Coverage Criteria for Search Based Automatic Unit Testing of Java Programs

Ina Papadhopulli, Elinda Meçe

Polytechnic University of Tirana

DAAD: 15th Workshop “Software Engineering Education and Reverse Engineering”

Outline

- Unit test case generation
- Evosuite tool
- Coverage criteria
- Combination of multiple coverage criteria
- Experiments and results
- Conclusions

Automatic Unit Test Case Generation

- Two important techniques:
  1. *Search-Based Software Testing (SBST)*
  2. *Symbolic Execution (SE)*

- SBST: meta heuristics approach
- Genetic algorithm: most used optimization method of SBST

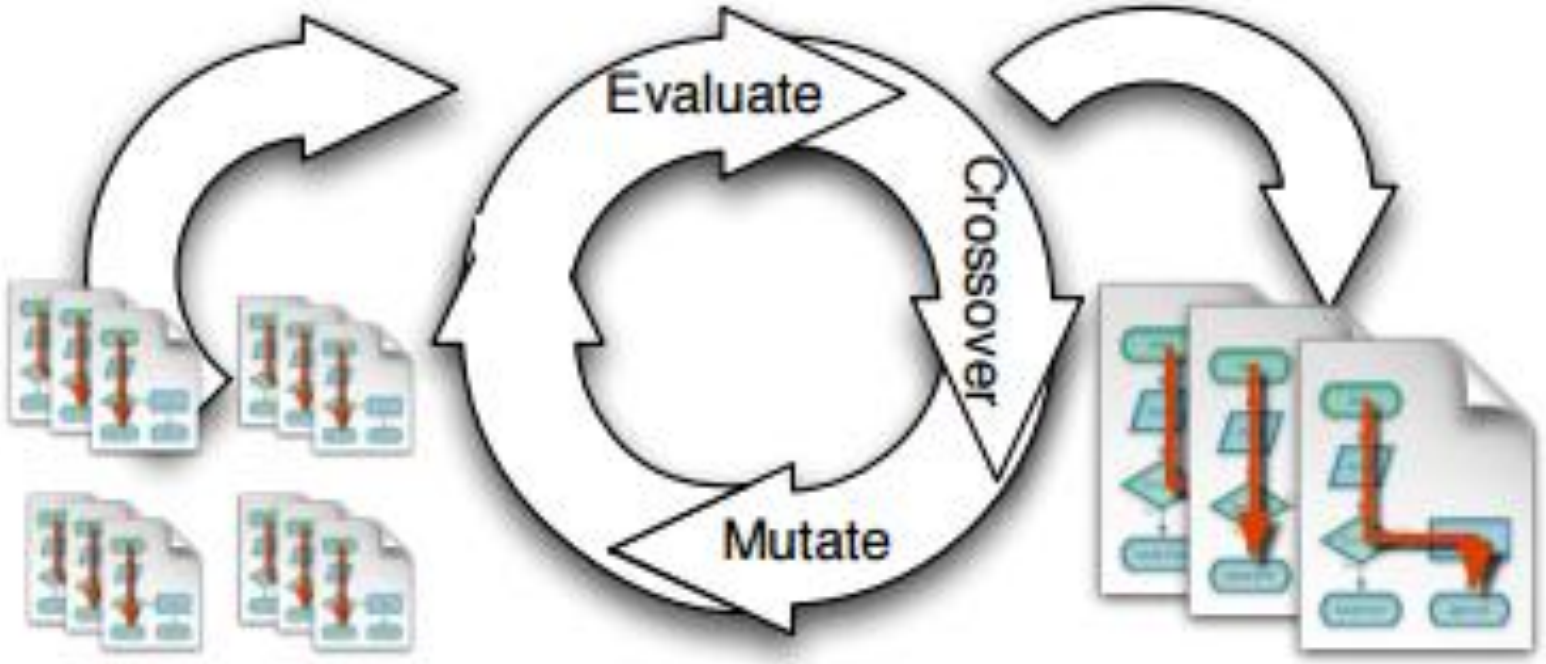
Test Suite of Java Programs

- A test suite **T** is a set of test cases **t**: $T = \{t_1, t_2, \ldots, t_n\}$
- Test case **t** in unit testing**:** program that executes the CUT
- Test case: sequence of statements $t = \{s_1, s_2, \ldots, s_m\}$
- The statements for Java test cases are:
  1. *Primitive*
  2. *Constructor*
  3. *Method*
  4. *Field*
  5. *Assignments*

