four values of t including 10%, 20%, 30% and 50% were tested and SEMS-GP is shortened as SEMS-GP10, SEMS-GP20, SEMS-GP30 and SEMS-GP50, respectively. Note that selecting $t = 0\%$ would be equivalent to GP. The training data was randomized into two sets: the fitness evaluation set (80%), and the validation set (20%). In each generation, the best individual on the training set is selected and evaluated on the validation set. The best individual on the validation set was recorded as the final solution and was tested on the testing set. For each problem and each parameter setting, 50 runs were performed.

For statistical analysis, a Wilcoxon signed rank test with a confident level of 95% was used on the results in the following tables. The test values (p-values) are attached

[1]This value was calibrated from the experiments.

Table 2. Evolutionary parameter values.

Parameter	Value
Population size	500
Generations	100
Selection	Tournament
Tournament size	3
Crossover, mutation probability	0.9; 0.1
Function set	$+, -, *, /, sin, cos$
Terminal set	$X_1, X_2, ..., X_n$
Initial Max depth	6
Max depth	17
Max depth of mutation tree	15
Raw fitness	mean absolute error on all fitness cases
Trials per treatment	50 independent runs for each value
Elitism	Copy the best individual to the next generation.

in each result tables, and if the test shows that SEMS-GP is significantly better than GP, this result is marked + at the end. Conversely, the result is marked - at the end.

5. Results and Discussion

To assess the performance of the tested methods, four popular performance metrics in GP including training error, testing error, the size of the solutions and the evolution time were used.

5.1. Training Error

The average of the training error over 50 runs is presented in Table *3*. The table shows that SEMS-GPs (abbreviated for all configurations of SEMS-GP) often achieved the best results. The values obtained by SEMS-GPs are usually smaller than GP (excepts on function F4 and F5). Precisely, SEMS-GP10, SEMS-GP20, SEMS-GP30 and SEMS-GP50 are better than GP on 14, 14, 14 and 12 problems. Conversely, the training error of Interleaved and RI 50% are much worse compared to GP. These results are consistent with the results in [12] where Interleaved and RI 50% were reported to reduce the running time of GP system but not to improve the training error.

The Wilcoxon signed rank test also confirms that SEMS-GPs are often significantly better than GP. Specially, SEMS-GP10 is significantly better than GP on 7 problems and the remaining configurations (SEMS-GP20, SEMS-GP30 and SEMS-GP50) are significantly better than GP on 10 problems. Conversely, GP is significantly better than SEMS-GPs on only one problem (F5 with SEMS-GP30). For Interleaved and RI 50%, the statistical test shows that their training error is significantly worse than GP on most tested problems.

Table 3. *The mean best fitness on all training data and p-values of Wilcoxon signed rank test*

