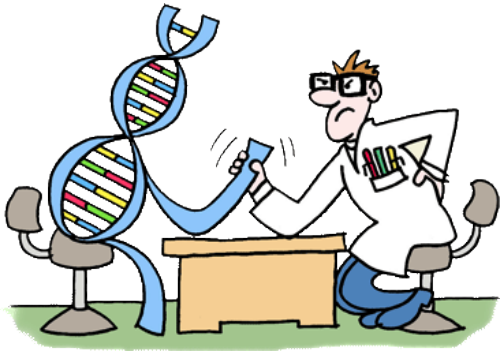
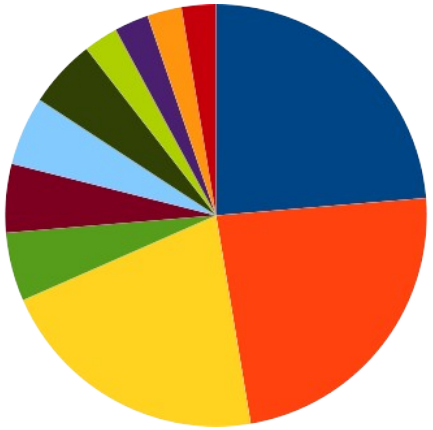
Information Theory and Robustness

Software Systems (SSY) Group Seminar, 4 March 2025, King's College London



$10000 "Humies"
Deadline 30 May



38% GP papers on Applications

"When I was your age I could think of six impossible things before breakfast."
GI@ICSE 2025 Sunday Apr 27

Software is not chaotic

Be shallow
Repair shallow code
Optimise shallow code

Information theory says
deep nested errors
tend not to impact outputs

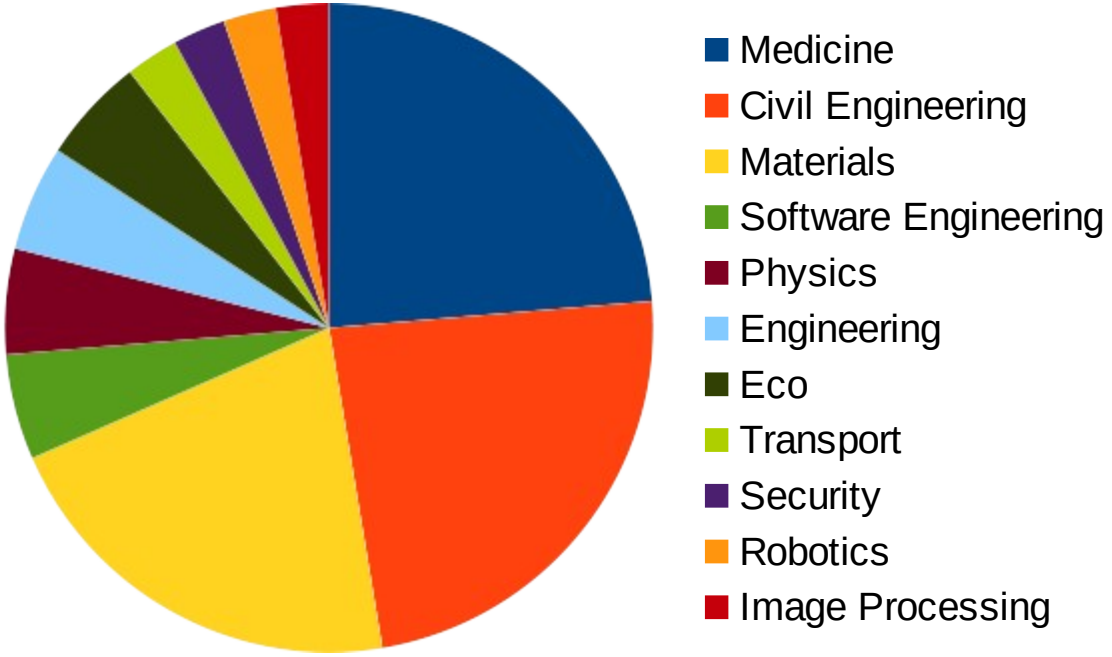
W. B. Langdon, UCL

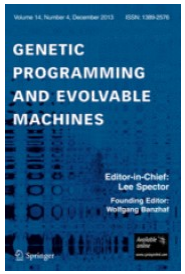
Information Theory and Robustness

- 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 means mutations have little impact in deeply nested trees
- Evidence that deep mutations have little impact in C/C++
- Implications of too much robustness
  - C++ deeper code harder to test, improve, optimise, repair
  - Sustained long term evolution needs limited depth of nesting:
    Open architecture with many small shallow programs

Applications of Genetic Programming

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

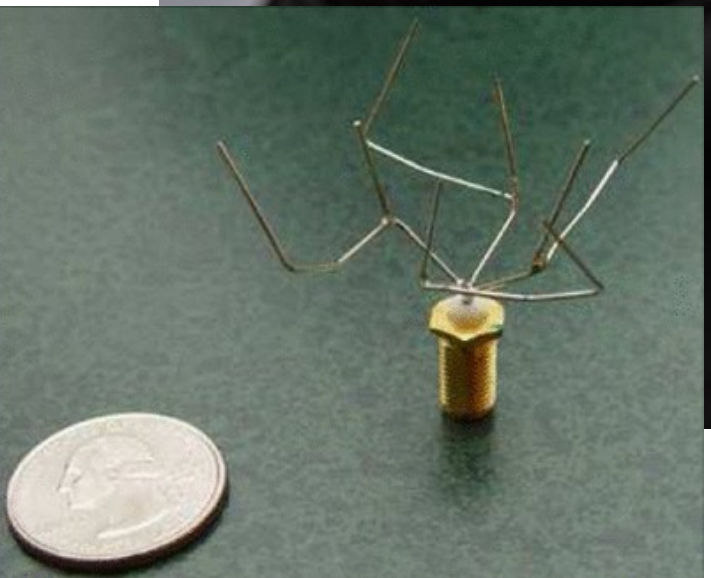


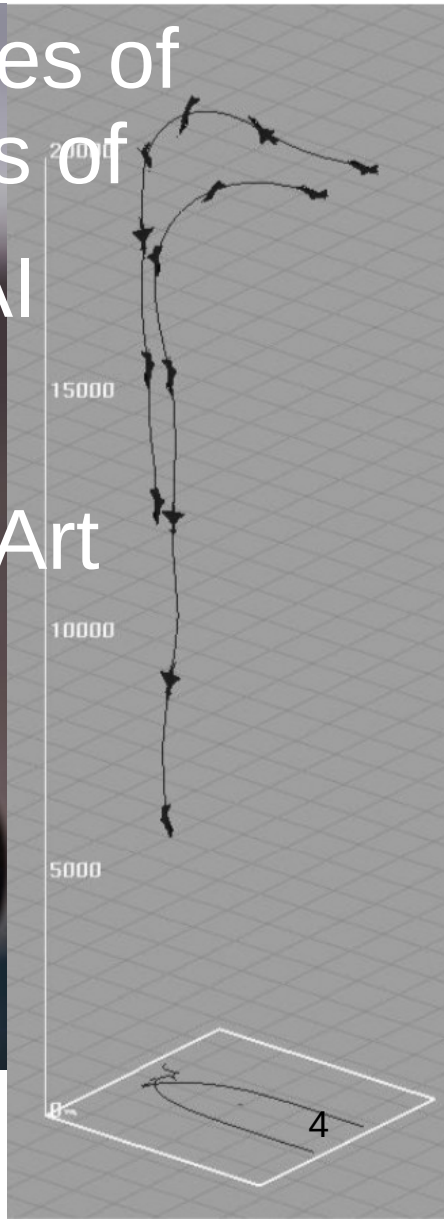


"Jaws 30"
Celebration of 30th anniversary Koza's GP book
Genetic Programming and Evolvable Machines

Creativeness of Evolutionary Computing

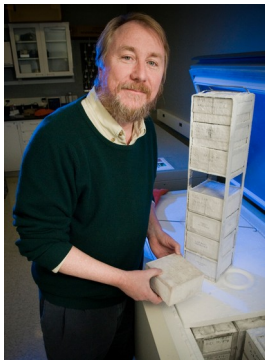
- Claim in 2019 “no working examples of creative AI” in fact several decades of
- Evolutionary Computing creative AI
  - patentable inventions
  - conference series on it EvoMusArt



