Improving the Parsimony of Regression Models for an Enhanced Genetic Programming Process

Alexandru-Ciprian Zăvoianu[1], Gabriel Kronberger[2], Michael Kommenda[2], Daniela Zaharie[1], Michael Affenzeller[2]

[1] Department of Computer Science, West University of Timişoara, Romania
[2] Heuristic and Evolutionary Algorithms Laboratory (HEAL), Upper Austrian University of Applied Sciences, Austria

Abstract. This research is focused on reducing the average size of the solutions generated by an enhanced GP process without affecting the high predictive accuracy the method exhibits when being applied on a complex, industry proposed, regression problem. As such, the effects the GP enhancements have on bloat have been studied and, finally, a bloat control system based on dynamic depth limiting (DDL) and iterated tournament pruning (ITP) was designed. The resulting bloat control system is able to improve by $\simeq 40\%$ the average GP solution parsimony without impacting average solution accuracy.

Keywords: genetic programming, symbolic regression, solution parsimony, bloat control

1 Introduction

For the concrete, industry proposed, system identification problem that has been considered for this research, an enhanced GP algorithm (Ehd-GP) developed by the HEAL team is able to produce high quality regression models that were of comparable, if not even better, accuracy than the regression models obtained using other, more well established, non-linear, data-mining techniques like support vector machines (SVMs) and artificial neural networks (ANNs) [6] [11]. Further evidence of the high quality of the solutions generated by Ehd-GP is summarized in [9].

The main advantage of using GP for the given regression problem task lies in the the ability of the this method to produce white-box, human interpretable, models than can be easily used by human domain experts to gain new insight into phenomena associated with the given industrial process. However, the degree of interpretability a GP generated regression model exhibits is proportional to its parsimony as trying to analyze large and/or highly complex models can be a quite tedious task.

A general threat to GP model interpretability comes in the form of a well studied phenomenon known as *bloat*: a generation-wise rapid growth in the size of evolved programs without any corresponding benefits in terms of accuracy. This phenomenon is very well known to GP practitioners and literature proposes a large set of studies regarding bloat. Controlling and combating bloat is an open and fairly complicated task as Silva and Costa [8] argue that the phenomenon is very likely to be a natural consequence of combining an evolutive (fitness driven) search with a variable-length solution representation (i.e. two of the main characteristics of the GP theory).

With the above mentioned evidence of the ability of Ehd-GP to produce high accuracy regression models, the purpose of this research is to analyze the impact the modifications in Ehd-GP have on bloat and, consequently, try to control the bloating phenomenon in this GP process such as to reduce the average size of the resulting solution models without affecting the high average quality of these models.

2 The Ehd-GP Process

The Offspring Selection Strategy (OS) The most important modification the Ehd-GP proposes over the standard GP process described by Koza [5] is the incorporation of the offspring selection strategy [1]. The idea behind this elitist selection method is that an offspring is accepted as a member of the population of the next generation if and only if it outperforms the fitness of its own parents. As with conventional GP, offspring are generated by parent selection, crossover and mutation. The difference consists in the introduction of a second (offspring) selection stage. A variable called success ratio indicates the ratio of the next generation members that must outperform their respective parents. As long as this ratio is not fulfilled, offspring are created and the successful ones are inserted into the next generation, while those that do not outperform their parents are stored in a rejection pool. After enough successful next generation offspring have been created, the rest of the members of the generation are randomly chosen from the rejection pool.

The Linear Scaled Error Measure The second important enhancement that Ehd-GP proposes is the use of a linear scaled error measure (as described by Keijzer in [4]) in the fitness evaluation function. The advantage of using a scaled mean squared error (MSE), as opposed to the traditional approach, lies in "the change of the view" that the selection operator has on the worth of an individual expression. As this error measure rescales the expression on the optimal slope and intercept, selection will favor expressions that are close in shape with the target, instead of demanding first that the scale is correct.

3 Test Setup

The industry proposed main modeling scenario used in our tests contains 44 dependent variables and has been split into a training set, a validation set and a test set. For a detailed description of the benefits of using this data partitioning strategy in GP based modeling please see [9].

Methodology Because GP remains at its core a stochastic method, we have performed 100 GP runs of a given configuration in order to determine the general behavior of that configuration (in terms of solution size and accuracy). We shall refer to the data set that contains all the 100 solutions obtained for a specific GP configuration as the *the full data set* of that configuration. Since solution accuracy is the first performance GP criteria, we are especially interested in analyzing and improving the best solutions that can be generated with a given GP configuration. As a result, for each GP configuration, we have also constructed a *top accuracy subset* that only contains the 25 most accurate solutions in the full data set. Ties were broken in favor of the more parsimonious model.

Throughout this paper, our comparisons are based on basic central tendency indicators (the average - μ, the standard deviation - σ and the median - $\mu_{1/2}$) for both the full solution data sets and the top accuracy subsets.

