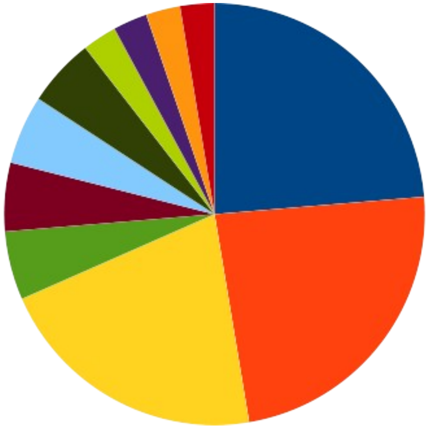
Software Evolutionary Robustness

UCL, Software Systems Engineering Seminar, 13 Nov 2024 13:00 Roberts 3.09

W. B. Langdon, UCL



38% GP papers on Applications

“When I was your age I could think of six impossible things before breakfast.”

Do the impossible

No chaos

Information theory says deep nested crossovers and mutations do not change fitness

Be shallow
Be close to fitness environment

14.11.2024

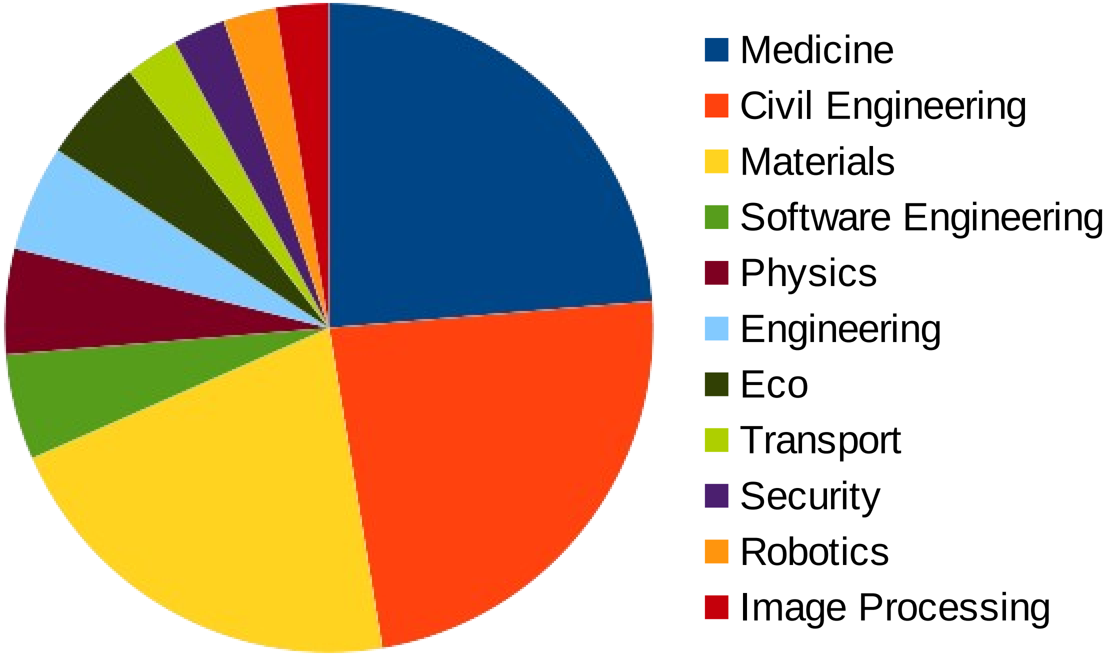
“Sustaining Evolution for Shallow Embodied Intelligence”
W.B. Langdon and Daniel Hulme

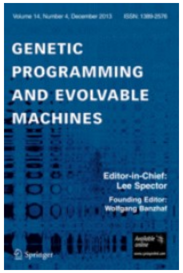
Software Evolutionary Robustness

- Applications of Genetic Programming, 38% of GP papers
- Creativeness of Evolutionary Computing [CACM June 2024]
- Run Genetic Programming to a million generations
  - A trillion GP operations per second
  - Exploits lost of disruption in deep trees (10000 levels, 1e9 nodes)
- Information Theory explains Failure of Disruption to Propagate
- FDP so crossover/mutation little impact in deeply nested trees
- Evidence that deep mutations have little impact in C/C++
- Implications of too much robustness
  - Sustained long term evolution needs limited depth of nesting:
    - Open architecture with many small shallow programs
  - C++ deeper code harder to test, improve, optimise, repair

Applications of Genetic Programming

- Industrial use not published but:
- Applications 38% of 2023 GP bibliography papers



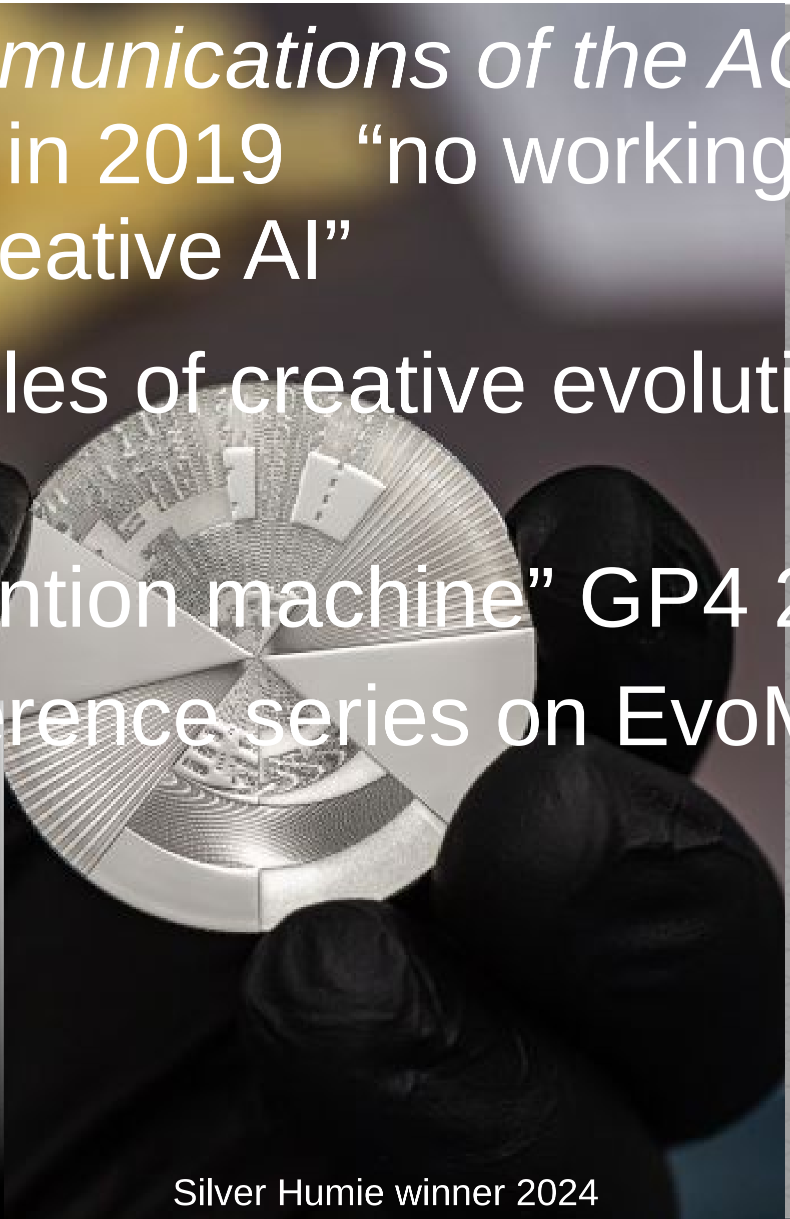

Volume 14, Number 4, December 2013 ISSN: 1389-2576
GENETIC PROGRAMMING AND EVOLVABLE MACHINES
Editor-in-Chief: Lee Spector
Founding Editor: Wolfgang Banzhaf
Springer

