A Guided Genetic Algorithm for Automated Crash Reproduction

Soltani, Panichella, & van Deursen
2017 International Conference on Software Engineering

Presented by: Katie Keith, Emily First, Pradeep Ambati

A Guided Genetic Algorithm for Automated Crash Reproduction

(2) Approach / Key idea

A Guided Genetic Algorithm for Automated Crash Reproduction

(1)Problem

Motivation of Automated Crash Reproduction

Reduce developers' effort for crash debugging

(Source: Google Images)

Overview of Automated Crash Reproduction

- Focus on "**post-failure**" approach = use data available after failure
- Narrow down to only using **crash stack traces**
- Generate **tests** that trigger the target failure

```
java.lang.NullPointerException:
  at org.apache.tools.ant.util.SymbolicLinkUtils.
       isSymbolicLink(SymbolicLinkUtils.java:107)
  at org.apache.tools.ant.util.SymbolicLinkUtils.
       isSymbolicLink(SymbolicLinkUtils.java:73)
  at org.apache.tools.ant.util.SymbolicLinkUtils.
       deleteSymbolicLink(SymbolicLinkUtils.java:223)
  at org.apache.tools.ant.taskdefs.optional.unix.
       Symlink.delete(Symlink.java:187)
```

Listing 1. Crash Stack Trace for ANT-49137.

```
public void test0()  throws Throwable  {
  Symlink symlink0 = new Symlink();
  symlink0.setLink("");
  symlink0.delete();
}
```

Listing 2. Generated test by EvoCrash for ANT-49137.

Limitations of other methods

1. Can't deal with **external environmental dependencies** (network inputs and files)

2. Finding test cases that trigger the crash is **computationally expensive** (usually large number of unnecessary mutated test cases)

EvoCrash built on EvoSuite

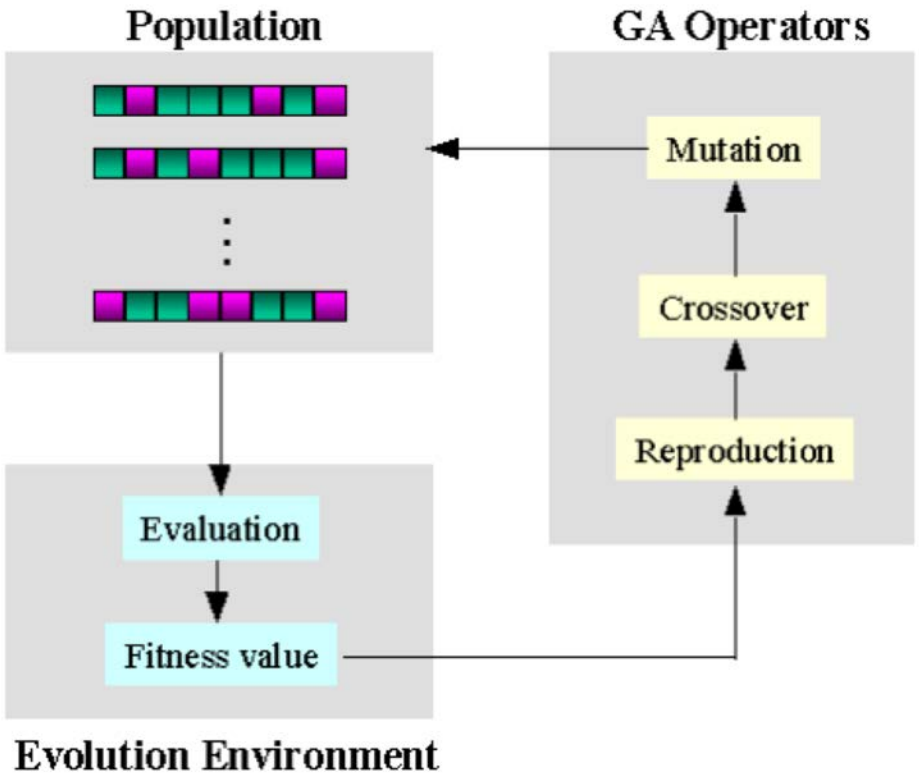
EvoSuite

(automatic test suite generation tool for Java)

EvoCrash

(built with stack-trace guided genetic algorithm)

Genetic algorithms refresher



(Source: Liao and Sun https://www.ewh.ieee.org/soc/es/May2001/14/Begin.htm)

Research questions

1. Under what circumstances does EvoCrash successfully/unsuccessfully **reproduce target crashes**?

1. How does EvoCrash **compare to state-of-the-art** automated reproduction approaches (that also only exclusively use stack traces)?