Pro	GP	Interleaved		RI 50%		SEMS-GP10		SEMS-GP20		SEMS-GP30		SEMS-GP50	
		Value	P-value	Value	P-value	Value	P-value	Value	P-value	Value	P-value	Value	P-value
A. **Benchmarking Problems**													
F1	0.13	0.15^{-}	0.00	0.14^{-}	0.00	**0.13**	0.30	$\mathbf{0.12}^{+}$	0.00	$\mathbf{0.12}^{+}$	0.00	$\mathbf{0.12}^{+}$	0.00
F2	0.09	0.15^{-}	0.00	0.15^{-}	0.00	**0.09**	0.90	**0.09**	0.33	**0.09**	0.91	0.10	0.11
F3	1.58	2.07^{-}	0.00	2.03^{-}	0.00	**1.52**	0.28	$\mathbf{1.41}^{+}$	0.01	$\mathbf{1.41}^{+}$	0.02	$\mathbf{1.36}^{+}$	0.00
F4	4.53	5.49^{-}	0.01	4.96	0.19	5.66	0.08	5.15	0.15	5.15	0.38	4.95	0.69
F5	2.09	9.27^{-}	0.00	9.01^{-}	0.00	2.11	0.75	3.26	0.43	3.75^{-}	0.02	3.08	0.31
F6	12.97	23.71^{-}	0.00	21.06^{-}	0.00	$\mathbf{7.14}^{+}$	0.00	$\mathbf{6.84}^{+}$	0.00	$\mathbf{7.22}^{+}$	0.00	$\mathbf{6.41}^{+}$	0.00
F7	0.10	0.13^{-}	0.03	0.22^{-}	0.00	0.10	0.26	**0.09**	0.32	**0.07**	0.19	**0.08**	0.35
F8	7.28	7.47^{-}	0.00	7.53^{-}	0.00	**7.17**	0.13	$\mathbf{7.08}^{+}$	0.00	$\mathbf{7.08}^{+}$	0.00	$\mathbf{6.96}^{+}$	0.00
F9	0.89	0.92^{-}	0.00	0.92^{-}	0.00	$\mathbf{0.86}^{+}$	0.00	$\mathbf{0.84}^{+}$	0.00	$\mathbf{0.83}^{+}$	0.00	$\mathbf{0.82}^{+}$	0.00
F10	102.96	116.14^{-}	0.05	116.75^{-}	0.00	$\mathbf{39.02}^{+}$	0.00	$\mathbf{37.73}^{+}$	0.00	$\mathbf{37.22}^{+}$	0.00	$\mathbf{36.03}^{+}$	0.00
F11	2.24	2.56	0.07	2.72	0.07	$\mathbf{1.80}^{+}$	0.04	$\mathbf{1.06}^{+}$	0.00	$\mathbf{0.73}^{+}$	0.00	$\mathbf{0.44}^{+}$	0.00
B. **UCI Problems**													
F12	11.55	18.95^{-}	0.00	17.85^{-}	0.00	**11.24**	0.87	**11.54**	0.55	**9.93**	0.18	**10.18**	0.16
F13	10.17	19.31^{-}	0.00	19.38^{-}	0.00	**9.98**	0.87	11.56	0.37	10.60	0.50	11.39	0.28
F14	13.64	17.87^{-}	0.00	17.26^{-}	0.00	$\mathbf{9.37}^{+}$	0.00	$\mathbf{8.28}^{+}$	0.00	$\mathbf{7.44}^{+}$	0.00	$\mathbf{6.04}^{+}$	0.00
F15	4.42	4.96^{-}	0.00	4.94^{-}	0.00	**4.42**	0.56	**4.35**	0.28	**4.40**	0.93	4.44	0.53
F16	12.81	24.43^{-}	0.00	19.40^{-}	0.00	$\mathbf{9.93}^{+}$	0.00	$\mathbf{9.30}^{+}$	0.00	$\mathbf{8.88}^{+}$	0.00	$\mathbf{7.40}^{+}$	0.00