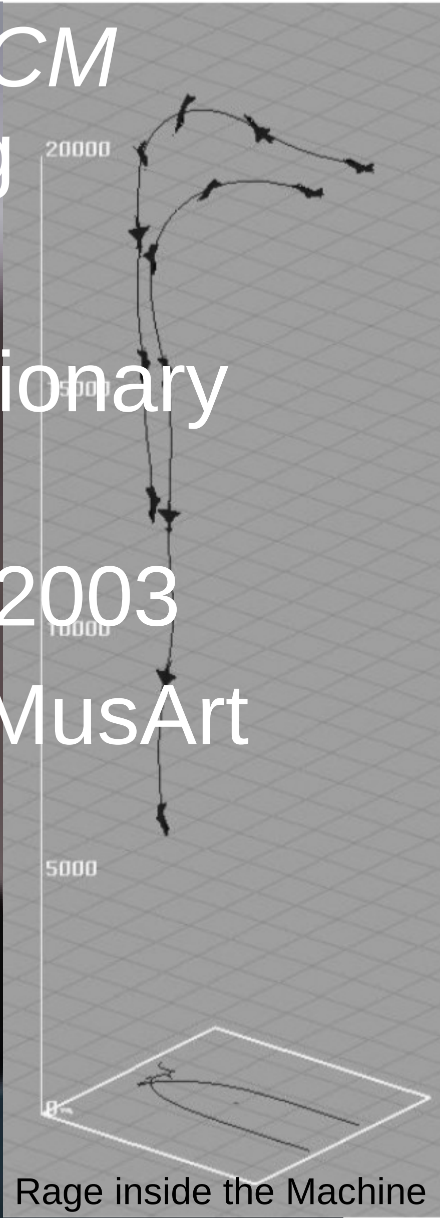
“Jaws 30”
Celebration of 30th anniversary Koza’s GP book
Genetic Programming and Evolvable Machines

Creativeness of Evolutionary Computing

- Last year *Communications of the ACM* included claim in 2019 “no working examples of creative AI”
- 100s of examples of creative evolutionary computing:
  - Koza’s “invention machine” GP4 2003
  - Whole conference series on EvoMusArt

Uusitalo, GPEM 2024

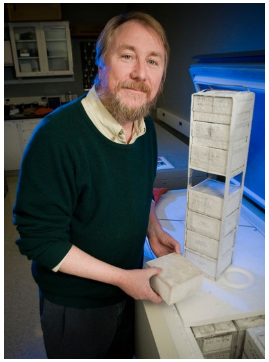

Silver Humie winner 2024



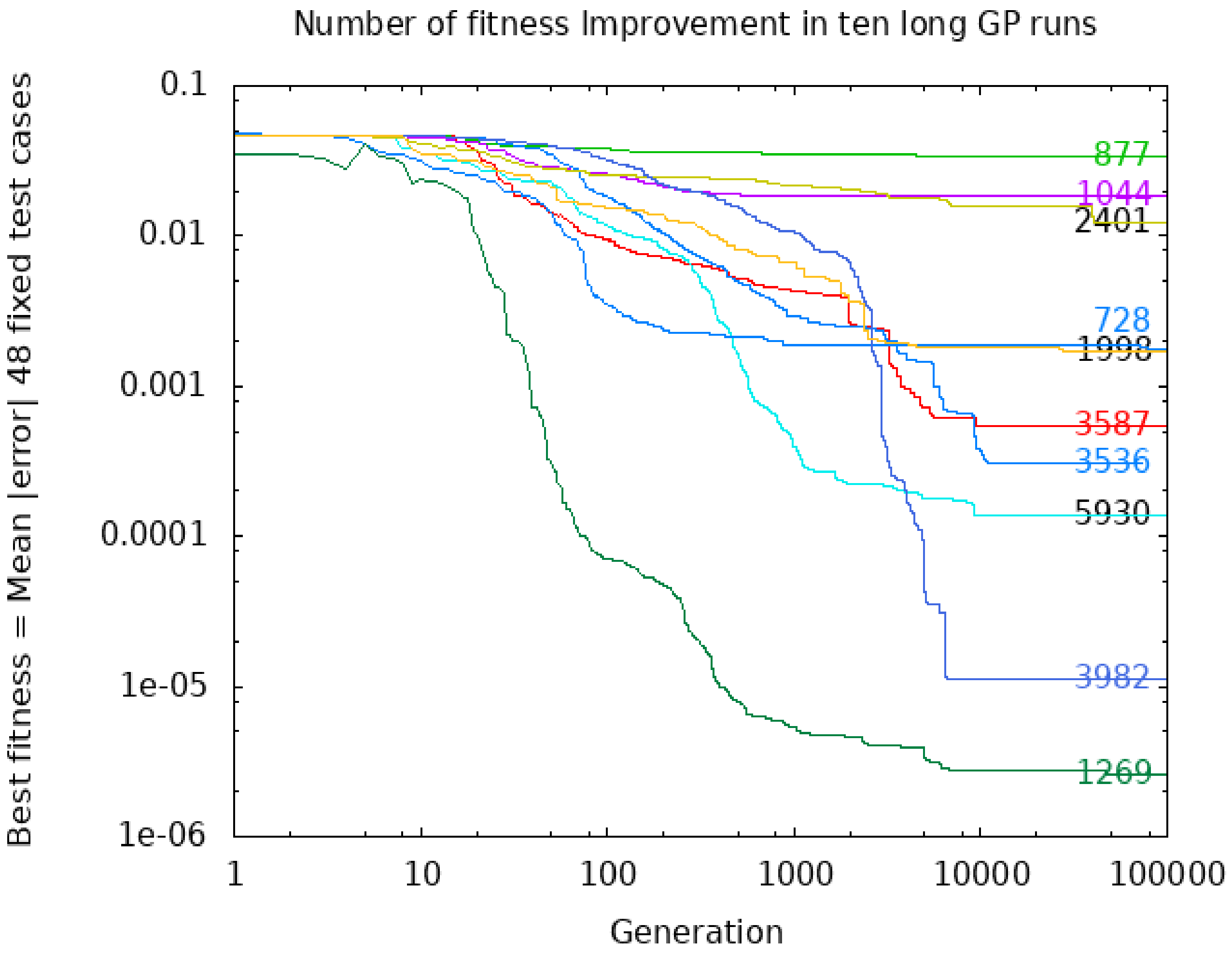
Rage inside the Machine

Long-Term Evolution Experiment with Genetic Programming

- Rich Lenski's Long Term Evolution Experiment 80000 generations of bacteria E.coli 4.6Mb shows continued evolution. (Homo Sapiens about 9300 generations old)
  - Experiment running since 1988 (36 years), MSU
- What happens in Genetic Programming?
  - Run up to 1 million generations 2 billion nodes
  - Run experiments in days/weeks



