Using Novelty Search to Debug Physical Systems Simulations

David Griffin, Susan Stepney, Ian Vidamour
Optimising search finds an “optimal” solution to a problem.

“Optimal” normally defined by some kind of metric:
- Cost
- Score
  “How well does it do on this problem”

Lots of different methods:
- Hill Climbing
- Simulated Annealing
- Genetic Algorithms
Standard problem: Local Optima vs Global Optima

What if you don’t know what is optimal?

Different solutions might solve different problems

Multiple metrics might describe the quality of a solution
Standard problem: Local Optima vs Global Optima

What if you don’t know what is optimal?

Different solutions might solve different problems

Multiple metrics might describe the quality of a solution
Standard problem: Local Optima vs Global Optima

What if you don’t know what is optimal?

Different solutions might solve different problems

Multiple metrics might describe the quality of a solution
Novelty Search is a partial solution to Optimising Search not being very good at some problems

- Defines a behaviour space given by metrics that describe behaviour of solutions
- Attempts to provide coverage of the behaviour space
- By covering the behaviour space, solutions with unusual or novel behaviours can be found
- Multiple methods are available to achieve this
  - This talk uses PyCHARC with a Microbial Genetic Algorithm
  - Which itself is a reworking of the CHARC framework
# Software Engineering: Debugging

<table>
<thead>
<tr>
<th>Debugging is a common activity</th>
<th>Remove software bugs from a piece of software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Bugs arise during software development</td>
<td>Typically defined as unintended, unexpected and incorrect behaviour</td>
</tr>
<tr>
<td>Software bugs can be found by a variety of approaches</td>
<td>Comparison against an ‘oracle’</td>
</tr>
<tr>
<td></td>
<td>Unit testing</td>
</tr>
<tr>
<td></td>
<td>Fuzzing</td>
</tr>
</tbody>
</table>
Software Engineering: Debugging

Debugging is a common activity

Remove software bugs from a piece of software

Software Bugs arise during software development

Typically defined as unintended, unexpected and incorrect behaviour

Software bugs can be found by a variety of approaches

Comparison against an ‘oracle’
Unit testing
Fuzzing
Software Engineering: Debugging

- **Debugging is a common activity**
- **Software Bugs arise during software development**
- **Software bugs can be found by a variety of approaches**

Remove software bugs from a piece of software

Typically defined as unintended, unexpected and incorrect behaviour

Comparison against an ‘oracle’, Unit testing, Fuzzing
Debug Workflow

- Development
- Simulator
- Testing
- Experiments

Bugs found

Bugs not found
Definitions from previous slides:

- Software bugs and unintended, unexpected and incorrect behaviour
- Novel behaviour is unexpected

Corollary: It is possible to use Novelty Search to find unexpected behaviours that may correspond to software bugs.
Novel Behaviour = Bug?

Definitions from previous slides:

- Software bugs and unintended, unexpected and incorrect behaviour
- Novel behaviour is unexpected

Corollary: It is possible to use Novelty Search to find unexpected behaviours that may correspond to software bugs
Application of Novelty Search to Debugging

Replace Metrics with tests or testable hypotheses

Novelty Search attempts to find where these tests or hypotheses yield the most “novel behaviours”

If set up correctly, these novel behaviours correspond to either:
  • A bug in the software
  • A bug in the metric
    • Misunderstanding of what the system can do

Advantage: If you’re already using Novelty Search to explore behaviours, it’s very easy to use it for debugging as well!
• Application of Novelty Search to Debugging
• Replace Metrics with tests or testable hypotheses
• Novelty Search attempts to find where these tests or hypotheses yield the most “novel behaviours”
• If set up correctly, these novel behaviours correspond to either:
  • A bug in the software
  • A bug in the metric
  • Misunderstanding of what the system can do
• Advantage: If you’re already using Novelty Search to explore behaviours, it’s very easy to use it for debugging as well!
DebugNS Workflow

Development → Simulator
Simulator → Novelty Search experiments
Novelty Search experiments → DebugNS
DebugNS → Development

Inconsistent
Consistent
Tests can be used for metrics

DebugNS tests shouldn’t be binary

• Want to be more informative than ‘did it crash’ or ‘right answer’
• Can convert X < Y to two metrics: X, Y

Criteria for a test are similar to normal unit tests

• Stress the system
• Try to exercise expected and corner cases
Tests can be used for metrics