F17	47.32	48.87^{-}	0.00	48.93^{-}	0.00	$\mathbf{46.54}^{+}$	0.00	$\mathbf{46.15}^{+}$	0.00	$\mathbf{45.84}^{+}$	0.00	$\mathbf{45.38}^{+}$	0.00

Table 4. *The median of testing error and p-values of Wilcoxon signed rank test*

Pro	GP	Interleaved		RI 50%		SEMS-GP10		SEMS-GP20		SEMS-GP30		SEMS-GP50	
		Value	P-value	Value	P-value	Value	P-value	Value	P-value	Value	P-value	Value	P-value
A. **Benchmarking Problems**													
F1	0.14	0.14^{-}	0.03	0.14^{-}	0.02	**0.14**	0.94	**0.13**	0.38	0.14	0.08	0.15^{-}	0.04
F2	0.21	0.28^{-}	0.00	0.28^{-}	0.00	**0.20**	0.97	**0.20**	0.48	0.21	0.32	0.23^{-}	0.01
F3	2.10	2.57^{-}	0.00	2.54^{-}	0.00	2.22	0.17	**2.08**	0.91	2.17	0.96	2.46^{-}	0.01
F4	7.08	7.23	0.20	7.21	0.75	7.43	0.16	**7.02**	0.43	7.29	0.50	7.38	0.55
F5	1.59	2.00^{-}	0.00	4.01^{-}	0.00	1.63	0.96	1.60	0.63	1.64^{-}	0.03	1.71	0.09
F6	48.05	99.25	0.07	98.37	0.12	**38.27**	0.19	**39.04**	0.47	90.38	0.40	77.78	0.34
F7	0.08	0.09	0.11	0.10^{-}	0.00	**0.05**	0.18	**0.08**	0.27	**0.06**	0.09	0.09	0.28
F8	7.32	7.36	1.00	7.43^{-}	0.03	**7.32**	0.36	**7.31**	0.38	7.34	0.39	7.37	0.51
F9	0.87	0.87^{-}	0.05	0.87^{-}	0.02	0.87^{-}	0.02	0.89^{-}	0.00	0.89^{-}	0.00	0.89^{-}	0.00
F10	129.64	$\mathbf{122.52}^{+}$	0.01	**122.97**	0.06	130.36^{-}	0.01	130.03	0.08	130.13^{-}	0.03	**128.48**	0.93
F11	4.98	5.38	0.07	5.60	0.10	**3.45**	0.11	$\mathbf{3.24}^{+}$	0.00	$\mathbf{3.24}^{+}$	0.00	$\mathbf{3.24}^{+}$	0.00
B. **UCI Problems**													
F12	25.26	27.78	0.06	26.45^{-}	0.05	**22.33**	0.92	**22.49**	0.22	**21.04**	0.98	**17.49**	0.91
F13	8.28	17.31^{-}	0.00	18.44^{-}	0.00	8.77	0.93	9.35	0.40	10.02	0.61	9.71	0.34
F14	10.10	**9.54**	0.97	**9.36**	0.72	**9.82**	0.86	**9.57**	0.45	**9.59**	0.23	12.63^{-}	0.00
F15	4.39	4.84^{-}	0.00	4.81^{-}	0.00	**4.36**	0.34	**4.37**	0.27	**4.39**	0.63	4.57^{-}	0.02
F16	17.48	21.14^{-}	0.04	19.23	0.37	**17.35**	0.81	**17.26**	0.75	17.51	0.55	20.05	0.30
F17	44.17	44.78^{-}	0.01	44.89^{-}	0.00	44.66	0.17	44.81	0.05	44.88^{-}	0.00	44.94^{-}	0.01