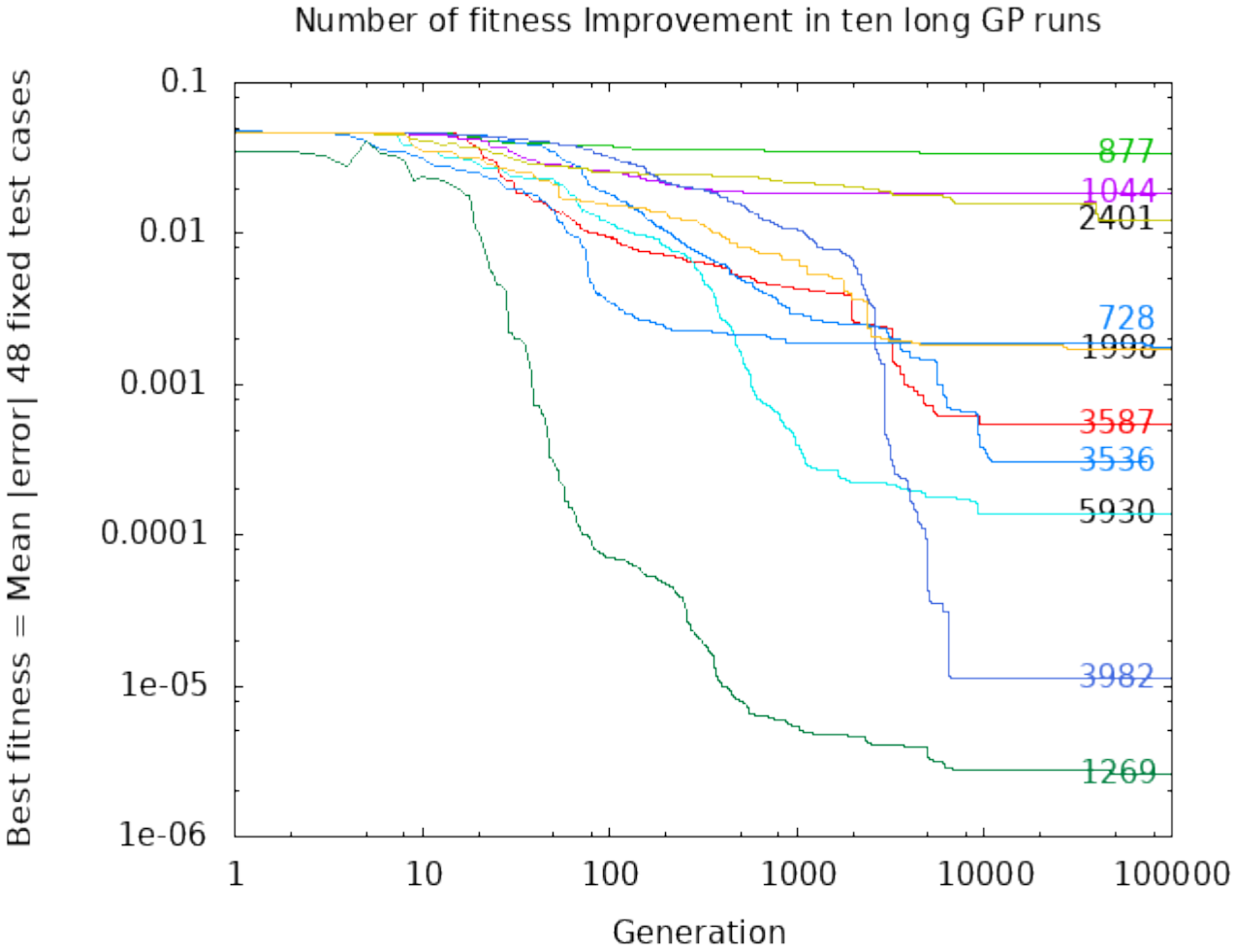


Long-Term Evolution Experiment with Genetic Programming

- Rich Lenski's Long Term Evolution Experiment LTEE 80000 generations of bacteria E.coli shows continued evolution. (Homo Sapiens about 9300 generations old)
  - Experiment running since 1988 (37 years)
- What happens in computer based evolution?
  - Run up to 1 million generations 2 billion nodes
  - Run experiments in days/weeks

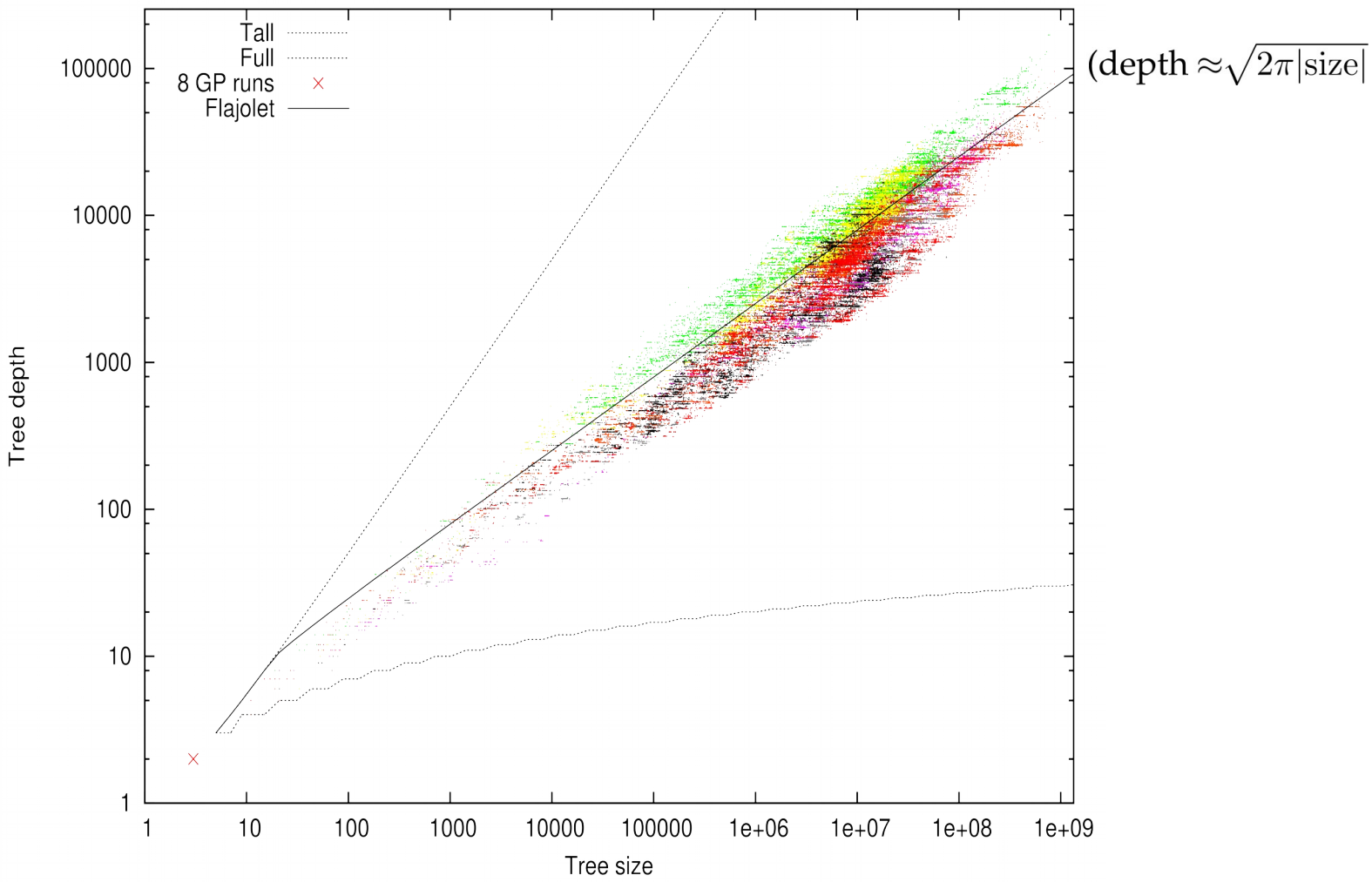


Fitness improvement continues but slows



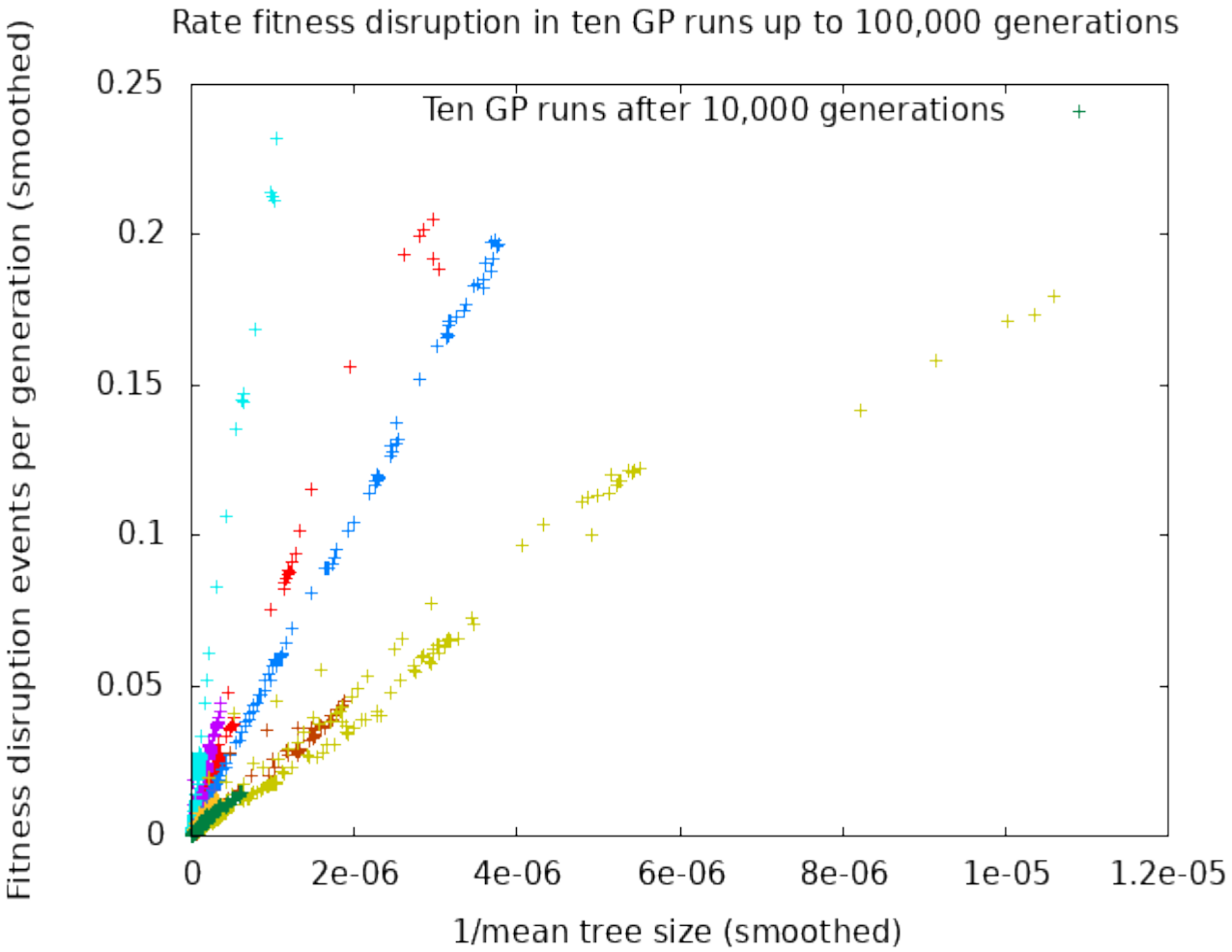
Evolution of mean absolute error in ten runs of Sextic polynomial with population of 500. Runs to 100,000 generations.
Thousands of fitness improvements found.
Note log scales.

