Transfer Learning in Genetic Programming

Thi Thu Huong Dinh
Faculty of IT
Thu Dau Mot University
Binh Duong, Vietnam
Email: huongdtt2011@gmail.com

Thi Huong Chu
Faculty of IT
Le Quy Don University
Hanoi, Vietnam
Email: huongktqs@gmail.com

Quang Uy Nguyen
Faculty of IT
Le Quy Don University
Hanoi, Vietnam
Email: quanguyhn@gmail.com

***Abstract*—Transfer learning is a process in which a system can apply knowledge and skills learned in previous tasks to novel tasks. This technique has emerged as a new framework to enhance the performance of learning methods in machine learning. Surprisingly, transfer learning has not deservedly received the attention from the Genetic Programming research community. In this paper, we propose several transfer learning methods for Genetic Programming (GP). These methods were implemented by transferring a number of good individuals or sub-individuals from the source to the target problem. They were tested on two families of symbolic regression problems. The experimental results showed that transfer learning methods help GP to achieve better training errors. Importantly, the performance of GP on unseen data when implemented with transfer learning was also considerably improved. Furthermore, the impact of transfer learning to GP code bloat was examined that showed that limiting the size of transferred individuals helps to reduce the code growth problem in GP.**

I. Introduction

Transfer learning has been considered as an emerging research topic in machine learning [1]. Transfer learning attempts to mimic the process in which human beings learn [2]. Human beings learn and solve new problems based on the knowledge that they perceived via solving the similar problems in the past [3]. Conversely, common machine learning techniques are not designed to do this. In other words, machine learning algorithms are often applied separately when dealing with a new problem [4]. This implementation of machine learning is unnatural and less effective in solving real-world applications. This is particularly true when machine learning is applied to dynamic environments where the objectives and the parameters of the problems are changed during the learning process. In order to amend this, transfer learning has been proposed and investigated recently [5]. The previous research showed that transfer learning approaches help to leverage and widen the ability of machine learning in real-world applications [5].

The goal of transfer learning is to improve learning in the target task by leveraging knowledge from the source task. There are three common beneficial aspects of applying transfer learning [1]. First is the initial performance in the target task. Thanks to the knowledge from the source problem, the starting performance of transfer learning is often better compared to the initial performance of a traditional learner. Second is the amount of time to fully learn the target task. Given the transferred knowledge, it is often possible that transfer learning can learn faster than learning from scratch. Third is the final performance achieved by transfer learning in the target task compared to the final level without transfer. Three these advantages of transfer learning methods have been evidenced in a number of preceding researches [6], [7].

In the field of evolutionary algorithms, Genetic Programming (GP) [8], [9] is a paradigm with the objective of evolving computer programs that perform a user-defined task. GP has often been seen as a machine learning method, as it aims to induce a functional expression or program that presents relations between input and output data. Various learning schemes for GP have been proposed and investigated [10], [11]. However, examining the efficiency of transfer learning in GP has not been known to date. This paper aims to propose several techniques to implement transfer learning in GP. These schemes are based on transferring a number of good solutions or sub-solutions from the source to the target problem. The experiments were conducted on two families of symbolic regression problems showing the advantages of implementing transfer learning in GP.

The remainder of the paper is organized as follows. In the next section, we present the background of transfer learning. A short review of the previous research on transfer learning in evolutionary algorithms is given in Section III. The proposed transfer learning methods are described in Section IV. The experimental settings are detailed in Section V. The performance of transfer learning methods is examined and compared with standard GP in Section VI. Section VII analyzes some limitations of the paper. Section VIII concludes the paper and highlights some potential future work.

II. Transfer Learning

Transfer learning attempts to enhance learning performance in the target problem based on the knowledge from the previous solved problems [1]. Usually, there are two tasks in transfer learning: the source and the target task. The source task is a classical machine learning task which is often simpler and has been learned previously. The target task is often more complex and is the task to learn. In fact, we can address the target task separately from the source task. However, due to the difficulty of this task and the relationship between the source and the target task, it is often easier if the knowledge and experiences gained in the source task can be applied to the target task. Formally, the definition of transfer learning is follows [1]:

Definition 1: Given a source domain D_S and learning task T_S, a target domain D_T and learning task T_T, transfer learning

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

aims to help improve the learning of the target predictive function $f_T(.)$ in D_T using the knowledge in D_S and T_S, where $D_S \neq D_T$, or $T_S \neq T_T$.

There are three important research issues in transfer learning: 1) what to transfer, 2) how to transfer, and 3) when to transfer [1]. "What to transfer" asks which part of knowledge can be transferred from the source to the target problem. Some knowledge is specific for the source task and should not be re-used. Other knowledge may be common between different domains and potentially help improve performance for the target domain. After transferred knowledge has been discovered, a learning algorithm need to be devised to transfer the knowledge to the target problem, which corresponds to the issue "How to transfer".

"When to transfer" asks in which situations, transferring skills will be beneficial and when knowledge should not be re-used. In some situations, when the source problem and target problem are entirely different, brute-force transfer may not be successful. In the worst case, it may even deteriorate the performance of learning in the target domain. This situation is referred to as negative transfer. Most previous work on transfer learning focused on two first issues: "What to transfer" and "How to transfer". They implicitly assumed that the source and target domains be related to each other. Recently, how to avoid negative transfer became an important issue that attracted more and more attention from the research community [12].

