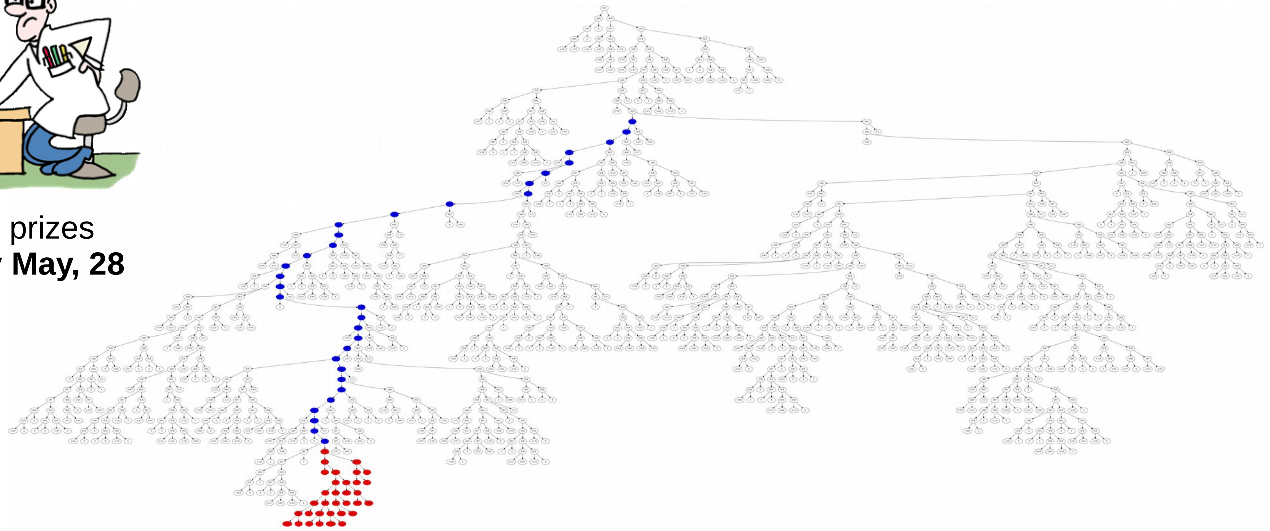
Fitness First

Genetic Programming Theory & Practice 2021

W. B. Langdon

Thursday 20 May, 2021 1:45pm

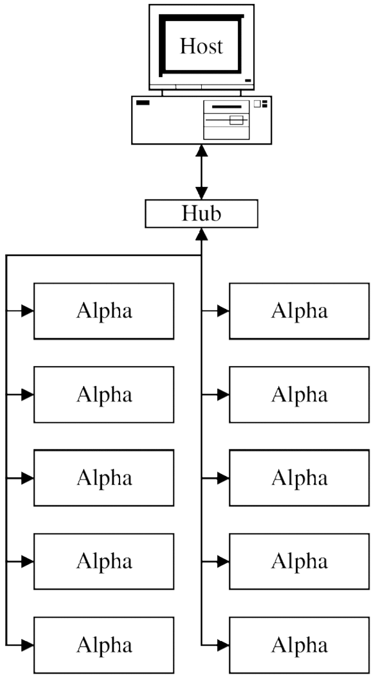
Humies $10000 prizes
Submit by **Friday May, 28**



slide 1 of 12

Fitness before Crossover

- Half a Peta Flops per day, [Bennett,1999]
- Half a Tera GP operation per second



16 core Intel 3.80GHz i7-9800X (2018)