GP trees converge centre of search space



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

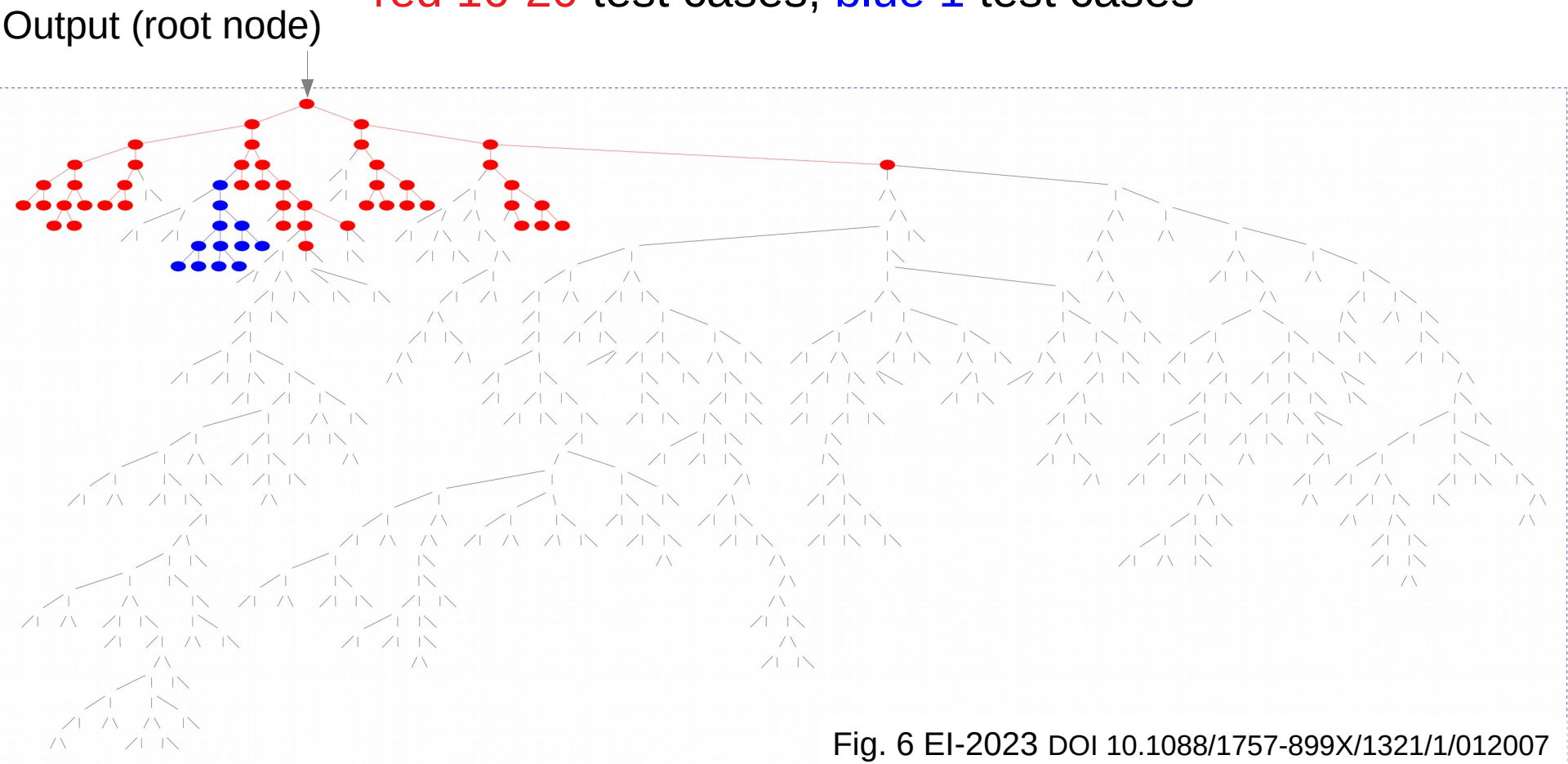
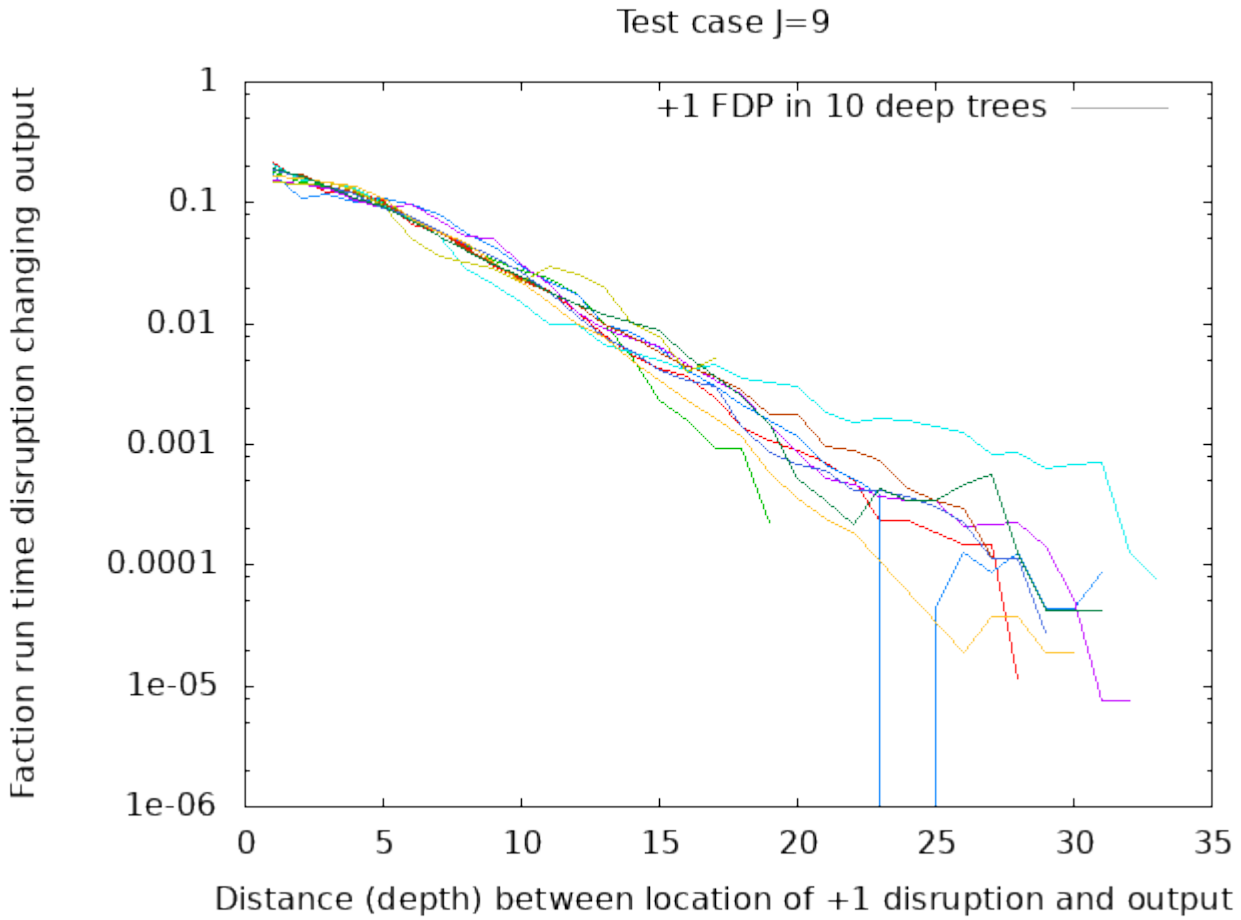