Based on "What to transfer", transfer learning methods can be classified into four categories. The first method is instance-based transfer learning which transfers some parts of the data in the source domain to the target domain [13], [14]. A second case is called feature-representation-transfer approach. In this approach, the knowledge transferred across problems is the learned feature representation [15], [16]. With the new knowledge presented in the form feature representation, the performance of the target problem is expected to enhance significantly.

The third approach can be referred to as the relational knowledge-transfer problem in which the relationship among the data is transferred [17]. The basic assumption behind this approach is that the data in the source and target domains share some common relationship. The last case is the approach used in this paper. This method is called parameter-transfer approach which assumes that the source tasks and the target tasks share some parameters of the learned models [18]. The transferred knowledge is the shared parameters. Thus, when the shared parameters are discovered, they can be transferred across problems.

III. Related Work

Although, transfer learning has received a considerable attention from the machine learning community, few researchers in evolutionary computation (EC) have addressed this research strand. As far as we know, there was only one related publication in the literature [19]. The authors proposed a transfer learning scheme for genetic algorithm (GA) where the knowledge from multi-source problems is transferred to one target problem [19]. When GA is applied to the source problems, some individuals (the best, middle and worst individuals) in each generation are copied to a pool. The individuals in the pool are then used to replace some of the randomly generated individuals at the first generation in the target problem. The results reported in [19] were promising.

In the field of GP, there has not be any report on the study of transfer learning to date. The closest research to transfer learning in GP was focused on its reuse ability. These researches are summarized into two classes: Developmental evaluation in GP and GP in dynamic environments. In developmental evaluation based GP, the objective is to evaluate the fitness of individuals during development. In other words, a part of individuals is evaluated on a simpler problem first. Only if this part is considered as a good solution for the simpler problem, it is developed (grew) to be come the solution for a more complex problem. The developmental evaluation has been widely studied in GP, their advantages have been evidenced by a number of research [20], [21]

In the second class, GP is adapted to apply to dynamic environments [22]. In dynamic environments, some components of the problems like objective function are changed during the evolutionary process. Traditionally, GP researchers adapted GP to dynamic environments by transferring (copying) the whole population when there is a change in the environments [23], [24]. In order to avoid the deterioration of GP performance when the environments are altered, some schemes such as memory call or diversity promoting are often used. These methods allow GP to better cope with the changing environments [25], [26].

Although, the method of applying GP to dynamic environments is similar to transfer learning, examining transfer learning has not clearly been addressed in GP. First, the objective of implementing transfer learning in GP is different from their usage in dynamic environments. In transfer learning, the objective is to enhance GP performance on a novel problem with the knowledge perceived from the previous solved problems but not to cope with changing environments. Second, three important questions in transfer learning has not fully been addressed when applying GP to dynamic environments. In the following section, we will propose some schemes that allow us to study the effectiveness of transfer learning in GP.

IV. Methods

This section presents a method to implement transfer learning in GP. This method aims to answers two above questions: "what to transfer" and "how to transfer". The question: "when to transfer" is not addressed in this paper. Based on "what to transfer", we classify our method into three schemes. The first schema is called *FullTree* and is presented in Figure 1. In this technique, a number of good individuals (solutions) in the last generation of the source problem are copied to the first generation of the target problem. This technique is similar to the method where GP is applied to dynamic environments. However, the difference lies in the amount of knowledge transferred. In dynamic environments, the whole population is re-used when the environments are changed. In FullTree, only k% best individuals in the last population is transferred. The goal is to avoid the problem of negative transferring when the target problem and the source problems is not strongly related.

The second transferring learning technique is called *SubTree*. The idea behind this method is that there is some com-

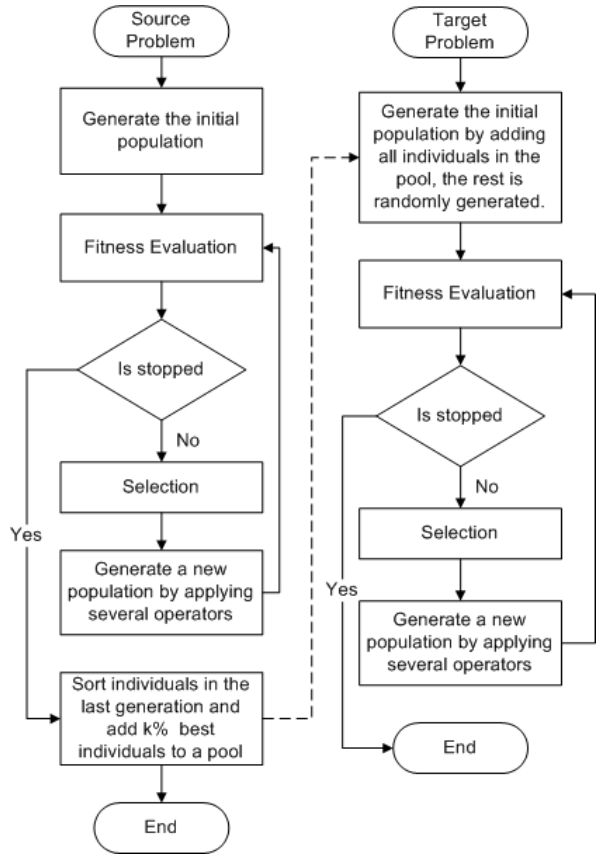

Fig. 1. Transferring by copy good individuals from the last generation of the source problem.