Fitness improvement continues but slows

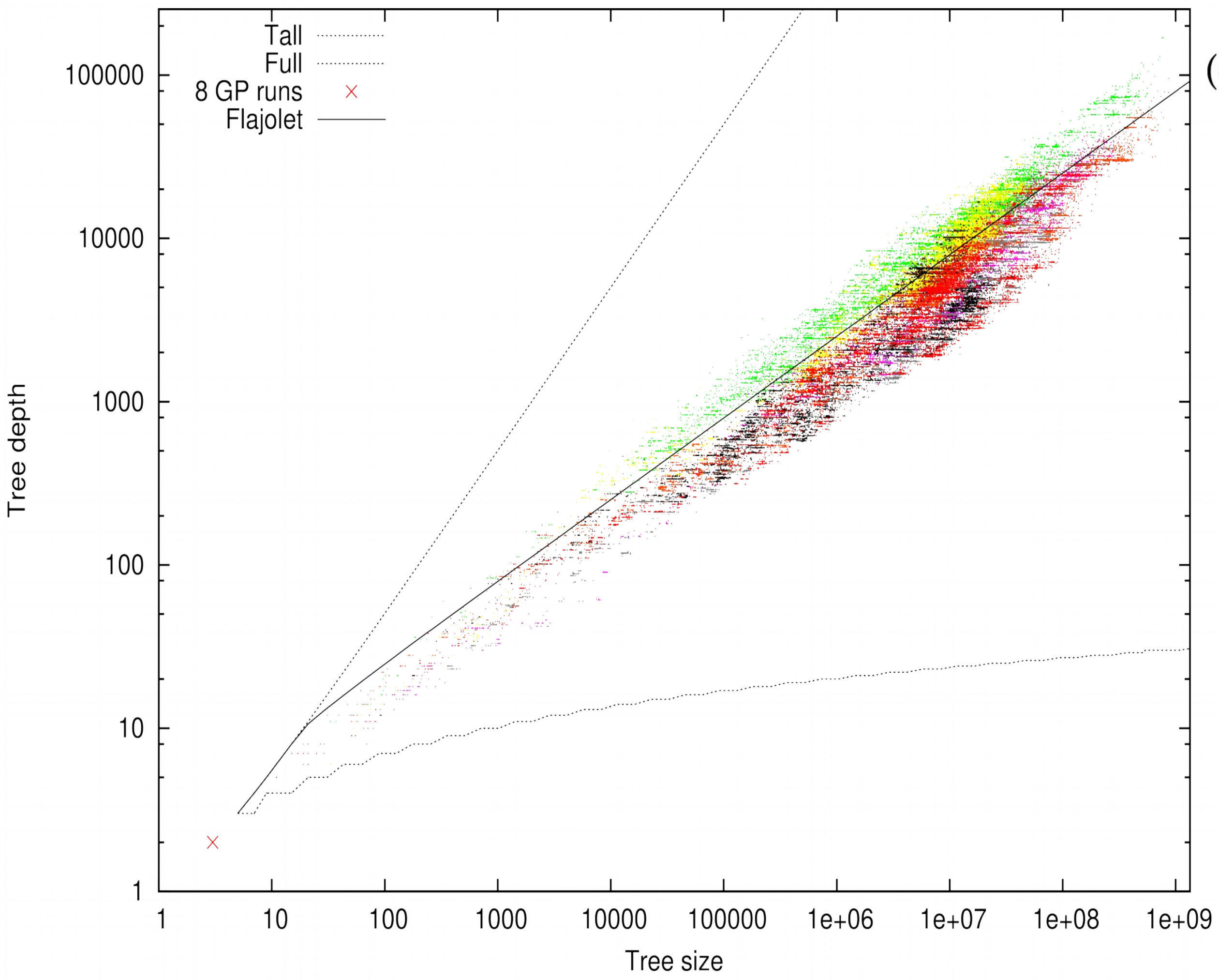


Evolution of mean absolute error in ten runs of Sextic polynomial with population of 500. Runs to 100,000 generations.

Thousands of fitness improvements found.

Note log scales.

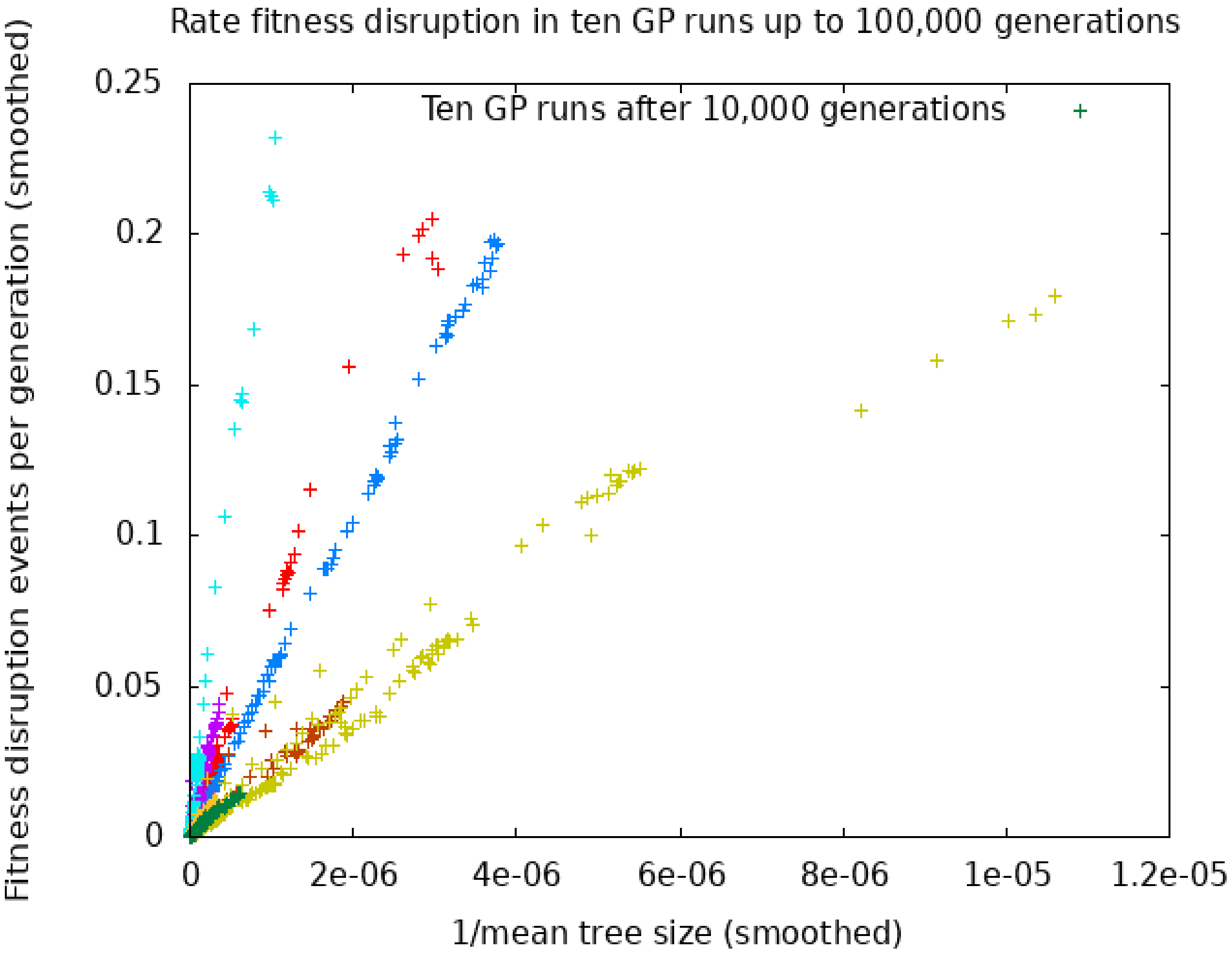
GP trees converge centre of search space



$(\text{depth} \approx \sqrt{2\pi|\text{size}|}$

Size and depth of the best individual in each of 100,000 generations for eight Sextic polynomial runs with population of 500.