16 x 16 AVX floats per clock tick

692 billion GP operations per second
~60 peta GPops per day

10 node Beowulf cluster
533MHz DEC alpha $18000

Reproducible.
Floating point operation v. GP instruction
+ - * / constant X

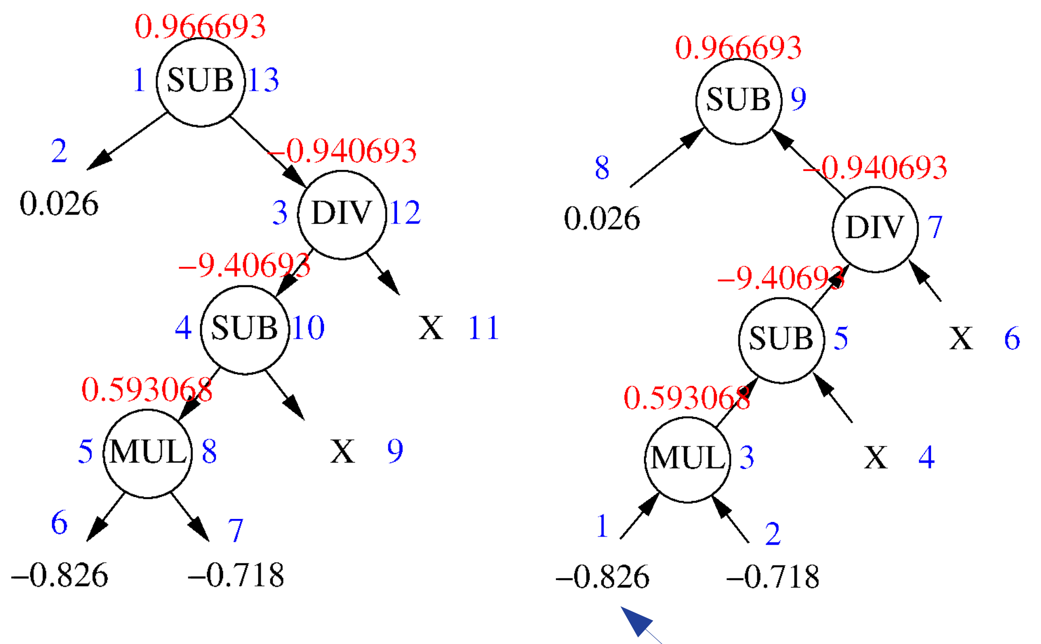
Why its fast: 692 billion GP operations/sec

- AVX evaluate 16 test cases simultaneously
- 16 cores parallel fitness and crossover
- Bloated trees, so crossover & convergence
  - Fitness 1st so avoid creating more than half children
  - Converged so fitness=mum's fitness stop evaluation once have proved child’s fitness is same as parent. Evaluate less than 1% of tree
- Result is identical to full evaluation (but faster)

Evaluate Fitness before Crossover

- Most GA individuals do not have children.
- Evaluating population fitness before crossover/mutation:
  - select by fitness
  - only fit parents (who will have children) need be created
- Evolution will be identical.

Top Down = Bottom Up



Left: Conventional top-down recursive evaluation of (SUB 0.026 (DIV(SUB (MUL -0.826 -0.718) X) X)). X=10.

Blue integers indicate evaluation order, red floats are node return values.

Right: an alternative ordering, starting with leaf -0.826 and working to root node.

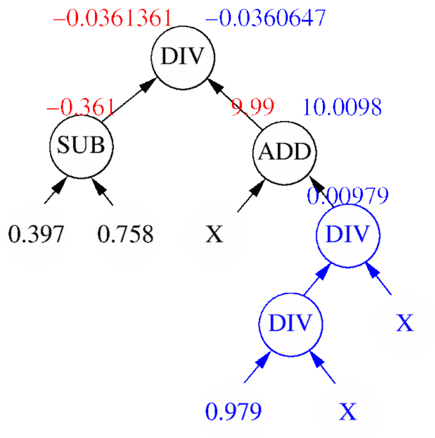
Both return exactly the same answer.

Genetic difference ≠> phenotypic difference

- Mum and child are identical except for inserted & removed subtrees (no side effects).
- If by chance inserted & removed subtrees are identical:
  - mum and child are identical and so have the same fitness
- If inserted subtree evaluates to same value as removed subtree on every test case:
  - mum and child (at root node) evaluate to same value on every test case
  - genetic difference => identical fitness
- What if the inserted subtree evaluates to different values to that given by remove subtree?
  - If we evaluate both child and mum starting at the change, there is a progressive fall in the number of test cases where the change is visible as we move towards the root node.