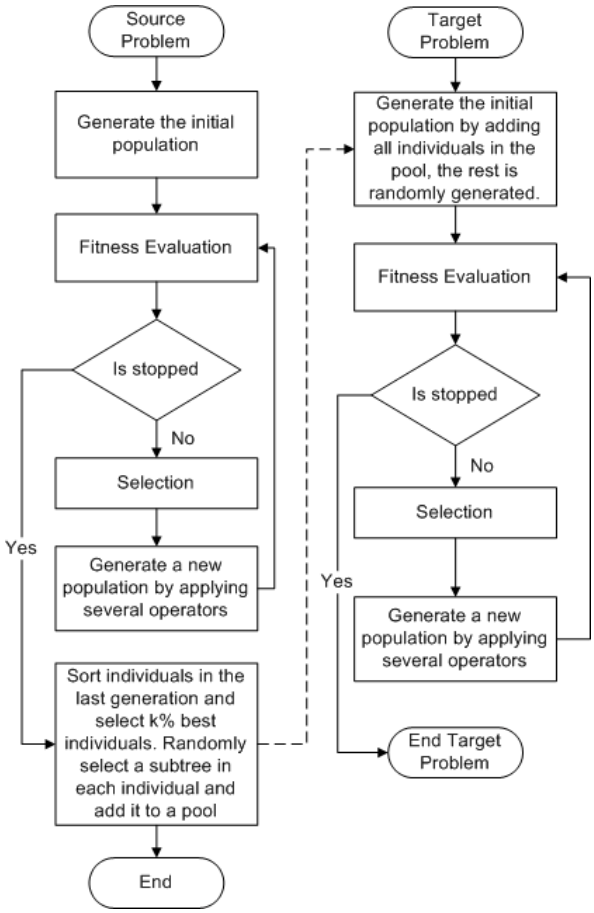

Fig. 2. Transferring by selecting a subtree in the individuals at the last generation of the source problem.

mon good subtree structures sharing between related problems. Moreover, we assume that these structures will exist in the last generation of the source problem. With this analysis, SubTree transferring learning method is implemented by randomly choosing a subtree in each solution at the end of evolutionary process (the last generation) in the source problem. These subtrees are added to a pool and then transferred to the target problem. They become the individuals in the first generation for the target problem. This scheme is presented in details in Figure 2.

The last transferring learning technique for GP is based on the assumption that good solutions for the target problem may appear in certain generations during the evolutionary process of the source problem. Therefore, if we can sample and store these solutions, they can be transferred to the target problem. This scheme is named *BestGen*. The detailed description of BestGen is presented in Figure 3. In each generation of the source problem, k best individuals are sampled and stored in a pool. The individuals in the pool are then copied to the first generation of the target problem.

V. Experimental Settings

In order to evaluate the performance of the transferring techniques, we tested them on two families of symbolic regression problems. These families include polynomial and trigonometric functions. The source problems is selected to be a simple function in each family. The target functions consist of four functions in each class. The target functions are always more complex than the source function. The source and the target function and their training and testing Data are presented in Table I. In this table, Poly-0 is the source problem for the polynomial functions and Trig-0 is the source task for trigonometric tasks. We assume that the target problems have a correlation to the source problem since their structure is similar to but more complex than the source problem.

The GP parameters used for our experiments are follows. The function set includes eight functions that are widely used by the GP research community [9], [27]. The terminal set for each problem includes one variable X. The raw fitness is the mean of absolute error on all fitness cases. Therefore, smaller values are better. Other parameters are presented in Table II. They are typical values for the experiments based on GP.

For each transfer learning technique, four configurations were tested. For FullTree method, k% (k=25, 50, 75 and 100) of the individuals in the last generation of the source problem was transferred to the the target problem. These configurations of FullTree method will be abbreviated as FullTree25,...,FullTree100. Similarly, k% (k =25, 50, 75, 100) of the individuals in the last generation was selected in SubTree technique. After than, a subtree in each individual was randomly selected and then transferred to the target problem. These schemes of SubTree technique will be referred to as

TABLE I. PROBLEMS FOR TESTING TRANSFER LEARNING METHODS.