Crossover fitness disruption 1/size

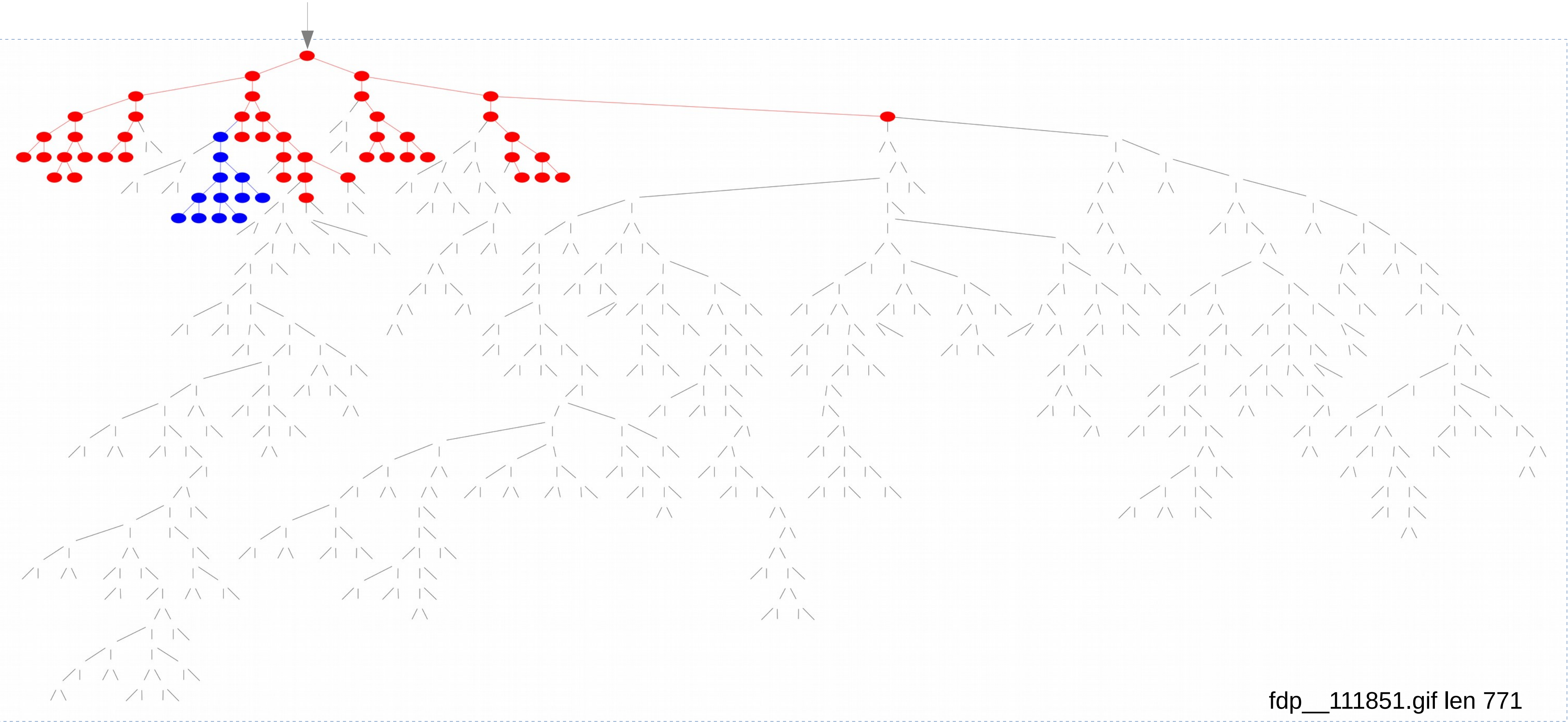


When programs are large chance crossover hitting sensitive area near output falls in proportion to size. (Evolution gives different ratios between runs.) Smoothed by taking running averages over a 100 generations.

+1 Disruption, Fibonacci run 7, depth 33

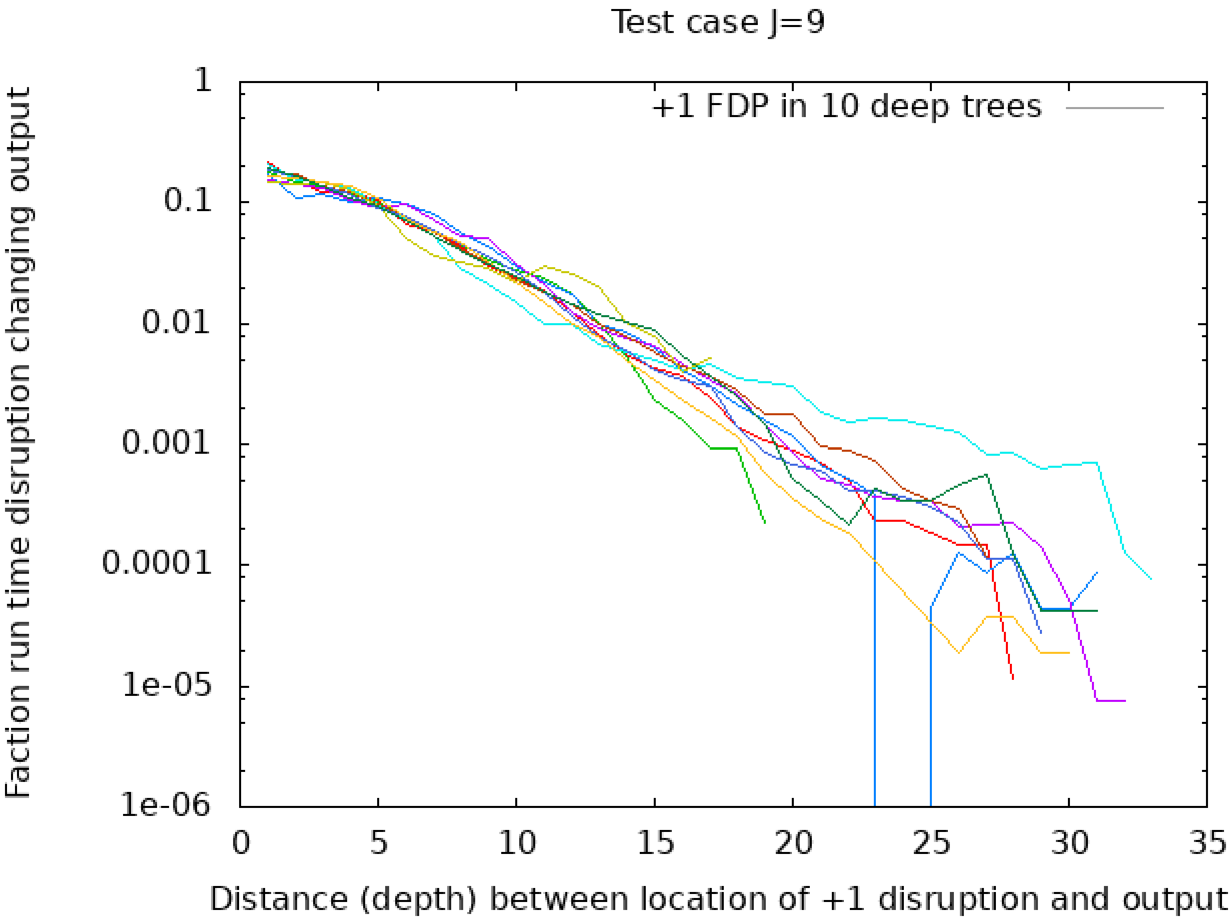
red 16-20 test cases, blue 1 test cases

Output (root node)



Only disruption near root node reaches output

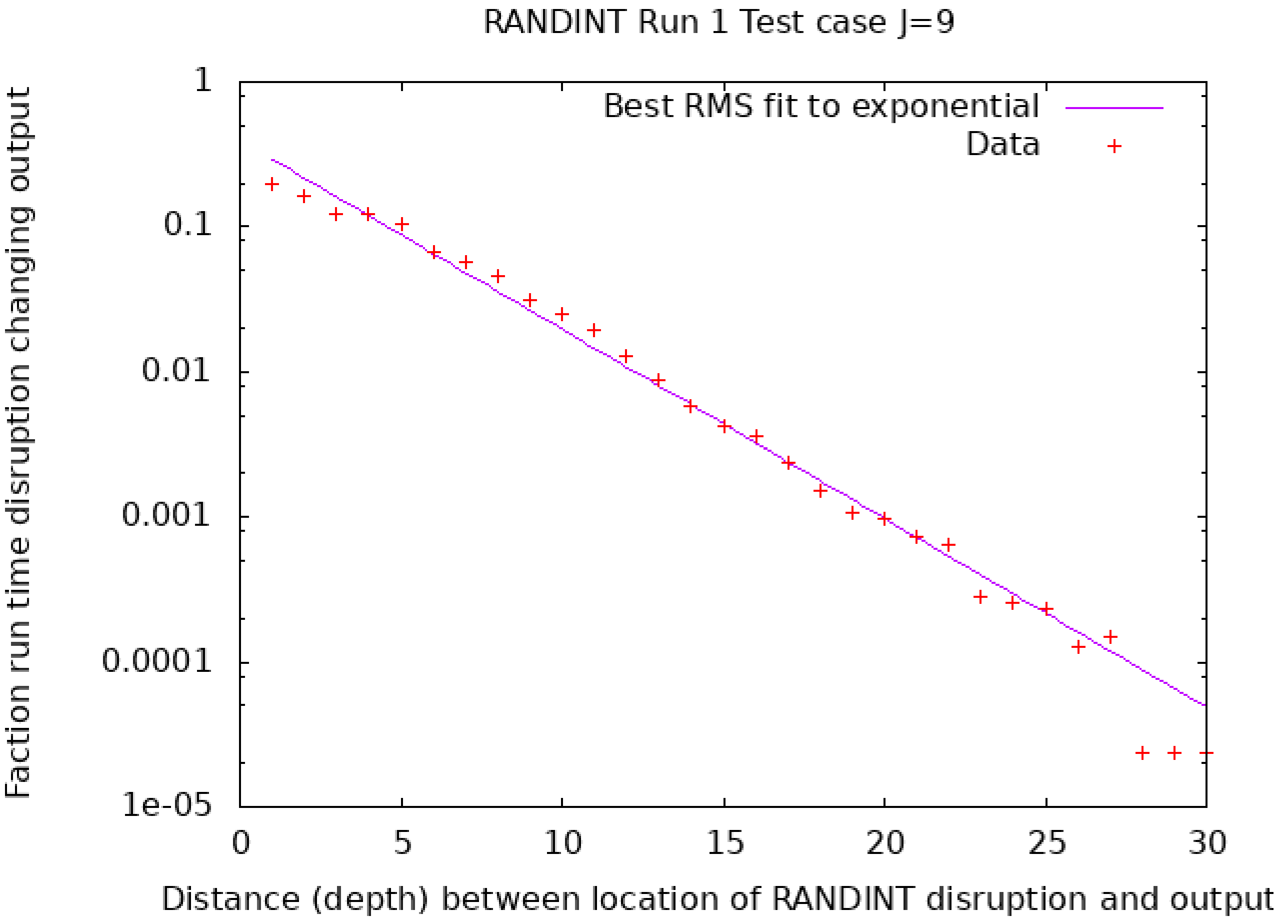
Exponential fall in fraction of run time disruption changing program output with depth