Fig. 6 EI-2023 DOI 10.1088/1757-899X/1321/1/012007

Only disruption near root node reaches output

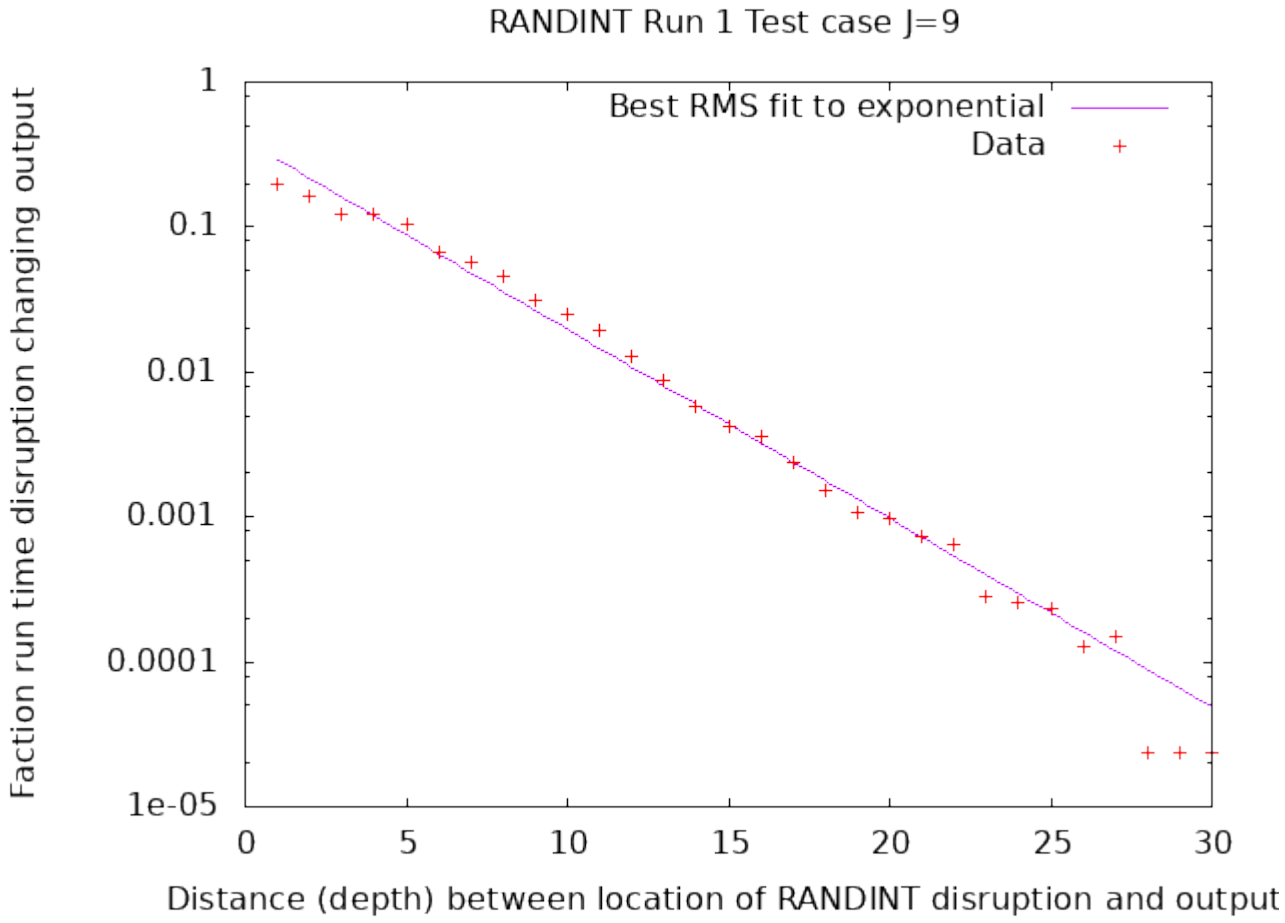
Exponential fall in fraction of run time disruption changing program output with depth



Slope: disruption falls by
≈ ½ per 2.0 levels

fdp.gif

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



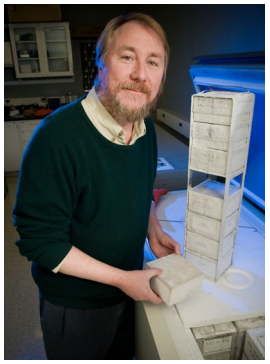
Slope: disruption falls by ½ per 2.3 levels

Deep floating point GP trees similar

params.gnu

Long-Term Evolution Experiment with Genetic Programming

- Rich Lenski's Long Term Evolution Experiment LTEE 80000 generations of bacteria E.coli shows continued evolution. (Homo Sapiens about 9300 generations old)
- What happens in computer based evolution?
  - 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



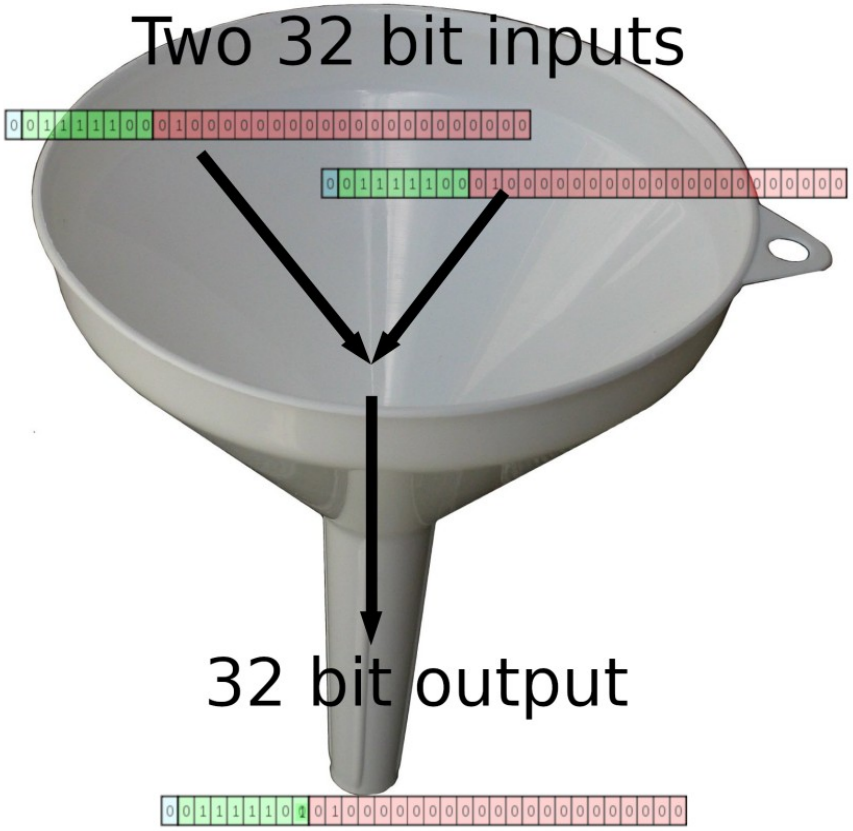
Long-Term Evolution Experiment with Genetic Programming
Artificial Life, 2022 28(2) pp173-204

Deep Mutations have Little Impact

- Voas 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 Progamming 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

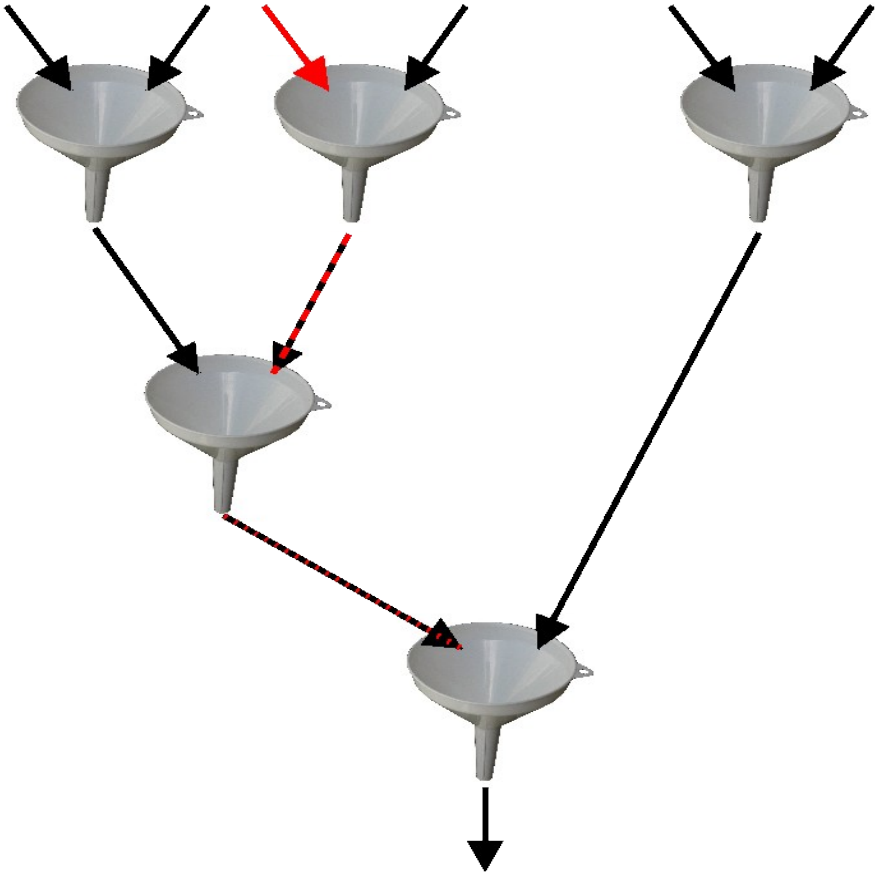


Information funnel

More information enters than leaves

Information flow in five nested functions

Potential information loss at each (irreversible) function



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
- 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 GDB 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

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

Deep imperative mutations have less impact,
Automated Software Engineering (2025) 32:6