Name	Definition	Training Set	Testing Set
Poly-0	$x^4 + x^3 + x^2 + x$	U[-1,1,100]	U[-3,3,100]
Poly-1	$x^6 + x^5 + x^4 + x^3 + x^2 + x$	U[-1,1,100]	U[-3,3,100]
Poly-2	$2x^4 + 3x^3 + 2x^2 + x$	U[-1,1,100]	U[-3,3,100]
Poly-3	$2x^5 + 3x^4 + 2x^3 + 3x^2 + x$	U[-1,1,100]	U[-3,3,100]
Poly-4	$x^6 + 2x^5 + x^4 + 3x^3 + 2x^2 + 5x$	U[-1,1,100]	U[-3,3,100]
Poly-5	$x^7 + 2x^6 + 3x^5 + 4x^4 + x^3 + x^2 + 3x$	U[-1,1,100]	U[-3,3,100]
Trig-0	$sin(x) + cos(x)$	U[-1,1,100]	U[-3,3,100]
Trig-1	$sin(x) + cos(x) + sin^2(x) + sin^3(x)$	U[-1,1,100]	U[-3,3,100]
Trig-2	$sin(x) + cos(x) + sin^2(x) + sin^3(x) + sin^4(x)$	U[-1,1,100]	U[-3,3,100]
Trig-3	$sin(x) + cos(x) + sin^2(x) + sin^3(x) + sin^4(x) + sin^5(x)$	U[-1,1,100]	U[-3,3,100]
Trig-4	$sin(x) + cos(x) + sin(2x) + sin(3x)$	U[-1,1,100]	U[-3,3,100]
Trig-5	$sin(x) + cos(x) + sin(2x) + sin(3x) + sin(4x)$	U[-1,1,100]	U[-3,3,100]

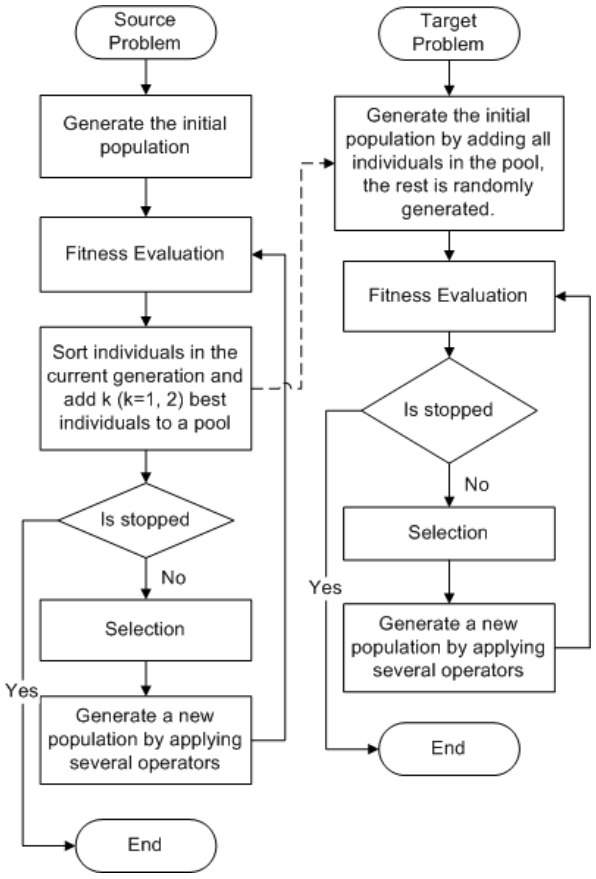

Fig. 3. Transferring by copy k best individuals at each generation of the source problem.

SubTree25, ..., SubTree100.

For BestGen method, we designed two experiments. In the first experiment, k (k=1, 2) best individuals in each generation of the source problem were transferred to the target problem. These schemes are shorted as BestGen1 and BestGen2. The second experiment was similar to the first one except that k (k=1, 2) best individuals are transferred to the target problems only if the size (the number of nodes) of these individuals is smaller than a threshold. In this experiment, the threshold was set at 50. The results of all experiments will be presented in the next section.

TABLE II. EVOLUTIONARY PARAMETER VALUES.

Parameter	Value
Population size	500
Generations	50
Selection	Tournament
Tournament size	3
Crossover probability	0.9
Mutation probability	0.1
Initial Max depth	6
Max depth	15
Max depth of mutation tree	5
Non-terminals	+, -, *, / (protected version), sin, cos, exp, log (protected version)
Terminals	X
Raw fitness	mean absolute error on all fitness cases
Trials per treatment	30 independent runs for each value

VI. RESULTS AND DISCUSSION

This section aims to analyze the performance of transfer learning techniques and to compare them with the standard GP where transfer learning techniques are not used. Three metrics were used for analyzing the effectiveness of transfer learning methods. The first metric is the mean best fitness on the training data. The second metric is the ability of transfer learning techniques to generalize on the unseen data. The last metric is the code bloat effect of these methods. They are detailed below.

A. Training Error Analysis

Although, the testing error is the most important indicator for the performance of a learner, the training error also provides some useful insight into the learning process. Thus, it is first analyzed in this section. The mean best fitness values of standard GP and transfer learning techniques are presented in Table III. The results for two source problems, Poly-0 and Trig-0 are not presented in this section since transfer learning methods were not executed on these problems.

It can be seen from Table III that most transfer learning technique helps GP to achieve better training errors. Apparently, the training errors obtained by transfer learning methods are often smaller than those of standard GP. However, the performance of transfer learning techniques are not consistent. Amongst three transfer learning techniques, the performance of SubTree and BestGen are usually better than the performance of FullTree.

Comparing between four configurations, we can see that transferring 25% the best individuals in the last generation

TABLE III. MEAN BEST FITNESS ON TRAINING DATA. LOWER IS BETTER. IF A RESULT OF TRANSFER LEARNING METHODS IS SIGNIFICANTLY BETTER THAN STANDARD GP, IT IS PRINTED BOLD FACE. IF IT IS SIGNIFICANTLY WORSE, IT IS PRINTED ITALIC FACE.