Slope: disruption falls by
≈ ½ per 2.0 levels

fdp.gif

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

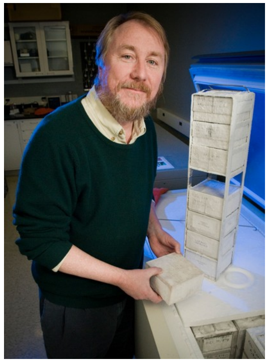


Slope: disruption falls by ½ per 2.3 levels

Deep floating point GP trees similar

params.gnu

Long-Term Evolution Experiment with Genetic Programming



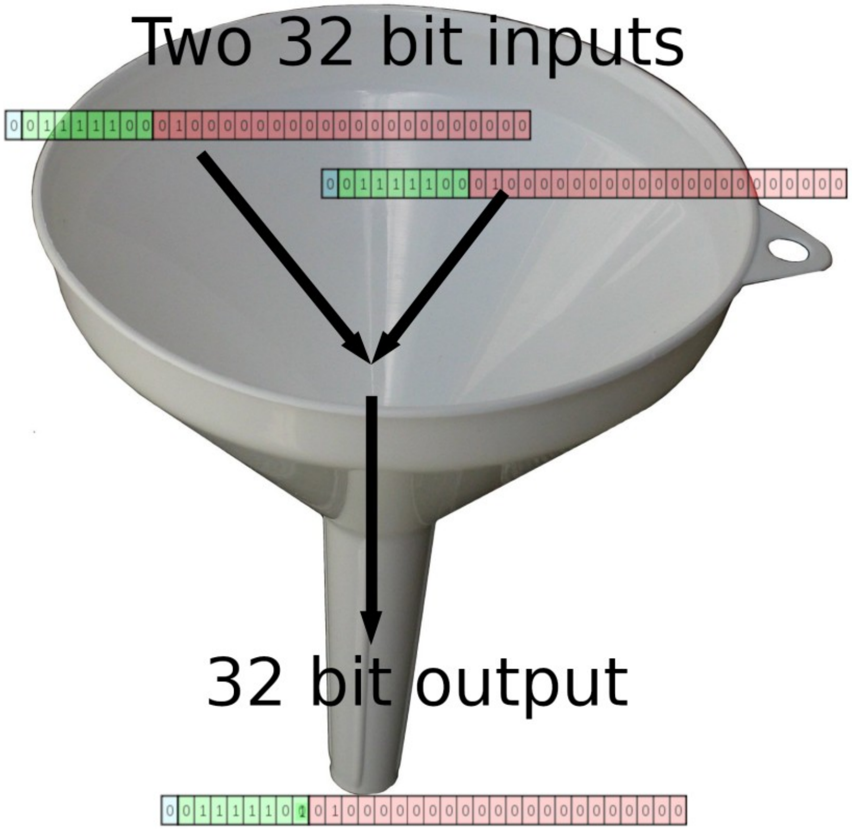
- Rich Lenski's Long Term Evolution Experiment 80000 generations of bacteria E.coli 4.6Mb shows continued evolution. (Homo Sapiens about 9300 generations old)
- What happens in Genetic Programming?
  - Run up to 1 million generations 2 billion nodes
- In GP fitness improvement continues but slows
- Information theory explains crossover Disruption Fails to Propagate (FDP) to output, so fitness is unchanged.
- Only crossover or mutation near output impacts fitness. Rate of fitness improvement O(1/size)
- True in any hierarchical system, shows up in C/C++
- Need evolvable code close to fitness environment

Deep Mutations have Little Impact

- PIE (Propagation Infection Execution) view of software bugs
  - **E** bug needs to be executed
  - **I** bug needs to change (disrupt) the program’s state (infection)
  - **P** the disruption needs to propagate to the program’s outputs
- If entropy loss causes Failure of the Disruption to Propagate FDP the error is invisible and the software is robust to it
- Information Theory says impact of disruptions lost with distance when nested
- Just seen deep Genetic Programming trees are robust
- PARSEC, VIPS, vipsthumbnail benchmark deeper C code more robust to mutations (also gem5 C++)

Information Funnel

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

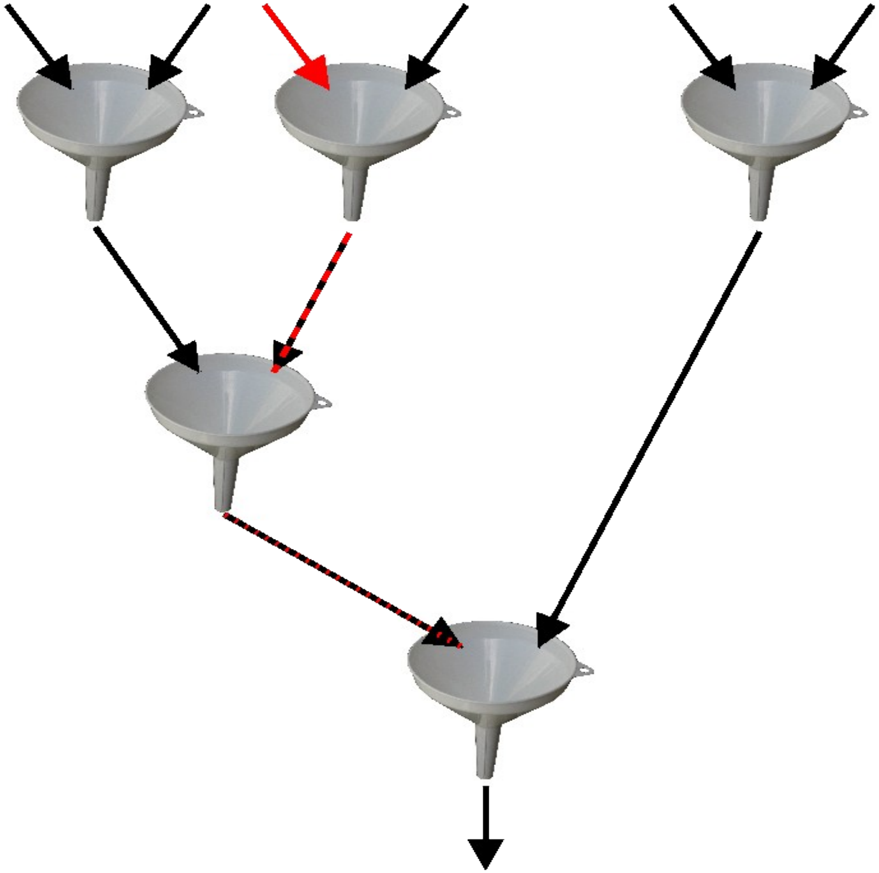


Information funnel

More information enters than leaves

Information flow in five nested functions

Potential information loss at each (irreversible) function



Output

Disruption may fail to reach reach output.

