Information theory suggests for most deeply nested mutations disruption fails to propagate to the output.

W.B. Langdon and D. Clark. In GI@ICSE 2024 Lisbon, DOI: 10.1145/3643692.3648259
Deep Mutations have Little Impact

- What is the PIE (Propagation Infection Execution) view of software bugs
  - How PIE leads to Failed Disruption Propagation and
  - Information Theory idea of software robustness.

- Information Theory says impact of disruptions lost with distance when nested

- Deep mutations have little impact in pure functions
  - Genetic programming deeply nested trees

- Traditional imperative software, C++, with side effects, including global variables

- Evidence that deep mutations have little impact in C++

- What is Magpie? Tutorial after lunch

- What is PARSEC, VIPS, C++ vipsthumbnail benchmark

- Implications:
  - unit testing v. system testing, flaky tests
  - Software code bloat, depth important
  - Software robustness: deeper harder to test, improve, optimise, repair?
Voas’ PIE and Silent Bugs, Information Theory, Robust Deeply Nested Software

• PIE framework suggested by Jeffrey M. Voas and Keith W. Miller, 1995:
  - For a software bug buried in code to have impact, it must:
    be Executed
    it must change (Infect) the program’s state
    the state change must Propagate from the bug to some externally visible point (eg a print statement)

• Expand to consider any type of Disruption, e.g. radiation, power glitch and mutations.

• Failure of Disruption to Propagate to output, means software is robust

• Information Theory helps to explain why FDP is common and software is often robust

• Initial experiments suggest FDP more common in deeply nested software
Computer operators are irreversible. Meaning input state cannot be inferred from outputs. Information is lost

Two 32 bit inputs

Information funnel

More information enters than leaves

32 bit output

W. B. Langdon, UCL
Information flow in five nested functions

Potential information loss at each (irreversible) function

Disruption may fail to reach output.

(No side effects.)
Entropy = Information content

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

<table>
<thead>
<tr>
<th>Number inputs</th>
<th>mean</th>
<th>sd $\sigma$</th>
<th>entropy</th>
<th>Gaussian entropy</th>
<th>Information loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.5</td>
<td>2.9</td>
<td>3.3</td>
<td>3.6</td>
<td>0%</td>
</tr>
<tr>
<td>2</td>
<td>9.0</td>
<td>4.1</td>
<td>4.0</td>
<td>4.1</td>
<td>39%</td>
</tr>
<tr>
<td>3</td>
<td>13.5</td>
<td>5.0</td>
<td>4.4</td>
<td>4.4</td>
<td>56%</td>
</tr>
<tr>
<td>4</td>
<td>18.0</td>
<td>5.7</td>
<td>4.6</td>
<td>4.6</td>
<td>66%</td>
</tr>
<tr>
<td>5</td>
<td>22.5</td>
<td>6.4</td>
<td>4.7</td>
<td>4.7</td>
<td>72%</td>
</tr>
<tr>
<td>$n$</td>
<td>4.5$n$</td>
<td>$\sqrt{(8.25n)}$</td>
<td></td>
<td>2+$\log_2 \sqrt{(8.25n)}^{1/2}$</td>
<td>$&lt; 100% = 1 - 2/(3.3n) - (1/3.3n) \log_2 \sqrt{(8.25n)^{1/2}}$</td>
</tr>
</tbody>
</table>
Entropy lost when adding digits

Add \( n \) non-uniformly distributed digits (0-9, left) quickly converges on Gaussian.

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

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

Evolve Deep Integer GP Trees

• Integer Koza’s Fibonacci problem [GECCO 2022 Companion pp574-577].
  • Recursive program to generate Fibonacci sequence
    \[ X_j = X_{j-1} + X_{j-2} \quad 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ... \]
  • \( 0 \ 1 \ 2 \ 3 \ J \ + \ - \ * \ SRF \)
    \( SRF(j,\text{default}) = j^{th} \text{ value. default applies if j is invalid} \)
• 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

W. B. Langdon, UCL
+1 Disruption, Fibonacci run 7, depth 33
red 16-20 test cases, blue 1 test cases

Only disruption near root node reaches output
Exponential fall in fraction of run time disruption changing program output with depth

Test case J=9

Fraction run time disruption changing output vs Distance (depth) between location of +1 disruption and output

W. B. Langdon
Exponential fall in fraction of run time disruption changing program output with depth

Deep floating point GP trees similar

W. B. Langdon
Using Magpie to Sample C++ Mutations

- Genetic Improvement tool 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 take large image create 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 files. srcml => 37 XML files
- 1000 random Magpie mutations, measure their impact, measure depth

W. B. Langdon, UCL
Magpie Mutating C++

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

3264 x 2448

128 x 96 thumbnail

W. B. Langdon, UCL
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, gives right answer 526
- 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

<table>
<thead>
<tr>
<th>Compiled, ran correct output</th>
<th>526</th>
<th>Correct output</th>
<th>438</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Mutation is identical to original code</td>
<td>88</td>
</tr>
<tr>
<td>Failed to compile</td>
<td>302</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Failed to run correctly or gave incorrect output</td>
<td>164</td>
<td>exception</td>
<td>127</td>
</tr>
<tr>
<td></td>
<td></td>
<td>output error</td>
<td>37</td>
</tr>
<tr>
<td>Magpie TypeError</td>
<td>8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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
Conclusions

• Information theory predicts failed disruption propagation.
  • In evolved pure nested functions (Genetic Programming)
    • Impact of most mutations lost. Exponential decay with depth
  • In deeply nested C++ code
    • Excluding segfault etc., most mutations >30 nested function calls did not change output

• Test oracle need to be close to error for tests to find them
• Unit testing v. system testing, flaky tests
• Software code bloat, depth important
• Software robustness: deeper code harder to test, improve, optimise, repair?
• Automatic bug fixing (APR): avoid deep mutations, make shallow changes near output? Add multiple test probes (test oracles)?

W. B. Langdon, UCL
Genetic Programming

W. B. Langdon
Human-Competitive results $10,000 prizes

Email your entry to goodman@msu.edu by Friday 31 May

W. B. Langdon
References

1) Evolving Open Complexity, W.B. Langdon, EI 2022
2) Long-Term Evolution Experiment with Genetic Programming, W.B. Langdon and W.Banzhaf, Artificial Life, 2022 28(2) pp173-204.
3) Dissipative Arithmetic, W.B. Langdon, Complex Systems. 2022, 31(3) 287-309
4) Deep Genetic Programming Trees are Robust, WB Langdon, ACM TELO, 2022 2(2) 6.
5) Genetic Programming Convergence, WB Langdon, GP+EM, 23(1) 71-104
9) Dissipative Polynomials, W.B. Langdon and J. Petke and D.Clark,in GECCO 2021 comp., pp1683-1691. DOI
The Genetic Programming Bibliography

16873 references, 16000 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

A personalised list of every author’s GP publications.

blog

Googling GP bibliography, eg:
Development and learning site:gpbib.cs.ucl.ac.uk