Evosuite

- Tool for automatic unit testing of Java programs
- Usage: Command Line or Eclipse plugin
- Input: Bytecode of the target class
- Output: JUnit test cases
- Technology: Genetic Algorithms
- Open Source
- www.evosuite.org
- Uses **whole test suite** generation: optimizes entire test suites towards satisfying all goals at the same time

SBST- Genetic Algorithms



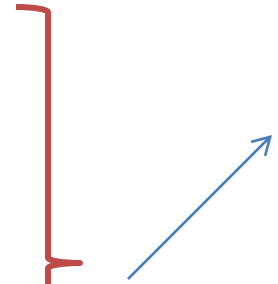
Random Test Suites *Genetic Algorithm* *Final Test Suite*

Coverage Criteria

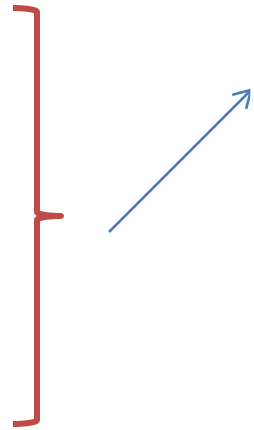
1. Line Coverage
2. Branch Coverage
3. Direct Branch Coverage
4. Output Coverage
5. Weak Mutation
6. Exception Coverage
7. Top-level Method Coverage
8. No-exception Top-level Method Coverage

Coverage Criteria

1. Line Coverage
2. Branch Coverage
3. Direct Branch Coverage
4. Output Coverage
5. Weak Mutation
6. Exception Coverage
7. Top-level Method Coverage.
8. No-exception Top-level Method Coverage



Fitness function with guidance



No guidance fitness function

Combination of coverage criteria

- Using more than one criterion to guide the search.
- All criteria considered here are non-conflicting
- Fitness function of combined coverage criteria: linear combination of each fitness function
- How does search-based testing scale to combinations of multiple criteria?

Research Questions

- *RQ1: How does the combination of multiple criteria affect the coverage of each criterion?*
- *RQ2: How does the combination of all the criteria affect the mutation score of the suite?*
- *RQ3: Which of the criteria (except Weak Mutation) used separately achieves the highest mutation score?*
- *RQ4: How does the number of mutants of the CUT affect the mutation score?*
- *RQ5: How does multiple criteria affect the number of suite's test cases and their size?*
- *RQ6: How does the adding of weak mutation criterion affect the size of the test cases?*

Experiment/System characteristics

Operating System: Linux 32 bit

Memory: 1 GB

Processor: Intel Core 2 Duo CPU E7400 2.8GHz x 2

Experiment/Evosuite Usage

- Run from the command line
- Eclipse plugin is not currently available for Linux OS

Experiment/Subject Selection

- Open source software

Project Name	**No. of Classes**	**Source**
MathPareser	48	SourceForge
MathQuickGame	25	SourceForge
java.util.Regex	92	jdk 1.8.0/src
Refactoring	87	SourceForge
Library	22	CodeCreator.org
StudentManagementSystem	24	CodeCreator.org
Total	**298**	

Experiment/Subject Selection

- 120 classes are chosen randomly
- Each experiment is repeated 5 times to overcome the randomness of the genetic algorithms.
- For 6 classes, Evosuite quits without generating any output.
- The results are in many cases affected by the *security manager* of Evosuite (restrict test execution)

Experiment/Parameters of GA

- Population size: 50 test suites
- Chromosome length: 40 test cases
- Probability of mutation: 0.75
- Probability of crossover: 0.75
- Search budget: 1min/5min

Research Question 1

- **RQ1: How does the combination of multiple criteria affect the coverage of each criterion?**

Experiment: For each of the classes we run Evosuite with the configurations:

1. All criteria with search budget of 1 min
2. All criteria with search budget of 5 min
3. Each criterion separately with search budget of 5 min