XOvers	Poly-1	Poly-2	Poly-3	Poly-4	Poly-5	Trig-1	Trig-2	Trig-3	Trig-4	Trig-5
GP	0.008	0.011	0.020	0.045	0.041	0.013	0.0124	0.014	0.023	0.056
FullTree25	**0.005**	0.009	0.022	0.039	0.035	0.013	0.019	*0.021*	**0.019**	**0.042**
FullTree50	0.008	0.011	0.020	0.045	0.040	*0.026*	0.020	0.019	0.031	0.055
FullTree75	0.006	0.009	0.020	**0.034**	0.037	0.020	0.018	0.015	*0.039*	0.065
FullTree100	0.008	0.011	0.020	0.045	0.040	*0.028*	0.016	*0.022*	*0.040*	0.054
SubTree25	**0.004**	0.008	0.026	0.039	0.033	0.012	**0.013**	**0.012**	**0.017**	**0.031**
SubTree50	**0.005**	**0.002**	0.018	**0.024**	**0.030**	**0.009**	**0.015**	0.015	**0.014**	**0.027**
SubTree75	0.007	**0.004**	**0.017**	**0.026**	0.035	0.016	**0.013**	0.015	**0.008**	**0.021**
SubTree100	**0.005**	**0.004**	0.020	**0.024**	0.031	0.011	**0.014**	0.014	**0.007**	**0.036**
BestGen1	0.006	0.010	0.027	0.042	**0.026**	0.013	**0.006**	**0.007**	**0.007**	**0.019**
BestGen2	0.006	**0.008**	**0.014**	**0.031**	**0.030**	0.011	0.012	**0.007**	**0.008**	**0.028**
BestLimit1	0.007	**0.008**	0.016	0.037	0.039	**0.007**	**0.007**	**0.006**	**0.007**	**0.011**
BestLimit2	0.006	**0.006**	**0.015**	0.034	0.048	**0.008**	**0.007**	**0.006**	**0.002**	**0.010**

TABLE IV. MEDIAN OF TESTING ERROR. LOWER IS BETTER. IF A RESULT OF TRANSFER LEARNING METHODS IS SIGNIFICANTLY BETTER THAN STANDARD GP, IT IS PRINTED BOLD FACE. IF IT IS SIGNIFICANTLY WORSE, IT IS PRINTED ITALIC FACE.

XOvers	Poly-1	Poly-2	Poly-3	Poly-4	Poly-5	Trig-1	Trig-2	Trig-3	Trig-4	Trig-5
GP	116.7	23.8	105.1	138.1	509.1	1.45	2.69	4.67	1.58	2.34
FullTree25	**86.5**	32.8	100.3	**100.3**	485.6	1.06	2.53	3.81	**0.91**	**1.11**
FullTree50	89.1	*44.3*	97.3	*168.7*	**429.0**	**0.94**	2.22	**3.37**	**1.04**	**0.91**
FullTree75	113.6	*39.1*	116.1	127.1	**402.9**	**0.80**	**1.44**	**2.21**	**0.83**	**1.10**
FullTree100	93.9	*33.4*	91.5	148.9	**423.5**	1.01	**1.18**	**2.24**	**1.22**	**1.15**
SubTree25	**1.9E-07**	**2.1E-07**	**87.5**	125.2	**384.9**	1.09	2.28	**3.12**	**1.22**	**1.09**
SubTree50	**1.9E-07**	**2.0E-07**	**48.1**	**106.5**	**380.0**	**1.03**	**1.74**	**2.23**	**0.63**	**0.78**
SubTree75	**1.8E-07**	**2.1E-07**	**34.1**	**92.0**	**304.9**	1.19	2.57	**2.4E-07**	**2.5E-07**	**0.65**
SubTree100	**1.8E-07**	**2.0E-07**	**69.6**	**89.0**	**170.3**	1.93	**1.15**	**2.01**	**2.5E-07**	**0.72**
BestGen1	*290.8*	*83.4*	*137.6*	137.6	**409.8**	1.03	2.52	**2.32**	**0.54**	**1.03**
BestGen2	95.9	29.5	131.8	134.1	**397.9**	1.56	2.75	4.41	**0.94**	**1.19**
BestLimit1	**35.9**	**6.64**	**67.2**	140.1	**355.6**	**0.83**	3.29	**2.75**	**2.5E-07**	**0.57**
BestLimit2	**1.9E-07**	**2.1E-07**	**36.1**	142.5	488.1	1.02	**1.02**	**2.23**	**0.12**	**0.59**

is the most successful schema for FullTree. When too much information was transferred like FullTree75 and FullTree100, the performance of FullTree is worse on some problems e.g. Trig-2, Trig-3 and Trig-4. Perhaps, the correlation between the source and the target problems is not strong on these problems.

