William B. Langdon

GGGP, Department of Computer Science, University College London, Gower Street, London WC1E 6BT, UK (Email: w.langdon@cs.ucl.ac.uk)

Long-Term Evolution of Genetic Programming Populations

1 Why

More challenging problems may require running evolution for longer. Hence the need to study what happens in long runs. Perhaps we can anticipate and solve problems that may occur

2 Benchmark: 6 input multiplexor

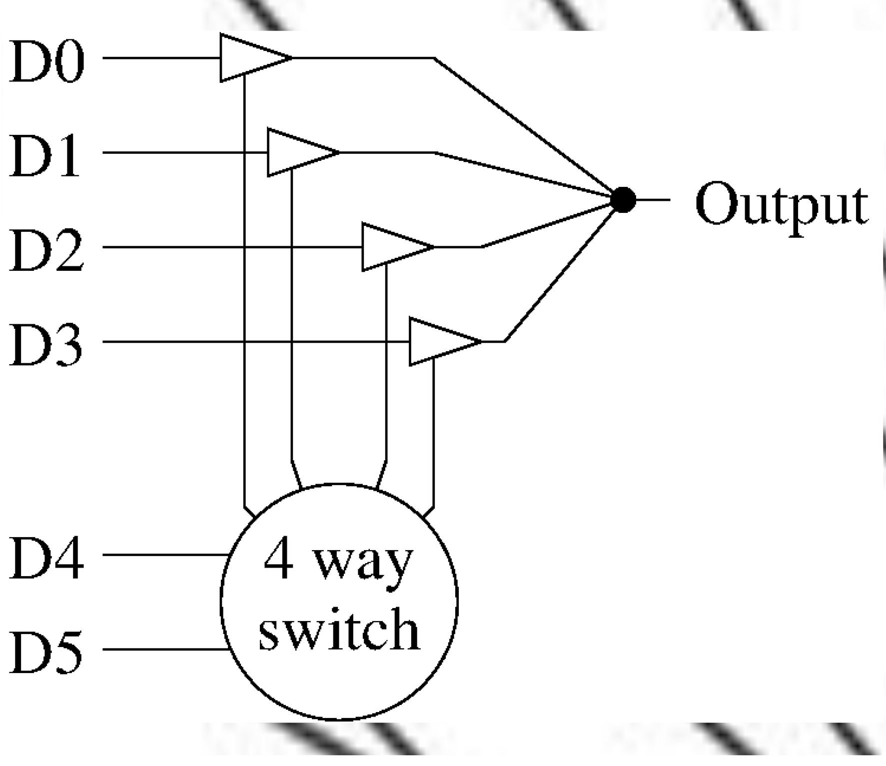


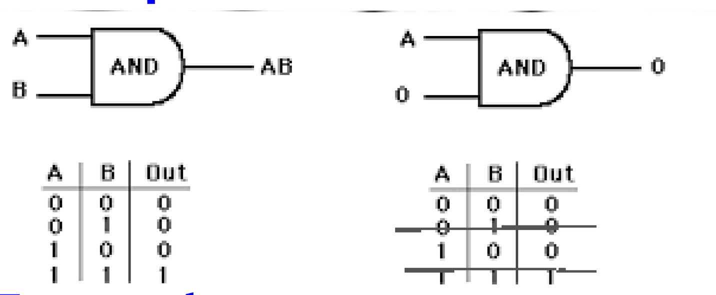
Figure 1: GP bench mark 6 multiplexor.

- Six inputs: Use two (D4 D5) as binary number to connect corresponding data lines (D0-D3) to the output
- Test on all 2^6=64 possible combinations
- Fitness score (0-64) is number correct

3 Introns, Constant, Effective code

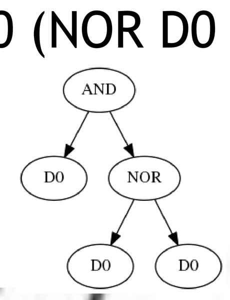
- Subtree like whole tree.
- Output of subtree is via its root node
- **Intron**: subtree which has no effect on overall fitness. I.e. its output does not impact on root node of whole tree.
- **Constant** subtree always has same output, i.e. same output on all 64 test cases.
- Remaining **effective code** has an impact on root node. Typically it is next root node