Evaluate both trees from change up

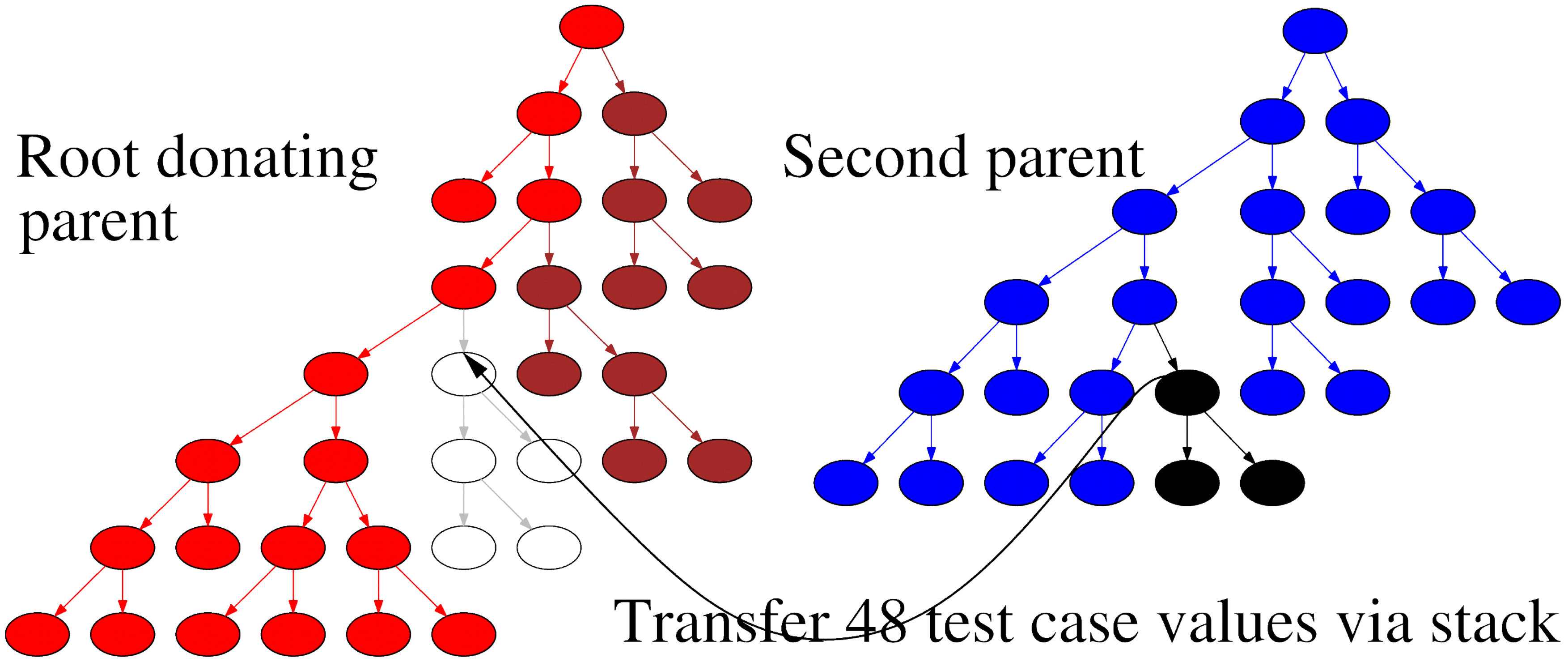
- Mum and child are identical above change, so use mum
- Fitness evaluation is identical except on route from change to root node.
- Evaluate both mum and child up this path.
- If they evaluate to identical values at any point then they evaluate to same value on the rest of the tree, including the root node:
    - semantic difference => identical fitness.



Evaluate mum in red. Evaluate child in blue.
Inserted code (DIV (DIV 0.979 X) X) in blue.
Here incremental evaluation proceeds 38 levels up the child tree before both mum and child evaluations are identical on all 48 test cases.

