Search-Based Software Testing Tool Competition 2021

Sebastiano Panichella
Zurich University of Applied Science (ZHAW)

Alessio Gambi
Passau University

Fiorella Zampetti
University of Sannio

Vincenzo Riccio
University of Lugano
## History SBST Tool Competition

<table>
<thead>
<tr>
<th>Round</th>
<th>Year</th>
<th>Venue</th>
<th>Coverage Tool</th>
<th>Mutation Tool</th>
<th>#CUTs</th>
<th>#Projects</th>
<th>#Participants (+ baseline)</th>
<th>Statistical Tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Round 1</td>
<td>2013</td>
<td>ICST</td>
<td>Cobertura</td>
<td>Javalanche</td>
<td>77</td>
<td>5</td>
<td>2</td>
<td>✗</td>
</tr>
<tr>
<td>Round 2</td>
<td>2014</td>
<td>FITTEST</td>
<td>JaCoCo</td>
<td>PITest</td>
<td>63</td>
<td>9</td>
<td>4</td>
<td>✗</td>
</tr>
<tr>
<td>Round 3</td>
<td>2015</td>
<td>SBST</td>
<td>JaCoCo</td>
<td>PITest</td>
<td>63</td>
<td>9</td>
<td>8</td>
<td>✗</td>
</tr>
<tr>
<td>Round 4</td>
<td>2016</td>
<td>SBST</td>
<td>DEFECT4J (Real Faults)</td>
<td>PITest + Our Env.</td>
<td>68</td>
<td>5</td>
<td>4</td>
<td>✗</td>
</tr>
<tr>
<td>Round 5</td>
<td>2017</td>
<td>SBST</td>
<td>JaCoCo</td>
<td>PITest + Our Env.</td>
<td>69</td>
<td>8</td>
<td>2 (+ 2)</td>
<td>✓</td>
</tr>
<tr>
<td>Round 6</td>
<td>2018</td>
<td>SBST</td>
<td>JaCoCo</td>
<td>PITest + Our Env.</td>
<td>59</td>
<td>7</td>
<td>2 (+ 2)</td>
<td>✓ + combined analysis</td>
</tr>
<tr>
<td>Round 7</td>
<td>2019</td>
<td>SBST</td>
<td>JaCoCo</td>
<td>PITest + Our Env.</td>
<td>69</td>
<td>8</td>
<td>2 (+ 2)</td>
<td>✓ + combined analysis</td>
</tr>
<tr>
<td>Round 8</td>
<td>2020</td>
<td>SBST</td>
<td>JaCoCo</td>
<td>PITest + Our Env.</td>
<td>69</td>
<td>8</td>
<td>1 (+ 1)</td>
<td>✓ + combined analysis</td>
</tr>
</tbody>
</table>
SBST Tool Competition - 2021

What is New?

Java tool competition: As for recent years, we invite researchers to participate in the competition with their unit test generation tool for Java. Tools will be evaluated against a benchmark with respect to code coverage and mutation score.

Cyber-physical systems (CPS) testing competition: In addition to the traditional Java tool competition, we also organize a CPS testing competition on self-driving cars simulation environments. Specifically, in collaboration with the BeamNG research team (https://beamng.gmbh/research/), this competition focuses on the

- Generation of scenarios using BeamNG self-driving cars simulator
SBST Tool Competition - 2021

What is New?

Java tool competition: As for recent years, we invite researchers to participate in the competition with their unit test generation tool for Java. Tools will be evaluated against a benchmark with respect to code coverage and mutation score.

Cyber-physical systems (CPS) testing competition: In addition to the traditional Java tool competition, we also organize a CPS testing competition on self-driving cars simulation environments. Specifically, in collaboration with the BeamNG research team (https://beamng.gmbh/research/), this competition focuses on the Generation of scenarios using BeamNG self-driving cars simulator.