4 Example Intron: AND function



5 Example constant: zero

(AND D0 (NOR D0 D0)) = 0

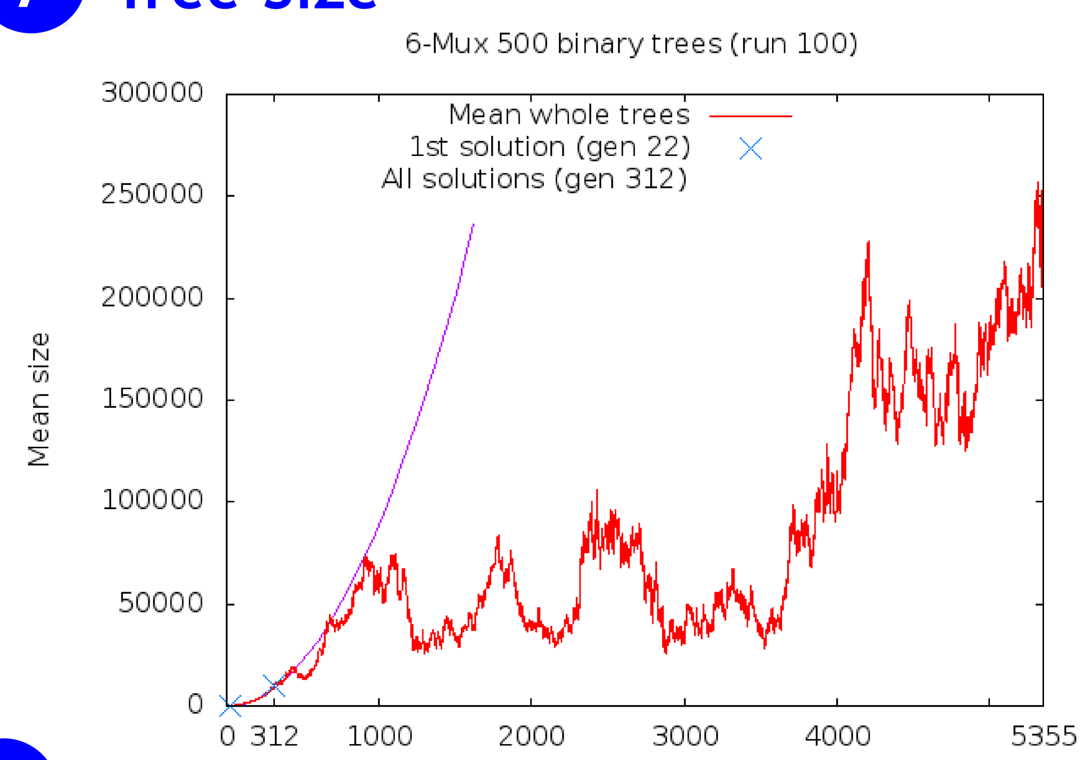


6 Long term evolution of 6-mux trees

Terminal set:	D0 D1 D2 D3 D4 D5
Function set:	AND OR NAND NOR
Fitness cases:	All 2^6 combinations of inputs
Selection:	tournament size 7
Population:	500. Panmictic, non-elitist, generational.
Parameters:	Ramped half-half depth 2-6. 100% subtree crossover. Stop first tree > 1million nodes

