Relaxed Genetic Programming

Luis E. Da Costa and Jacques-André Landry
Lab. for Imagery, Vision and Artificial Intelligence (LIVIA)
École de Technologie Supérieure
Montréal (Québec), Canada
costa@livia.etsmtl.ca, jacques.landry@etsmtl.ca

ABSTRACT

A study on the performance of solutions generated by **G**enetic **P**rogramming (**GP**) when the training set is *relaxed* (in order to allow for a wider definition of the desired solution) is presented. This performance is assessed through 2 important features of a solution: its *generalization error* and its *bloat*, a common problem of **GP** individuals. We show how a small degree of relaxation improves the generalization error of the best solutions; we also show how the variation of this parameter affects the bloat of the solutions generated.

Categories and Subject Descriptors

I.2.2 [**Artificial Intelligence**]: Automatic Programming – Program Synthesis

General Terms

Algorithms, Measurement, Performance

Keywords

Genetic Programming, Bloat, Generalization Error

1. INTRODUCTION

Genetic **P**rogramming (**GP**) solves complex problems by evolving populations of computer programs (details in [2] and [1]). In this paper, we present a study of 2 aspects of the **GP** method: the *generalization error* and the problem of *bloat*.

The **generalization error** of a machine learning model is a function that estimates how far a candidate solution is from a target solution, over the *"entire"* set of possible data that could be generated by this target solution. **Bloat** is *an excess of code growth caused by the genetic operators in search of better solutions, without a corresponding improvement in fitness* [7]. It is a serious problem in **GP** [1]; several approaches have been proposed for dealing with it (refs. [2, 8, 5, 3, 4, 7] and references within).

Part of our research involves the use of **GP** for modelling the behavior of human subjects performing a particular classification task: **GP** performs regression on a mathematical function f mimicking the answers given by the subjects. Because of the inherent variability in human responses, the data set contains several *correct* answers for each row of entry data. So we have $f(\mathbb{X}) \in [y_{min}, y_{max}]$: the data set has been *relaxed*; instead of asking the **GP** algorithm to obtain the *perfect* answer y, we allow anything in the range $[y_{min}, y_{max}]$ **Does this *relaxation* improve the generalization error of GP? What effect does it have on the bloat behavior of the algorithm?**

GECCO'06, July 8–12, 2006, Seattle, Washington, USA.
ACM 1-59593-186-4/06/0007.

2. EXPERIMENTAL SETTING

2.1 The problem

The symbolic regression of the quartic polynomial $Q(x) = x^4 + x^3 + x^2 + x$, with 201 equidistant points in the interval $x \in [-10, +10]$ was used for the experiments. A training set of 20 points was randomly chosen; the other 181 points were assigned to the test set (Fig. 1). The *training points*

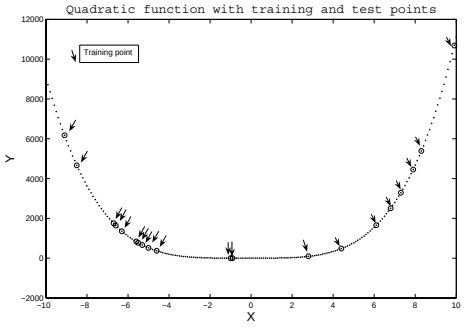

Figure 1: Training set (arrows) and test set

in Fig. 1 are also presented in Table 1.

Table 1: Training points

x	y	x	y	x	y	x	y
-9.10	6178	-8.50	4670	-6.70	1753	-6.60	1647
-6.30	1359	-5.60	834	-5.50	773	-5.30	663
-5.00	520	-4.60	367	-1.00	0	-0.90	0
2.80	94	4.40	484	6.10	1655	6.80	2506
7.30	3289	7.90	4458	8.30	5395	9.90	10684

A *relaxation* for a training point $x_p$is defined as the interval $[f(x_p) - \delta, f(x_p) + \delta]$. We defined 9 values for δ, calculated as a part of the total interval defined by $Q(x)$; as $x \in [-10, +10]$, $Q(x) \in [-0.33, 11110]$ and the width of the *interval* occupied by $Q(x)$ is $I(Q(x)) = 11110.67$. The

chosen values for δ (presented as absolute values δ_{abs} and as percentage of $I(Q(x))$, δ_{perc}) are presented in Table 2.

