Differential Evolution of Constants in Genetic Programming Improves Efficacy and Bloat

Shreya Mukherjee
Dept. Computer Science
Univ. Vermont, Burlington, VT 05405
Shreya.Mukherjee@uvm.edu

Margaret J. Eppstein
Dept. Computer Science
Univ. Vermont, Burlington, VT 05405
Maggie.Eppstein@uvm.edu

ABSTRACT
We employ a variant of Differential Evolution (DE) for co-evolution of real coefficients in Genetic Programming (GP). This GP+DE method is applied to 30 randomly generated symbolic regression problems of varying difficulty. Expressions were evolved on sparsely sampled points, but were evaluated for accuracy using densely sampled points over much wider ranges of inputs. The GP+DE had successful runs on 25 of 30 problems, whereas GP using Ephemeral Random Constants succeeded on only 6 and the multi-objective GP Eureqa on only 18. Although nesting DE slows down each GP generation significantly, successful GP+DE runs required many fewer GP generations than the other methods and, in nearly all cases, the number of nodes in the best evolved trees were smaller in GP+DE than with the other GP methods.

Categories and Subject Descriptors
I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods and Search

General Terms
Algorithms, Performance, Design, Reliability.

Keywords
Code Bloat, Constant Optimization, Differential Evolution, Genetic Programming, Symbolic Regression,

1. INTRODUCTION
Genetic programming (GP) is a powerful tool for symbolic regression, however the difficulty of finding numeric constants has long been recognized as a weakness [3]. Various approaches for constant estimation have been tried including ephemeral random constants (ERCs), persistent random constants, numeric mutation, gradient descent, digit concatenation, and evolutionary approaches (see [2] for a relatively recent review). Differential evolution (DE) [5] has been shown to be a simple but effective method for evolving real-valued variables, and two alternative approaches using DE to estimate constants in fixed-length chromosome variants of GP have yielded encouraging results [1,6]. However, in both cases only 2-3 problems were tested, each including only a single variable and a few constants within a small range, and the testing range of the variable was the same as the range used for training.

Copyright is held by the author/owner(s).
GECCO'12 Companion, July 7–11, 2012, Philadelphia, PA, USA.
ACM 978-1-4503-1178-6/12/07.

2. METHODS
In basic DE [5], new vectors $\boldsymbol{v}_i$ are first produced using weighted difference vectors as follows:

$$\boldsymbol{v}_i = \boldsymbol{x}_1 + F \cdot (\boldsymbol{x}_2 - \boldsymbol{x}_3) \quad (1)$$

where $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, and $\boldsymbol{x}_3$ are random and mutually exclusive members of the current population that are distinct from the target vector $\mathbf{x}_i$, and F is a control parameter typically less than 2. After performing crossover between the vector $\boldsymbol{v}_i$ and target vector $\boldsymbol{x}_i$, the new individual then replaces $\boldsymbol{x}_i$ if its fitness is at least as good. Numerous variants of Eq. (1) have been tried (in [5] and elsewhere) and found to work well under different conditions. During preliminary testing of several DE variants for constants estimation in randomly generated expression trees, we found that the following new variant performed most consistently and efficiently on these problems. Specifically, for some prespecified maximum number of DE generations ($DEgen_{max}$), we decrease the step size control parameter F linearly each generation from an initial value of $F = 1.25$ to a minimum of $F = 0.5$ at $DEgen_{max}$. The vector $\boldsymbol{x}_1$ is randomly selected from the top 20% most fit individuals with probability p, otherwise it is randomly selected from the entire population. We initialize $p = 0$, and increase it linearly to 1 by generation $DEgen_{max}/5$. Varying F and p in this manner over the course of DE evolution facilitates a transition from exploration to exploitation and helps the DE to converge. We used DE population sizes of $N_{DE} = 10 \times$ number of constants in the DE chromosome. DE was terminated when either the mean squared error (MSE) was < 1e-8, after 25 generations without improvement, or after a maximum of $DEgen_{max} = 250$ generations.