Guided Genetic Algorithm

Overview:

- Initial Population
- Fitness
- Operators: Crossover & Mutation
- Post Processing

Example

Stack Trace

```
java.lang.NullPointerException:
  at org.apache.tools.ant.util.SymbolicLinkUtils.
      isSymbolicLink(SymbolicLinkUtils.java:107)
  at org.apache.tools.ant.util.SymbolicLinkUtils.
      isSymbolicLink(SymbolicLinkUtils.java:73)
  at org.apache.tools.ant.util.SymbolicLinkUtils.
      deleteSymbolicLink(SymbolicLinkUtils.java:223)
  at org.apache.tools.ant.taskdefs.optional.unix.
      Symlink.delete(Symlink.java:187)
```

Initial Populaton

Stack Trace

```
java.lang.NullPointerException:
  at org.apache.tools.ant.util.SymbolicLinkUtils.
      isSymbolicLink(SymbolicLinkUtils.java:107)
  at org.apache.tools.ant.util.SymbolicLinkUtils.
      isSymbolicLink(SymbolicLinkUtils.java:73)
  at org.apache.tools.ant.util.SymbolicLinkUtils.
      deleteSymbolicLink(SymbolicLinkUtils.java:223)
  at org.apache.tools.ant.taskdefs.optional.unix.
      Symlink.delete(Symlink.java:187)
```

class under test

Guided Initial Population

Stack Trace

```
java.lang.NullPointerException:
  at org.apache.tools.ant.util.SymbolicLinkUtils.
      isSymbolicLink(SymbolicLinkUtils.java:107)
  at org.apache.tools.ant.util.SymbolicLinkUtils.
      isSymbolicLink(SymbolicLinkUtils.java:73)
  at org.apache.tools.ant.util.SymbolicLinkUtils.
      deleteSymbolicLink(SymbolicLinkUtils.java:223)
  at org.apache.tools.ant.taskdefs.optional.unix.
      Symlink.delete(Symlink.java:187)
```

call to this method is *target call*

Fitness

```
java.lang.NullPointerException:
  at org.apache.tools.ant.util.SymbolicLinkUtils.
      isSymbolicLink(SymbolicLinkUtils.java:107)
  at org.apache.tools.ant.util.SymbolicLinkUtils.
      isSymbolicLink(SymbolicLinkUtils.java:73)
  at org.apache.tools.ant.util.SymbolicLinkUtils.
      deleteSymbolicLink(SymbolicLinkUtils.java:223)
  at org.apache.tools.ant.taskdefs.optional.unix.
      Symlink.delete(Symlink.java:187)
```

Single-point Crossover

Parent Test 0

```
public void test0()  throws Throwable  {
  Symlink symlink0 = new Symlink();
  symlink0.setLink("");
  symlink0.delete();
}
```

Parent Test 1

```
public void  test1()  throws Throwable  {
  Symlink symlink0 = new Symlink();
  symlink0.delete();
  Symlink symlink1 = new Symlink();
}
```