Background evolution of life
https://doi.org/10.1371/journal.pone.0002566

7 Tree Size



8 Evolution of Effective Code

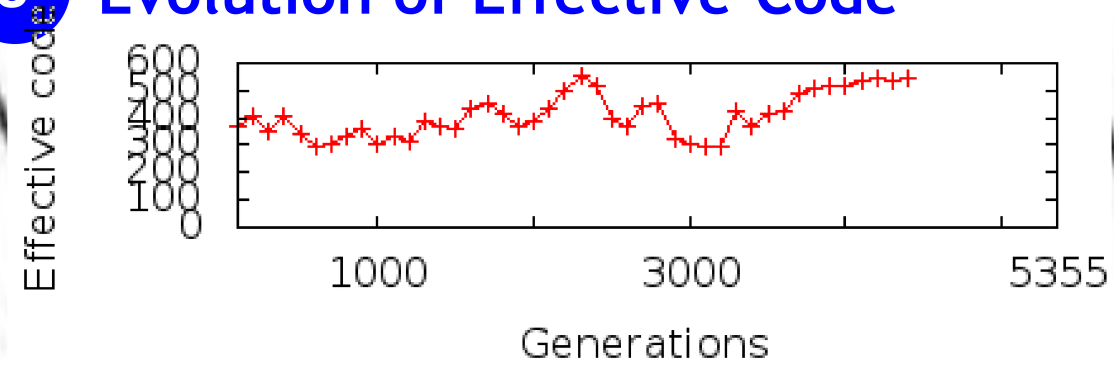
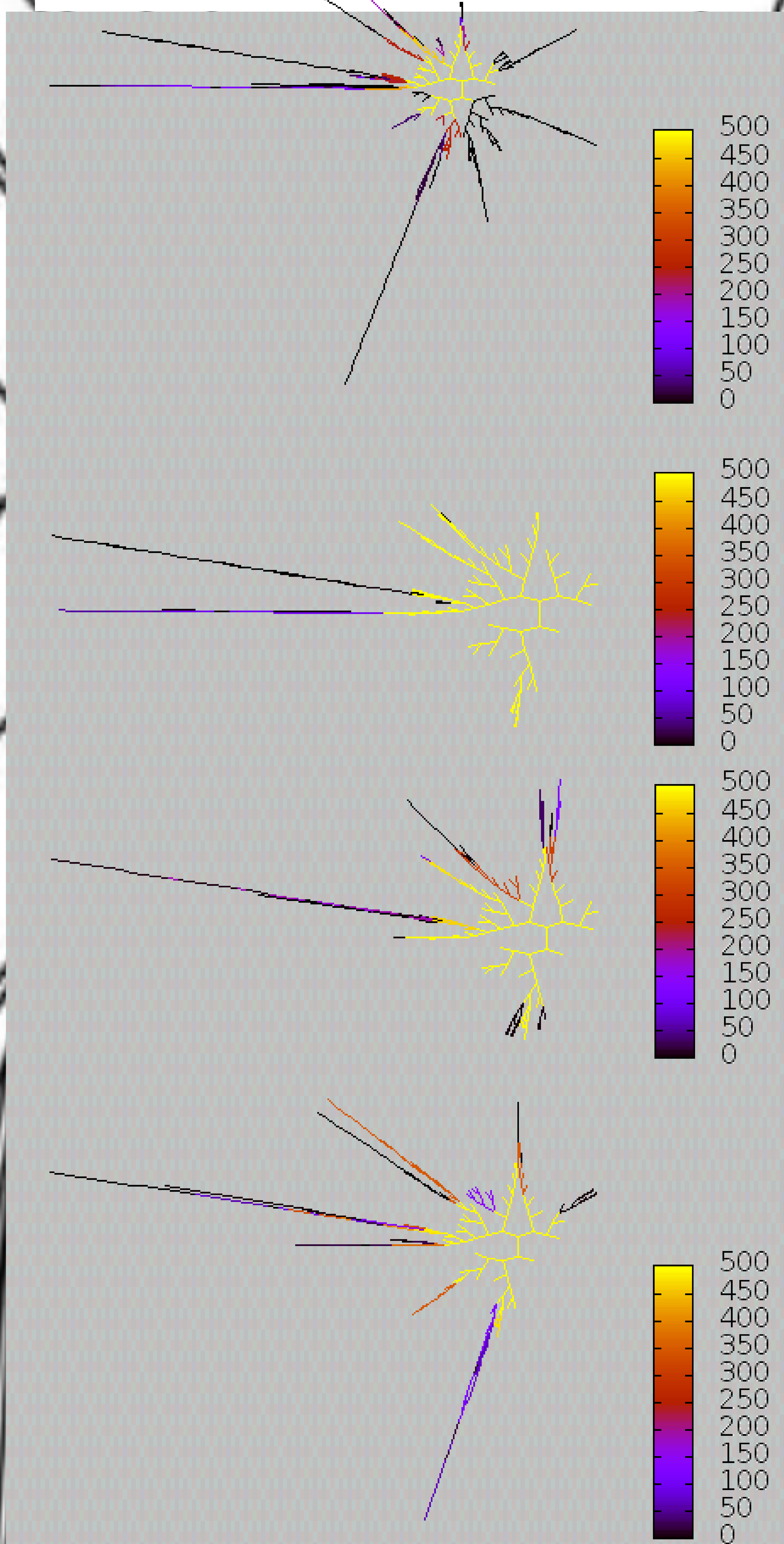