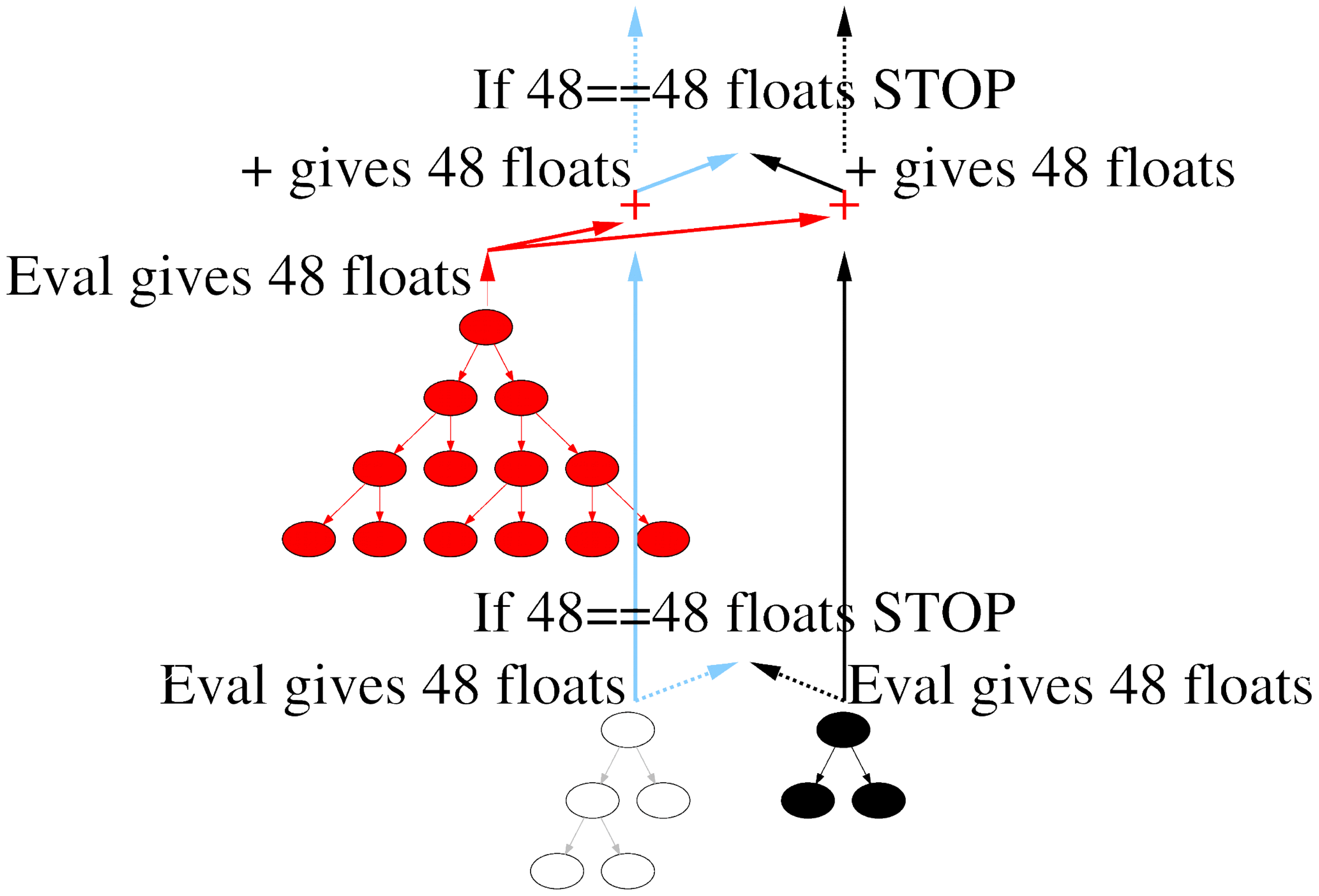
Functions lose information and so can give same output even with different inputs.

Fitness is evaluated using only parents



Subtree to be inserted (black) is evaluated on all test cases and values are transferred to evaluation of mum. Use incremental evaluation, so differences between original code (white subtree) and new (unborn) are propagated up 1st parent (mum) until either all differences are zero or we reach the root node.