5.2. Testing Error

In machine learning, the generalization ability is perhaps the most desirable property of a leaner. In each GP run, the best individual on the validation set was selected and evaluated on the test data set. The median value of the testing error over 50 runs is shown in Table *4*. It can be seen that SEMS-GPs achieved better results than GP. In fact, SEMS-GP10, SEMS-GP20, SEMS-GP30 and SEMS-GP50 are better than GP on 10, 12, 5 and 3 problems, respectively. However, the performance of SEMS-GPs on the testing data are not as convincing as on the training data. One possible reason is that SEMS-GPs are overfitted on some problems. Further research will be conducted to investigate this hypothesis.

Table *4* also shows that two interleaved sampling techniques are often significantly worse than GP. Particularly, Interleaved is significantly worse than GP on 9 problems and RI 50% is significantly worse than GP on 11 problems. In general, the results showed that SEMS-GPs achieved better performance on unseen data compared to GP and two recently proposed sampling techniques (Interleaved and RI 50%) on the tested problems.

Table 5. The average of solution's size and p-values of Wilcoxon signed rank test

Pro	GP	Interleaved		RI 50%		SEMS-GP10		SEMS-GP20		SEMS-GP30		SEMS-GP50	
		Value	P-value	Value	P-value	Value	P-value	Value	P-value	Value	P-value	Value	P-value
A. **Benchmarking Problems**													
F1	84.4	**63.7**$^{+}$	0.03	**81.2**	0.75	**61.1**$^{+}$	0.04	**83.0**	0.85	**71.3**	0.21	**83.0**	0.94
F2	120.3	**110.1**	0.42	128.8	0.40	**98.5**$^{+}$	0.04	129.7	0.40	**114.7**	0.40	**113.9**	0.51
F3	158.1	**118.3**$^{+}$	0.00	**92.9**$^{+}$	0.00	**100.6**$^{+}$	0.00	**131.9**$^{+}$	0.02	**126.8**	0.06	**135.6**$^{+}$	0.01
F4	159.4	**156.0**	0.79	**155.8**	0.69	160.0	0.76	**144.2**	0.43	**138.2**	0.23	**126.1**$^{+}$	0.01
F5	122.3	**80.0**$^{+}$	0.00	**84.1**$^{+}$	0.01	**120.8**	0.92	**118.4**	0.96	**109.8**	0.20	**114.2**	0.45
F6	126.6	148.5	0.11	143.9	0.21	**105.3**	0.09	129.9	0.82	**101.8**$^{+}$	0.05	**118.5**	0.33
F7	131.2	136.9	0.84	**127.2**	0.63	**102.5**$^{+}$	0.01	131.3	0.98	138.5	0.36	**120.3**	0.19
F8	138.7	141.6	0.63	**116.5**	0.09	**103.0**$^{+}$	0.01	166.5	0.15	**137.2**	0.60	162.1	0.19
F9	72.6	88.4	0.17	90.6^{-}	0.04	**34.0**$^{+}$	0.00	**54.3**	0.08	77.5	0.80	94.0^{-}	0.01
F10	178.3	232.4^{-}	0.03	194.6	0.45	**21.3**$^{+}$	0.00	**79.2**$^{+}$	0.00	**91.8**$^{+}$	0.00	**134.9**	0.08
F11	150.1	**135.0**	0.20	**139.9**	0.32	**117.9**	0.06	**93.4**$^{+}$	0.00	**59.2**$^{+}$	0.00	**49.1**$^{+}$	0.00
B. **UCI Problems**													
F12	164.1	**142.2**	0.11	**155.5**	0.83	**156.2**	0.55	175.0	0.59	**152.2**	0.62	**142.1**	0.10
F13	180.0	**147.0**$^{+}$	0.01	**150.8**$^{+}$	0.02	**166.4**	0.24	**152.5**$^{+}$	0.02	**154.8**$^{+}$	0.03	**145.5**$^{+}$	0.00
F14	126.0	**119.3**	0.90	**87.8**$^{+}$	0.01	**23.1**$^{+}$	0.00	**53.4**$^{+}$	0.00	**67.2**$^{+}$	0.00	**100.7**	0.05
F15	173.6	**120.8**$^{+}$	0.00	**110.2**$^{+}$	0.00	**115.8**$^{+}$	0.00	**171.7**	0.71	**167.5**	0.74	**144.8**	0.17
F16	74.5	**63.9**	0.55	**59.0**	0.13	**72.7**	0.91	91.9	0.19	91.3	0.07	85.6	0.21
F17	67.3	**27.2**$^{+}$	0.00	**27.2**$^{+}$	0.00	**56.2**	0.28	**39.4**$^{+}$	0.00	**26.2**$^{+}$	0.00	**40.1**	0.26

5.3. Size of the solutions

We recorded the size of the solutions found by tested methods and calculated the average value of them over 50 runs. The results are then presented in Table *5*. Apparently, SEMS-GPs found smaller solutions on most tested problems compared to GP. In fact, SEMS-GP10, SEMS-GP20, SEMS-GP30 and SEMS-GP50 found simpler solutions than

GP on 16, 11, 14 and 14 problems, respectively. In terms of statistical test, SEMS-GPs found significantly smaller solution than GP on most tested problems while the vice versa is very rare. Similarly, Interleaved and RI 50% also achieved simpler solutions than GP. The statistical test shows that both Interleaved and RI 50% found significantly simpler solutions than GP on 6 problems while their solutions are more complex than GP on only one problem (F10 with Interleaved and F9 with RI 50%).

Figure *2* presents the average size of the population over the generations on two typical problems [2]. We can see that the population size of SEMS-GPs are often much smaller than GP. This is particularly true with two last configurations: SEMS-GP30 and SEMS-GP50. Interleaved and RI 50%, also reduce the population size at the end of the learning process. However, their population size is always greater than that of SEMS-GPs.

Table 6. Average running time in seconds and p-values of Wilcoxon signed rank test