When comparing among GP configuration results based on the full data sets, we also make use of *significance testing* to confirm our empirically based hypotheses. The significance test we use is the *Mann-Whitney-Wilcoxon test* (also know as the Mann-Whitney U test). The used significance level is $\alpha = 0.05$ in the case of one-tailed tests. The choice for this particular non-parametric test was made because we do not wish to presume that our solution data set is normally distributed, either according to size or accuracy.

In order to provide a simple but accurate and suggestive overview of the performance of the GP solutions in the full and top accuracy data sets, for each GP configuration, we construct *comparative kernel density estimation* plots of the the solution size and of the solution MSE.

GP Configurations The Ehd-GP was configured to enforce a strict OS thus forcing every member of the next generation to outperform both its parents. When generating a new offspring, one parent was chosen using proportional selection and the other was chosen using random selection.

For our tests we have also used a Koza style standard GP process (Std-GP) in which both parents were selected according to proportional selection.

Both GP processes used only point mutation and in both cases constants were initialized and modified uniformly (between -20 and 20), whilst variables were initialized and modified according to a normal distribution $N(0, 1)$. The other, more common, GP algorithm parameters that were used by both the Std-GP and the Ehd-GP processes are: an arithmetic function library ($+$, $-$, $*$, %, *power*, *sqrt*), a population size of 1000 individuals, a mutation rate of 15%, a max tree height of 15 and a maximum number of evolved generations of 1000.

The initial population initialization method was PTC2 [7] for both GP processes and the stopping criterion stated that a run should be terminated if no validation wise improvement was found in the last 100 generations.

The max tree height limit value was chosen empirically after performing systematic tests on the modeling scenarios considered in this research and observing that the very good solutions have a depth smaller than 10.

4 Ehd-GP performance analysis

The performances of Std-GP and of Ehd-GP with regards to average solution accuracy and average solution size are summarized in Table 1. The corresponding comparative kernel density plots are presented in Fig. 1.

Our first empirical observation that the Ehd-GP is far superior in terms of average solution accuracy is statistically significant as the *Mann-Whitney-Wilcoxon test* yielded a one-tailed *p-value* smaller than 0.001.

The second empirical observation that the Ehd-GP tends to produce solutions of lower parsimony than the Std-GP is also statistically significant, being confirmed as the one-tailed *Mann-Whitney-Wilcoxon test* produced a one-tailed *p-value* of 0.005.

5 A Bloat Control System for Ehd-GP

The main challenge of fitting an efficient bloat control mechanism in Ehd-GP came from trying to integrate this such a modification with the existing OS strategy.

For instance, anti-bloat selection methods [10][3] proved fairly difficult to combine with the offspring selection enhancement because their most effective implementations are based on dynamical adjustments of their control parameters. When using OS with a linear scaled error measure, the GP algorithm converges quite fast (in $> 90\%$ of the runs the solution was found before the 22^{nd} generation) and this means that there are far fewer opportunities for parameter adjustment.

In the case of anti-bloat genetic operators [2], our educated guess is that for Ehd-GP, the combination between size and/or depth limitations imposed to genetic operators (i.e. restricting the number of offspring than can be generated) and OS has a high chance of leading to premature convergence (because of a rapid drop in genetic diversity).

Dynamic Depth Limits - DDL In [8], Silva and Costa present a simple yet effective solution for overcoming most of the shortcomings of static depth limitation. Their idea is to dynamically adjust the value of the depth limit during the run. Compared to the original method, we have made a series of modifications in order to integrate DDL into the Ehd-GP process. In our implementation of the concept (Algorithm 1), the dynamic limit is initially set to a low value ($InitialDepthLimit$), that is usually 20-30% higher than that of the maxium depth in the initial population. An offspring is automatically accepted in the next generation if it satisfies the accuracy constraint and at the same time does not infringe the depth limit. If an offspring infringes the depth limit but is the best individual found so far, then it is accepted in the next generation if the increase in size is matched by the increase in accuracy. In the latter case, the limit is raised to match the depth of the new best-of-the-run individual. If during

Algorithm 1 The dynamic depth limiting module (DDL module)

```
 1: AcceptStatus = $true$
 2: if OffDepth $\leq$ DepthLimit then
 3:    if (BestMSE / OffspringMSE - 1) $\geq$ (DLimit - OffspringDepth) * $C_{lower}$ then
 4:       if OffspringDepth > $InitialDepthLimit$ then
 5:          DLimit = OffspringDepth
 6:       else
 7:          DLimit = $InitialDepthLimit$
 8:       end if
 9:    end if
10: else
11:    if (BestMSE / OffMSE - 1) $\geq$ (OffspringDepth - DLimit) * $C_{raise}$ then
12:       DLimit = OffspringDepth
13:    else
14:       AcceptStatus = $false$
15:    end if
16: end if
17: return AcceptStatus
```

the run the best found individual at a given time has a depth that is significantly lower than the current depth limit, the limit will be lowered.