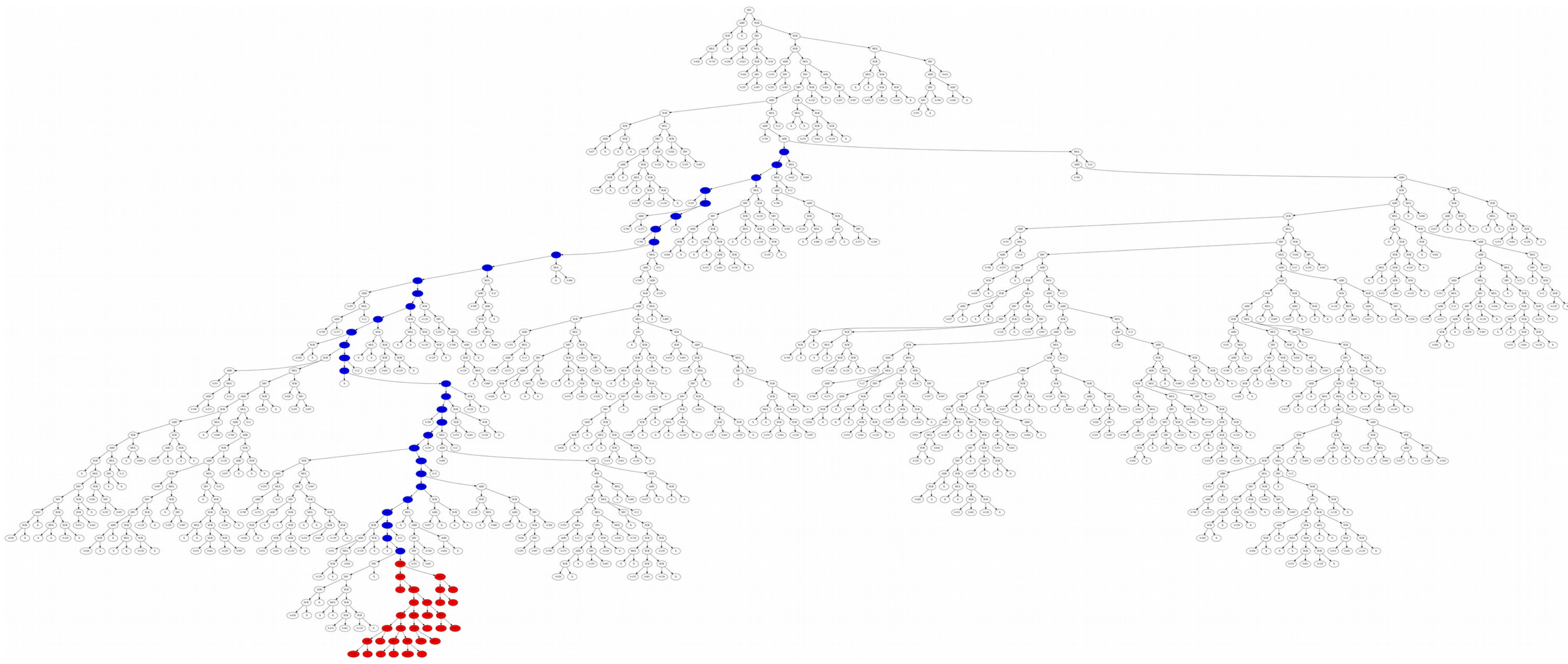
Interpret child using parents (without child)



Evaluating the subtree to be removed from the mum (white) and the subtree to be inserted (black) on all (48) test cases. The interpreter proceeds up the mum's tree until either the evaluation on all test cases in the mum and unborn child are the same or it reaches the root node.

