Ramped Half-n-Half Initialisation Bias in GP

Conference Paper *in* Lecture Notes in Computer Science · July 2003

DOI: 10.1007/3-540-45110-2_71 · Source: DBLP

CITATIONS
4

READS
833

3 authors, including:

Steven Gustafson
General Electric
59 PUBLICATIONS **1,373** CITATIONS

SEE PROFILE

Graham Kendall
University of Nottingham, Malaysia Campus
329 PUBLICATIONS **11,246** CITATIONS

SEE PROFILE

Ramped Half-n-Half Initialisation Bias in GP

Edmund Burke, Steven Gustafson*, and Graham Kendall

School of Computer Science & IT
University of Nottingham
{ekb,smg,gxk}@cs.nott.ac.uk

Tree initialisation techniques for genetic programming (GP) are examined in [4,3], highlighting a bias in the standard implementation of the initialisation method *Ramped Half-n-Half* (RHH) [1]. GP trees typically evolve to random shapes, even when populations were initially full or minimal trees [2]. In canonical GP, unbalanced and sparse trees increase the probability that bigger subtrees are selected for recombination, ensuring code growth occurs faster and that subtree crossover will have more difficultly in producing trees within specified depth limits. The ability to evolve tree shapes which allow more legal crossover operations, by providing more possible crossover points (by being bushier), and control code growth is critical. The GP community often uses RHH [4]. The standard implementation of the RHH method selects either the **grow** or **full** method with 0.5 probability to produce a tree. If the tree is already in the initial population it is discarded and another is created by **grow** or **full**. As duplicates are typically not allowed, this standard implementation of RHH favours **full** over **grow** and possibly biases the evolutionary process.

The **full** and **grow** methods are similar algorithms for recursively producing GP trees. The **full** algorithm makes trees with branches extending to the maximum initial depth. The **grow** algorithm does not require this and allows branches of varying length (up to the maximum initial depth). As many more unique trees exist which are full (as full trees contain more nodes), there is a tendency, especially with particular function and terminal sets, to produce more duplicate trees with the **grow** method.

To estimate the bias of the RHH method with a particular function and terminal set, we use the results from Luke [3] (Lemma 1). The expected number of nodes E_d at depth d is defined as: $E_d = \{1 \text{ if } d = 0, \text{else } E_{d-1}pb \text{ if } d > 0\}$ where pb is the expected number of children of a new node (p is the probability of picking a nonterminal, and b is the expected number of children of a nonterminal). Luke [3] uses this to calculate the expected size of trees in the infinite case. Here we bound the calculation to depth $d = 4$. For our analysis, E_d correctly predicted the **full** method would contribute more trees to the initial population whenever the expected size of the two methods was not similar (i.e. **grow** made smaller trees, causing more duplicates and rejected trees).

Canonical GP trees grow in size to maximum depth limits, making the initial trees seeds for the evolutionary process. As the **full** algorithm is more likely to evolve larger and more bushier trees with more nodes than the **grow** method, we conduct an experimental study to observe these differences in the evolutionary

* corresponding author

E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2724, pp. 1800–1801, 2003.

process on standard problems. We use RHH and the **full** and **grow** methods exclusively. Initial depths of 4 and 6, with maximum tree depths of 10 and 15, respectively, a population size of 500, a total of 51 generations, and standard subtree crossover is used for recombination on the artificial ant, even-5-parity, and symbolic regression problem with the quartic polynomial. 50 random runs were performed for each of the 6 experiments for each problem.

On the ant problem, GP produced the best fitness with smaller initial trees (smaller initial depth or those produced by the **grow** method). Initial trees in the ant problem are particularly important as they encode the initial path that the ant takes. The parity problem experiments produced better fitness with bigger and more fuller trees; the populations created only by **grow** had the worst fitness, followed by the smaller depth limit populations. The RHH method causes 70% of the initial ant and parity population to be created by the **full** method, a negative bias for the ant problem and a positive bias for the parity problem. The regression problem always produced trees which were sparse due to the number of unary functions in the regression function set (log, exp, sin, cos), which is typical in function sets for more complex regression problems. While the RHH method was not overly biased (**full** produced 55% of initial trees in the RHH experiments and **full** and **grow** had similar fitness), the sparseness and unbalancedness of trees caused a significant loss of genotypic diversity (the number of unique trees).

The bias of the RHH method can be positive or negative, but it can also effect diversity. A loss of genotypic diversity can result from two factors here: the creation of trees by crossover already in the population, or the failure of crossover to find acceptable crossover points (leading to the duplication of one of the parents). The ant and parity populations produced with **grow** were sparser, more unbalanced and lost diversity. All the regression populations had similar behaviour due to the function set.

A disadvantage exists for GP trees which are unable to grow effectively. GP trees which grow sparse and unbalanced will cause more code growth, less genotypic diversity, and search a smaller space of possible programs. These populations will be less effective in the evolutionary process. Current research is investigating various ways to adapt and detect program shapes for improved performance of fitness and recombination operators.

References

1. J.R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, Cambridge, MA, USA, 1992.
2. W. Langdon, T. Soule, R. Poli, and J.A. Foster. The evolution of size and shape. In Lee Spector et al., editors, *Advances in Genetic Programming 3*, chapter 8, pages 163–190. MIT Press, Cambridge, MA, USA, June 1999.
3. S. Luke. Two fast tree-creation algorithms for genetic programming. *IEEE Transactions on Evolutionary Computation*, 4(3):274–283, September 2000.
4. S. Luke and L. Panait. A survey and comparison of tree generation algorithms. In L. Spector et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 81–88, San Francisco, USA, 7-11 July 2001. Morgan Kaufmann.