The condition on Line 11 states that the DDL should be raised if each extra depth level is matched by an increase in training accuracy of at least $C_{raise}\%$. Analogously, the condition on Line 3 states that the DDL should be lowerd if each decreased depth level is matched by an increase in training accuracy of at least $C_{lower}\%$. As an empirical rule, tests have shown that the relation $C_{lower} = 2 * C_{raise}$ enables the DDL to have a stable behaviour throughout the run for all the considered test scenarios. Furthermore, after testing on three other modeling scenarios, we discovered that $C_{raise} = 0.015$ is also a stable setting.

Iterated Tournament Pruning - ITP Largely inspired by [9], and taking into consideration all the particularities of the GP process we are trying to enhance, we also decided to implement and test a bloat control strategy based on syntax-tree pruning. Our ITP strategy is described in Algorithm 2 and is based on a series of consecutive pruning tournaments. In each tournament we create several pruning candidates of the syntax-tree model we wish to prune in that tournament (the pruning base model). A pruning candidate for a model is created through a very simple process (Line 6) that randomly selects a subtree from the given model an replaces it with the mean value obtained by evaluating that subtree over all the records in the training set. The size of the excised subtree is limited to at most $MaxPruning\%$ with regards to the pruning base model. At the end of each tournament the pruning candidate with the highest accuracy (minimum MSE) will be selected as the next pruning base.

Initial tests performed with various settings for the configuration parameters ITP supports indicate that: the $MaxPruning$ and $IterationsCount$ parameters should be set such as to generally avoid the possibility of reducing a model to a single node; ITP displays the best results when applied only to the individuals that are not among the best 10-15% nor among the worst 20-40% according to accuracy; an increased mutation rate of 25-50% would improve solution accuracy;

Algorithm 2 The iterated tournament pruning module (ITP module)

```
1: BestMSE = $\infty$
2: PrunSolution = $\Phi$
3: PrunBase = OriginalModel
4: for $i = 0$ to $IterationsCount$ do
5:    for $j = 0$ to $TournamentSize$ do
6:       PrunCandidate = StochasticPrune(PrunBase, $MaxPruning$)
7:       PrunMSE = ComputeMSE(PrunCandidate)
8:       if PrunMSE < BestMSE then
9:          PrunSolution = PrunCandidate
10:         BestPrunMSE = PrunMSE
11:      end if
12:   end for
13:   PrunBase = PrunSolution
14: end for
15: return PrunSolution
```

The Resulting Bloat Control System - BCS From the previous descriptions, one can observe that DDL and ITP have two complementary ways of fighting bloat. Whilst the former tries to prevent the creation and propagation of bloat, the latter directly attempts to remove superfluous code from the population members. As such, trying to combine both methods in a single integrated solution aimed at combating bloating, seemed a very natural approach.

In our initial tests of the combined bloat control system, we decided to use for the two bloat control methods the same parameter settings from the stand-alone configurations. To our surprise, the approach proved quite successful.

6 Conclusions

Individually both the DDL and ITP modules managed to reduce the average Ehd-GP solution size (see Table 1). Both improvements were statistically significant with *Mann-Whitney-Wilcoxon test p-values* smaller than 0.0001.

The combination of the two bloat control methods proved to reduce even more the average Ehd-GP solution size (see Table 1). The average decrease of solution size determined by BCS was statistically significant both with regards to individual DDL (*p-value* = 0.0183) and to individual ITP (*p-value* < 0.0001).

While empirical observations may suggest that BCS also slightly increased the average solution quality of the Ehd-GP, there was no solid statistic proof