Evaluate both trees from change up

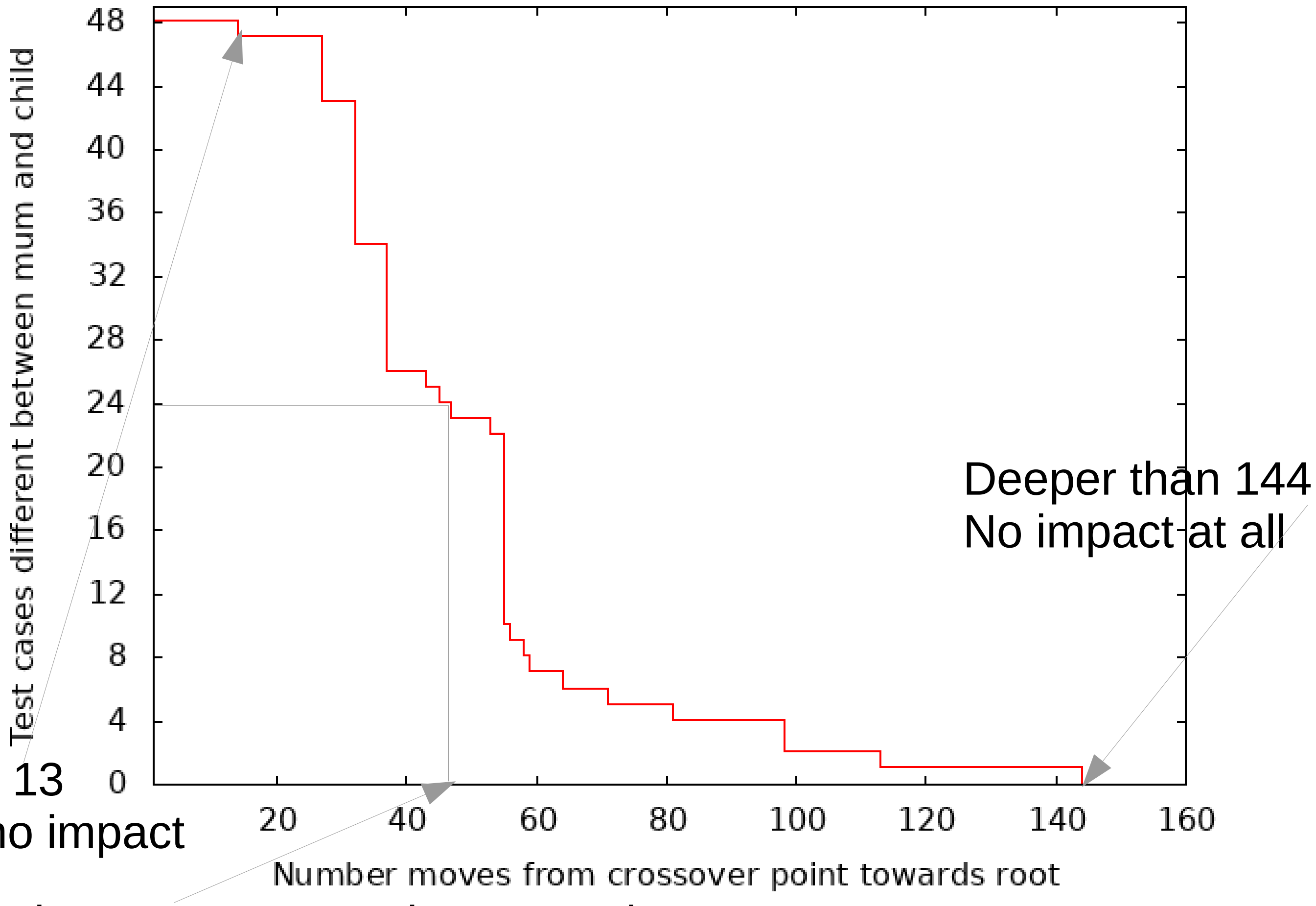
New code in red. Can stop fitness evaluation early as mum and child are phenotypically identical, due to information loss.



Example of bloated tree (chosen as it has no zeros, ie no “introns”). Bloat due to information loss.

Side Effect Free: Disruption Falls Monotonically

Deeper disruption tends to have less impact on fitness



Deeper than 144
No impact at all

Deeper than 13
3 tests see no impact

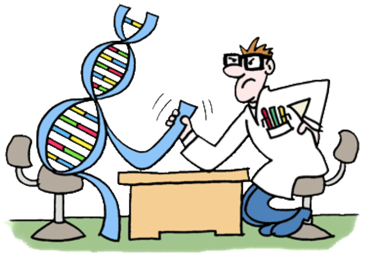
Deeper than 44 ½ tests observe no impact

At each GP node: 32 bits + 32 bits => 32 bits
Information funnel. Information is lost.

Conclusions

Test Case Disruption Falls Monotonically

1) More fitness test cases have only small effect, log(n)
   - 1000 test cases only marginally more effective than 48
2) Deeper crossover or mutation may have less effect
   - Design your new crossover & mutation operators
3) Some functions lose information faster, eg division
4) If no disruption gets to root, crossover/mutation no effect
   - fitness identical => GP pop converges & evolution stops
5) If on some test cases, disruption does not reach root, crossover/mutation has less impact on fitness:
   - Information loss gives smoother fitness landscape.