1000 random 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

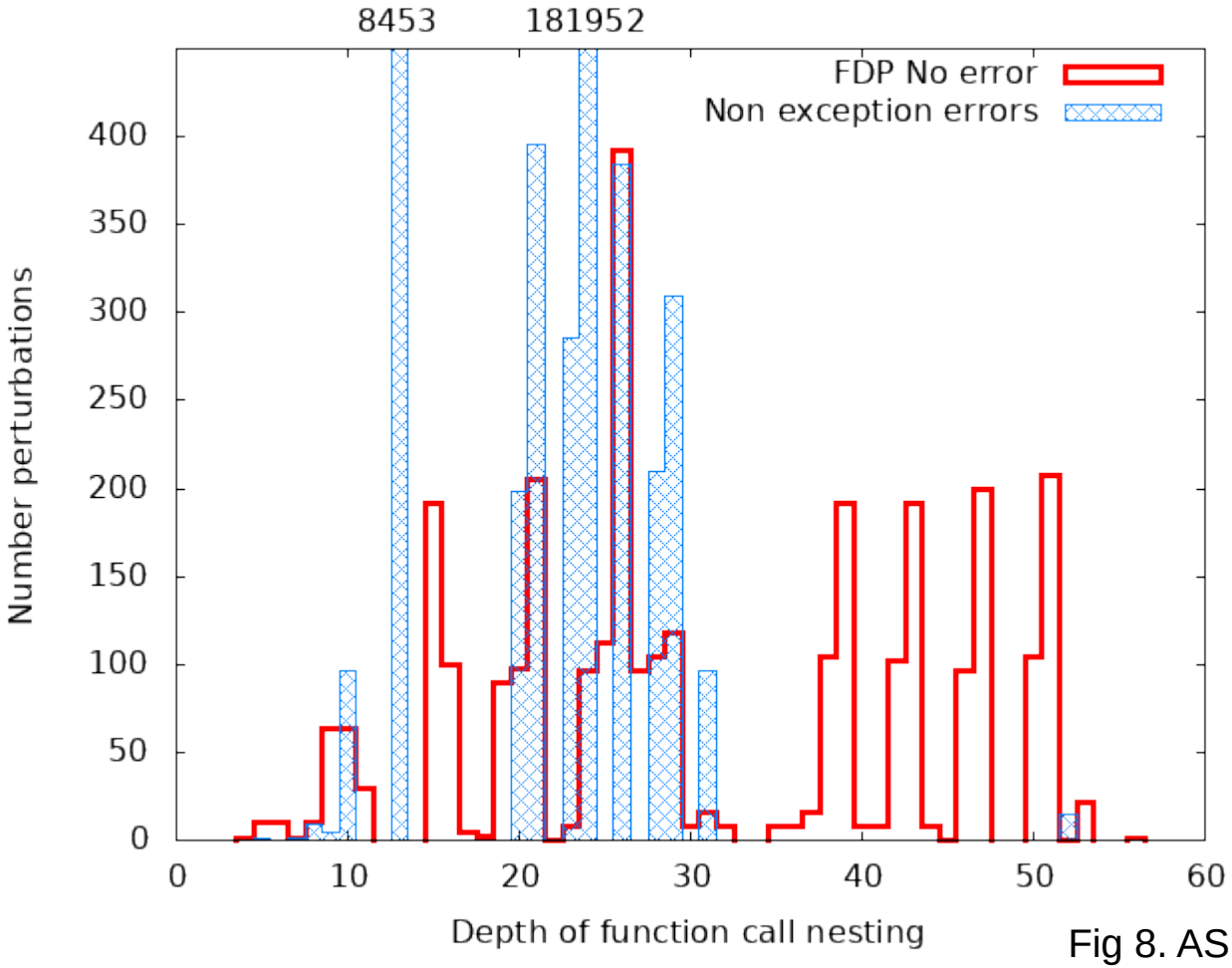
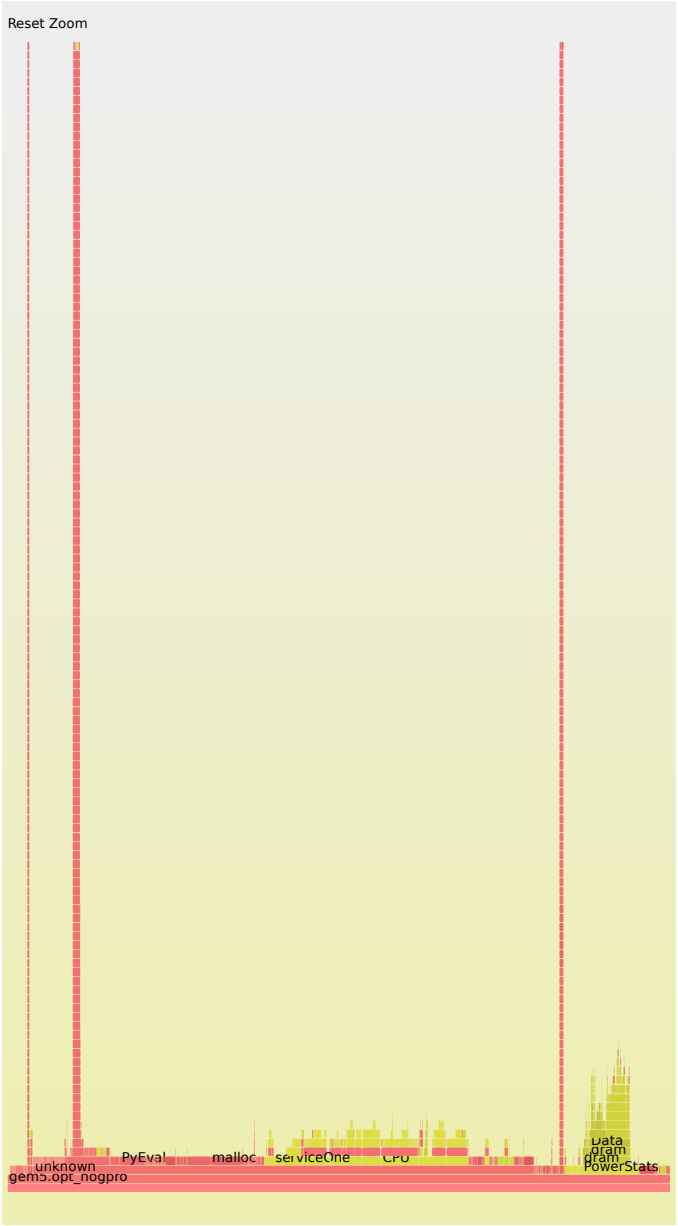


