CREST

Butterflies Considered Harmful

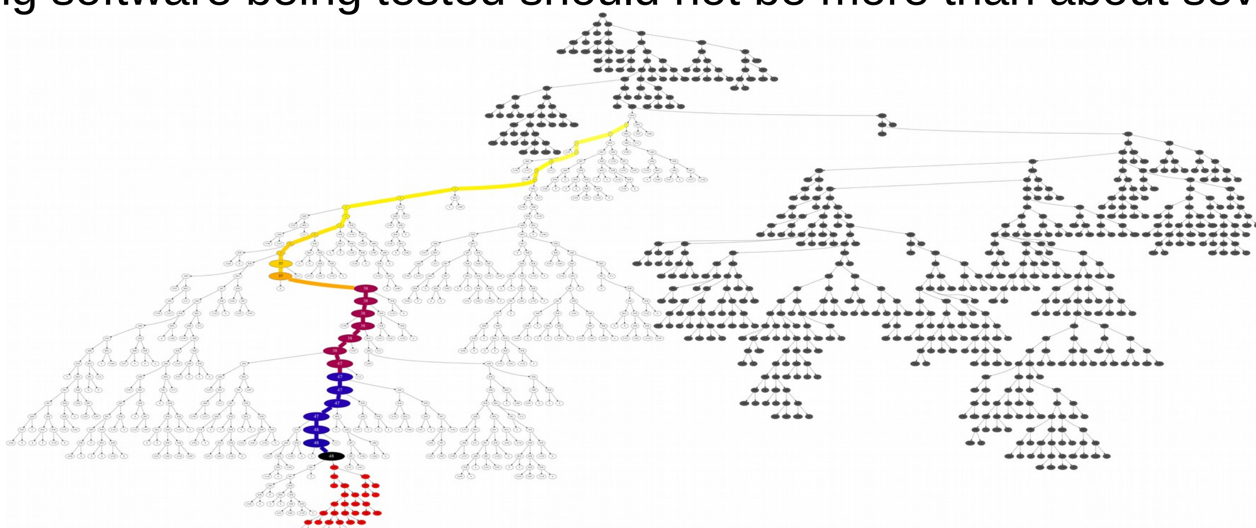
Wednesday 12 January 2022

Information theory suggests software is **not chaotic**.

Instead in deeply nested programs
most disruption fails to propagate to the output.

Exponential decay of failed disruption propagation
says **optimal test oracles** are at the error,
but next to the error is only 18% to 28% worse than optimal.
Suggesting software being tested should not be more than about seven levels deep

W. B. Langdon

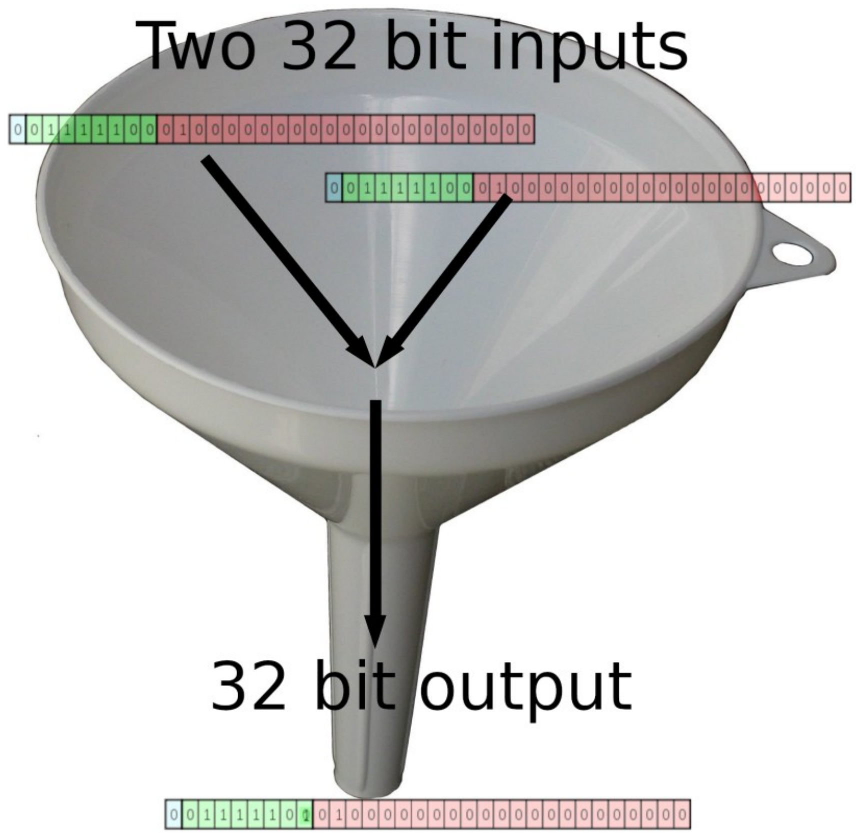


Information Theory and Experiments on Deep Genetic Programming Trees

- Information theory and failed disruption propagation
- Started with deep floating point polynomials
  - Injected errors lost mostly due to rounding error
- Evolve deep integer trees
  - Inject run time error everywhere, retest
  - 92% to 99.97% of errors have no effect
- Variation between programs
- Exponential decay with depth
  - Need to be close to error for tests to find them
  - On average <7 more than 50% errors detected
- Conclude by drawing lessons for programming

Information Funnel

Computer operators are irreversible. Meaning input state cannot be inferred from outputs. Information is lost