10 Tools Participating to the Competition

- Five times more tools than last 2020!!!
- 2 Tools from Industrial Organizations!!!
SBST Tool Competition - 2021

Co-chairs 2021

Sebastiano Panichella
Zurich University of Applied Science (ZHAW)

Fiorella Zampetti
University of Sannio

Alessio Gambi
Passau University

Vincenzo Riccio
University of Lugano

Java tool competition

Cyber-physical systems (CPS) testing competition
SBST Tool Competition - 2021

Co-chairs 2021

Sebastiano Panichella
Zurich University of Applied Science (ZHAW)

Fiorella Zampetti
University of Sannio

Alessio Gambi
Passau University

Vincenzo Riccio
University of Lugano

Java tool competition

Cyber-physical systems (CPS) testing competition
Java tool competition Infrastructure

JUnit Tool Contest Infrastructure

Randoop

runtool

....

runtool

Tool

runtool

CUT

Time budget

@Test

@Test

@Test
JUnit Tool Contest Infrastructure

- Randoop
- runtool
- Tooln

CUT

Time budget

@Test

Filtering Flaky Tests

@Test

pitest.org

JACOCO
Java Code Coverage

Metrics
Scoring Formula

\[ \text{covScore}(T, B, C, R) = 1 \times \text{Cov}_i + 2 \times \text{Cov}_b + 4 \times \text{Cov}_m \]

\[ \text{tScore}(T, B, C, R) = \text{covScore}(T, B, C, R) \times \min \left( 1, \frac{2 \times B}{\text{genTime}} \right) \]

\[ \text{Score}(T, B, C, R) = \text{tScore}(T, B, C, R) + \text{penalty}(T, B, C, R) \]

- \( T \) = Generated Test
- \( B \) = Search Budget
- \( C \) = Class under test
- \( R \) = independent Run

- \( \text{Cov}_i \) = statement coverage
- \( \text{Cov}_b \) = branch coverage
- \( \text{Cov}_m \) = Strong Mutation

- \( \text{getTime} \) = generation time

- \( \text{penalty} \) = percentage of flaky test and non-compiling tests
Class Under Test (CUT)

class Triangle {
    int a, b, c; // sides
    String type = "NON_TRIGANGLE";
    Triangle (int a, int b, int c){...}
    void computeTriangleType() {
        1. if (a == b) {
            2. if (b == c)
                3. type = "EQUILATERAL";
            else
                4. type = "ISOSCELES";
        } else {
            5. if (a == c) {
                6. type = "ISOSCELES";
            } else {
                7. if (b == c)
                8. type = "ISOSCELES";
                else
                    9. type = "SCALER";
            }
        }
    }
}

Test Case

@Test
public void test(){
    // Constructor (init)
    // Method Calls
    // Assertions (check)
}

@Test
public void test(){
    Triangle t = new Triangle (1,2,3);
    t.computeTriangleType();
    String type = t.getType();
    assertTrue(type.equals("SCALER"));
}
Benchmark Projects

• **Selection criteria**
  • GitHub repositories
  • Project builds using Maven or Gradle
  • Contains JUnit 4 test suite

• **6 projects selected**

<table>
<thead>
<tr>
<th>Guava</th>
<th>Seata</th>
<th>Okio</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="https://github.com/google/guava">GitHub</a></td>
<td><a href="https://github.com/seata/seata">GitHub</a></td>
<td><a href="https://github.com/square/okio">GitHub</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Spoon</th>
<th>FastJSON</th>
<th>Weka</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="https://github.com/INRIA/spoon/">GitHub</a></td>
<td><a href="https://github.com/alibaba/fastjson">GitHub</a></td>
<td><a href="https://github.com/Waikato/weka-3.8">GitHub</a></td>
</tr>
</tbody>
</table>
## Contest Methodology

<table>
<thead>
<tr>
<th>Search budgets</th>
<th>Classes under test</th>
<th>Repetitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>30 seconds</td>
<td>98 classes</td>
<td>10 repetitions</td>
</tr>
</tbody>
</table>

### Execution environment
- ![Docker logo]

### Statistical analysis
- Friedman’s test
- Post-hoc Conover
The Tools

Baseline

Randoop
Automatic unit test generation for Java

V.S.

Competitors

EVA SUITE

UtBot

Kex

EVA SUITE - DSE
Results (1)

Fig. 1: Line Coverage for Randoop, Evosuite(DSE), Utbot, Kex and Evosuite for 30 and 120 seconds.

Fig. 2: Branch Coverage for Randoop, Evosuite(DSE), Utbot, Kex and Evosuite for 30 and 120 seconds.
Results (2)

Fig. 3: Mutant Coverage for Randoop, Evosuite (DSE), Utbot, Kex and Evosuite for 30 and 120 seconds.

Fig. 4: Coverage for Evosuite (DSE), Utbot and Evosuite on a time budget of 5 minutes.
Final Ranking

Baseline

V.S.