We implemented a standard GP [3] with variable-length tree-based chromosomes, a population size of $N_{GP} = \{150, 200, 250\}$ for 1-, 2-, and 3-variable problems, respectively, a maximum tree depth of 8, and mild linear parsimony pressure (0.01) after achieving a 50% improvement. GP was terminated when either the MSE was < 1e-8, after 20 generations without improvement, or after a maximum of $GPgen_{max} = 150$ generations. In GP+DE, constants in the evolving expression trees are represented by placeholders in the chromosome. To evaluate the fitness of an individual, the DE described above is used to optimize the constants in the tree, and the best fitness found by the DE is used as the fitness in the GP. We also implemented GP with ERCs generated in the range $[-b\ b]$, where $b \in \{1, 10, 100, 500\}$; these ERC methods are denoted GP [b].

We tested the GP methods on 30 randomly generated expression trees (10 each with 1-, 2- and 3- variables) with up to 6 real-valued constants randomly generated from the range [-500 500], using a function set of {+, - ,*, /, ^}. The resulting trees had varying difficulty, with up to 11 operators and 23 nodes. Training sets comprised 100 × number of variables points randomly sampled over the input variable ranges [-2 2]. The best evolved trees were then evaluated over more than 6.4e4, 1.3e6, and 1.2e7

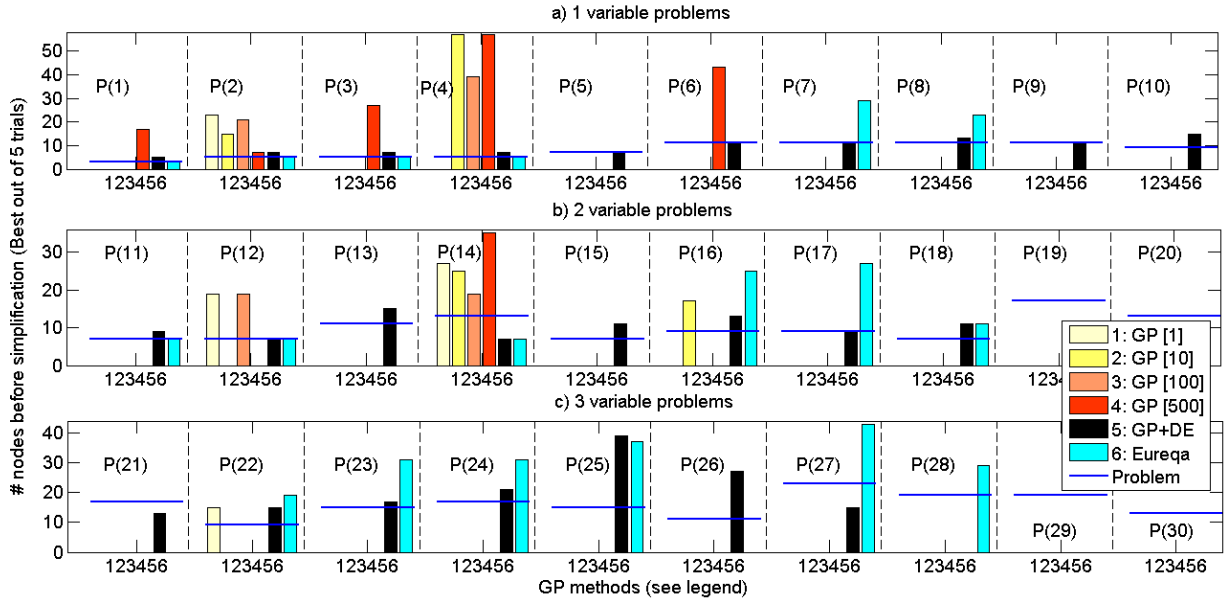

Figure 1: Number of nodes (before simplification) in the best successful solutions (of 5 trials) on (a) 1-variable, (b) 2-variable and (c) 3-variable problems. Missing bars signify that no successful solutions were found on a problem by the method. Horizontal lines show the number of nodes in the problem expression. None of the methods solved problems 19, 20, 29, 30 (hidden by the legend).