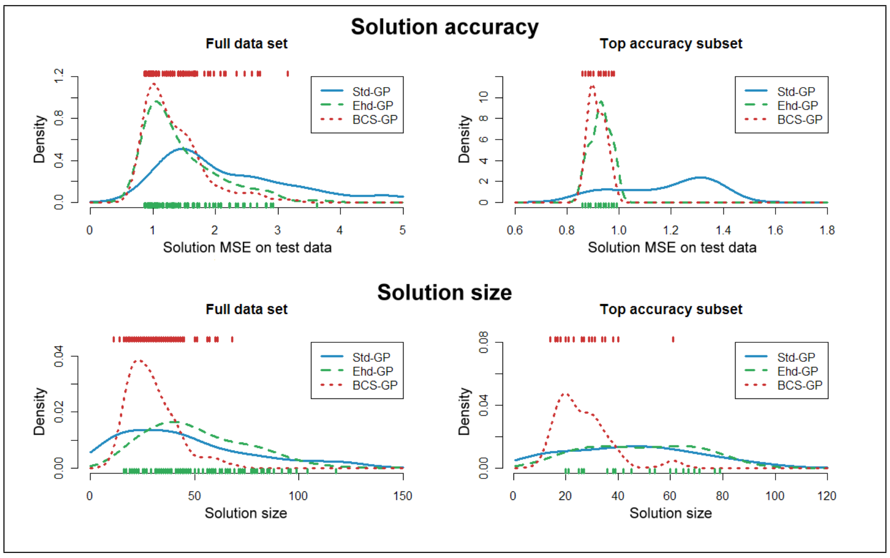

Fig. 1. Kernel density estimation plots for Std-GP and Ehd-GP

Table 1. Accuracy and size information regarding Std-GP and of Ehd-GP solutions

	Full data set			**Acc. subset**		
	μ	σ	$\mu_{1/2}$	μ	σ	$\mu_{1/2}$
Std-GP MSE	2.178	1.046	1.821	1.177	0.177	1.232
Ehd-GP MSE	1.448	0.584	1.280	0.929	0.037	0.930
DDL-GP MSE	1.514	0.643	1.345	0.919	0.049	0.940
ITP-GP MSE	1.228	0.925	1.065	0.962	0.035	0.970
BCS-GP MSE	1.339	0.470	1.210	0.915	0.032	0.900
Std-GP size	45.44	33.50	39.00	44.84	24.55	43.00
Ehd-GP size	52.60	28.11	47.00	53.36	29.94	52.00
DDL-GP size	32.92	16.30	30.50	34.16	14.32	34.00
ITP-GP size	39.34	18.65	35.00	42.16	18.43	38.00
BCS-GP size	29.17	11.18	28.00	26.36	10.21	23.00

that the average solution accuracy of the BCS augmented Ehd-GP process (BCS-GP) was any different from that of original Ehd-GP. The BCS-GP process was further tested on two more symbolic regression problems in [11] and the results confirmed the findings reported in this paper. The corresponding comparative kernel density plots of Ehd-GP and BCS-GP are presented in Fig. 1.

In spite of the very good results obtained by the final bloat control system, we consider that, at the current stage of development, its main function is that of a proof of concept with regards to the successful combination of two bloat combating methods that are based on different but complementary paradigms.

Acknowledgments

The work described in this paper was done within the Josef Ressel Centre for Heuristic Optimization Heureka! and sponsored by the Austrian Research Promotion Agency (FFG). For more information about Heureka! please visit http://heureka.heuristiclab.com.

The stay of Mr. Alexandru-Ciprian Zăvoianu at the Heuristic and Evolutionary Algorithms Laboratory has been supported by the EU through the SPRERS project (FP7 reference number 246839).

References

1. Affenzeller, M., Wagner, S.: Offspring selection: A new self-adaptive selection scheme for genetic algorithms. In: Ribeiro, B., Albrecht, R.F., Dobnikar, A., Pearson, D.W., Steele, N.C. (eds.) Adaptive and Natural Computing Algorithms. pp. 218–221. Springer (2005)
2. Crawford-Marks, R., Spector, L.: Size control via size fair genetic operators in the PushGP genetic programming system. In: GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 733–739. Morgan Kaufmann Publishers, New York (9-13 Jul 2002)
3. Ekart, A., Nemeth, S.Z.: Selection based on the pareto nondomination criterion for controlling code growth in genetic programming. Genetic Programming and Evolvable Machines 2(1), 61–73 (Mar 2001)
4. Keijzer, M.: Scaled symbolic regression. Genetic Programming and Evolvable Machines 5(3), 259–269 (Sep 2004)
5. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA, USA (1992)
6. Kronberger, G., C.Feilmayr, Kommenda, M., Winkler, S., Affenzeller, M., Burgler, T.: System identification of blast furnace processes with genetic programming. In: Logistics and Industrial Informatics - LINDI. pp. 1–6. IEEE Press (2009)
7. Luke, S.: Two fast tree-creation algorithms for genetic programming. IEEE Transactions on Evolutionary Computation 4(3), 274–283 (Sep 2000)
8. Silva, S., Costa, E.: Dynamic limits for bloat control in genetic programming and a review of past and current bloat theories. Genetic Programming and Evolvable Machines 10(2), 141–179 (2009)
9. Winkler, S.M.: Evolutionary System Identification. Ph.D. thesis, Johannes-Kepler-Universitat, Linz, Austria (2008)
10. Zhang, B.T., Mühlenbein, H.: Balancing accuracy and parsimony in genetic programming. Evolutionary Computation 3(1), 17–38 (1995)
11. Zăvoianu, A.C.: Towards solution parsimony in an enhanced genetic programming process. Master's thesis, International School Informatics: Engineering & Management, ISI-Hagenberg, Johannes Kepler University, Linz (2010)