Competitors

1

Evosuite

UtBot

Kex

Evosuite - DSE
Lessons Learnt

- Identified **aspects** to **improve** and **bugs** that could be **fixed** in the infrastructure

- Docker **simplifies** the evaluation procedure
  - **More participants to the competition!**
    - From Academia & Industry
What’s Next?

• Contest Infrastructure
  • https://github.com/JUnitContest/junitcontest
  • Improve usability
    • Facilitate setup of an evaluation
    • Facilitate evaluation in other contexts
    • Update the user documentation
  • Storage and versioning of the results (and participating tools?)

• For the next edition
  • More tools
  • More CUTs
  • **Python as new language to experiment!**
SBST Tool Competition - 2021

Co-chairs 2021

Sebastiano Panichella
Zurich University of Applied Science (ZHAW)

Fiorella Zampetti
University of Sannio

Alessio Gambi
Passau University

Vincenzo Riccio
University of Lugano

Java tool competition

Class Under Test (CUT)

```java
public class Triangle {  
  static double a, b, c;  
  public void computeTriangleType() {  
    if (a > b)  
      if (a > c)  
        switch (a) {  
          case "ACUT":  
            System.out.println("ACUT");  
            break;  
          case "BOCA":  
            System.out.println("BOCA");  
            break;  
          case "BCUNCTION":  
            System.out.println("BCUNCTION");  
            break;  
          case "ABC":  
            System.out.println("ABC");  
            break;  
          default:  
            System.out.println("Invalid");  
        }  
    }  
  }  
}  
```

Test Case

```java
@Test  
public void test(){  
  Triangle t = new Triangle();  
  t.computeTriangleType();  
  System.out.println("The triangle is of type "+t.getType());  
}  
```

Cyber-physical systems (CPS) testing competition
Cyber-physical systems (CPS) testing competition: In addition to the traditional Java tool competition, we also organize a CPS testing competition on self-driving cars simulation environments. Specifically, in collaboration with the BeamNG research team (https://beamng.gmbh/research/), this competition focuses on the

- Generation of scenarios using BeamNG self-driving cars simulator

Figure 2: Example of CPS testing tool simulation environment.
A Dream Come True
Friday briefing: Uber's self-driving software was responsible for pedestrian fatality

Crash investigators have found that disabled features and poor object identification led to the killing of a pedestrian by one of Uber's autonomous vehicles, vast canyons are buried in the ice between Antarctica's mountains

By WIRED
25 May 2018
Testing Self-Driving Cars

Time-consuming
Limited realism
Impractical
Simulation-based Testing

BeamNG.drive

REVIEW

“The Most Impressive Physics Engine You’ve Never Seen”
IGN

“BeamNG’s Amazingly Realistic Car Crashes”
Gameinformer

“Amazing Car Crashes + Hilarious Greenlight Trailer = Magic”
Kotaku

CUSTOMER REVIEWS

Overall Reviews:
Overwhelmingly Positive (40,939 reviews)
Lane Keeping Assist System
What is a Test Case?

- Start = (10.0, 20.0)
- A = (30.0, 20.0)
- B = (40.0, 30.0)
- C = (50.0, 40.0)
- D = (150.0, 100.0)
- End = (30.0, 180.0)
What is a Failure?

Out of Bound Episode
What is a Failure?

Out of Bound Episode
What is a Failure?

Out of

Bound

Episode

% car outside the lane > threshold
Infrastructure

Tools (Test Generators)

Test Case

Simulation Report

Simulation Data

Test Scenario

Driving Simulator

CPS Tool Contest Code Pipeline

Results

Budget

Max Speed

OBE Tolerance

Map Size

Infrastructure

CPS Tool Contest Code Pipeline
Metrics: # OBEs

- # Failure-inducing generated test cases

# OBEs = 3
Metrics: Failure Sparseness

- Average maximum distance of the relevant road sectors
Metrics: Efficiency and Effectiveness

- Number of generated **valid** test cases within the time budget

- A valid road should:
  - not self-intersect
  - not contain overly sharp turns
  - be fully contained in the specified map
# Contest Methodology