Pro	GP	Interleaved		RI 50%		SEMS-GP10		SEMS-GP20		SEMS-GP30		SEMS-GP50	
		Value	P-value	Value	P-value	Value	P-value	Value	P-value	Value	P-value	Value	P-value
A. **Benchmarking Problems**													
F1	21.3	**10.4**$^+$	0.00	**11.3**$^+$	0.00	**16.1**$^+$	0.00	**18.1**$^+$	0.03	**18.2**$^+$	0.01	**13.1**$^+$	0.00
F2	13.1	**5.3**$^+$	0.00	**8.2**$^+$	0.00	**9.3**$^+$	0.00	**11.5**	0.06	**10.9**$^+$	0.00	**7.6**$^+$	0.00
F3	18.1	**9.8**$^+$	0.00	**10.2**$^+$	0.00	**14.4**$^+$	0.00	**14.2**$^+$	0.00	19.9	0.94	**9.4**$^+$	0.00
F4	33.7	**18.2**$^+$	0.00	**18.5**$^+$	0.00	**26.7**$^+$	0.01	**24.0**$^+$	0.00	**22.7**$^+$	0.00	**20.1**$^+$	0.00
F5	39.6	**20.5**$^+$	0.00	**20.2**$^+$	0.00	**36.7**	0.29	**34.2**$^+$	0.03	**32.6**$^+$	0.01	**26.2**$^+$	0.00
F6	52.3	**33.4**$^+$	0.00	**33.2**$^+$	0.00	**43.7**$^+$	0.02	**46.6**	0.35	**37.8**$^+$	0.00	**33.0**$^+$	0.00
F7	56.7	**31.0**$^+$	0.00	**29.2**$^+$	0.00	**44.1**$^+$	0.00	**52.8**	0.15	**51.0**$^+$	0.03	**37.3**$^+$	0.00
F8	76.8	**32.3**$^+$	0.00	**29.9**$^+$	0.00	**60.5**$^+$	0.01	77.6	0.67	**67.2**	0.18	**53.4**$^+$	0.00
F9	59.0	**28.7**$^+$	0.00	**27.0**$^+$	0.00	**45.1**$^+$	0.00	**54.9**	0.09	**44.7**$^+$	0.00	**34.2**$^+$	0.00
F10	79.3	**39.8**$^+$	0.00	**36.0**$^+$	0.00	**75.9**	0.66	89.3	0.34	**67.2**	0.13	**59.2**$^+$	0.00
F11	40.1	**24.2**$^+$	0.00	**24.0**$^+$	0.00	**24.8**$^+$	0.00	**25.0**$^+$	0.00	**20.4**$^+$	0.00	**16.3**$^+$	0.00
B. **UCI Problems**													
F12	32.5	**15.9**$^+$	0.00	**16.9**$^+$	0.00	**28.7**	0.12	32.7	0.71	**28.6**	0.09	**22.5**$^+$	0.00
F13	31.2	**18.2**$^+$	0.00	**18.6**$^+$	0.00	**22.3**$^+$	0.00	**27.6**	0.15	**27.2**	0.07	**20.4**$^+$	0.00
F14	56.4	**33.4**$^+$	0.00	**49.2**$^+$	0.00	**51.6**	0.24	**51.2**	0.57	**54.3**	0.63	**41.4**$^+$	0.00
F15	46.4	**24.8**$^+$	0.00	**35.1**$^+$	0.01	**32.4**$^+$	0.00	**46.2**	0.76	**45.0**	0.49	**30.3**$^+$	0.00
F16	17.4	**10.0**$^+$	0.00	**10.1**$^+$	0.00	**14.4**$^+$	0.01	**14.7**$^+$	0.01	**13.2**$^+$	0.00	**10.9**$^+$	0.00
F17	131.3	**41.7**$^+$	0.00	**29.0**$^+$	0.00	**113.5**	0.21	**93.3**$^+$	0.01	135.9	0.74	**116.8**	0.46

5.4. Running Time

The amount of time needed to complete a GP run is averaged over 50 runs and presented in Table *6*. This table shows that SEMS-GPs executed much faster than GP on most tested problems. This is the result of using a smaller training dataset in the first phase of SEMS-GPs. Furthermore, the population size of SEMS-GPs is also smaller than GP.

Compared to interleaved sampling techniques, SEMS-GPs executed slightly slower. This is not surprising since the interleaved techniques evaluated individuals on only

[2]Due to the space limitation, the figures of other problems are showed in the supplement at: https://github.com/chuthihuong/SEMS-GP