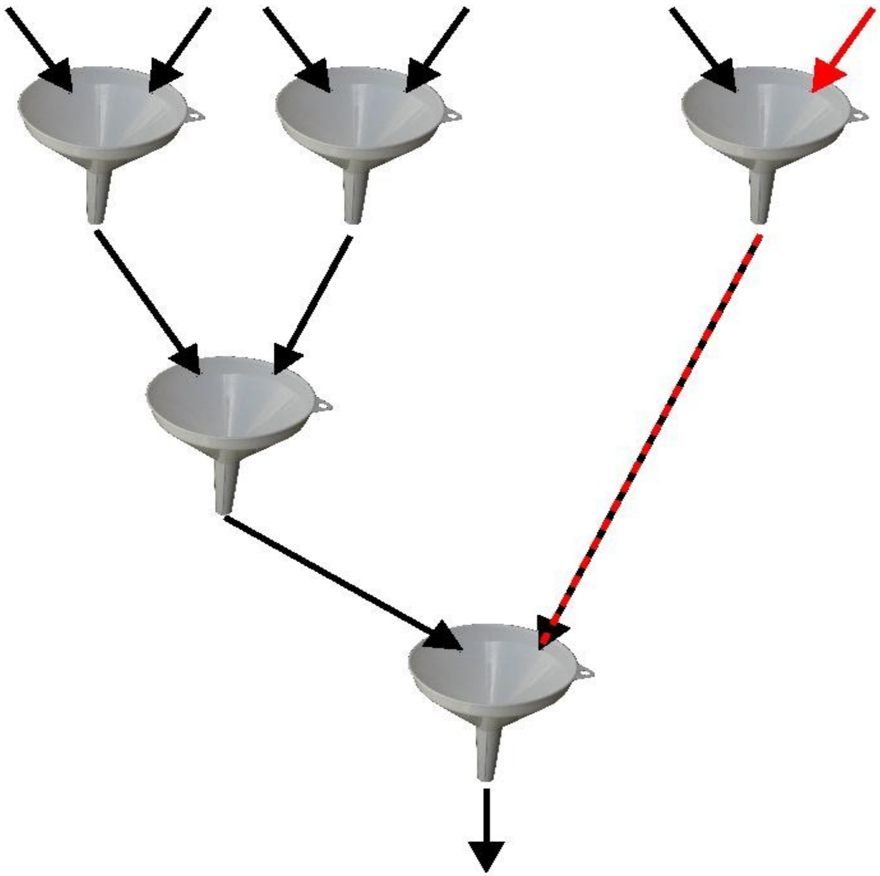


Information funnel

More information enters than leaves

Information flow in five nested functions

Potential information loss at each (irreversible) function



Disruption may fail to reach reach output.

(No side effects.)

Output
(often drawn at top of picture)

Evolve 10 Deep Integer GP Trees

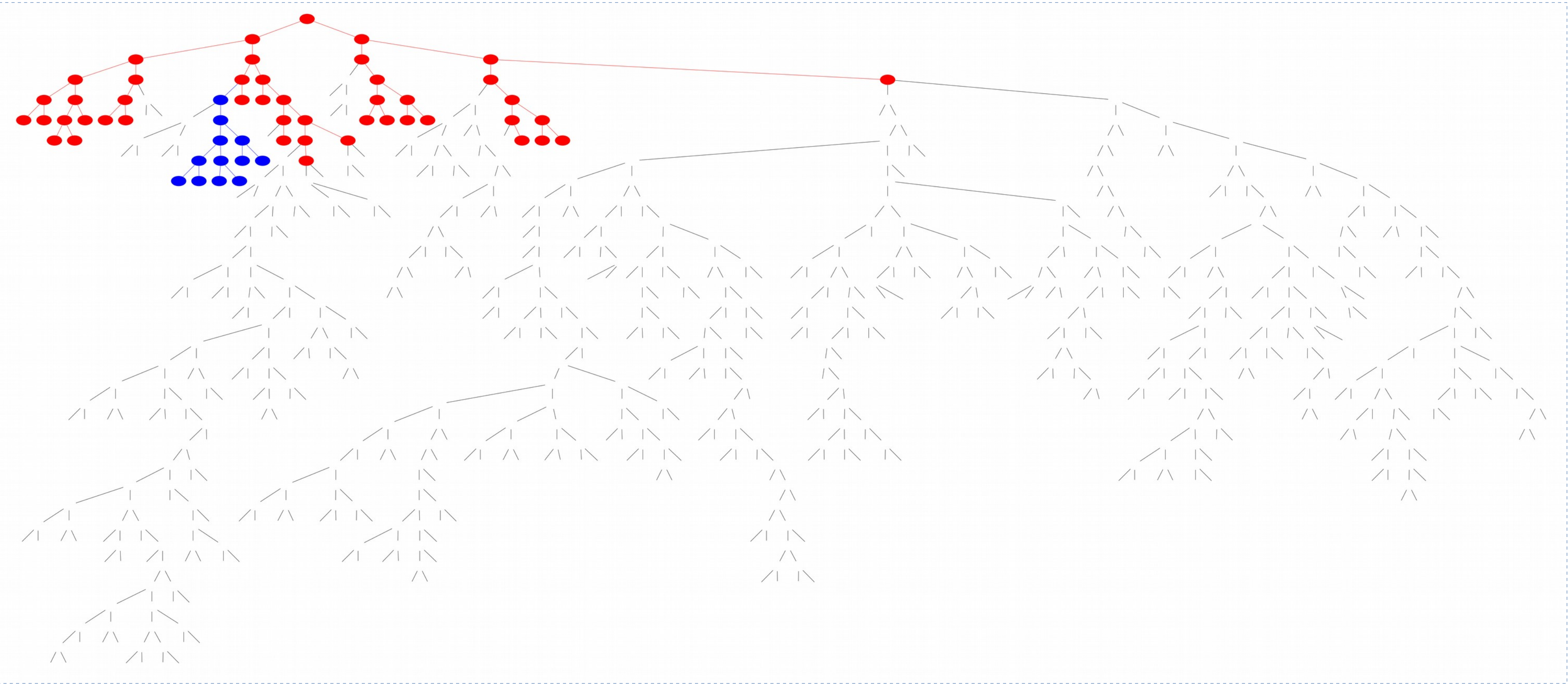
- Most GP experiments use float or Boolean, choose Koza's Fibonacci Problems.
  - Recursive program to generate Fibonacci sequence

    $X_J = X_{J-1} + X_{J-2}$ 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...
  - 0 1 2 3 J + - * SRF

    SRF(j,default) = jth value. default applies if j is invalid
  - Twenty tests J=0 … 19
  - Population 50000, 1000 generations
  - Ten runs
- Change at run time each point in tree on each of the 20 tests
  - Two run time disruptions: +1, replace with random int
  - +1 and RANDINT very similar
- Almost all run time disruptions make no difference