Fig 8. ASE (2025) 32:6

gem5 depth of runtime nesting



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

2500 random 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

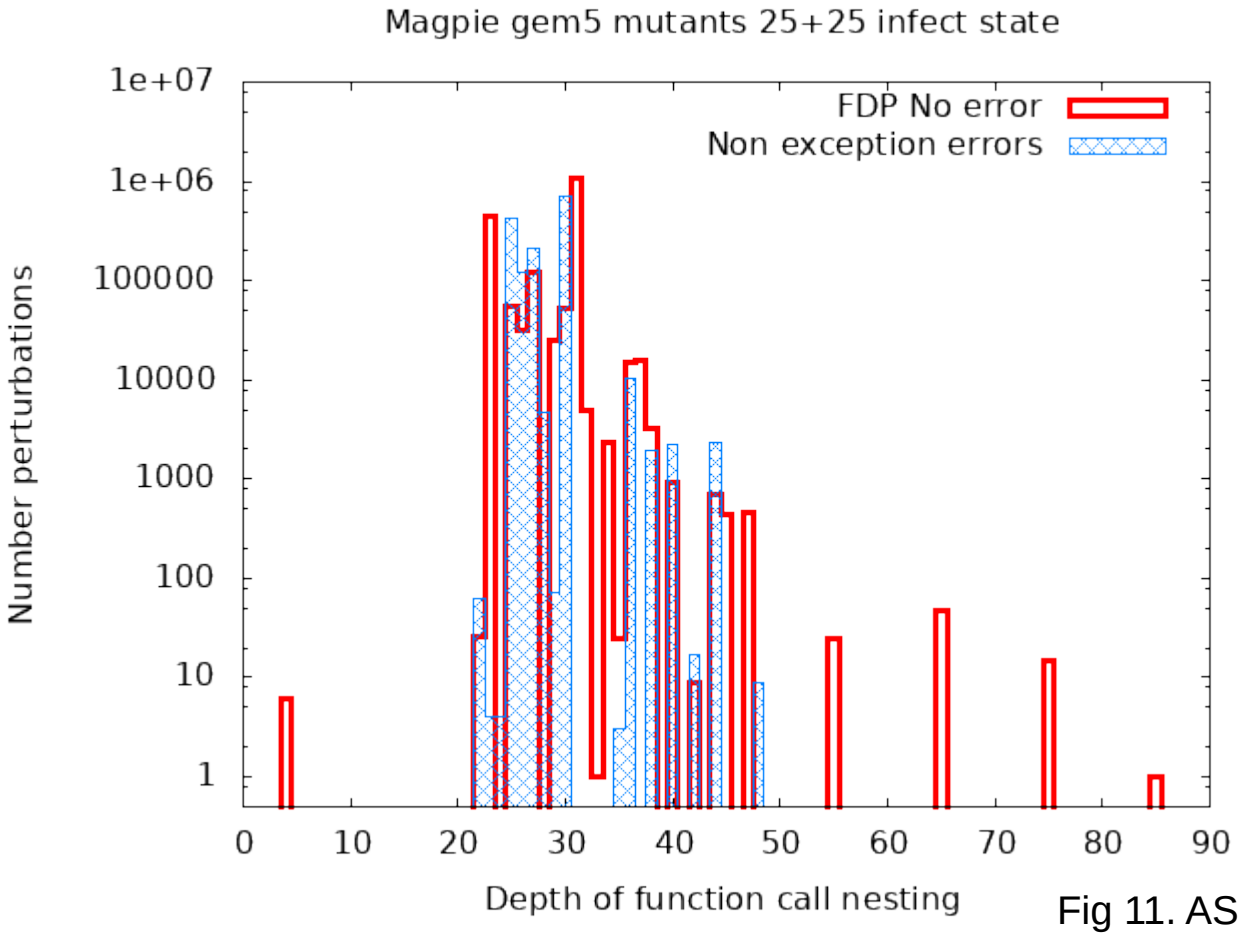


Fig 11. ASE (2025) 32:6

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

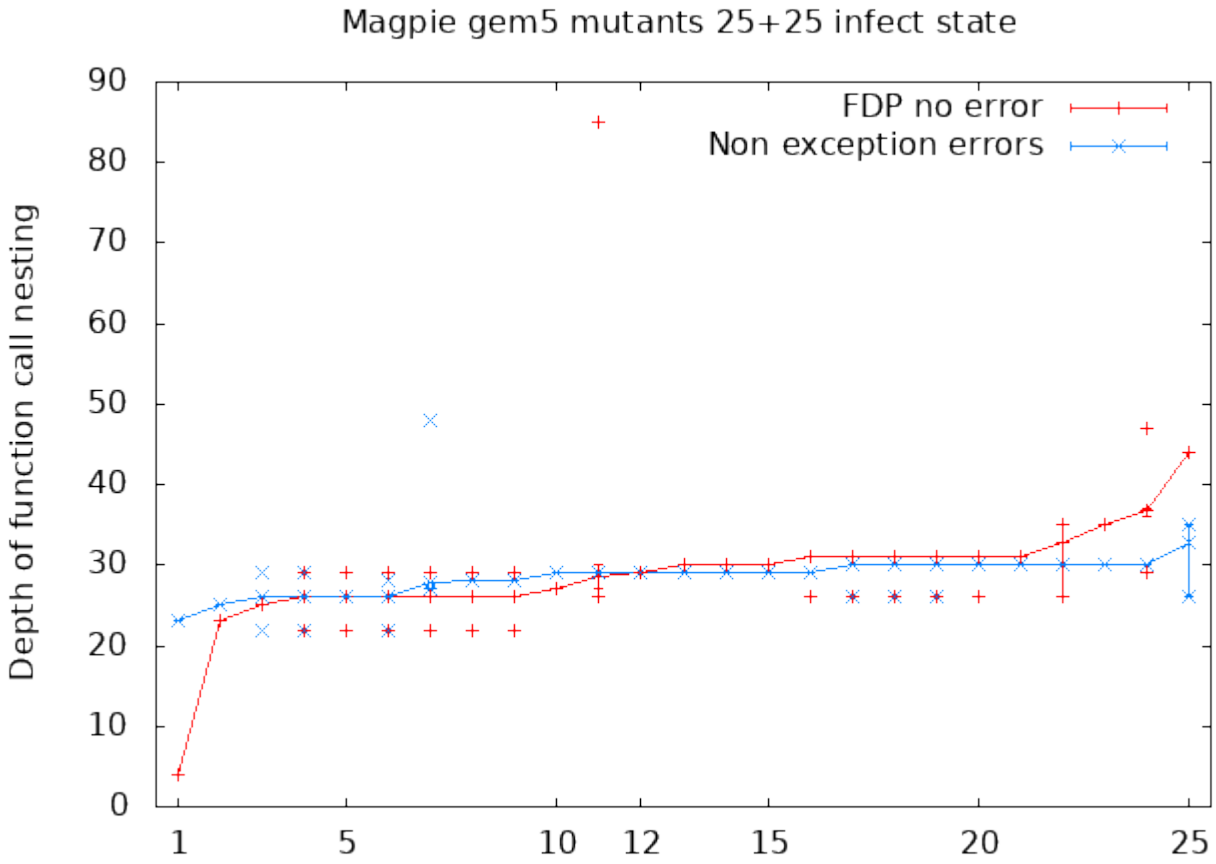
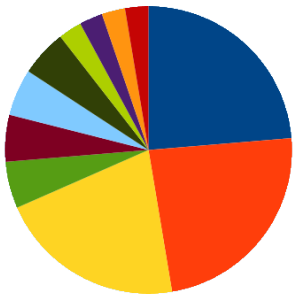


Fig 12. Deep imperative mutations have less impact
Automated Software Engineering (2025) 32:6

Conclusions: entropy loss gives robust code

- Importance of reminding everyone of GP achievements
- Genetic Programming can be creative, GP is in use
- Computing is not reversible, causing entropy loss, which gives failed disruption propagation, which makes GP and software robust
- Excluding segfault etc., most C/C++ mutations nested >30 function calls deep did not change output
- In GP exponential decay with depth, impact of mutations lost
  - Need shallow code for prolonged evolution
  - Shallow code unit tests preferred to deep system tests
- Be ambitious

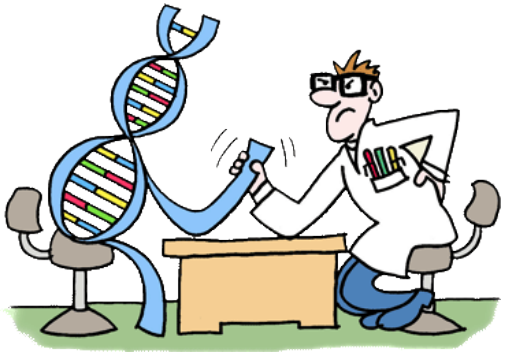


Do the impossible

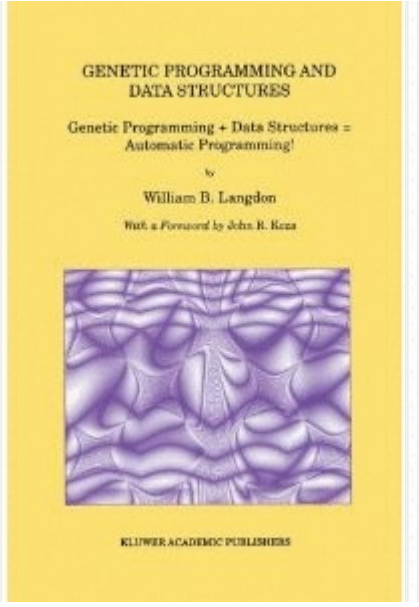
GI 2025

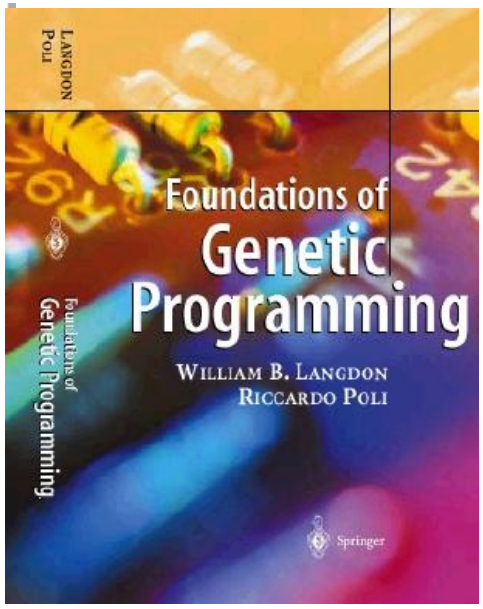
Genetic Improvement Workshop ICSE 2025

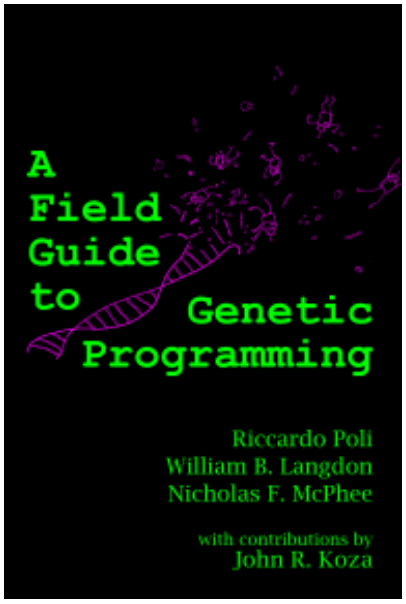
Sunday 27 April



$10000 "Humies"
Deadline 30 May







38% of 2023 GP papers on Applications

- 100 GP papers last full year in GP bib
- Classify by hand: “A” paper is primarily about an application (not GP)
- Classify “A” into topics
- Cluster topics
  - Breast Cancer ► Medicine
  - Humanitarian disaster risk reduction ► Medicine
  - Refrigerants ► Eco
  - Physics(graphene) ► Materials

Medicine	9
Civil Engineering	9
Materials	8
Software Engineering	2
Physics	2
Engineering	2
Eco	2
Transport	1
Security	1
Robotics	1
Image Processing	1

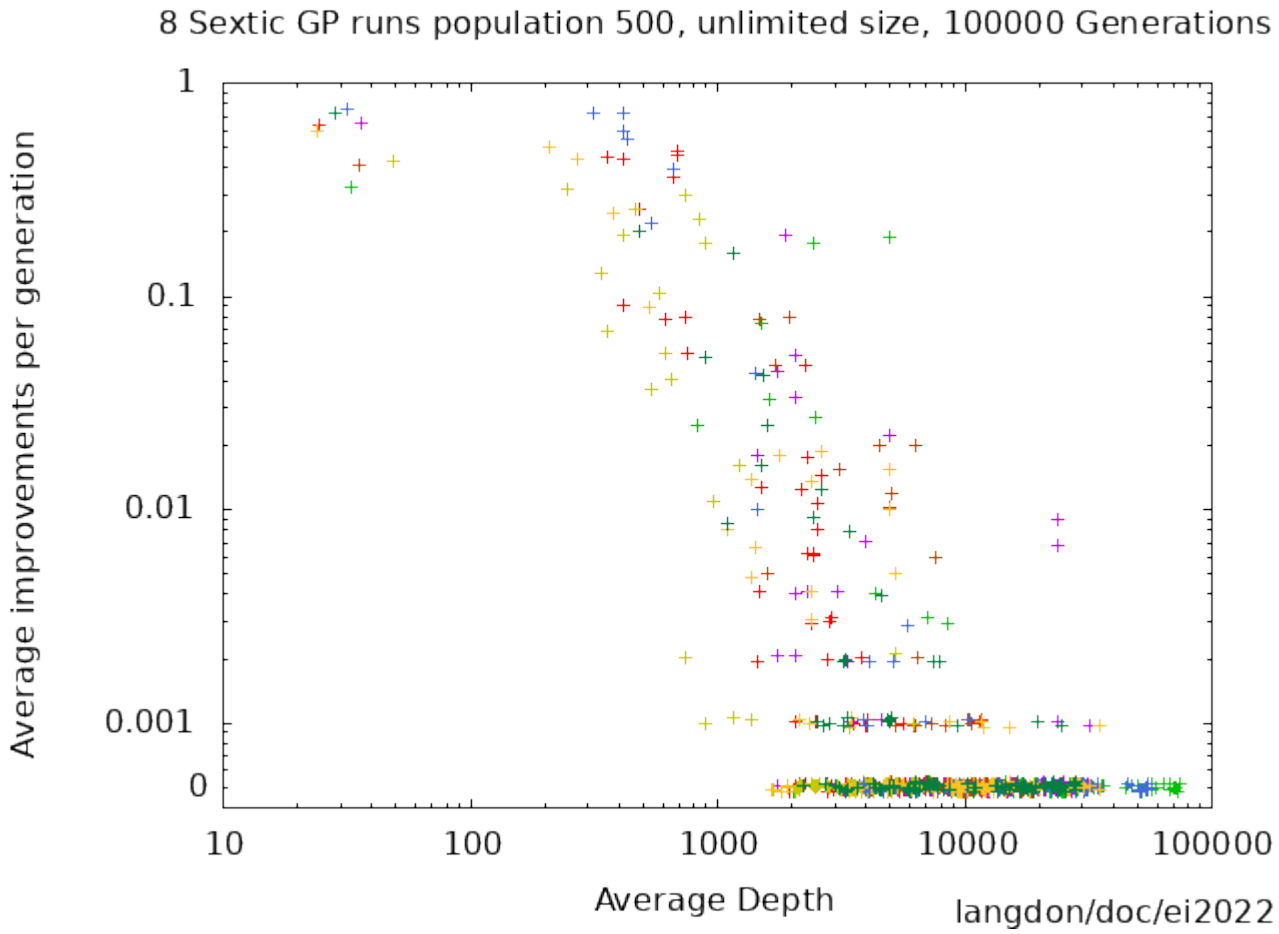
How Deep is Deep?