For SubTree transferring technique, its performance is much consistent than FullTree method. On most problem, SubTree technique achieved better training error than standard GP. Moreover, two configurations of SubTree50 and SubTree75 are better than SubTree25 and SubTree100. It is understandable since in SubTree method, a subtree in each individual (not the individual itself) is transferred to the target problem. This helps SubTree technique to avoid the problem of negative transfer when the target problem is not strongly correlated to the source problem. The performance of BestGen method is also as convincing as SubTree method. For BestGen technique, its performance is consistent on all four settings. Particularly, the performance of BestGen technique is very good on trigonometric problems. On these problems, the best results (the smallest training error) always belong to one of the BestGen configurations.

We also conducted a statistical test to compare between the performance of standard GP with transfer learning methods using a Wilcoxon signed rank test with a confidence level of 95%. In Table III if a result of transfer learning methods is better than standard GP, this result is printed bold faced. Conversely, if a result of these method is significant worse than standard GP, it is printed italic faced. The statistical test shows that FullTree method is ocationally significantly better than standard GP. This method is even worse than standard GP on some problems like Trig-2, Trig-3 and Trig-4. In contrast to FullTree, two transfer learning methods (SubTree and BestGen) frequently produce the better results than standard GP. Moreover, these two methods are not significantly worse than GP on any problem.

B. Generalization Ability

The second metric used to measure the performance of the tested methods is their ability to generalize beyond the training data. The generalization ability is perhaps the most desirable property of a learner. In each run, the best solution was selected and evaluated on an unseen data set (the testing set). The median of these values across 30 runs was calculated and the results are shown in Table IV.

The results in this table are consistent with those in Table III, confirming the superiority of transfer learning techniques to standard GP. On the tested problems, the ability of all transfer learning methods to generalize was often better than standard GP. Although FullTree transfer learning technique is sometimes worse than standard GP on the training data (shown in Table III), its performance was more convincing on testing data when compared to GP without transferring. Particularly, on trigonometric problems, the testing errors of FullTree scheme are often much smaller than those of GP. For SubTree and BestGen, their performance on the unseen data was very impressive. On some problems such as Poly-1, Poly-2, Trig-3 and Trig-4, SubTree and BestGen methods achieved very small values of testing error (nearly zero). On other problems, the testing errors of these two methods were also much smaller than those of GP. Comparing between SubTree and BestGen, it can been seen that SubTree is slightly better than BestGen on the testing data.

TABLE V. AVERAGE OF THE INDIVIDUAL SIZE IN THE LAST GENERATION. IF A RESULT OF TRANSFER LEARNING METHODS IS SIGNIFICANTLY SMALLER THAN STANDARD GP, IT IS PRINTED BOLD FACE. IF IT IS SIGNIFICANTLY GREATER, IT IS PRINTED ITALIC FACE.

XOvers	Poly-1	Poly-2	Poly-3	Poly-4	Poly-5	Trig-1	Trig-2	Trig-3	Trig-4	Trig-5
GP	86.8	92.5	104.6	117.9	119.8	87.2	87.0	80.5	91.3	111.9
FullTree25	*128.3*	*141.0*	*149.3*	*154.2*	*174.0*	*107.0*	*118.2*	*136.2*	*119.6*	*145.4*
FullTree50	*131.6*	*138.1*	*155.2*	*167.2*	*176.9*	*96.6*	*114.1*	*126.0*	*144.2*	*182.9*
FullTree75	*134.8*	*155.1*	*160.4*	*163.9*	*175.6*	*102.8*	*124.2*	*140.8*	*125.2*	*158.2*
FullTree100	*126.3*	*143.4*	*143.4*	*179.1*	*188.8*	*108.3*	*130.2*	*147.2*	*145.8*	*169.5*
SubTree25	82.5	**81.7**	100.3	**93.3**	**104.4**	79.8	84.0	86.3	**80.0**	**90.9**
SubTree50	80.3	**81.5**	97.4	**90.3**	**98.3**	78.7	**78.4**	85.4	83.4	**82.0**
SubTree75	84.9	**88.8**	**89.7**	**85.1**	**91.1**	**71.1**	**78.0**	78.7	**63.9**	**76.3**
SubTree100	83.2	**80.0**	**84.0**	**84.5**	**75.6**	**54.2**	**62.4**	**66.8**	**53.6**	**70.8**
BestGen1	*120.6*	*125.9*	*138.6*	126.6	*138.7*	92.4	*107.4*	*110.3*	98.1	120.9
BestGen2	*113.9*	*130.4*	*140.3*	*150.9*	*156.0*	82.5	100.1	*102.9*	101.9	*131.6*
BestLimit1	82.1	**76.06**	**78.9**	**102.5**	**82.5**	79.0	78.0	75.3	**79.1**	**77.2**
BestLimit2	81.7	**82.38**	**77.4**	**94.1**	**93.5**	**74.91**	**75.4**	75.5	**80.0**	**84.1**

We also conducted a statistical test for the testing errors using the Wilcoxon test. Similar to the results on the training data, if a result of transfer learning methods is better than standard GP, this results is printed bold face and if it is worse, italic face is used. The statistical test showed the superior performance of all transfer learning techniques on most tested problems. Obviously, transfer learning methods not only improve GP performance on training data, on unseen data, their performance is even more convincing.