+1 Disruption. Run 7, tree depth 33

red 16-20 test cases, blue 1 test cases

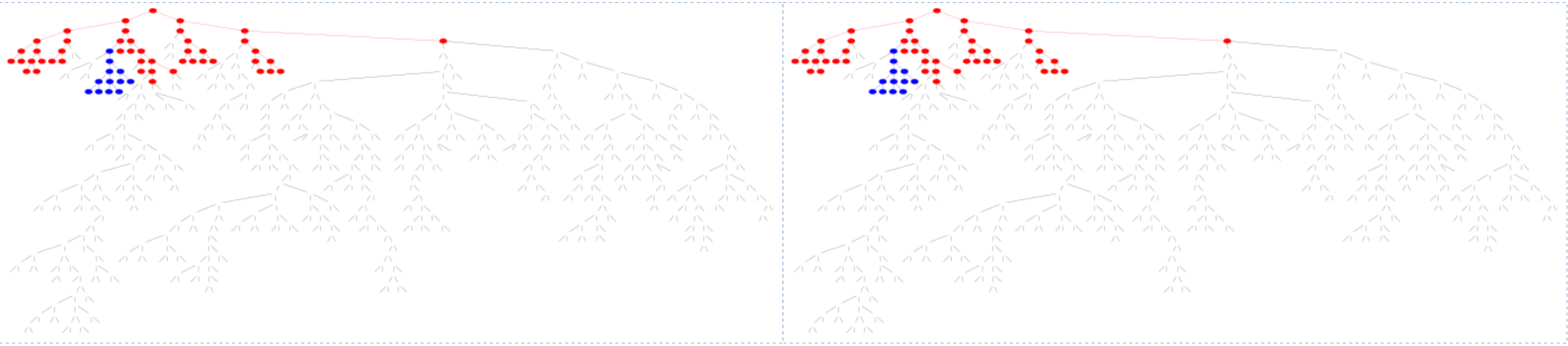


Only disruption near root node reaches output

CREST

Run 7, tree depth 33

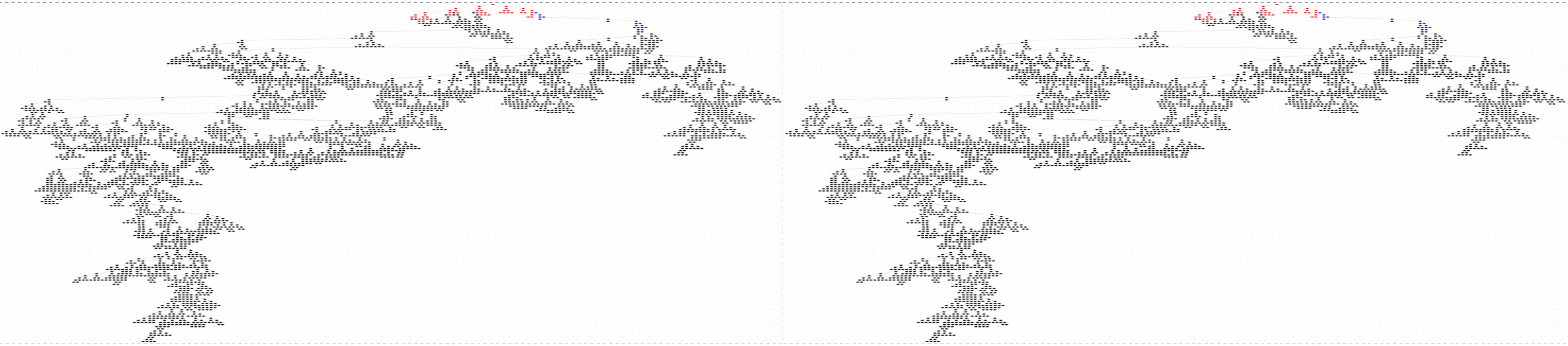
Red 26-20 test cases, blue 1 test cases



Same tree, +1 left, RANDINT right.
Almost identical response to different disruptions

Run 2, tree depth 160

Red 6-20 test cases, blue 1-2 test cases

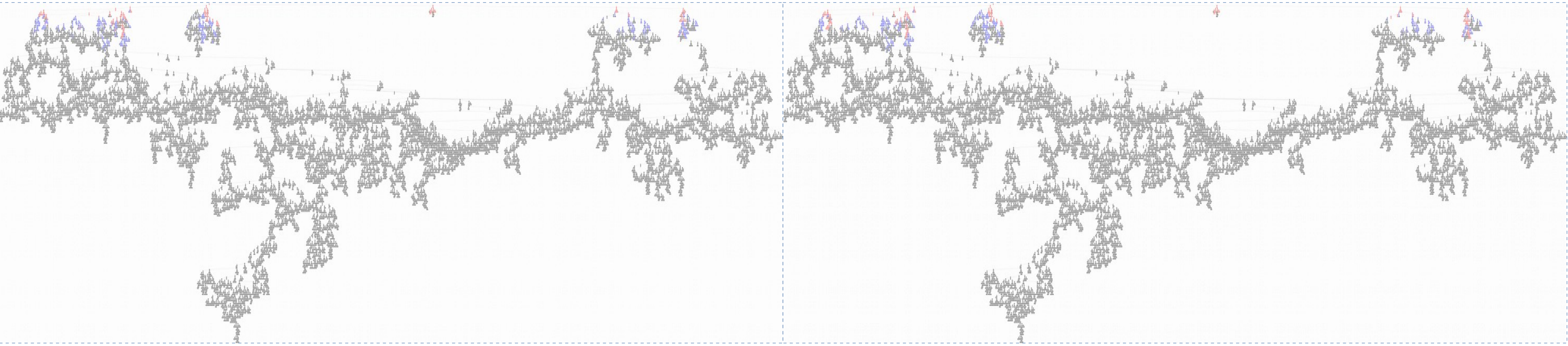


Same tree, +1 left, RANDINT right.

Almost identical response to different disruptions

Run 3, tree depth 220

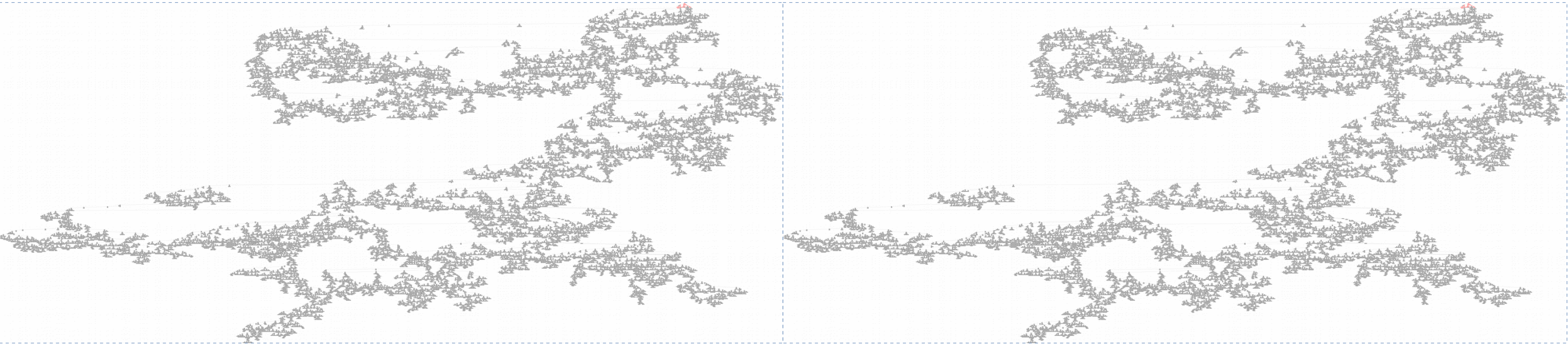
Red 4-20 test cases, blue 1-3 test cases



Same tree, +1 left, RANDINT right.
Almost identical response to different disruptions