- 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 GP, 50% fall for ~7 levels?
- “Normal” Multi Layer Perceptron has 3 or 4 levels, Wikipedia says “deep” means more than 4 levels

Long Term Evolution Experiments

- **LTEE** shows E.Coli continued innovation 80000 generations
- Genetic Programming continued fitness improvement a million generations BUT GP slows as expressions get deeper
  - Impact of mutations lost, e.g. due to rounding error
  - In deep integer trees 92% to 99.97% of evaluation changes have no effect
- Exponential decay with depth
  - Need to be close to error for tests to find them
  - On average <7 more than 50% errors detected
- Each case may differ in detail but information theory says entropy loss will occur and will lead to robustness

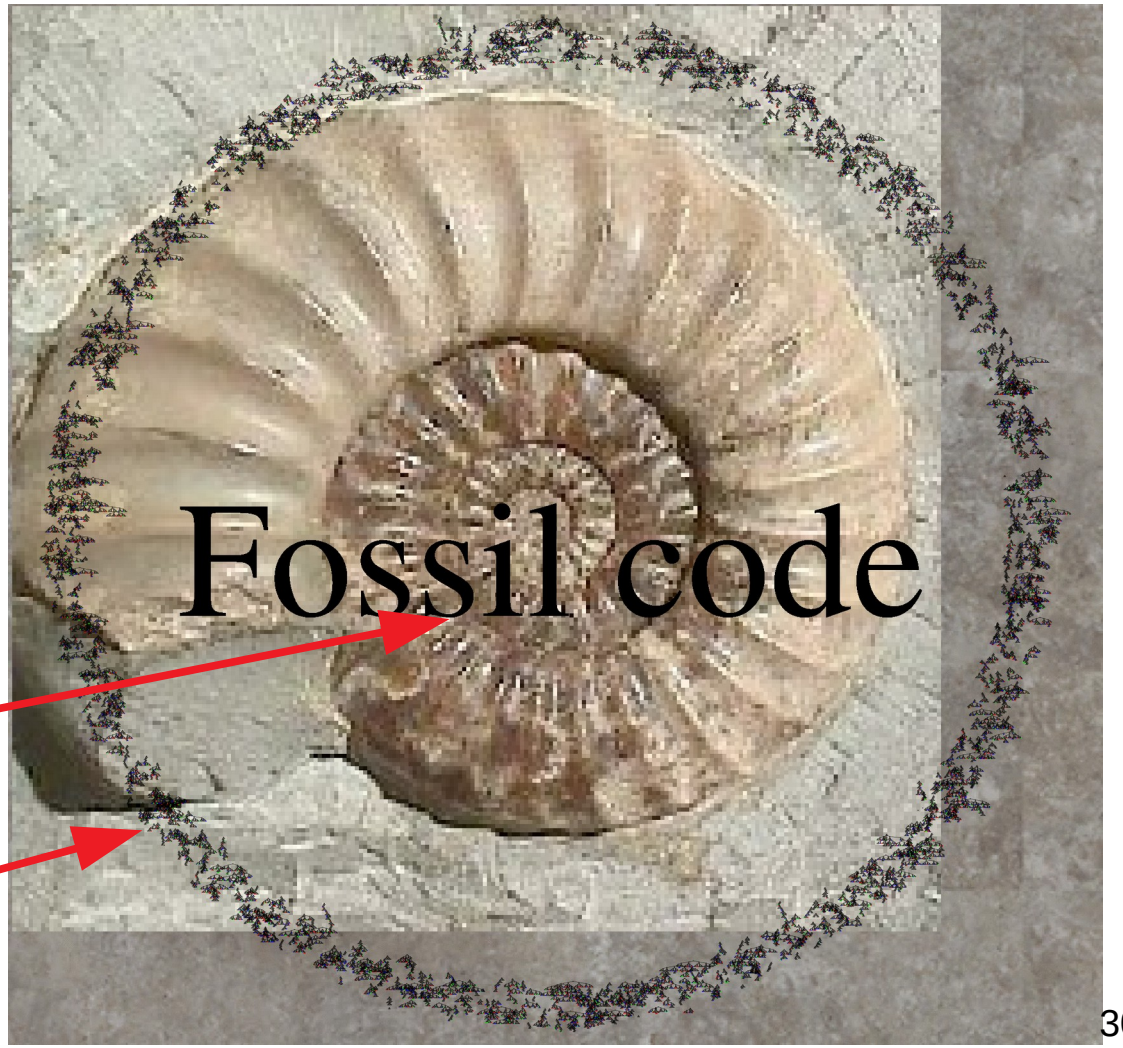
Deeper programs harder to evolve



As the GP populations evolve they find thousands of improvements but at a slower rate as the trees get deeper. Note log scales.

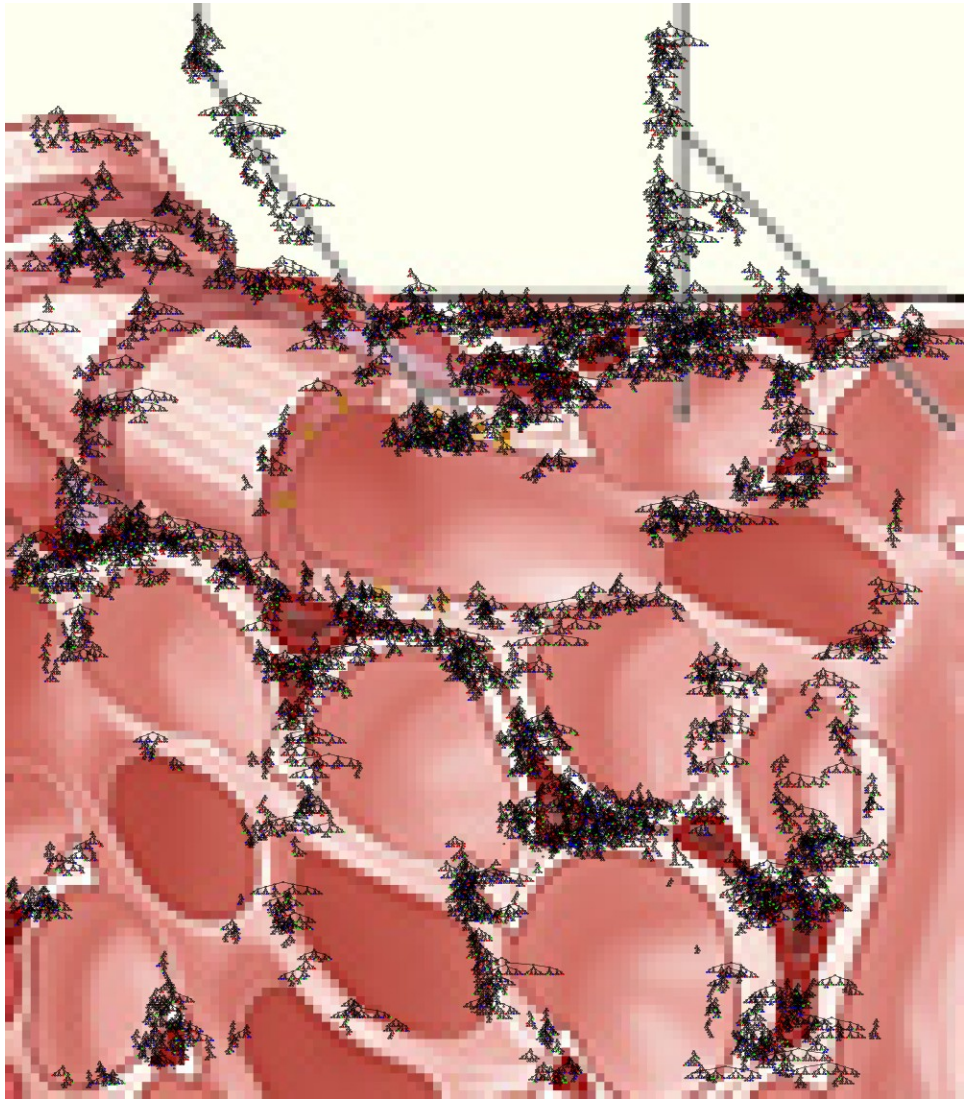
To evolve large complex code, Must **AVOID** large fossil of dead code

- With **deep code** most crossovers and mutations make **no difference**.
- Leading to random drift
- Not directed evolution
- To avoid dead center evolving code must be near environment.