Human-Competitive results $10,000 prizes

Email your entry to goodman@msu.edu
by **Friday May, 28**

So What

Test Case Disruption Falls Monotonically

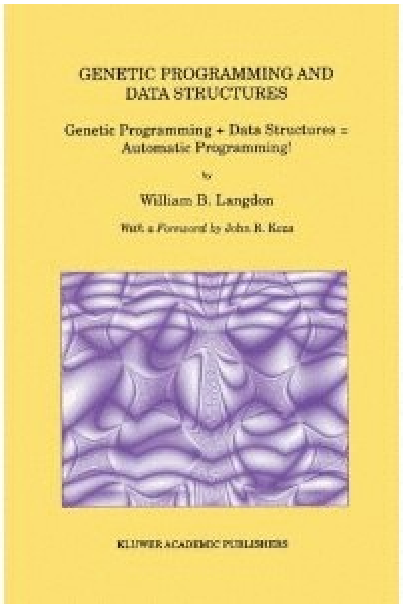
1) More fitness test cases have only small effect, log(n)
   - 1000 test cases only marginally more effective than 48
2) Deeper crossover or mutation may have less effect
   - Design your new crossover & mutation operators
3) Some functions lose information faster, eg division
4) If no disruption gets to root, crossover/mutation no effect
   - fitness identical => GP pop converges & evolution stops
5) If on some test cases, disruption does not reach root, crossover/mutation has less impact on fitness:
   - Information loss gives smoother fitness landscape.
6) Information loss gives bloat without introns
7) Fitness evaluation not always dominant cost
8) Multiple ways to evaluate tree

CREST

EPSRC

Genetic Programming

W. B. Langdon



GENETIC PROGRAMMING AND DATA STRUCTURES

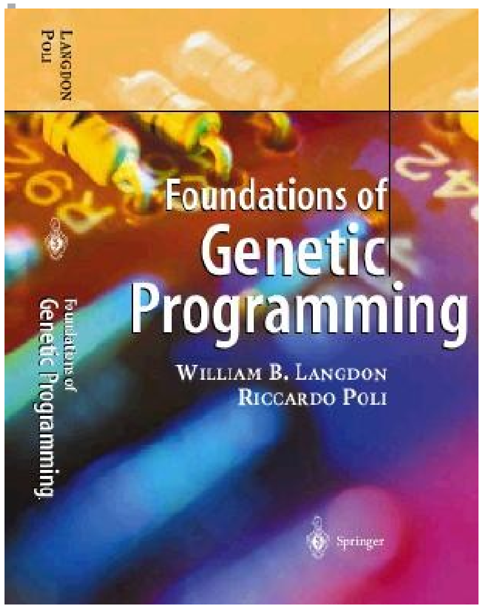
Genetic Programming + Data Structures = Automatic Programming!