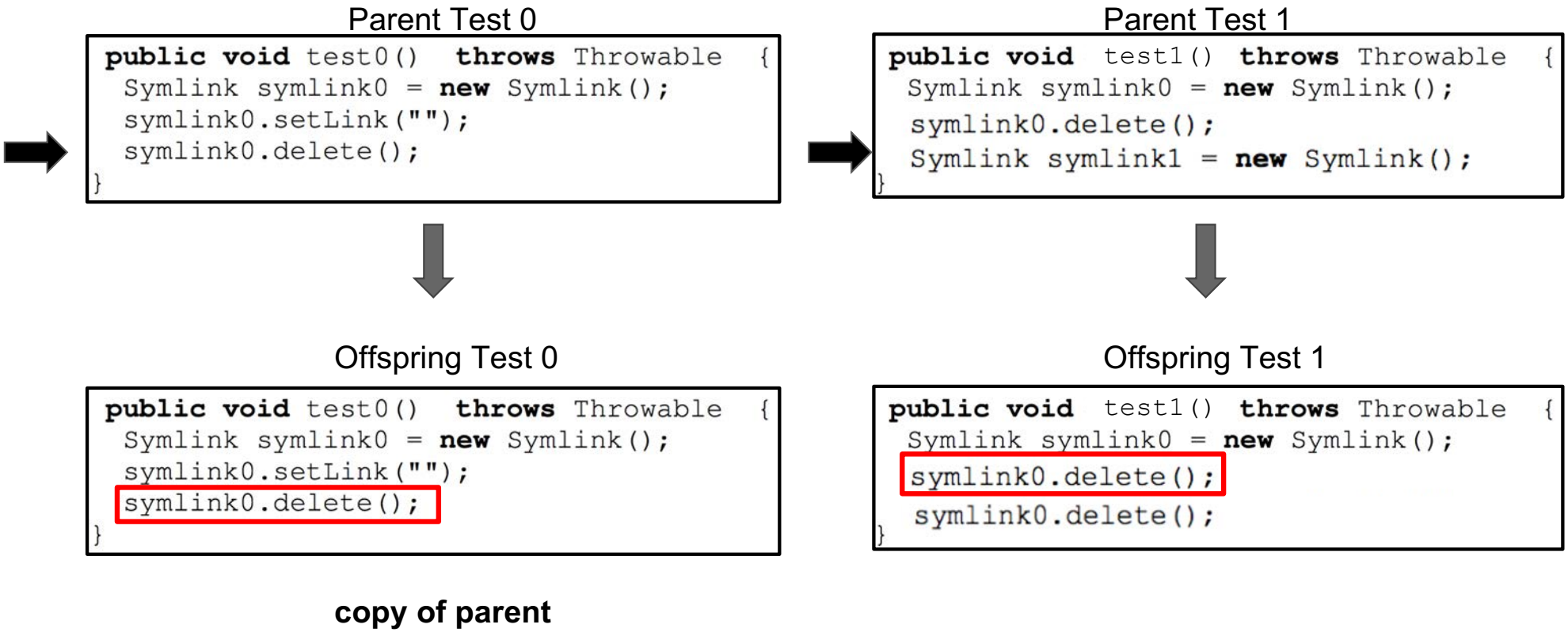
Offspring Test 0

```
public void test0()  throws Throwable  {
  Symlink symlink0 = new Symlink();
  symlink0.setLink("");
  Symlink symlink1 = new Symlink();
}
```