Table 2: Relaxation values

δ_{perc}	δ_{abs}	δ_{perc}	δ_{abs}	δ_{perc}	δ_{abs}
0.125%	13.89	0.25%	27.78	0.5%	55
0.75%	83.33	1%	110	5%	555
10%	1110	15%	1666.07	20%	2222.07

2.2 The algorithm

Populations of 400 individuals (randomly initialized, maximum depth of 6) were evolved for 50 generations. The **GP** algorithm uses standard tree crossover and tree mutation (details in [2]) as genetic operators, a **function set** F = $\{+, -, \times, \div, log, exp, sqrt\}$ and a **terminal set** T = $\{x$ (constant), $r \in \mathbb{R} \wedge r \in [0, 1], n \in \mathbb{N} \wedge n \in [1, 10]\}$. **Selection** for reproduction used the lexicographic parsimony pressure tournament [3] and no elitism in survival. The **fitness** of a candidate individual $\overline{\mathcal{S}} = \{\overline{s_1}, \overline{s_2}, \cdots, \overline{s_{20}}\}$ is its distance d_A to the middle point of each of the relaxed intervals. This distance is 0 if the candidate actually belongs to the target interval

3. RESULTS

A total of 40 runs were performed with each of the intervals. We used **GPLAB** [6], modified in order to allow optimization by intervals. The population of candidate solutions was optimized with the training set, and then the generalization error $\mathcal{G}_e$ (Eq. 1) was calculated on the validation set $\mathcal{S}^G$ (181 points).

$$\mathcal{G}_e(\overline{\mathcal{S}}, \mathcal{S}^G) = \sum_{i=1}^{181} d(\overline{s_i}, s_i^G) \quad (1)$$

Eq. 1 stresses the fact that the relaxation is performed on the training set, not on the test set.

3.1 Generalization error

Results for the generalization error are shown in Fig. 2. The generalization error drops from the 0% relaxation mark,

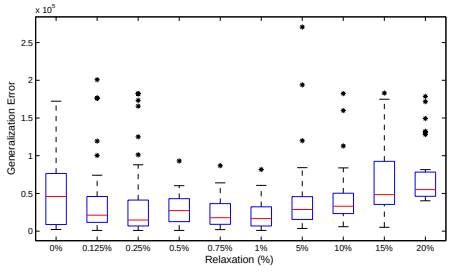

Figure 2: Boxplot generalization error

hitting a low value at 1% relaxation. The values for the average generalization error are under those for 0% relaxation until the 10% mark: so even a small relaxation causes the generalization error to drop. Although we need more experiments to confirm these results (in order to make them statistically significant), they point to the fact that there is an advantage in using a little relaxation in order to allow the algorithm some latitude.

3.2 Resources used

The resources used by the candidate solutions are the *number of nodes* in the tree. Fig. 3 presents the resources used on average for the 40 runs by the best solution, expressed along the generations of the run. For relaxation values up to 1%, the number of resources used are greater than that without relaxation. For values of relaxation greater than 1% the best solutions actually used up *less* resources than the solutions generated with the exact training set.

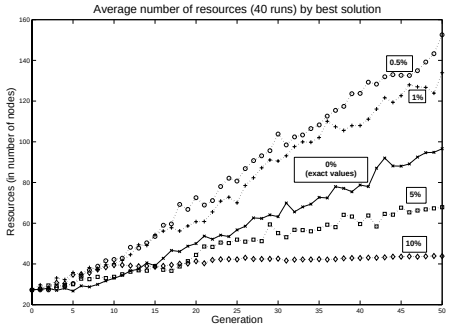

Figure 3: Resources used

4. DISCUSSION AND FUTURE WORK

Our results show that the generalization error for relaxation $\leq 1\%$ was better than the error for the exact-valued solution (Fig. 2) but, at the same time their bloat values were larger (Fig. 3). The opposite scenario holds for the results for relaxation greater than 1%. So there seems to be a compromise between the *quality* (in terms of approximation) of the solutions and their *quality*, in terms of bloat. More experiments with relaxation in the range $[1\%, 10\%]$ would be important in order to quantify this compromise, if there is any.

5. REFERENCES

[1] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone. *Genetic Programming An Introduction*. Morgan Kauffman Publishers, San Francisco, 1998.

[2] J. Koza. *Genetic Programming-On the programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, London, 1992.

[3] S. Luke and L. Panait. Lexicographic parsimony pressure. In W. Langdon and e. al, editors, *Proceedings of GECCO-2002*, pages 829–836. Morgan Kauffmann, San Francisco, CA, 2002.

[4] L. Panait and S. Luke. Alternative bloat control methods. In K. Deb, editor, *GECCO-2004*, pages 630–641, Berlin, 2004. Springer.

[5] R. Poli. A simple but theoretically-motivated method to control bloat in genetic programming. In R. C. and e. al, editors, *Proceedings of EuroGP-2003*, pages 204–217. Springer, Berlin, 2003.

[6] S. Silva. Gplab-a genetic programming toolbox for matlab, 2004. http://gplab.sourceforge.net.

[7] S. Silva, P. J. Silva, and E. Costa. Resource-limited genetic programming: Replacing tree depth limits. In B. Ribeiro, editor, *Adaptive and Natural Computing Algorithms*, pages 243–246, Coimbra, Portugal, 2005. Springer. http://icannga05.dei.uc.pt/.

[8] T. Soule and J. Foster. Effects on code growth and parsimony pressure on populations in genetic programming. *Evolutionary Computation*, 6(4):293–309, 1999.