Run 8, tree depth 425

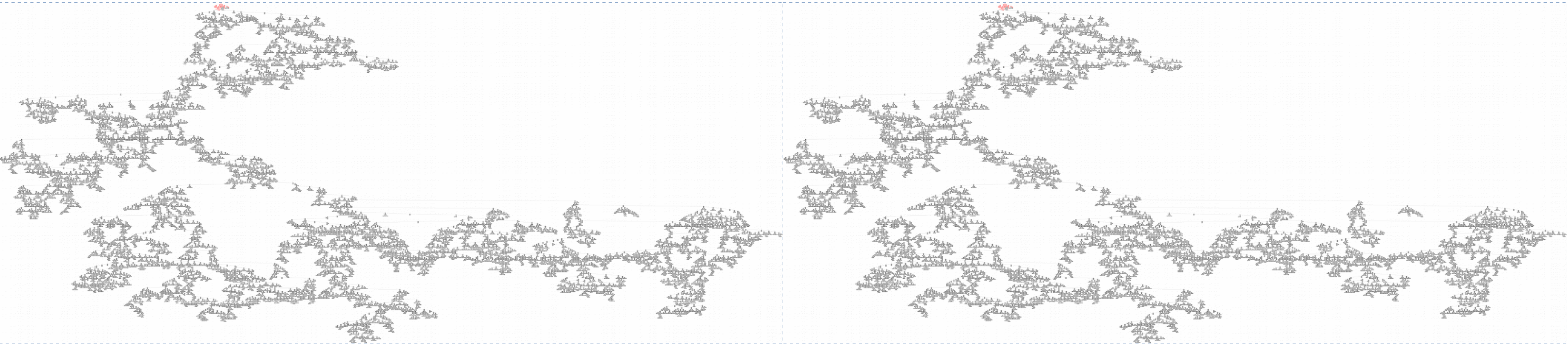
Red 17-20 test cases, no blue



Same tree, +1 left, RANDINT right.
Almost identical response to different disruptions