C. Code Bloat Effect

The last metric used to analyze the effect of transfer learning techniques is their level of code growth in the evolutionary process. It is important to examine this property since transfer learning techniques are implemented by copying good individuals at the later generations of the source problem. Earlier researches in GP have shown that the individual size tends to increase during the evolutionary process [28]. Therefore, transfer learning methods might incur more code growth in the target problems. In order to measure the code bloat impact of transfer learning schemes and compared them with standard GP, in each run we recorded the mean of the individual size at the last generation in the target problem. These values are then averaged over 30 runs and they are presented in Table V.

It can be seen from Table V that FullTree is only the scheme among three transfer learning methods that often leads to the increase of code growth compared to standard GP. Obviously, the average size of individuals of FullTree is much greater than GP. The statistical test using the Wilcoxon signed rank test showed that all difference between FullTree and GP is significant. In contrast to FullTree, SubTree technique does not incur the code growth to the population. Moreover, this method often helps to reduce the code growth in GP population. The statistical test evidenced that SubTree significantly reduced the average size of individuals in most problem.

For BestGen method, we can observe two different trends. For two configurations (BestGen1 and BestGen2) where the selected individuals for transferring are not restricted in size, they slightly increases the individual size of GP. Conversely, when they are constrained in size (BestLimit1 and BestLimit2), they help to significantly reduce GP code bloat. Overall, the results in this section show that transfer learning techniques helped GP to achieve better training error. Especially, the performance of transfer learning methods on unseen data was also significantly improved. Moreover, if we imposed a size restriction on transferred individuals, the resulting techniques often helped to reduce the code growth in GP.

VII. ASSUMPTIONS AND LIMITATIONS

Although this paper has shown that several benefits are to be gained from implementing transfer learning in GP, there are some limitations. First, the tested problems are some simple symbolic regression problems. Moreover, their relationship is based on the assumption that the target problems have similar but more complicated structure than the source problem. For a real-world application, a concrete structure is often not previously known. In order to apply transfer learning based GP to that problem, other methods to measure the relationship between the source and the target task is necessary.

Second, the performance of three transfer learning techniques proposed in this paper depends on some parameters (the amount of knowledge is transferred or a threshold to restrict the size of transferred individuals). Although, some different values for these parameters have been investigated, better methods such as self-adapting them could lead to a better performance. Last but not least, the transfer learning techniques are based on copying the good individuals (or sub-individuals) from the source to the target problem. This might result in a decrease in the diversity of the population in the target problem. Consequently, the negative transfer might happen. Therefore, the diversity analysis for transfer learning techniques is important to understand their advantages and limitations. Based on this, the problem of negative transfer could be avoided.

VIII. CONCLUSIONS AND FUTURE WORK

In this paper, the effectiveness of transfer learning in Genetic Programming (GP) was examined. Three methods to implement transfer learning in GP were proposed. These methods are based on transferring some good individuals (or sub-individuals) from the source to the target problem. The performance of transfer learning based GP were tested on two families of symbolic regression problems. The experimental results showed that transfer learning techniques helped GP to obtain better training error and significantly improved GP performance on unseen data. Moreover, using a scheme to restrict the size of transferred individuals helped to alleviate code bloat problem in GP.

There are a number of research areas for future work which arise from this paper. First, we would like to study some techniques to predict the relationship between the source and the target problems [29]. With this information, it is possible to select an appropriate amount of knowledge that is transferred to the target problem. Second, it will be very interesting to analyze some important aspects of transfer learning techniques such as diversity. If transfer learning methods affect negatively to diversity, it is important to use some schemes to maintain their population diversity. Better diversity preserving might result in better performance of transfer learning based GP. Last, we want to test the performance of these methods in some real-world application domains. In some problem domains such as time series forecasting and network security, it is natural that different problems have strong correlations. Thus, applying transfer learning methods to these domains could be more beneficial.

ACKNOWLEDGMENT

This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.01-2014.09. The first author is partly funded by Thu Dau Mot University, Binh Duong, Vietnam.

REFERENCES

[1] S. J. Pan and Q. Y. 0001, "A survey on transfer learning," *IEEE Trans. on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[2] M. E. Taylor, *Transfer in Reinforcement Learning Domains*, ser. Studies in Computational Intelligence. Springer, 2009, vol. 216.

[3] B. J. D., A. L. Brown, and R. R. Cocking, *How people learn: Brain, mind, experience, and school*. Washington D.C: National Academy Press, 200.

[4] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.

[5] S. Gutstein, O. Fuentes, and E. Freudenthal, "Knowledge transfer in deep convolutional neural nets," *International Journal on Artificial Intelligence Tools*, vol. 17, no. 3, pp. 555–567, 2008.

[6] W. Dai, G.-R. Xue, Q. Yang, and Y. Yu, "Transferring naive bayes classifiers for text classification," in *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, Vancouver, British Columbia, Canada, Jul. 2007.

[7] A. Niculescu-Mizil and R. Caruana, "Inductive transfer for bayesian network structure learning," in *ICML Unsupervised and Transfer Learning*, vol. 27. JMLR.org, 2012, pp. 167–180.

[8] R. Poli, W. B. Langdon, and N. F. McPhee, *A Field Guide to Genetic Programming*. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`, 2008, (With contributions by J. R. Koza). [Online]. Available: http://www.gp-field-guide.org.uk

[9] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, Massachusetts: The MIT Press, 1992.