Evolve Large Open, Lung Like, Open Architecture

- Make code shallow.
- Shallow code does not suffer failed disruption propagation.
- Instead fitness disruption caused by mutations and crossover do have impact.
- Fitness can direct evolution.
- Suggest large porous code
- All code near organism's environment.
- Communication between code internally & externally eased by globals, side effects, pipes, TCP/IP etc.



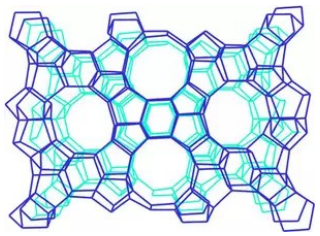
W.B. Langdon,
Embodied Intelligence 2022

Evolve **Open** Complexity

1) Information theory predicts, without side effects, nested irreversible computation will loose information and so
2) nested expressions suffer failed disruption propagation.
3) Meaning impact of deep code changes does not reach output
4) **Deep mutations do not change fitness**
5) Without fitness changes there is no evolution
6) To avoid code fossilising, changes must impact performance
7) To **evolve** code it must be **shallow**, close to environment
8) Open porous lung like code, possibly in many dimensions, with open channels between shallow <7 code modules

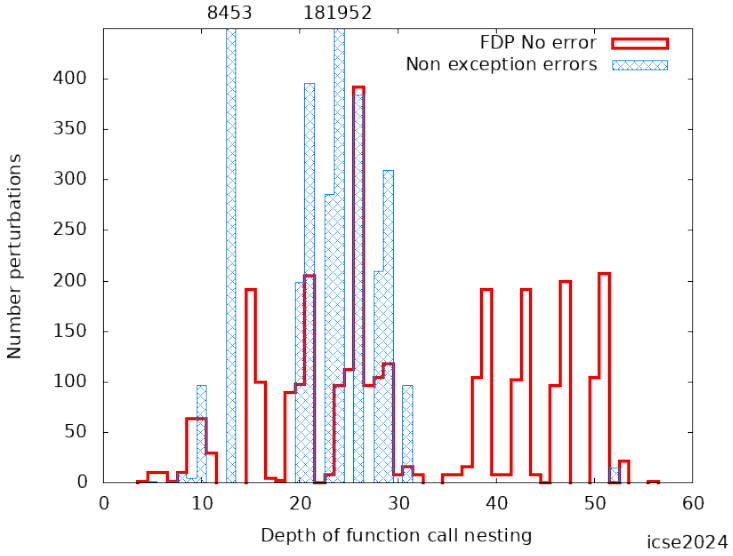
Deep Mutations have Little Impact

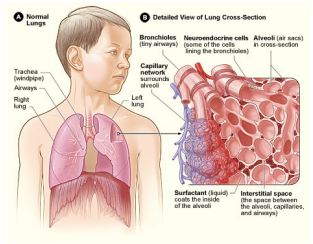
- Information theory says impact of disruptions lost with distance when nested
- True in pure functions
- Evidence true in real C++ software with side effects, globals
- Implications:
  - White box: Put test probes near changes
  - Black box: limit depth of nesting
  - Evolved embodied intelligent code must have high surface area
    - lungs, sponge, coral, pumice, zeolites. porous, low density.



Deep structures are robust, hard to adapt: Shallow code

Many, small, interlinked, low density, high surface area, codes close to fitness ecosystem



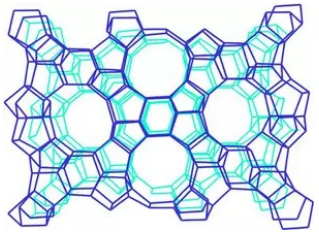


Lungs

Sponge

Coral

Pumice



Zeolite

Entropy = Information content

- Simple example, function = addition, inputs random 0-9 digits
- 1 digit mean 4.5, standard deviation σ = √8.25 entropy=$\log_2 10$
- n digit mean 4.5 n, σ = $(n\ 8.25)^{½}$,
  - large n distribution tends to Gaussian entropy=$2.047+\log_2 \sigma$
  - I.e, information content falls from 3.3n to $3.6+\log_2(n)/2$
  - Adding many digits loses almost all the information
  - Impossible to infer inputs from their sum

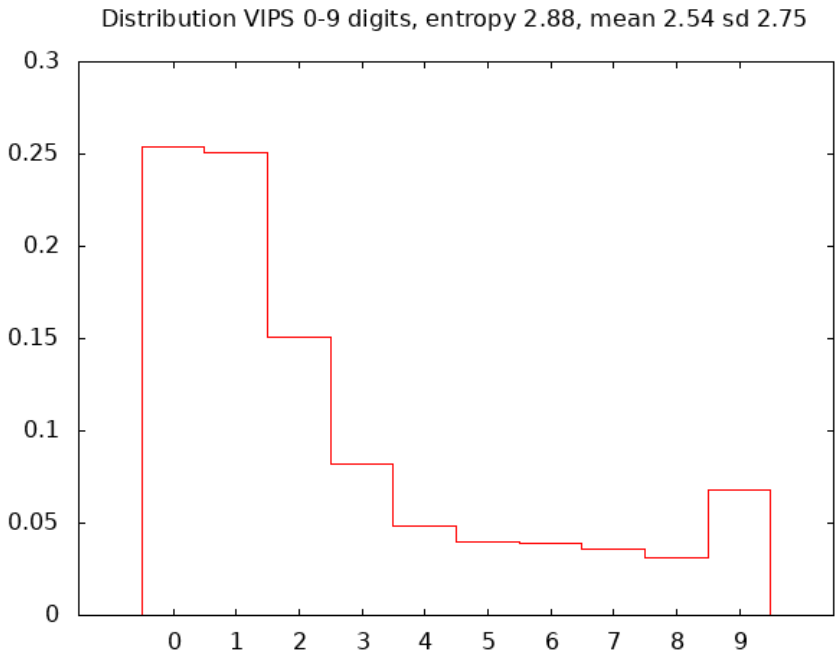
Number inputs	mean	sd σ	entropy	Gaussian entropy	Information loss
1	4.5	2.9	3.3	3.6	0%
2	9.0	4.1	4.0	4.1	39%
3	13.5	5.0	4.4	4.4	56%
4	18.0	5.7	4.6	4.6	66%
5	22.5	6.4	4.7	4.7	72%
n	4.5n	√(8.25n)		$2+\log_2\sqrt{(8.25n)}$	$< 100\% = 1 - 2/(3.3n) - (1/3.3n)\log_2\sqrt{(8.25n)}$

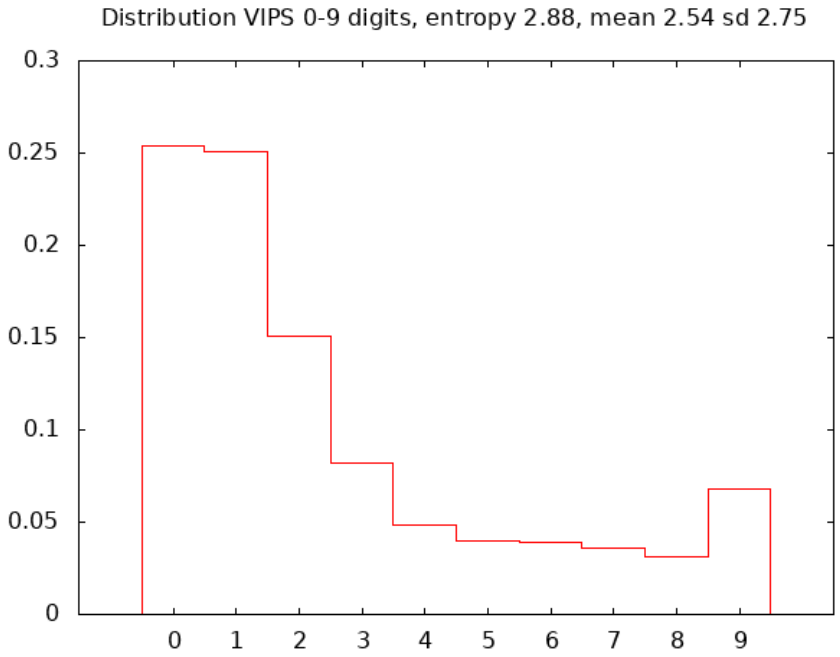
Entropy lost when adding digits

Add n non-uniformly distributed digits (0-9, left)

Quickly converges on Gaussian distribution.

Addition not reversible (given output do not know inputs)
Addition loses information (entropy)





Entropy falls from 2.88n in inputs to ≈$2+\log_2(7.7n)/2$ in output

http://www.cs.ucl.ac.uk/staff/W.Langdon/icse2024/langdon_2024_GI/add10.gif

Evolve Deep Integer GP Trees

- Integer Koza's Fibonacci problem [GECCO 2022 Companion pp574-577].
  - 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) = j^{th} 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 or replace with random int
  - +1 and RANDINT very similar
- Almost all run time disruptions make no difference

Mutated C can give: no change, small change, poor output

128 x 96 thumbnail

128 x 96 thumbnail
Rejected.
But acceptable?

128 x 96 thumbnail
Rejected.

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

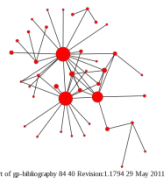
The Genetic Programming Bibliography

17539 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

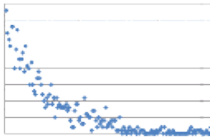
Co-authorship community.



Downloads

Downloads by day

A personalised list of every author's GP publications.



Your papers

blog

Googling GP bibliography, eg:
software testing site:gpbib.cs.ucl.ac.uk

Text search