Figure 2: Size of effective code is fairly stable



Effective code in population of 500 binary trees after 500,1000, 2000 and 4000 gens. Note similarity. In gen 4000 almost all the population have 82 effective nodes in common (yellow). Darker colours indicate effective code which occurs in <148 (blue) or <22 (black) trees. Ineffective code not plotted.

Animations: http://www.cs.ucl.ac.uk/staff/W.Langdon/gggp/bmux6.100.gif

9 A Limit to Bloat

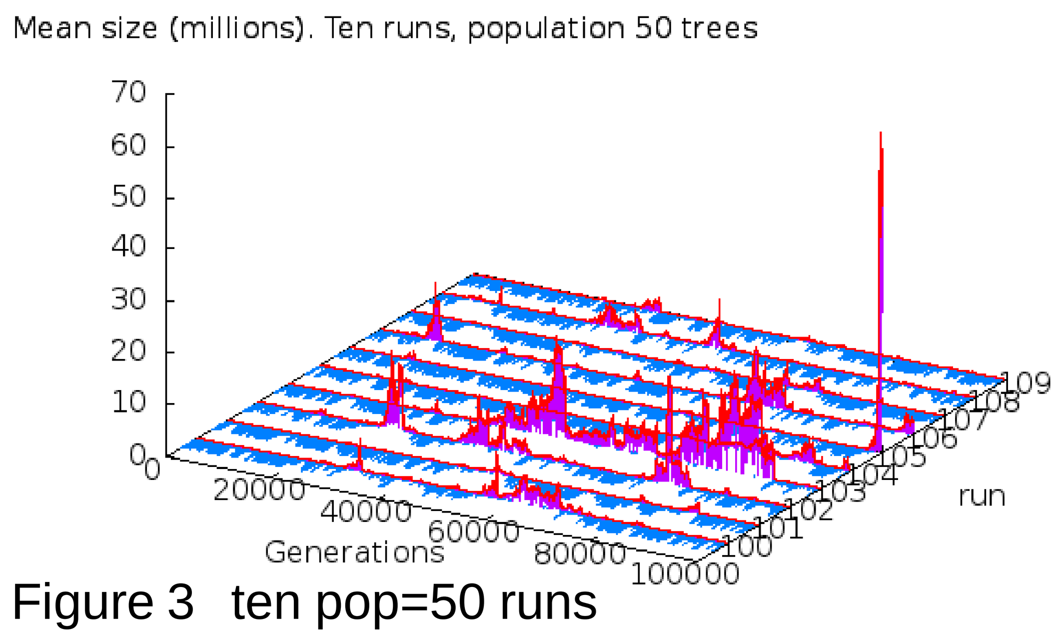


Figure 3 ten pop=50 runs

An Edge to Bloat

Bloat cannot continue forever. Even after thousands of generations fitness selection is needed to sustain tree size. When everyone in the population has same fitness, there is no selection. Tree size wanders apparently at random (Gambler's ruin).

In a finite population trees may become so large that effective code is never disrupted by crossover.

Number runs ≈ popsize × coresize / treesize

Runs < 1 suggests

treesize > popsize × coresize

Ten smaller pop=50 runs, estimate coresize 500

Estimate treesize about 25 000

The median of ten runs of the mean tree size over 100 000 generations is 42 507.

10 Summary

Run GP for many thousand of generations. See **convergence** in that almost all of population has same fitness, but every tree is unique. However every tree is the **same near their roots** and **effective code is conserved** across many generations.

11 Next

Next:

- Why quadratic increase in size < gen 350
  - Existing theory
- differences from crossover only limit
- Formalise random drift
- Which types of GP will converge like this?

Reference: Long-Term Evolution of Genetic Programming Populations, W.B. Langdon.
In GECCO comp 2017, 235-236. 15-19 July.
RN/17/05 code GPbmux6.tar.gz gp2lattice