densely and uniformly spaced points over the range [-4 4] for the 1-, 2-, and 3-variable problems, respectively. Our intention in extrapolating well outside the training range is to discriminate which expressions were correctly estimated from those that simply over-fit the data in the training range. Resulting predicted and actual values were normalized to the maximum actual value over this larger range; evolved solutions were considered successful if their maximum normalized absolute error was < 1%.

For comparison, we also tried to solve the same 30 problems with Eureqa [4], although we were not able to specify (or even ascertain) many of the control parameters, including population size. Eureqa was allowed to evolve until the best MSE < 1e-8, or until no further improvement was seen (in the latter case, no runs were terminated before 28,500 generations). We then selected the smallest tree on the non-dominated front returned by Eureqa that met our success criterion. For all methods, we report results for the best individual returned out of 5 independent trials.

3. RESULTS AND DISCUSSION

As shown in Fig. 1, GP+DE (black bars) had successful runs on 25 of the 30 problems, whereas GP using ERC was only able to solve {4, 4, 4, 6} of the problems with $b \in \{1, 10, 100, 500\}$, respectively. Furthermore, the heights of the bars indicate that successful GP+DE solutions were generally much smaller than successful GP solutions using ERC. This implicit parsimony occurs because the nested co-evolution of constants with DE enables GP+DE to converge in many fewer GP generations, since correct expression trees can be more readily identified when the constants are also correct. In this study, successful GP+DE solutions required a median of only 8 GP generations. Despite this, GP+DE was still relatively slow, since each GP generation required a median of 149 DE generations per individual in the GP population. Eureqa succeeded on 18 of the 30 problems. Of these, 7 were approximately double the size of the GP+DE solutions (even though minimizing both size and MSE are explicit objectives in Eureqa) and 10 were similarly sized. Eureqa succeeded on 1 problem (28) that GP+DE was not able to solve within the allowed stall generations (although this required 31,724 Eureqa generations). However, GP+DE succeeded on 8 problems that Eureqa did not. In a few cases (problems 14, 21, and 27), solutions considered successful based on our extrapolation error criterion had fewer nodes than the target problems; these were actually slightly suboptimal solutions, but the missing terms contributed negligibly to the function values.

In summary, using nested DE to co-evolve constants enables GP to evolve parsimonious and accurate solutions to more, and more complex, symbolic regression problems, albeit with significant computational overhead. As computational resources are becoming increasingly cheap while problems are becoming increasingly complex, this trade-off may often be warranted.

4. REFERENCES

[1] B.M. Cerny, P.C. Nelson and C. Zhou. Using differential evolution for symbolic regression and numerical constant creation, *Proc. Of the 10th Annual Conf. on Genetic and Evol Comp, pp.* 1195-1202, 2008.

[2] I. Dempsey, M. O'Neill and A. Brabazon. Constant Creation and Adaptation in Grammatical Evolution. *Foundations in Grammatical Evolution for Dynamic Environments,* pages 69-104. Springer. 2009.

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

[4] M. Schmidt and H. Lipson. Distilling Free-Form Natural Laws from Experimental Data, *Science*, Vol. 324, no. 5923, pp. 81-85, 2009.

[5] R. Storn and K. Price. Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces, *J. Global Optimiz*. vol. 11, pp.341 – 359, 1997.

[6] Q. Zhang, C. Zhou, W. Xiao, and P.C. Nelson. Improving Gene Expression Programming Performance by Using Differential Evolution, *Proc. of the 6th Intl Conf on Mach Learn and Appl*, p.31-37, 2007.