(No side effects.)

Using Magpie to Sample C Mutations

- Genetic Improvement tool Magpie https://github.com/bloa/magpie
- PARSEC suite of benchmarks to test parallel super computers for NASA. Mostly numeric but includes some image processing, including VIPS
- VIPS C image processing library 90,000 lines of code (LOC)
- Chose vipsthumbnail, parallel multi-threaded, takes large image creates small image 128 pixels wide
- Use linux perf to profile vipsthumbnail select all VIPS functions perf reports
- Use debugger to select all functions called enroute to top CPU using function
- Remove unused functions (a few unused lines, eg if/switch case included)
- 90,000 => 7328 lines, in 37 C files. srcml => 37 XML files
- 1000 random Magpie mutations, measure their impact, measure execution depth

Magpie Mutating C

- Genetic Improvement Magpie https://github.com/bloa/magpie
- VIPS image thumbnail benchmark (use 37 files 7328 LOC)

3264 x 2448

128 x 96
thumbnail

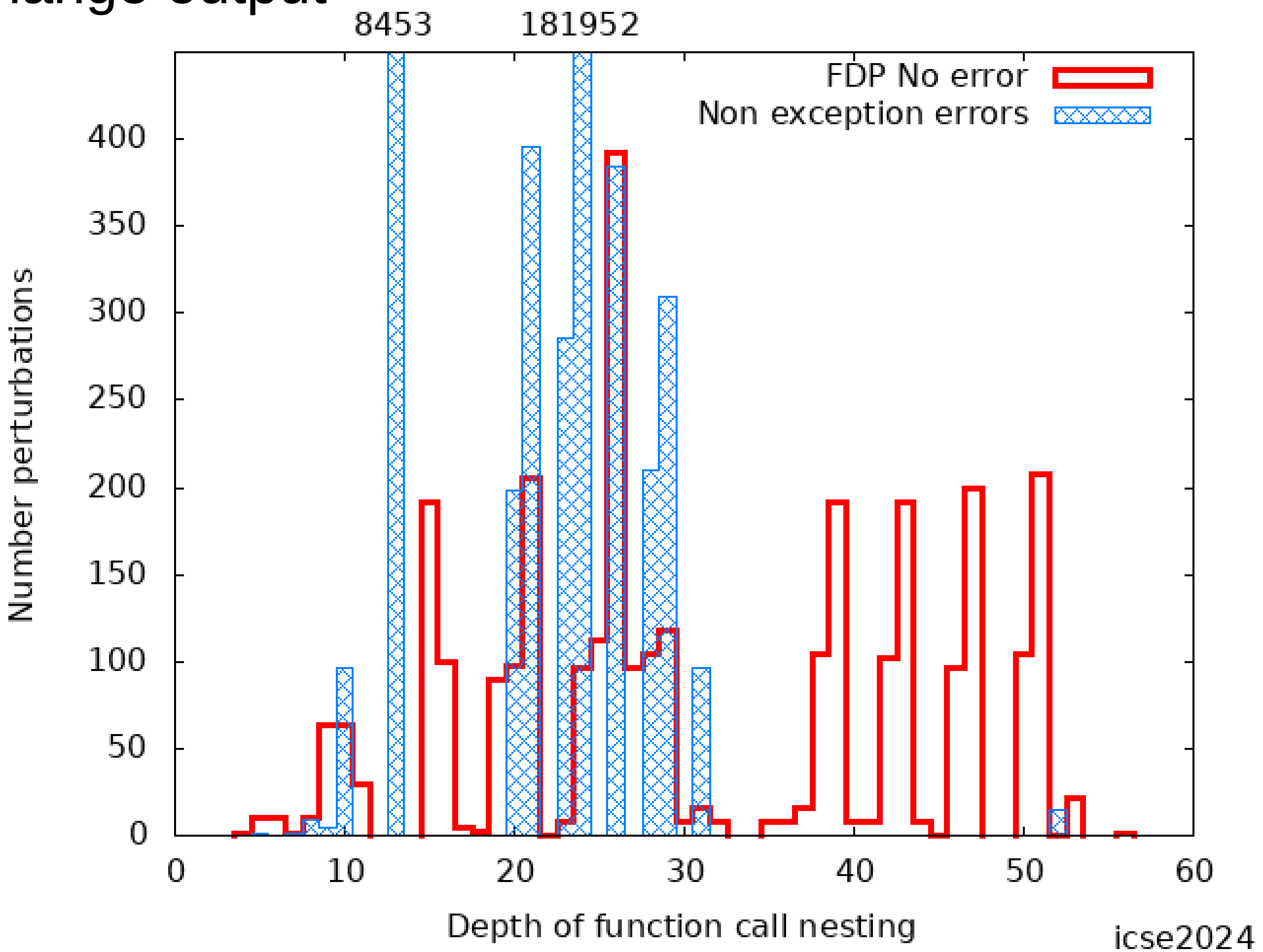
1000 random Magpie VIPS mutants

- VIPS image thumbnail benchmark (use 37 files 7328 LOC)
  - try to exclude unused code
- Magpie mutating source code as XML, mostly syntax preserving, mostly compiles, runs, 526 give right answer
- 37 cases output wrong but no exception.
  - Randomly choose 25 of 37, compare with 25 where mutant code is run, changes state but output is unchanged

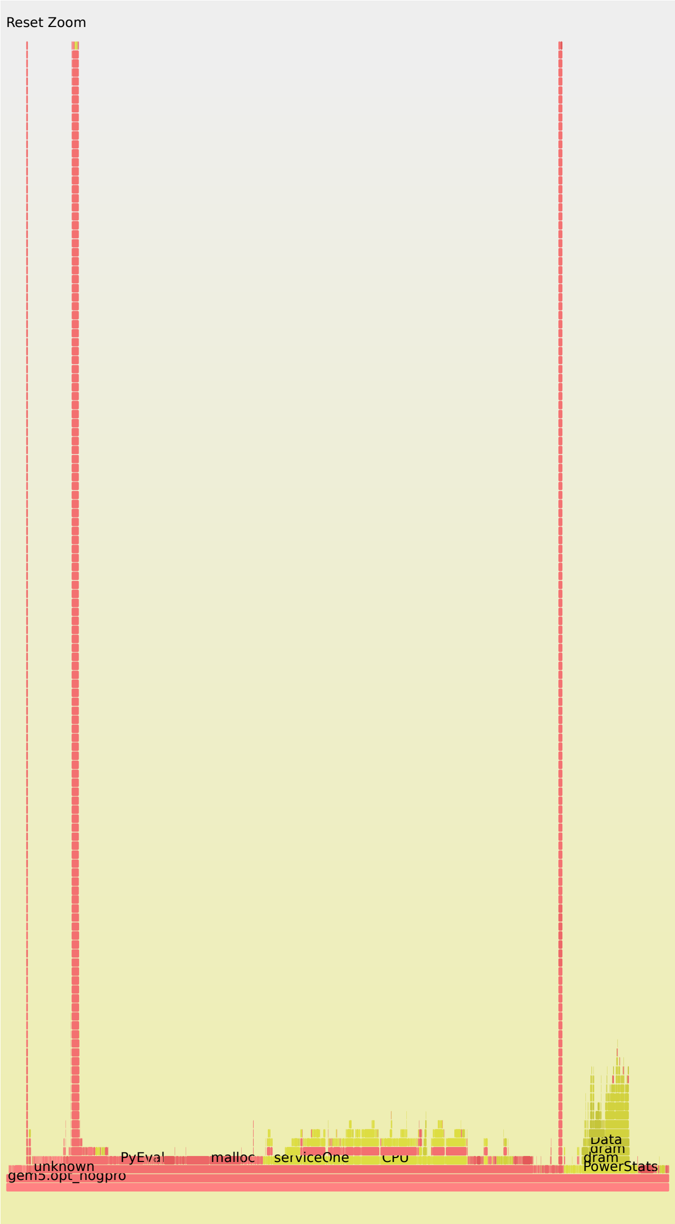
Compiled, ran correct output	526	Correct output	438
		Mutation is identical to original code	88
Failed to compile	302		
Failed to run correctly or gave incorrect output	164	exception	127
		output error	37
Magpie TypeError	8		