DebugNS tests shouldn’t be binary

• Want to be more informative than ‘did it crash’ or ‘right answer’
• Can convert \( X < Y \) to simply \( X \)

Criteria for a test are similar to normal unit tests

• Stress the system
• Try to exercise expected and corner cases
More abstract than using a test as a metric

Take outputs from system, evaluate it according to a hypothesis

- “No more than five types of output”
- “Output follows a known distribution”

Expose the underlying metrics to DebugNS

- Number of output types
- Multiple outputs from individuals

Easier to construct tests with a wider set of values to search over

- Can search over more of the behavior space
DebugNS
Metrics - Hypotheses

More abstract than using a test as a metric

Take outputs from system, evaluate it according to a hypothesis

- “No more than five types of output”
- “No object travels faster than the speed of light”

Expose the underlying metrics to DebugNS

- Number of output types
- Speed of objects

Easier to construct tests with a wider set of values to search over

- Can search over more of the behavior space
Case Study: RingSim

RingSim is a simulator for nanoscale magnetic ring arrays (used for exploring possibilities with reservoir computing)

Simulates the progression of domain walls within the magnetic ring array

| Domain walls are moved around the rings by an external magnetic field | They can get pinned at junctions | They can annihilate if two opposite domain walls touch | They can spawn new domain walls |
RingSim is a simulator for nanoscale magnetic ring arrays (used for exploring possibilities with reservoir computing).

Simulates the progression of domain walls within the magnetic ring array

- Domain walls are moved around the rings by an external magnetic field
- They can get pinned at junctions
- They can annihilate if two opposite domain walls touch
- They can spawn new domain walls into empty rings
During development of RingSim there was a need to refactor the simulator.

During this refactoring, DebugNS was used to examine RingSim for potential bugs.

Experiments were carried out on a square grid arrays of varying size:
- Graphs here use 5x5 for clarity.
RingSim
DebugNS
hypotheses

Hypothesis
• behaviour of rings would fall into a class given by the local topography

Novelty Search Input
• Input magnetic fields to the ring array

Novelty Search Metric
• classes of each ring (determined by metrics from rings and clustering)

Expected behaviours
• Number of ring classes is much smaller than number of rings
• Ring class is mirror and rotationally symmetric
<table>
<thead>
<tr>
<th>Hypothesis</th>
<th>• behaviour of rings would fall into a class given by the local topography</th>
</tr>
</thead>
<tbody>
<tr>
<td>Novelty Search Input</td>
<td>• Input magnetic fields to the ring array</td>
</tr>
<tr>
<td>Novelty Search Metric</td>
<td>• classes of each ring (determined by metrics from rings and clustering)</td>
</tr>
<tr>
<td>Expected behaviours</td>
<td>• Number of ring classes is much fewer than number of rings</td>
</tr>
<tr>
<td></td>
<td>• Ring class is mirror and rotationally symmetric</td>
</tr>
</tbody>
</table>
Ring Classes in RingSim v1
What happened?

Observations were used to develop RingSim v2.

Multiple bugs were found from the relatively simple hypotheses that were tested.

- Domain walls could pass through each other rather than annihilating.
- Assumption of a clockwise direction for driving magnetic field.
- Behaviour dependent on edge due to RingSim v1 updating rings sequentially.
What happened?

Multiple bugs were found from the relatively simple hypotheses that were tested:

- Domain walls could pass through each other rather than annihilating.
- Assumption of a clockwise direction for driving magnetic field.
- Behaviour dependent on edge due to RingSim v1 updating rings sequentially.

Observations were used to develop RingSim v2.
Ring Classes in RingSim v2
Conclusions & Further Work

Novelty Search, with appropriate metrics, can be used as a debugging tool.

And an easy-to-use debugging tool, if you’re already planning on using Novelty Search.

Potentially most powerful if there is a trusted oracle to compare results against – a slow but accurate simulator.

But can also be used to get a population of varied individuals and check that a hypothesis holds.

Future work: investigate using Novelty Search to automatically test equivalence between two systems. E.g. two implementations of an experimental setup.
Conclusions & Further Work

Novelty Search, with appropriate metrics, can be used as a debugging tool.

And an easy-to-use debugging tool, if you’re already planning on using Novelty Search.

Potentially most powerful if there is a trusted oracle to compare results against – a slow but accurate simulator.

But can also be used to get a population of varied individuals and check that a hypothesis holds.

Future work: investigate using Novelty Search to automatically test equivalence between two systems.

E.g. two implementations of an experimental setup.
Questions