Experiment/Results

CRITERION	ALL 1min)	ALL (5 min)	Only one (5 min)
LINE	59.7	60.7	61.1
BRANCH	52.6	53.4	53.5
EXCEPTION	83.7	83.9	83.9
WEAK MUTATION	62.4	69.7	74.3
OUTPUT	47.9	48.2	48.6
TOP-LEVEL	93.4	93.8	81
METHOD-NOEXCEPTION	89.6	90.7	80.5
DIRECT BRANCH	49.8	53.1	52.8

Table 1: Coverage results for each configuration, average of all runs for all CUTs

- RQ1: For criteria whose fitness guides the search, the performance between the combination of all criteria and each criterion separately converges nearly to the same value. For criteria with low guidance the ALL - combination increases the performance.

Mutation Criterion

- It is the gold criterion in unit testing for predicting suite quality by researchers.
- It is difficult to apply and computationally expensive.
- Problems:
  1. The number of mutants for a given system can be huge
  2. Equivalent mutants

Research Question 2/3

- **RQ2: How does the combination of all the criteria affect the mutation score of the suite?**
- **RQ3: Which of the criteria (except Weak Mutation) used separately achieves the highest mutation score?**

Experiment: For each of the classes we run Evosuite with the configurations:

1. All criteria with search budget of 5 min
2. All criteria (except Weak Mutation) with search budget of 5 min
3. Each criterion separately with search budget of 5 min

Experiment/Results

CRITERION	**Mutation Score**
LINE	**15.6**
BRANCH	**25.8**
EXCEPTION	**0.2**
WEAK MUTATION	**28.3**
OUTPUT	**16.6**
TOP-LEVEL	**23**
METHOD-NOEXCEPTION	**15.3**
DIRECT BRANCH	**21.2**
All (without Weak Mutation)	**24.5**
All (5 min)	**26.1**

Table 2: Mutation scores for each configuration, average of all runs for all CUTs with search budget of 5 min

- RQ2: Given enough time the combination of all criteria achieves higher mutation score than each criterion separately (except Weak Mutation).
- RQ3: In our experiments the next best criterion to achieve high mutation scores is branch coverage.

