10 Years After: Automatic Software Generation and Improvement Through Search Based Techniques

Dr. Andrea Arcuri
Westerdals Oslo ACT
University of Luxembourg
arcand@westerdals.no
While in my first year of PhD in 2006-2007

- Working on SBST for unit testing
  - tools and theory
- Research group on evolutionary computation, not software engineering (SE)
- Meeting with supervisor/post-docs, discussion about co-evolution
  - Q: “Any problem in SE that can be addressed with co-evolution???”
Literally, a solution in search of a problem...
Hmmmm, co-evolving **tests** with what???
Hmmmm, **programs**?

- Use Genetic Programming (GP) to evolve programs
  - Eg, for machine-learning classifiers
- Use SBST to evolve test cases
- But co-evolving them to achieve what?
  - What is the fitness function here???
What about automatically generate code from formal specifications?

• You use SBST to generate some tests
• Formal specification used as *oracle* for the tests
• Use GP to evolve programs to pass all tests
• Then evolve new tests, and keep on as co-evolution...
Did it work???

• Already struggled on toy examples like a bubble-sort...
• ... and anyway, who the heck writes formal specifications???
• But could still publish a couple of papers though...
  • ASE’07 and INS
• Time for (yet) a new PhD topic???
From failure to “simpler” problems...

• Full code generation didn’t work
• But what if I try from a simpler case where program is nearly complete?
• ... actually not a so silly question as it sounds...
Bug Fixing

• A program with bugs is a “nearly” complete program
• Use it as seeding for first generation of GP
• It worked fine on some toy examples...
• But still issues with formal specifications... without it cannot use co-evolution (as used for automated oracle)
• Who cares about the *hammer/co-evolution*?
• ... throw it away if it doesn’t work!!!
• “repairing” a program on a fixed set of test cases is a real SE problem
• ... done manually by millions of engineers every day...
• A lot of work done in the last 10 years by different researchers
Can get something useful out of co-evolution?

• Discussion with **David R. White**... 2007-2008
  • Both in SEBASE project
  • Me PhD student in Birmingham, he in York

• ... after few beers... what if:
  • original program as oracle for functional correctness
  • second objective: optimize for *performance*
  • would *co-evolution* work here?
  • can we *genetically improve* an existing program to make it faster?
Quite different from what compilers do
Did it work?

- *Small* functions
- *Found quite a few non-trivial improvements that compilers cannot do*
- ... so, somehow **yes**

<table>
<thead>
<tr>
<th>Name</th>
<th>LOC</th>
<th>GP Nodes</th>
<th>Input</th>
<th>LV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triangle1</td>
<td>35</td>
<td>107</td>
<td>int, int, int</td>
<td>1</td>
</tr>
<tr>
<td>Triangle2</td>
<td>38</td>
<td>175</td>
<td>int, int, int</td>
<td>1</td>
</tr>
<tr>
<td>Sort1</td>
<td>11</td>
<td>63</td>
<td>int[], int</td>
<td>3</td>
</tr>
<tr>
<td>Sort2</td>
<td>18</td>
<td>69</td>
<td>int[], int</td>
<td>4</td>
</tr>
<tr>
<td>Factorial</td>
<td>7</td>
<td>16</td>
<td>int</td>
<td>0</td>
</tr>
<tr>
<td>Remainder</td>
<td>40</td>
<td>208</td>
<td>int, int</td>
<td>3</td>
</tr>
<tr>
<td>Swi10</td>
<td>22</td>
<td>68</td>
<td>int</td>
<td>1</td>
</tr>
<tr>
<td>Select</td>
<td>94</td>
<td>392</td>
<td>int[], int, int</td>
<td>9</td>
</tr>
</tbody>
</table>
In Retrospective...

• Haven’t worked on such topics since finishing PhD in 2009
  • got more focused on software testing
• Curiosity-driven research on uncharted territories is high risk
  • I got lucky to get publishable results
  • Maybe not really best for a PhD main topic...

• In industry, plenty of “problems” that need a “solution”
• Need to find right balance in SE research
Conclusion

• Bug Fixing and Genetic Improvement were “lucky” accidents
• At that time 2007-2009 results just on small functions
• But shown it was feasible
• Looking forward to next talks to see the current status ;(
References

• *Coevolving programs and unit tests from their specification*, Arcuri and Yao, ASE’07

• *A novel co-evolutionary approach to automatic software bug fixing*, Arcuri and Yao, CEC’08

• *Multi-objective improvement of software using co-evolution and smart seeding*, Arcuri, White, Clark and Yao, SEAL’08

• *Automatic software generation and improvement through search based techniques*, Arcuri, PhD 2009

• *Evolutionary repair of faulty software*, Arcuri, ASoC’11

• *Evolutionary improvement of programs*, White, Arcuri and Clark, TEVC’11