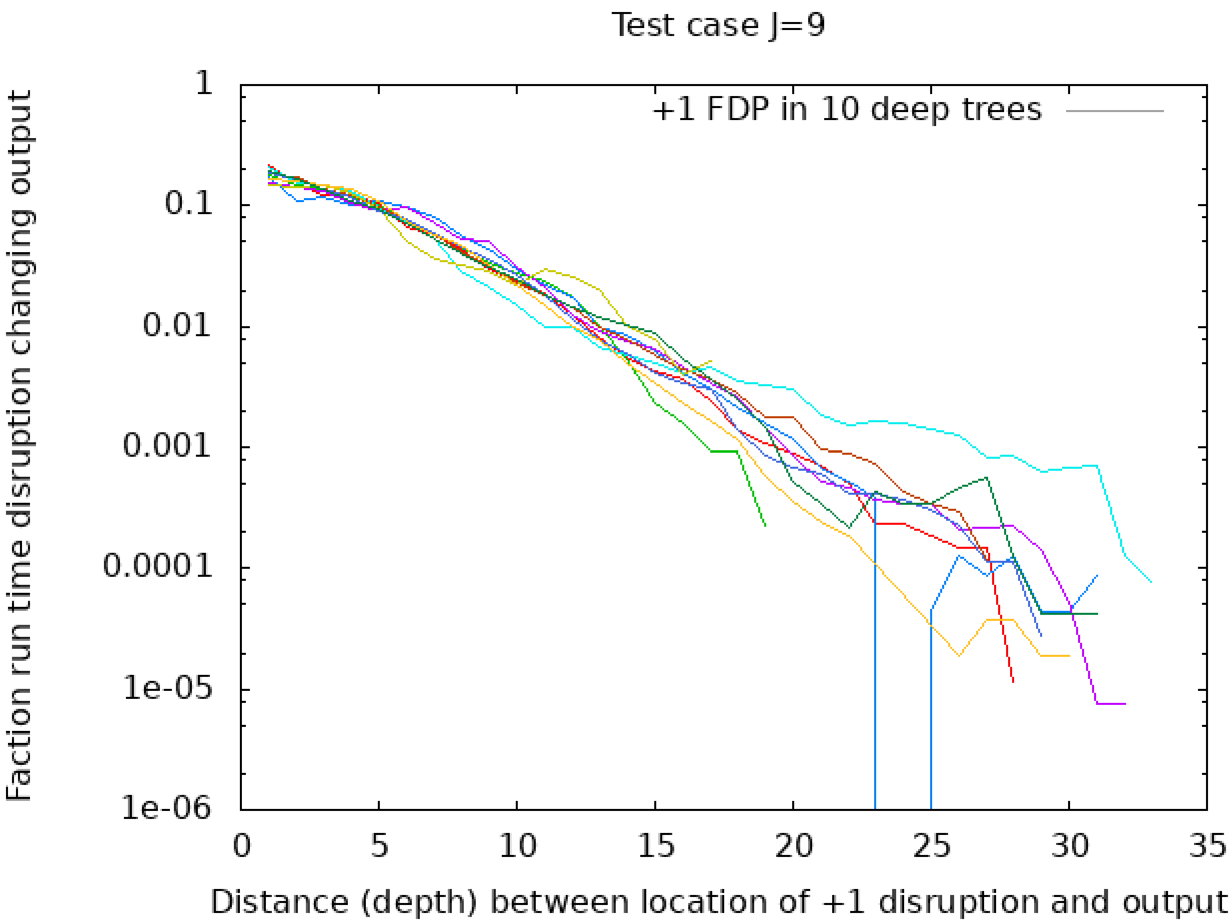
Run 10, tree depth 360

Red 10-20 test cases, no blue

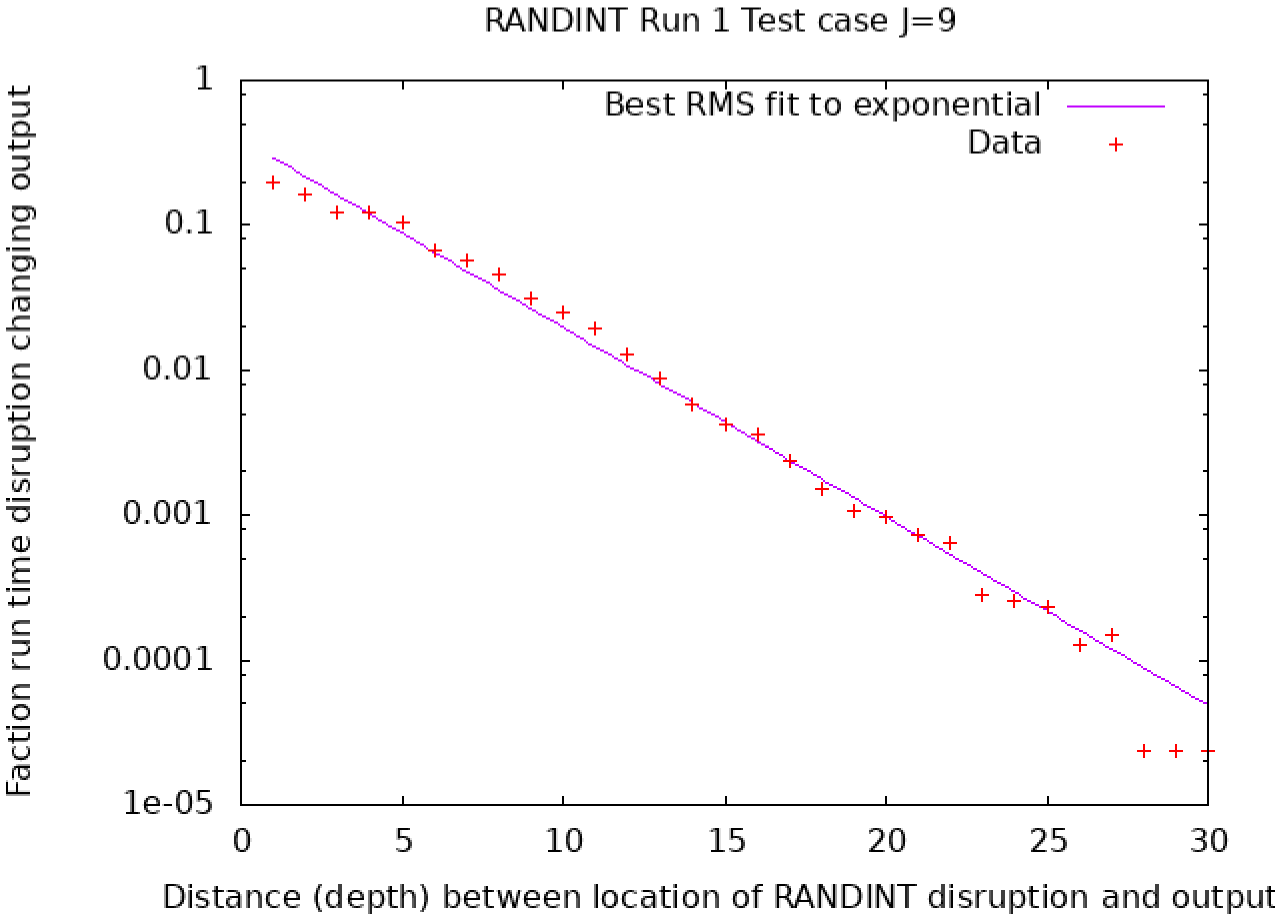


Same tree, +1 left, RANDINT right.
Almost identical response to different disruptions

Exponential fall in fraction of run time disruption changing program output with depth



Exponential fall in fraction of run time disruption changing program output with depth



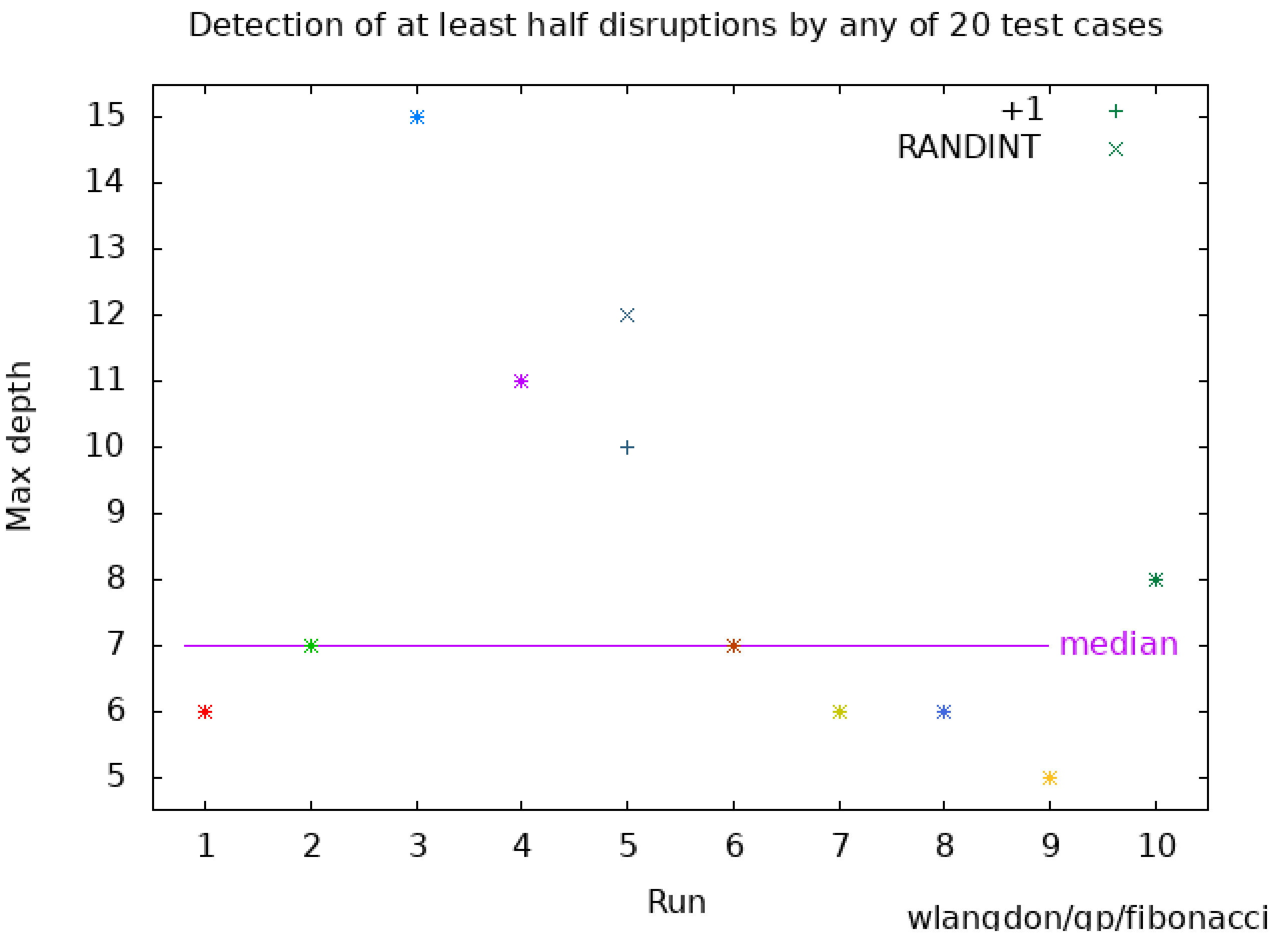
Fraction disruption reaching output in deep Fibonacci trees

depth	sum |error|	+1		RANDINT	
663	20	0.114%	-0.31	0.092%	-0.31
160	10	1.449%	-0.30	1.449%	-0.33
220	184	3.010%	-0.27	3.053%	-0.27
449	130	0.127%	-0.28	0.121%	-0.29
454	632	0.253%	-0.20	0.256%	-0.20
626	0	0.056%	-0.27	0.056%	-0.27
33	0	7.523%	-0.21	7.523%	-0.22
425	0	0.073%	-0.30	0.073%	-0.30
485	0	0.032%	-0.33	0.032%	-0.33
360	0	0.137%	-0.26	0.137%	-0.26