25 v 25 Mutants. Deep less impact

- 25 mutants change execution but no change to output
- 25 mutants which change execution (without causing segfault) but change output



gem5 depth of runtime nesting



SVG flamegraph
produced by Linux perf
https://github.com/wblangdon/
Deep-Imperative-Mutations-have-Less-Impact

In web browser cursor shows details of
performance of each C++ function

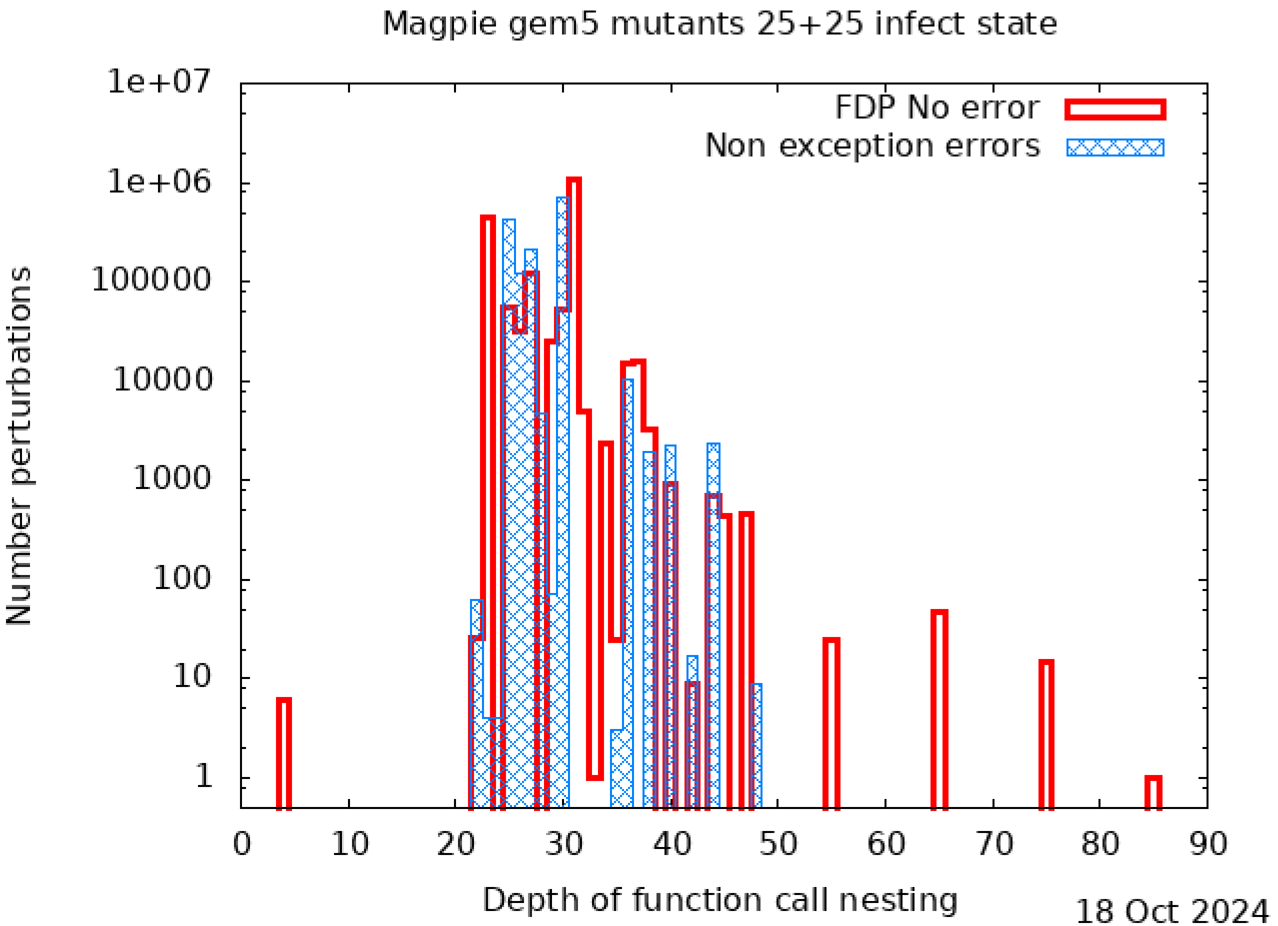
2500 random Magpie gem5 mutants

- gem5 X86 discrete time simulator 1 million lines C++
- Magpie mutating source code as XML, mostly syntax preserving, mostly compiles, runs, 397 give right answer
  - Ignore 1730 mutants to unused code
- 55 cases output wrong but no exception.
  - Randomly choose 25 of 55, compare with 25 where mutant code is run, changes state but output is unchanged

Compiled, ran correct output	397	Correct output	142
		Mutation is identical to original code	255
Failed to compile	228		
Failed to run correctly or gave incorrect output	145	exception	90
		output error	55

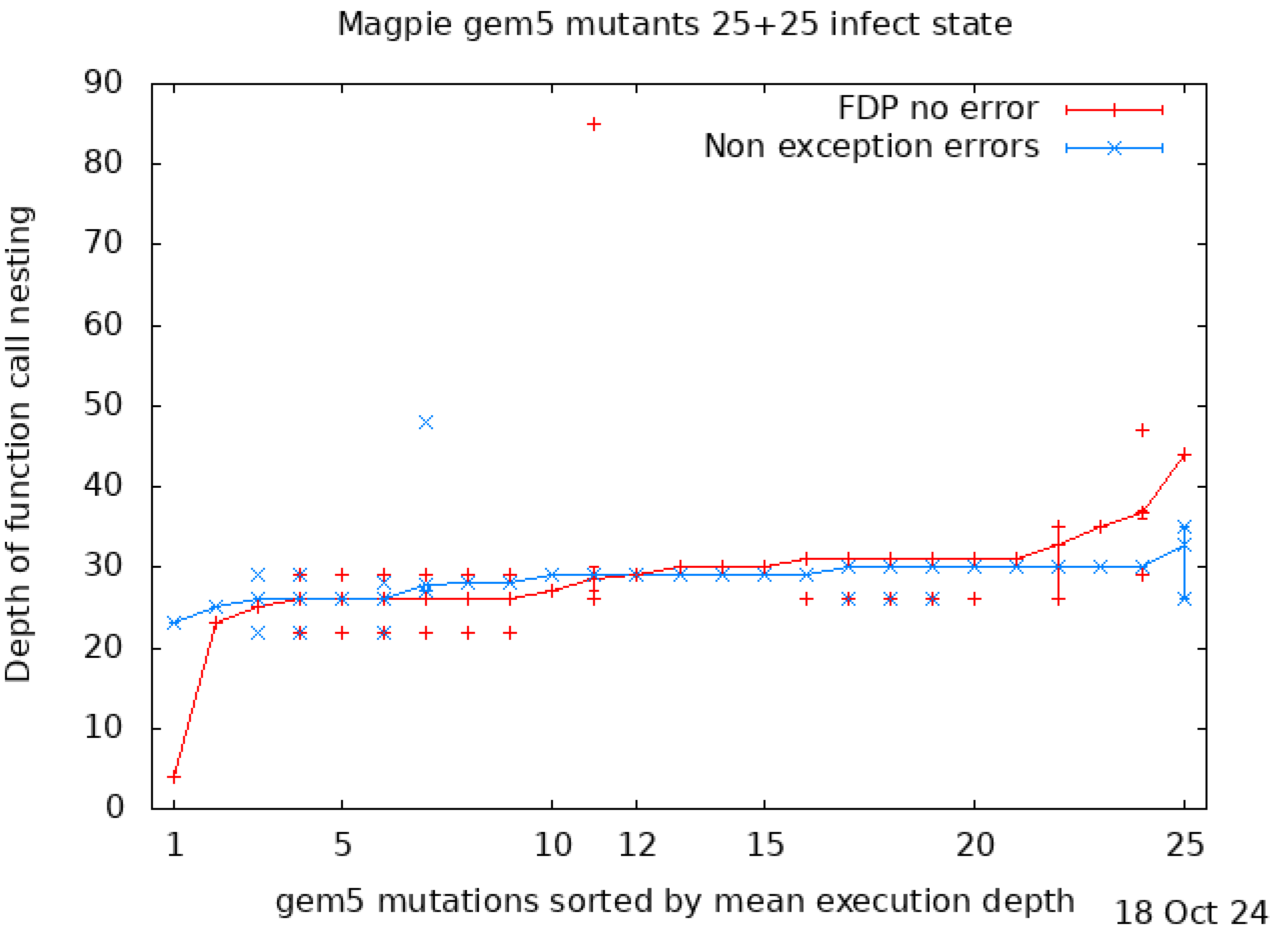
25 v 25 gem5 Mutants. Deep less impact

- 25 mutants change execution but no change to output
- 25 mutants which change execution (without causing segfault) but change output