<table>
<thead>
<tr>
<th>Test Subject</th>
<th>Default</th>
<th>SBST21</th>
</tr>
</thead>
<tbody>
<tr>
<td>BeamNG AI</td>
<td>Search budget: 5 hours</td>
<td>Search budget: 2 hours</td>
</tr>
<tr>
<td>Driving Simulator</td>
<td>Map Size: 200 X 200</td>
<td>Map Size: 200 X 200</td>
</tr>
<tr>
<td>BeamNG tech</td>
<td>Max Speed: None</td>
<td>Max Speed: 70 km/h</td>
</tr>
<tr>
<td></td>
<td>OBE Tolerance: 0.95</td>
<td>OBE Tolerance: 0.85</td>
</tr>
</tbody>
</table>
Competitors

- Deeper (MDH+RISE+HSU)
- Frenetic (NII)
- GABExplore (TU Graz)
- GABExploit (TU Graz)
- Swat (PolyMtI)
Results: # OBEs

![Box plot comparing # OBEs across different settings: Default and SBST21. The box plot shows the distribution of # OBEs with quartiles and outliers for each setting. Each setting has a different color and is labeled as: Deeper, Frenetic, GABExploit, GABExplore, and Swat. The x-axis represents the settings (Default and SBST21), and the y-axis represents the number of OBEs.]
Results: Failure Sparseness

![Box plot showing sparseness comparison between Default and SBST21 with different strategies: Deeper, Frenetic, GABExploit, GABExplore, Swat.](image)
Results: Efficiency and Effectiveness

The diagram shows the number of valid and invalid test cases for different tools: Deeper, Frenetic, GABExploit, GABExplore, and Swat.

- **Default** vs. **SBST21**
  - Green bars represent valid test cases.
  - Blue bars represent invalid test cases.

- **# Valid Test Cases**
- **# Invalid Test Cases**
Towards Interpretable Failures

- Evaluate test input generators for self-driving software using interpretable feature maps (e.g., map coverage)

DEEPHYPERION: Exploring the Feature Space of Deep Learning-Based Systems through Illumination Search

Tabereh Zohbaninashab
Università della Svizzera Italiana
Lucerne, Switzerland
tabereh.zohbaninashab@usi.ch

Alessio Gambi
University of Passau
Passau, Germany
alessio.gambi@uni-passau.de

Vincenzo Riccio
Università della Svizzera Italiana
Lucerne, Switzerland
vincenzo.riccio@usi.ch

Paolo Tonella
Università della Svizzera Italiana
Lucerne, Switzerland
paolo.tonella@usi.ch

ABSTRACT

Deep Learning (DL) has been successfully applied to a wide range of application domains, including safety-critical ones. Several DL testing approaches have been recently proposed in the literature, but none of them aims to assess how different interpretable feature sets of the generated inputs affect the system's behavior.

In this paper, we resort to IlluminationSearch to find the highest-performing test cases (i.e., misbehaving and closest to misbehavior), exploiting the cells of a map representing the feature space of the system. We introduce a methodology that guides the users of our approach in the task of identifying and quantifying the dimensionality of the feature space for a given domain. We developed, Illumina, a research tool for DL systems that illuminates, i.e., presents a path of exploration of the feature space, through the visualization of the behavior of the system.

1 INTRODUCTION

Deep Learning (DL) has become an essential component of complex software systems, including autonomous vehicles and medical diagnosis systems. As a consequence, the problem of ensuring the reliability of these systems is relevant.

Unlike traditional software, in which developers explicitly program the system's behavior, the complexity of DL systems is that they mimic the human ability to learn how to perform a task from training examples. Therefore, it is essential to understand in which extent they can be trusted in response to the diversity of inputs they will process once deployed in the real world, as they could face scenarios that might be not sufficiently represented in the data from which they have learned [1].
Lessons Learnt

• Adopt open infrastructure and intuitive APIs

• Involve and grow the community

• Pull requests are welcome: https://github.com/se2p/tool-competition-av

• Join the discussion on: https://join.slack.com/t/droversity
What’s Next?

• New test subjects
  • Learning-Based driving agents
  • Path/Trajectory planners

• Training test subjects based on competitors to avoid (representation) bias

• Larger test space/new driving tasks:
  • Environment, weather, 3D roads
  • Obstacles, traffic

• "Open" submission (continuous evaluation)
Search-Based Software Testing Tool Competition 2022

Co-chairs 2021

Sebastiano Panichella
Zurich University of Applied Science (ZHAW)

Alessio Gambi
Passau University

Fiorella Zampetti
University of Sannio

Vincenzo Riccio
University of Lugano

Co-chairs 2022

Fiorella Zampetti
University of Sannio

Vincenzo Riccio
University of Lugano

Co-chair(s): You? Co-chair(s): You?