Offspring Test 1

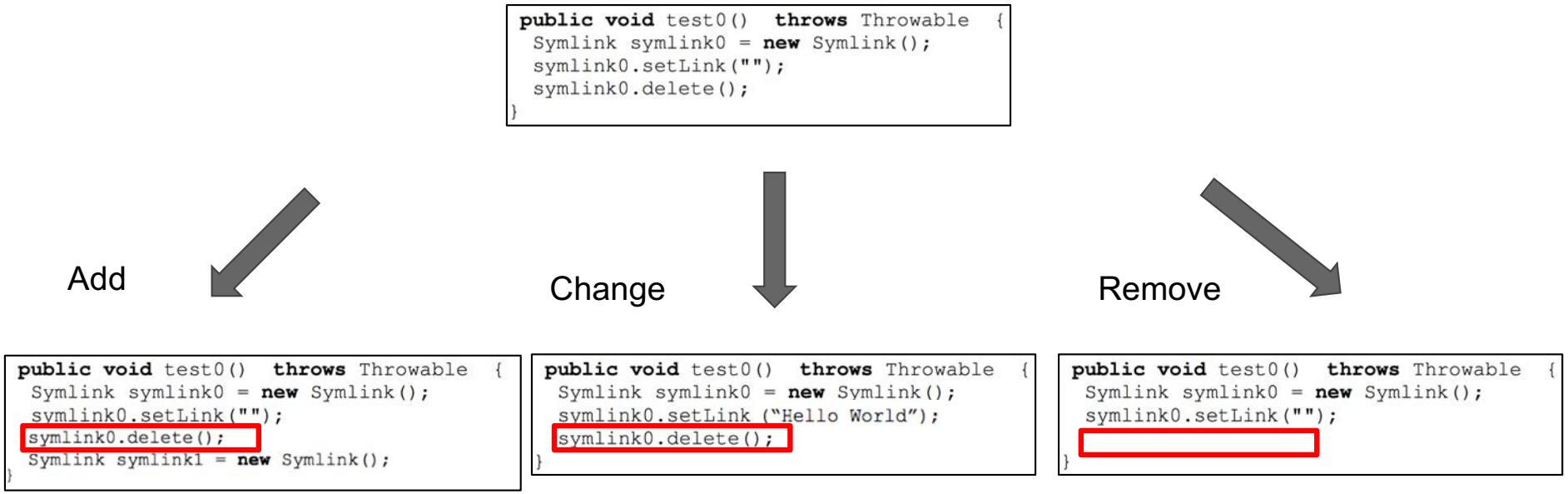
```
public void  test1()  throws Throwable  {
  Symlink symlink0 = new Symlink();
  symlink0.delete();
  symlink0.delete();
}
```

Guided Single-point Crossover



copy of parent

Mutation


public void test0() throws Throwable {
Symlink symlink0 = new Symlink();
symlink0.setLink("");
symlink0.delete();
}
Add
Change
Remove
public void test0() throws Throwable {
Symlink symlink0 = new Symlink();
symlink0.setLink("");
symlink0.delete();
Symlink symlink1 = new Symlink();
}
public void test0() throws Throwable {
Symlink symlink0 = new Symlink();
symlink0.setLink ("Hello World");
symlink0.delete();
}
public void test0() throws Throwable {
Symlink symlink0 = new Symlink();
symlink0.setLink("");
}

Guided Mutation

While target call is not in test case,

Continue mutation until re-inserted

Post Processing

- May add irrelevant statements
- Test minimization
  - Greedy Algorithm

Evaluation : Definition and Context

- Test setup consists of 50 bugs from 3 real-world open source projects
  - Apache Commons Collections (ACC) 12 bugs
  - Apache Ant (ANT) 20 bugs
  - Apache Log4j (LOG) 18 bugs

- Severity of these real-world bugs varies between
  - Medium (50%)
  - Major (36%)
  - Critical (6%)

Evaluation: Experimental Procedure

- To evaluate the successful crash reproduction rate
  - Crash Coverage
  - Test usefulness

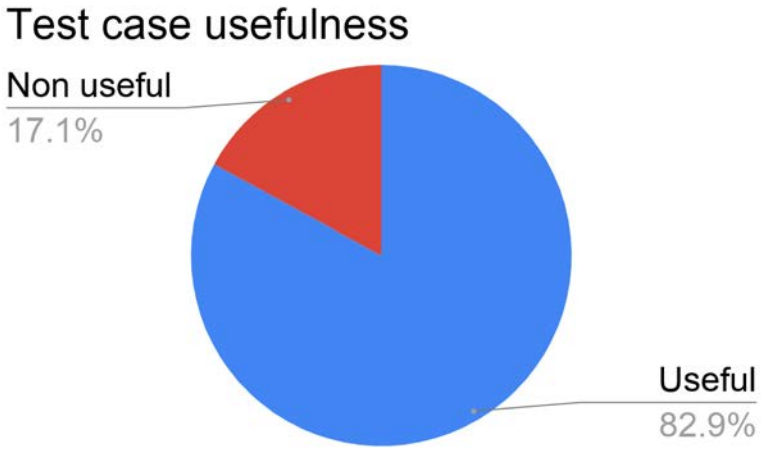
- Compare the EvoCrash to
  - STAR (covers all 50 crashes)
  - MuCrash (covers 12 crashes)
  - JCHARMING (covers 8 crashes)
- Comparison is made relying on the published data in the respective work.