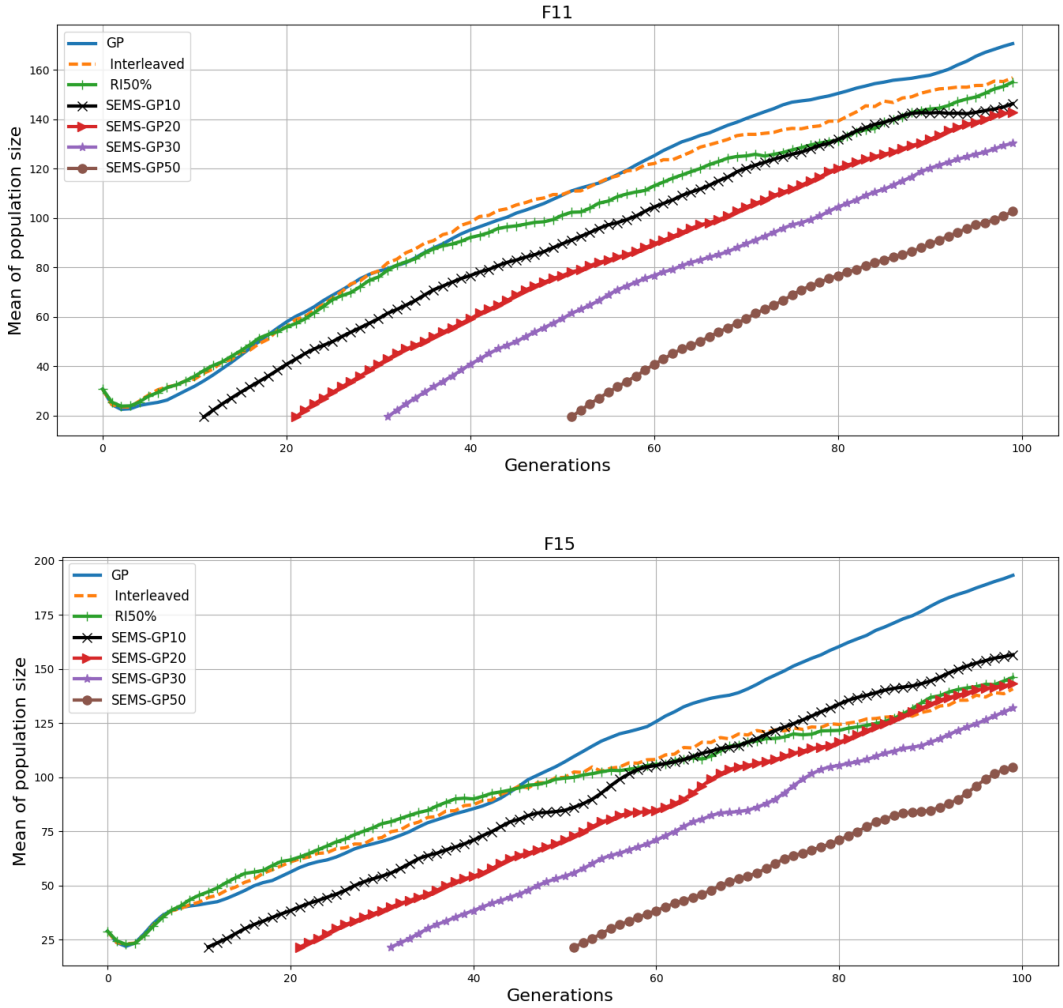

Fig. 2. The average size of population over the generations on F11 and F15.

single sample on half of the generations. Overall, the results of this section indicated that SEMS-GPs achieved better performance than GP. Moreover, SEMS-GPs found simpler solutions and executed faster than GP and two recently proposed dynamic sampling techniques (Interleaved and RI 50%).

6. Conclusions and Future Work

In this paper, we proposed a sampling technique for Genetic Programming (GP) called SEMS-GP. To assess the performance of SEMS-GP, four configurations of SEMS-GP were tested on seventeen regression problems. Experimental results showed that SEMS-GP achieved significantly better performance compared to GP and two recently proposed sampling techniques.

There are several new ideas for future research. First, we would like to examine methods for automatically self-adapting the number of generations for evolving the subpopulations in SEMS-GP. Second, the previous results showed that SEMS-GP are overfitted on some problems. Therefore, we want to combine SEMS-GP with a method for reducing overfitting [17] to improve its generalization ability. Last but not least, we would like to apply SEMS-GP to a wider range of problems including real-world applications such as time series forecasting.