by

William B. Langdon

With a Foreword by John R. Koza

KLUWER ACADEMIC PUBLISHERS



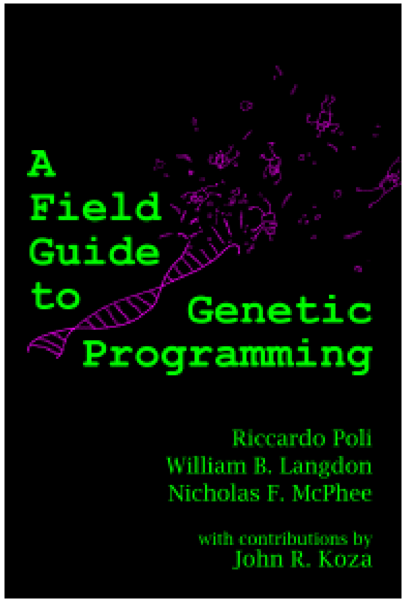
LANGDON POLI

Foundations of Genetic Programming

WILLIAM B. LANGDON RICCARDO POLI

Foundations of Genetic Programming

Springer

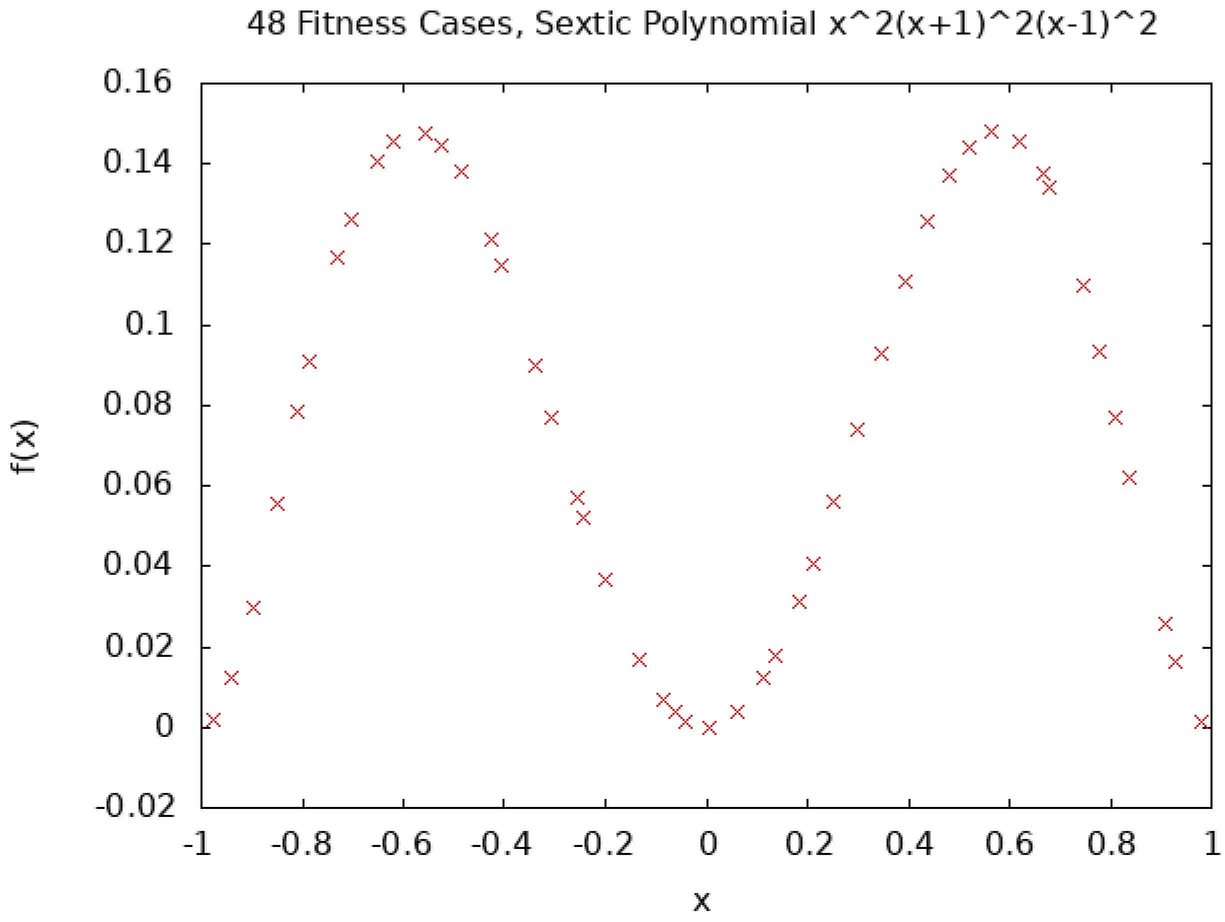


A Field Guide to Genetic Programming

Riccardo Poli
William B. Langdon
Nicholas F. McPhee

with contributions by John R. Koza

Genetic Programming Benchmark



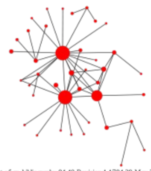
Sextic polynomial: match curve at 48 points

The Genetic Programming Bibliography

14493 references, 13000 authors

Make sure it has all of your papers!

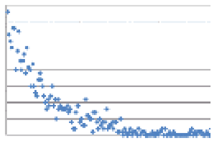
E.g. email W.Langdon@cs.ucl.ac.uk or use | Add to It | web link


Part of gp-bibliography 84 40 Revision:1.1794 29 May 2011

Co-authorship community.
Downloads

Downloads by day

A personalised list of every author's
GP publications.


Your papers

blog

Googling GP bibliography, eg:
Evolutionary Medicine site:gpbib.cs.ucl.ac.uk