Variation between trees but smallest +1 and large RANDINT %disruption and exponential regression (-0.33 to -0.20) are both similar

Effectiveness of whole test suite varies with depth

50% chance of detecting disruption depth 5 to 15

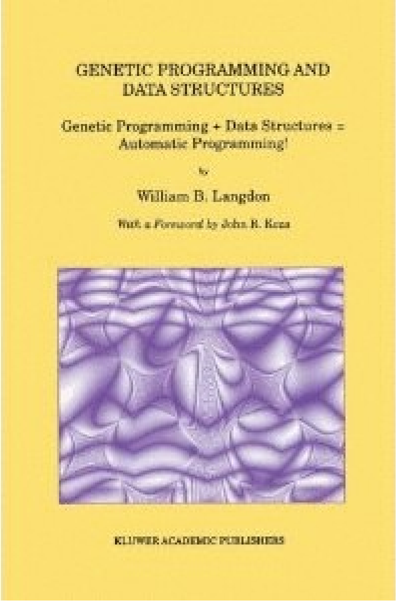


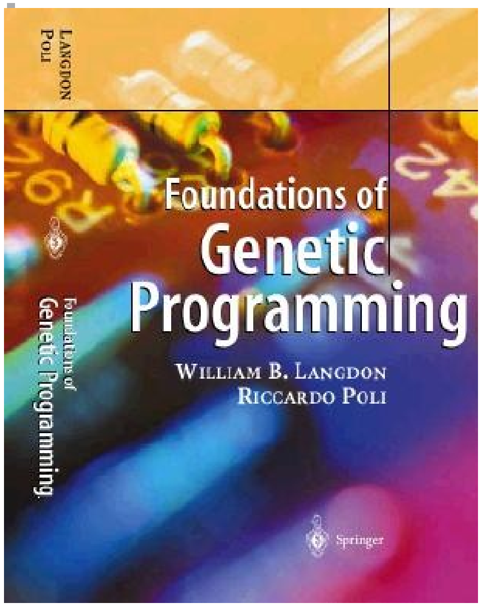
Conclusion: Deep nesting hides errors

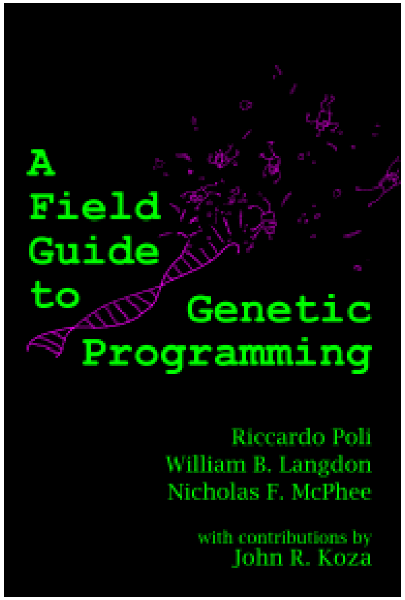
1) More fitness test cases has only small effect, <= log(n)
    - 1000 test cases only marginally more effective than 48
    - Test value 0.0f can be least effective
2) Testing is hard. Need to place test probe near error
    - Problem dependent but next to 18 – 28% reduction
    - Try to minimise *depth* of software being tested.
    - Problem dependent but here on average 7 levels
3) Write testable code: ie write units which are <7 levels deep
4) Programs are not chaotic, tiny errors often have no effect. Instead programs are robust because most (large or small) errors fail to propagate.

Genetic Programming

W. B. Langdon







The Genetic Programming Bibliography

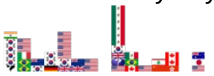
14736 references, [13000 authors]

Make sure it has all of your papers!

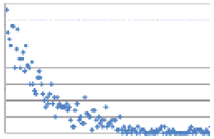
E.g. email W.Langdon@cs.ucl.ac.uk or use | [Add to It] | web link

Co-authorship community.
Downloads

Downloads by day



A personalised list of every author's GP publications.