Acknowledgment

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

References

[1] J. R. Koza 1994. Genetic programming as a means for programming computers by natural selection, *Statistics and Computing*, 4(2), 87–112.
[2] R. Poli, W. B. Langdon, N. F. McPhee, and J. R. Koza (2008). *A field guide to genetic programming*.
[3] H. Hmida, S. B. Hamida, A. Borgi, and M. Rukoz (2016). Sampling methods in genetic programming learners from large datasets: A comparative study, in *INNS Conference on Big Data*, 50–60.
[4] Y. Martínez, E. Naredo, L. Trujillo, P. Legrand, and U. López (2017). A comparison of fitness-case sampling methods for genetic programming, *Journal of Experimental & Theoretical Artificial Intelligence*, 29(6), 1203–1224.
[5] C. Gathercole and P. Ross (1994). Dynamic training subset selection for supervised learning in genetic programming, in *International Conference on Parallel Problem Solving from Nature*, 312–321.
[6] H. Iba (1999). Bagging, boosting, and bloating in genetic programming, in *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation*, 1053–1060.
[7] C. Gathercole (1998). *An investigation of supervised learning in genetic programming*.
[8] J. Zegklitz and P. Posík (2015). Model selection and overfitting in genetic programming: Empirical study, in *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, 1527–1528.
[9] R. Curry and M. Heywood (2004). Towards efficient training on large datasets for genetic programming, in *Conference of the Canadian Society for Computational Studies of Intelligence*, 161–174.
[10] R. Curry, P. Lichodzijewski, and M. I. Heywood (2007). Scaling genetic programming to large datasets using hierarchical dynamic subset selection, *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(4), 1065–1073.
[11] I. Gonçalves, S. Silva, J. B. Melo, and J. M. Carreiras (2012). Random sampling technique for overfitting control in genetic programming, in *European Conference on Genetic Programming*, 218–229.
[12] I. Gonçalves and S. Silva (2013), "Balancing learning and overfitting in genetic programming with interleaved sampling of training data," in *European Conference on Genetic Programming*, 73–84.
[13] R. M. A. Azad, D. Medernach, and C. Ryan (2014). Efficient interleaved sampling of training data in genetic programming, in *Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation*, 127–128.
[14] E. Galván-López, L. Vázquez-Mendoza, M. Schoenauer, and L. Trujillo (2017). Dynamic gp fitness cases in static and dynamic optimisation problems, in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 227–228.
[15] D. R. White, J. McDermott, M. Castelli, L. Manzoni, B. W. Goldman, G. Kronberger, W. Jaskowski, U.-M. O'Reilly, and S. Luke (2013). Better GP benchmarks: community survey results and proposals, *Genetic Programming and Evolvable Machines*, 14(1), 3–29.
[16] K. Bache and M. Lichman (2013). UCI machine learning repository, http://archive.ics.uci.edu/ml.
[17] J. Fitzgerald and C. Ryan (2011). Validation sets, genetic programming and generalisation, in *Research and Development in Intelligent Systems XXVIII*, 79–92.

Manuscript received 18-01-2018; Accepted 16-10-2018.

Chu Thi Huong received her B. Eng. degree in Applied Mathematics and Informatics from Hanoi University of Science and Technology and MSc Degree in Computer Science from Le Quy Don Technical University. Since 2002, she has been teaching at Faculty of IT, Le Quy Don Technical University. She is currently working on a PhD program at Faculty of IT, Le Quy Don Technical University. Her research interest are in the domain of Evolutionary Algorithms, Genetic Programming and Machine Learning.

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

PHƯƠNG PHÁP LẤY MẪU CHO TIẾN HÓA NHIỀU QUẦN THỂ CON TRONG LẬP TRÌNH DI TRUYỀN

Tóm tắt

Các kỹ thuật lấy mẫu là các kỹ thuật sử dụng tập con dữ liệu của tập dữ liệu huấn luyện thay cho toàn bộ dữ liệu. Gần đây, các cách tiếp cận này được sử dụng trong lập trình di truyền (GP) để tăng tốc độ tiến hóa và cải tiến hiệu suất của nó. Trong bài báo này, chúng tôi đề xuất một kỹ thuật lấy mẫu mới cho phép tiến hóa các quần thể con trên các tập dữ liệu lấy mẫu. Một số tập con dữ liệu được lấy mẫu từ tập dữ liệu huấn luyện và mỗi tập con dữ liệu đó được sử dụng để tiến hóa một quần thể con với một số thế hệ định nghĩa trước. Sau đó, các quần thể con được kết hợp thành một quần thể và được tiến hóa trên toàn bộ dữ liệu với các thế hệ còn lại. Chúng tôi kiểm định kỹ thuật này (ký hiệu là SEMS-GP) trên mười bảy bài toán hồi quy và so sánh hiệu suất của nó với hiệu suất của GP chuẩn và hai kỹ thuật lấy mẫu gần đây (Interleaved Sampling và Random Interleaved). Các kết quả thực nghiệm chỉ ra SEMS-GP đã đạt được hiệu suất tốt hơn các phương pháp khác. Đặt biệt, lỗi huấn luyện và kích thước lời giải của SEMS-GP thường tốt hơn so với các phương pháp khác.