Generalisation in Genetic Programming

W. B. Langdon
Dept. of Computer Science, University College London
Gower Street, WC1E 6BT, UK
W.Langdon@cs.uc1.ac.uk

ABSTRACT

GP can evolve large general solutions using a tiny fraction of possible fitness test sets. Just one test may be enough.

Categories and Subject Descriptor I.2.8 [search]: heuristic
General Terms: Theory

Natural evolution has been applied to computer programs for twenty years. However GP still lacks rigorous widely applicable predictive mathematical foundations [1, 6]. We investigate artificially evolved computer program's ability to correctly answer even when neither they nor any of their ancestors have seen the problem before. (Cf. "delta evaluation" [2, 7].)

In Software Engineering it is impossible to test exhaustively. So there is considerable interest in the number of tests needed and the level of confidence they confer on the code under test. GP can create software which works even on examples where it has not been tested but this is taken for granted. As yet there is no rigorous proof of how well GP will generalise or why or when randomly assembled computer programs will predict unknown cases. Typically machine learning assumes that bigger solutions are liable to over fit the training data, i.e. to not generalise, and use information theory, regularisation, etc., to reduce the size of models. However evolution was successful for billions of years before William of Occam [4].

Figure 1 shows typical evolution of a GP population which reliably evolves complete solutions to the 11-multiplexor problem using one eighth of the normal training data. I.e. it correctly guesses the other $7/8^{th}$, yet these correct solutions are far from parsimonious. The third plot shows the typical evolution of GP on 20-Mux where each individual is only shown 32 tests but bloated programs are evolved which can guess correctly the other 1 048 544 answers. (The 11-Mux can be solved with one test. However we did try the 20-Mux with only one test, and a GP population of up to 4 million, but the run ran out of power and did not find a general solution.) Plot 4 shows the 37-Mux can be solved when using only $1/17^{th}$ million part of the whole test suite.

As expected in successful evolved populations training and validation performance is correlated ($0.19 \leq r \leq 1$). What is it about the multiplexor problems that allows GP to evolve generalising solutions? They are non-trivial high dimensional deceptive and discrete. They have no parameters

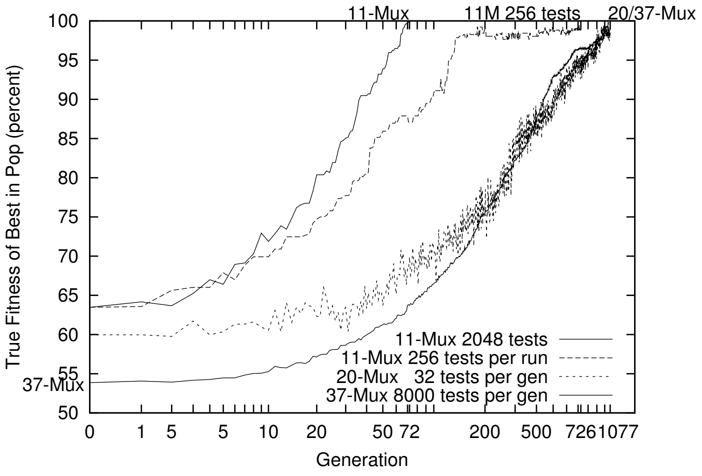

Figure 1: Evolving general multiplexers. 1) conventional GP (all 2048 tests), 2) 256 randomly chosen tests (11-Mux pops 16 384). 3) 32 unique random tests (of 1 048 576) 4) 8000 tests (of $1.37\ 10^{11}$) each gen (pops 262 144). Note "logarithmic" fitness increase is reminiscent of the coupon collector. This suggests building blocks are equally difficult.

suitable for continuous gradient ascent. GP's ability to generalise has been known for some time [3]. How many other problems shared this ability to generalise? We should certainly expect problems where evolved programs do not generalise. Code available via FTP `cs.ucl.ac.uk` directory `genetic/gp-code/gp32cuda.tar.gz` [5]. UCL technical report RN/11/10 investigates further.

[1] H.-G. Beyer and W. Langdon, editors. *Foundations of Genetic Algorithms*, Austria, 5-9 Jan 2011. ACM.

[2] T. C. Fogarty, F. Vavak, and P. Cheng. Use of the genetic algorithm for load balancing of sugar beet presses. In L. J. Eshelman, editor, *ICGA* pp617–624, 1995.

[3] W. B. Langdon. *GP and Data Structures.* Kluwer, 1998.

[4] W. B. Langdon. Was Occam wrong? Blunting Occam's razor. *BNVKI newsletter*, 19(3):56–57, June 2002.

[5] W. B. Langdon. A many threaded CUDA interpreter for genetic programming. *EuroGP 2010*, p146–158.

[6] W. B. Langdon and R. Poli. *Foundations of Genetic Programming.* Springer-Verlag, 2002.

[7] P. Ross, D. Corne, and H.-L. Fang. Improving evolutionary timetabling with delta evaluation and directed mutation. In *PPSN III*, *LNCS 866*, pp556–565. 1994.

GECCO'11, July 12–16, 2011, Dublin, Ireland.
ACM 978-1-4503-0690-4/11/07.