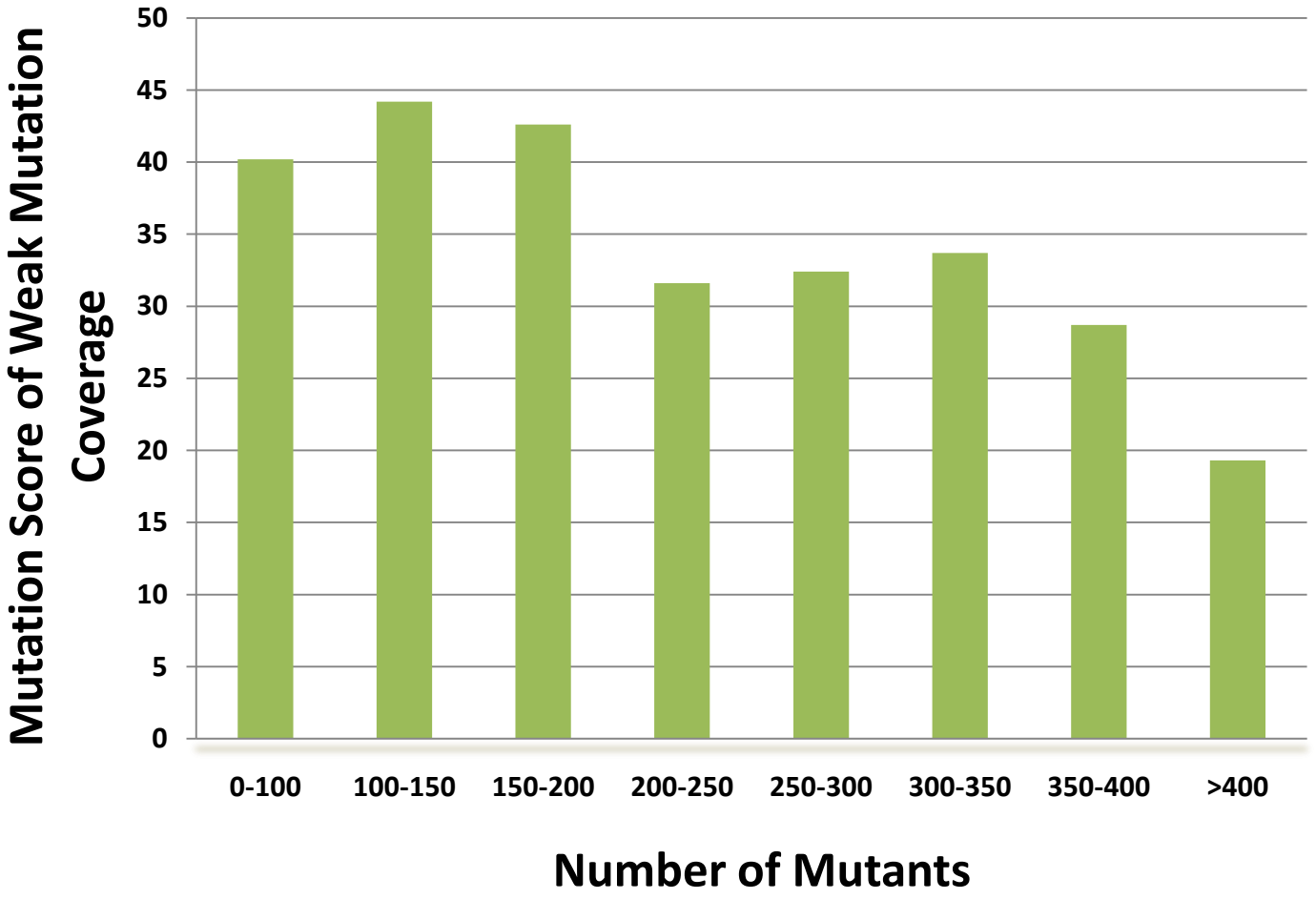
Research Question 4

- **RQ4: How does the number of mutants of the CUT affect the mutation score?**

Experiment: For each of the classes we run Evosuite with the configuration:

1. Weak Mutation criterion with search budget of 5 min

Experiment/Results



RQ4: Given the limited budget the number of mutants in the CUT affects the mutation scores achieved by Evosuite.

Research Question 5/6

- Automatically generated JUnit tests need to be manually checked in order to detect faults.

- **RQ5: How does multiple criteria affect the size of the test cases?**
- **RQ6: How does the adding of weak mutation criterion affect the size of the test cases?**

- Size of a test case: the number of statements after the minimization (without assertions)

Research Question 5/6

Experiment: For each of the classes we run Evosuite with the configurations:

1. All criteria with search budget of 5 min
2. All criteria (except Weak Mutation) with search budget of 5 min
3. Each criterion separately with search budget of 5 min

(In 35 runs of Evosuite the minimization phase timeout)

Experiment/Results

CRITERION	Coverage	Mutation Score	Test suite size
LINE	**61.1**	**15.6**	**14.2**
BRANCH	**53.5**	**25.8**	**15.7**
EXCEPTION	**83.9**	**0**	**6.7**
WEAK MUTATION	**74.3**	**28.3**	**19.4**
OUTPUT	**48.6**	**16.6**	**10**
TOP-LEVEL	**81**	**23**	**10.3**
METHOD-NOEXCEPTION	**80.5**	**15.3**	**12.4**
DIRECT BRANCH	**52.8**	**21.2**	**10.2**
All (without Weak Mutation)	**59.7**	**24.5**	**22.3**
All (5 min)	**69.3**	**26.1**	**27.8**

- Table 3: Suite size for each configuration, average of all runs for all CUTs with search budget of 5 min

- RQ5: Using all the criteria increases the test suite size by more than 50% that the average test suite size of each constituent criterion used separately.
- RQ6: Adding weak mutation criterion increases the test suite size approximately with 20%.

Conclusions

- Using multiple criteria increases the overall coverage and mutation score with the cost of a considerable increase in test suite length.
- A lot of improvements need to be made to enable the usage of automation for unit testing support.
- To increase the coverage there is still necessary to find different fitness functions or to adapt them during optimization.
- There is the need to further investigate the conditions that make certain problems more difficult to be optimized with a genetic algorithm.

“Coverage Criteria for Search Based Automatic Unit Testing of Java Programs”

Thank You!

Questions?

Suggestions?