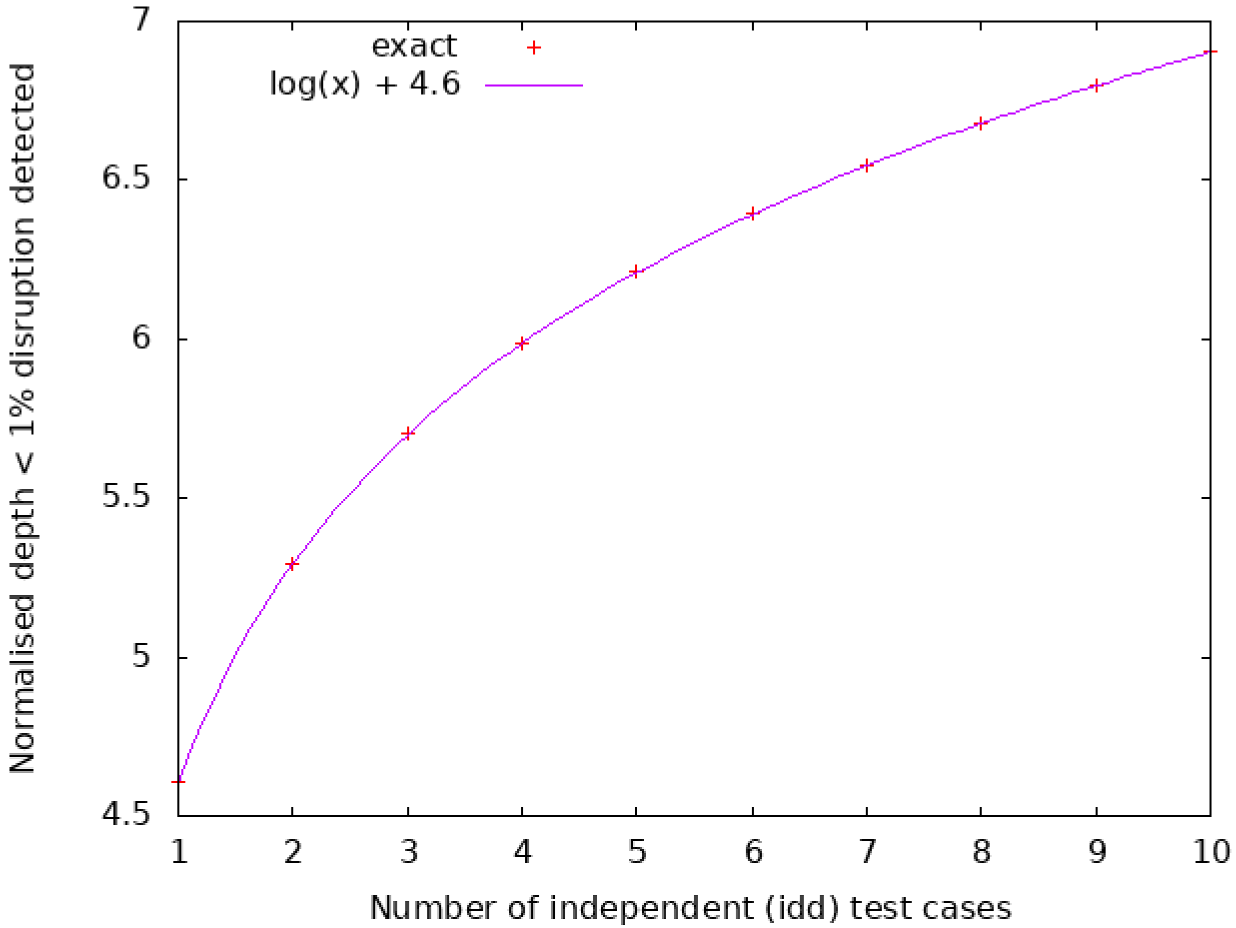


Your papers

[blog]

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

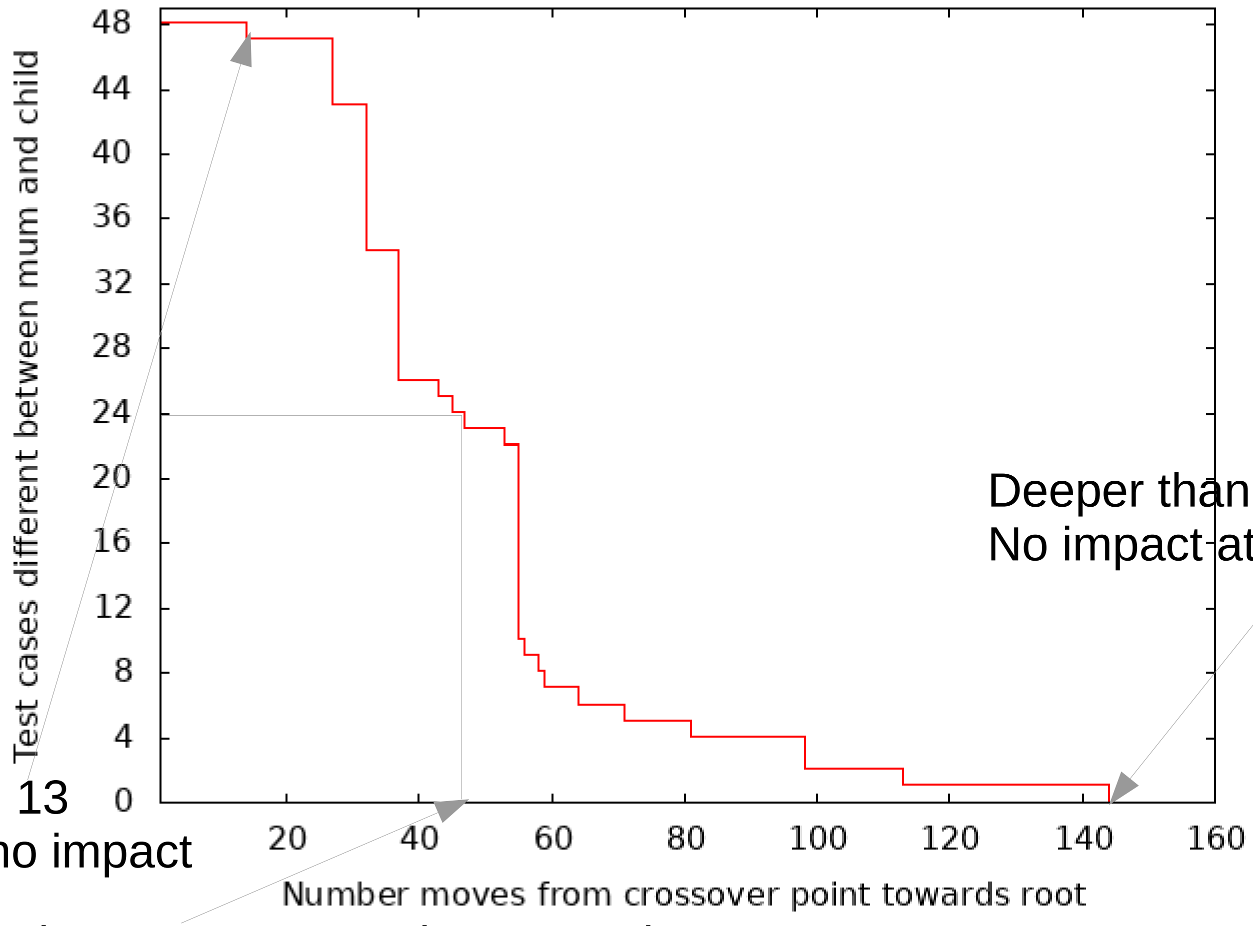
Best independent tests but test suite effectiveness only log(n)



Number of functions disruption must pass through before reaching the root node before the chance of detection is less than 1% v. test suite size. (Vertical axis *normalised* by dividing by mean of geometric distribution.)