Evaluation: Successful crash reproduction

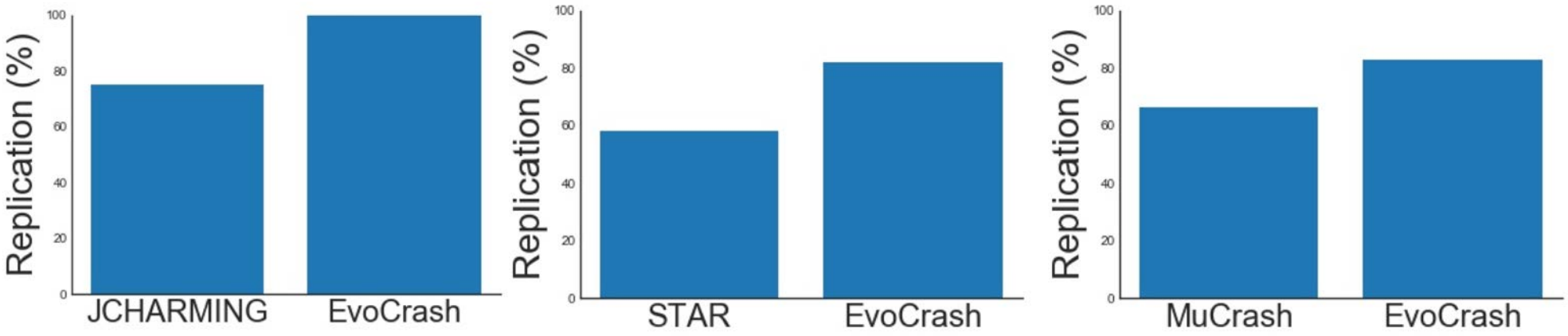
- Crash Coverage: Total of 41 crashes were reproduced out of 50 (82%).

	ACC	ANT	LOG
Evocrash	10/12 (83%)	14/20 (70%)	17/18 (94%)

- Test Usefulness



Evaluation : Comparison to the state-of-art



EvoCrash outperformed the state-of-art

Contributions

1. Guided genetic algorithm (GGA) for crash reproduction
    - Only generates/evolves tests that have at least one method involved in the failure (stack trace)
2. EvoCrash, Java tool that implements GGA
3. Empirical study on 50 real-world software crashes
    - EvoCrash replicates 82% of cases (82.9% of those useful for debugging)
4. Compared EvoCrash to three other state-of-the-art approaches

Discussion Questions

Consider the following fitness function:

$$f(t) = 3 \times d_s(t) + 2 \times d_{except}(t) + d_{trace}(t)$$

1. How did the authors come up with the coefficients in the fitness function? How could this heuristic be improved?

We suggest the the coefficients could be learned or the authors could do empirical studies to determine the robustness of results to changes in the coefficients.

Discussion Questions

2. What are limitations of genetic algorithms? Are there better search alternatives to genetic algorithms?

Genetic algorithms are limited by the amount of computation time one has.

Discussion Questions

3. The methodology in this paper centers on guiding test cases via the stack trace. How reliable are stack traces? What could be limitations of this reliance?

> Connecting this to Homework 1 where we were provided traces to help us debug, stack traces are not always the most useful for debugging and could distract focus from finding the true bug.

Discussion Questions

4. The guided genetic algorithm has a “max time” parameter; however, the authors do not describe how they select it. Clearly this threshold will give a time vs. recall tradeoff, but how should one select values for the parameter?

Vary the threshold and see how this affects accuracy on the test set.

Discussion Questions

5. The authors mention that for some cases with dependencies on external files they can generate successful test cases by increasing the population size. However, the authors concede they do not handle ALL crash cases with environmental dependencies. What are other ways that one could handle this?

External files can be provided with bug reports. However, this requirement is unlikely and has privacy concerns.