25 v 25 Mutants. gem5

- 25 mutants change execution but no change to output
- 25 mutants which change execution (without causing segfault) but change output

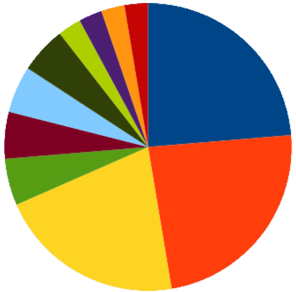


18 Oct 24

Conclusions: Software is Robust

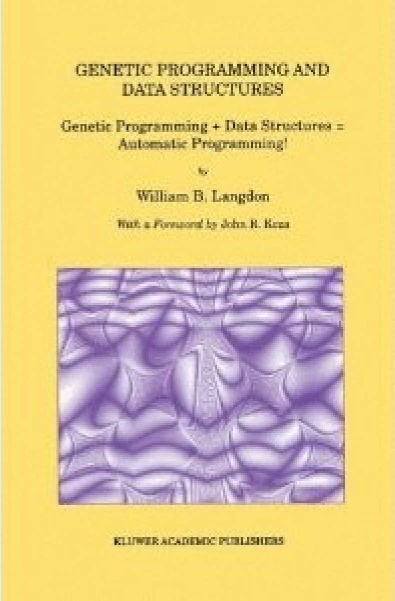
- Importance of reminding everyone of GP achievements
- Genetic programming can be creative, GP is in use
- Information theory predicts failed disruption propagation FDP, which makes genetic programming and software robust
- In GP exponential decay with depth, impact of mutations lost
  - Need shallow code for prolonged evolution
- Excluding segfault etc., most C/C++ mutations nested >30 function calls deep did not change output
- Be ambitious

Genetic programming as an invention machine



Mangroves in Bali

Do the impossible



GENETIC PROGRAMMING AND DATA STRUCTURES

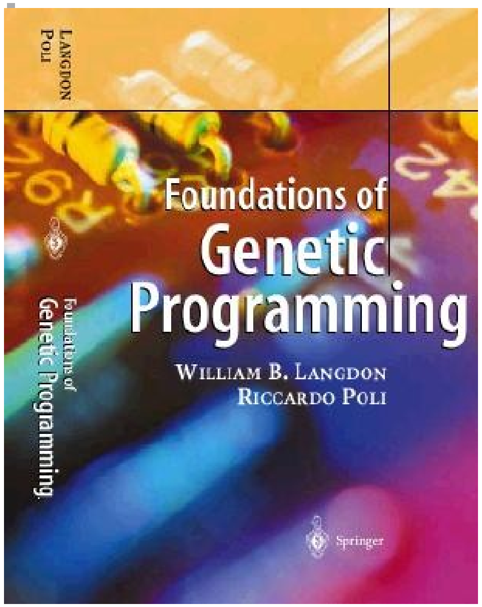
Genetic Programming + Data Structures = Automatic Programming!

by

William B. Langdon

With a Foreword by John R. Koza

KLUWER ACADEMIC PUBLISHERS



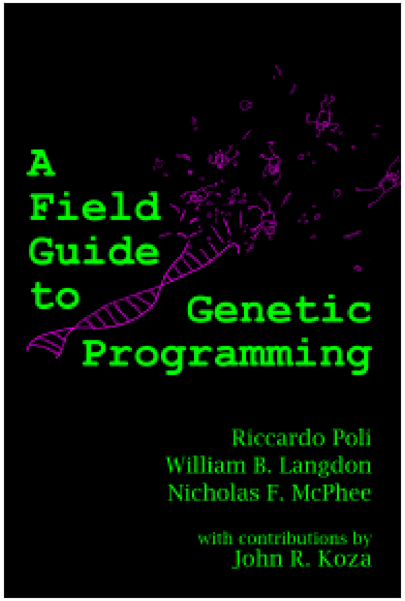
LANGDON
POLI

Foundations of Genetic Programming

Foundations of
Genetic Programming

WILLIAM B. LANGDON
RICCARDO POLI

Springer



A Field Guide to Genetic Programming

Riccardo Poli
William B. Langdon
Nicholas F. McPhee

with contributions by
John R. Koza

How Deep is Deep?

- depends…
  - Internal program data connectivity ?
  - Information lost by functions (Boolean >> int > float) ?
- In VIPS thumbnail and gem5 deep ~30 levels of function nesting
  - Mostly functions are small with calls to other functions often at function level or one level in (eg inside if).
- In integer Genetic Programming Fibonacci, the fraction of large effective mutants falls by 50% for each additional ~2.2 levels
- In floating point Genetic Programming, 50% fall for ~7 levels?
- “Normal” Multi Layer Perceptron has 3 or 4 levels,
  Wikipedia says “deep” means more than 4 levels

References

1) Sustaining Evolution for Shallow Embodied Intelligence, W.B. Langdon and Daniel Hulme, EI 2023

2) Evolving Open Complexity, W.B. Langdon, EI 2022

3) Long-Term Evolution Experiment with Genetic Programming, W.B. Langdon and W.Banzhaf, Artificial Life, 2022 28(2) pp173-204.

4) Dissipative Arithmetic, W.B. Langdon, Complex Systems. 2022, 31(3) 287-309

5) Deep Genetic Programming Trees are Robust, WB Langdon, ACM TELO, 2022 2(2) 6.

6) Genetic Programming Convergence, WB Langdon, GP+EM, 23(1) 71-104

7) Measuring Failed Disruption Propagation in Genetic Programming, GECCO 2022, 964-972, WB Langdon, *et al.*

8) Failed Disruption Propagation in Integer Genetic Programming, GECCO comp 2022, 574-577, WB Langdon, *et al.*

9) Information Loss Leads to Robustness, W.B. Langdon and J.Petke and D. Clark, IEEE Software Blog, 12 Sept. 2021.

10) Dissipative Polynomials, W.B. Langdon and J. Petke and D.Clark,in GECCO 2021 comp., pp1683-1691. DOI

11) Software Robustness: A Survey, a Theory, and Some Prospects, J.Petke, D.Clark and W.B. Langdon, in ESEC/FSE 2021 (IVR), pp 1475-1478, DOI

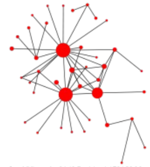
12) Long-Term Evolution of Genetic Programming Populations, W.B. Langdon. In GECCO 2017 Comp., 235-236. DOI

The Genetic Programming Bibliography

17436 references, 17000 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

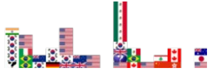
Co-authorship community.



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

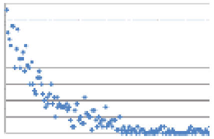
Downloads

Downloads by day



bibliography

A personalised list of every author's GP publications.



Your papers

blog

Googling GP bibliography, eg:
Graph-based Genetic Programming site:gpbib.cs.ucl.ac.uk

Find medicine ≤hits 10 medicine 10 of 28
○ brief ● terse ○ full
Before= After= 2014

Text search