Side Effect Free: Disruption Falls Monotonically

Deeper disruption tends to have less impact on fitness



Deeper than 144
No impact at all

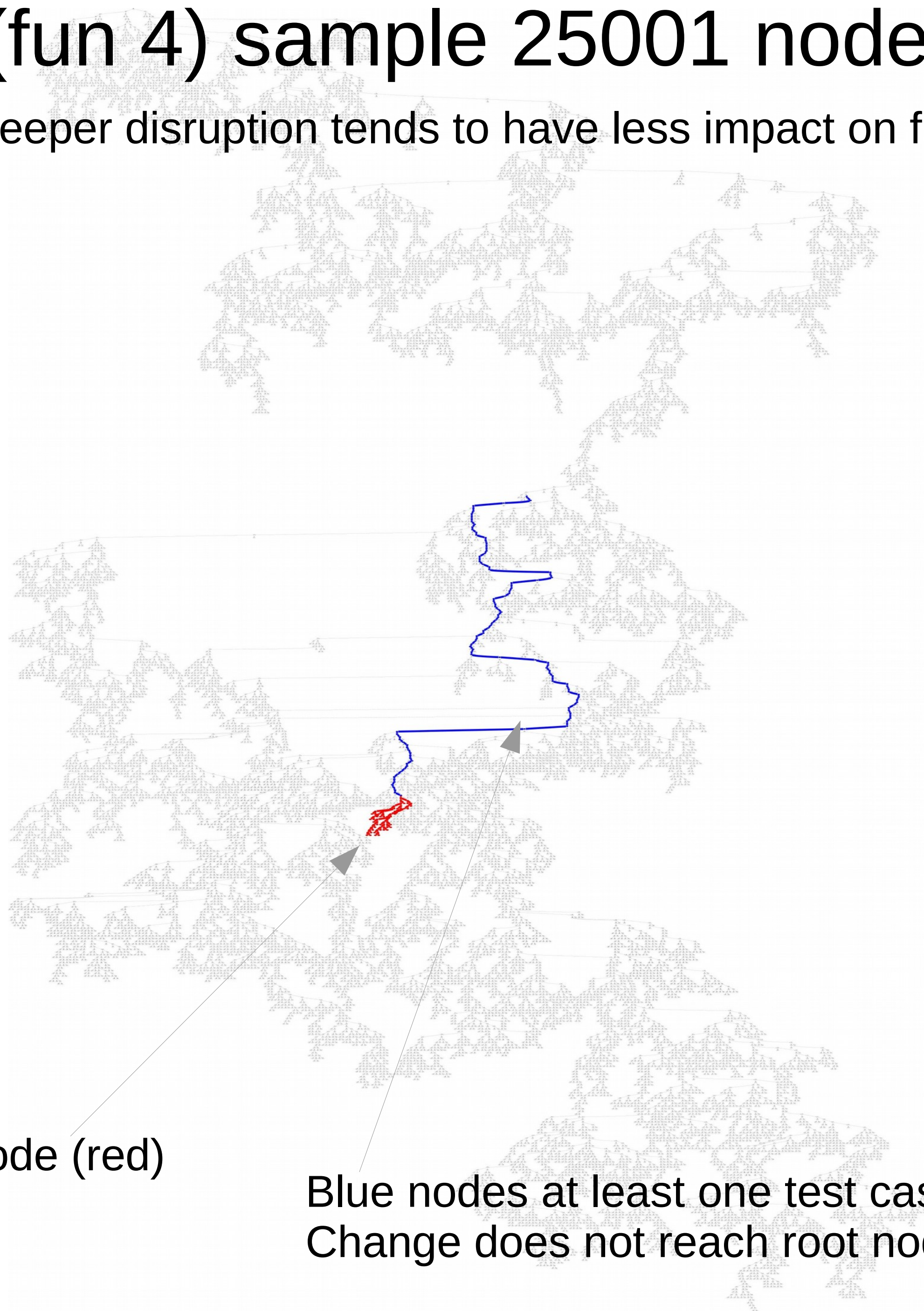
Deeper than 13
3 tests see no impact

Deeper than 44 ½ tests observe no impact

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

Random (fun 4) sample 25001 nodes depth 491

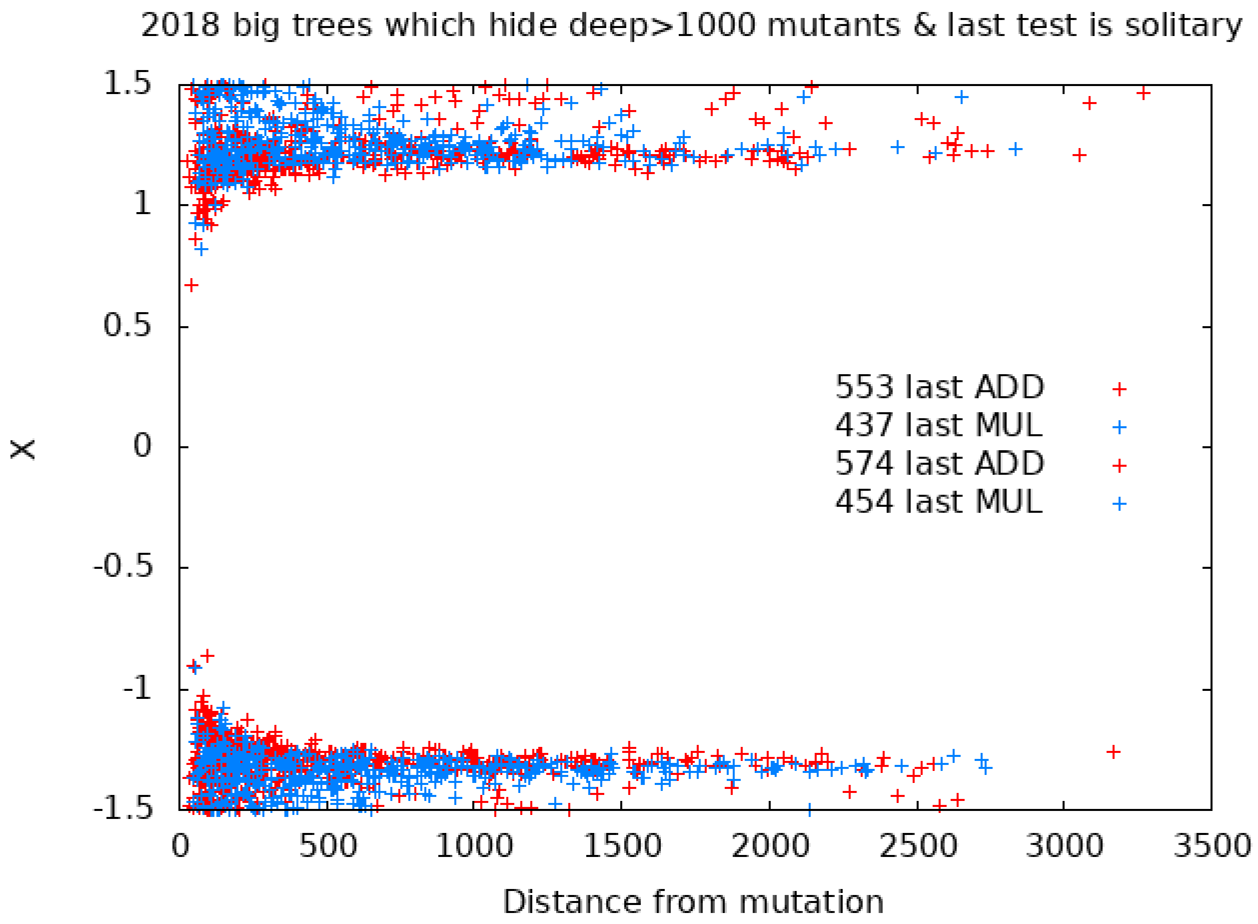
Deeper disruption tends to have less impact on fitness



Changed code (red)

Blue nodes at least one test case is different.
Change does not reach root node.

Most Difficult to Conceal Polynomial test case



For large random polynomials, most effective test cases |X| ~ 1.3