[10] F. de Lima Arcanjo, G. L. Pappa, P. V. Bicalho, Wagner Meira, Jr., and A. S. da Silva, "Semi-supervised genetic programming for classification," in *GECCO '11: Proceedings of the 13th annual conference on Genetic and evolutionary computation*. Dublin, Ireland: ACM, 12-16 Jul. 2011, pp. 1259–1266.

[11] F. Wang2 and X. Xu, "Adagp-rank: Applying boosting technique to genetic programming for learning to rank," in *IEEE Youth Conference on Information Computing and Telecommunications (YC-ICT)*, Nov. 2010, pp. 259–262.

[12] M. T. Rosenstein, Z. Marx, L. P. Kaelbling, and T. G. Dietterich, "To transfer or not to transfer," in *In NIPS05 Workshop, Inductive Transfer: 10 Years Later*, 2005.

[13] P. Huang, G. Wang, and S. Qin, "Boosting for transfer learning from multiple data sources," *Pattern Recognition Letters*, vol. 33, no. 5, pp. 568–579, 2012.

[14] X. Liao, Y. Xue, and L. Carin, "Logistic regression with an auxiliary data source," in *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005*, ser. ACM International Conference Proceeding Series, vol. 119. ACM, 2005, pp. 505–512.

[15] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, "Self-taught learning: transfer learning from unlabeled data," in *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, ser. ACM International Conference Proceeding Series, vol. 227. ACM, 2007, pp. 759–766.

[16] J. Blitzer, R. T. McDonald, and F. Pereira, "Domain adaptation with structural correspondence learning," in *EMNLP 2007, Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, 22-23 July 2006, Sydney, Australia*. ACL, 2006, pp. 120–128.

[17] L. Mihalkova and R. J. Mooney, "Transfer learning from minimal target data by mapping across relational domains," in *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, 2009, pp. 1163–1168.

[18] J. Gao, W. Fan, J. Jiang, and J. Han, "Knowledge transfer via multiple model local structure mapping," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*. ACM, 2008, pp. 283–291.

[19] B. Koçer and A. Arslan, "Genetic transfer learning," *Expert System and Applications*, vol. 37, no. 10, pp. 6997–7002, 2010.

[20] R. I. McKay, T. H. Hoang, D. L. Essam, and X. H. Nguyen, "Developmental evaluation in genetic programming: the preliminary results," in *Proceedings of the 9th European Conference on Genetic Programming*, ser. Lecture Notes in Computer Science, vol. 3905. Budapest, Hungary: Springer, 10 - 12 Apr. 2006, pp. 280–289.

[21] N. F. McPhee, E. F. Crane, S. E. Lahr, and R. Poli, "Developmental plasticity in linear genetic programming," in *GECCO '09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation*. Montreal: ACM, 8-12 Jul. 2009, pp. 1019–1026.

[22] I. Dempsey, M. O'Neill, and A. Brabazon, *Foundations in Grammatical Evolution for Dynamic Environments*, ser. Studies in Computational Intelligence. Springer, Apr. 2009, vol. 194.

[23] Q. U. Nguyen, E. Murphy, M. O'Neill, and X. H. Nguyen, "Semantic-based subtree crossover applied to dynamic problems," in *The Third International Conference on Knowledge and Systems Engineering, KSE'2011*. Hanoi University: IEEE, 14–16 Oct. 2011, pp. 78–84.

[24] E. Murphy, "Examining grammars and grammatical evolution in dynamic environments," in *GECCO 2011 Graduate students workshop*, M. Nicolau, Ed. Dublin, Ireland: ACM, 12-16 Jul. 2011, pp. 779–782.

[25] C. N. Bendtsen and T. Krink, "Dynamic memory model for non-stationary optimization," in *Proceedings of the Fourth Congress on Evolutionary Computation (CEC-2002)*, vol. 1, 2002, pp. 145–150.

[26] M. Riekert, K. M. Malan, and A. P. Engelbrecht, "Adaptive genetic programming for dynamic classification problems," in *2009 IEEE Congress on Evolutionary Computation*, IEEE Computational Intelligence Society. Trondheim, Norway: IEEE Press, 18-21 May 2009, pp. 674–681.

[27] D. R. White, J. McDermott, M. Castelli, L. Manzoni, B. W. Goldman, G. Kronberger, W. Jaskowski, U.-M. O'Reilly, and S. Luke, "Better GP benchmarks: community survey results and proposals," *Genetic Programming and Evolvable Machines*, vol. 14, no. 1, pp. 3–29, Mar. 2013.

[28] S. Silva, S. Dignum, and L. Vanneschi, "Operator equalisation for bloat free genetic programming and a survey of bloat control methods," *Genetic Programming and Evolvable Machines*, vol. 13, no. 2, pp. 197–238, Jun. 2012.

[29] G. Kuhlmann and P. Stone, "Graph-based domain mapping for transfer learning in general games," in *Machine Learning: ECML 2007, 18th European Conference on Machine Learning, Warsaw, Poland, September 17-21, 2007, Proceedings*, ser. Lecture Notes in Computer Science, vol. 4701. Springer, 2007, pp. 188–200.