# Evosuite

Search-based testing tool

http://www.evosuite.org

# Content

- Overview
- Evosuite Process
- Genetic Algorithm
- Demo

# Overview Evosuite




# Overview

- Produce coverage test suites for Java classes fully automatically
- Work with file bytecode (.class or .jar)
- Test suites, that achieve high code coverage, are as small as possible
- Use search-based approach integrating with other techniques such hybrid-search...

# Evosuite Process

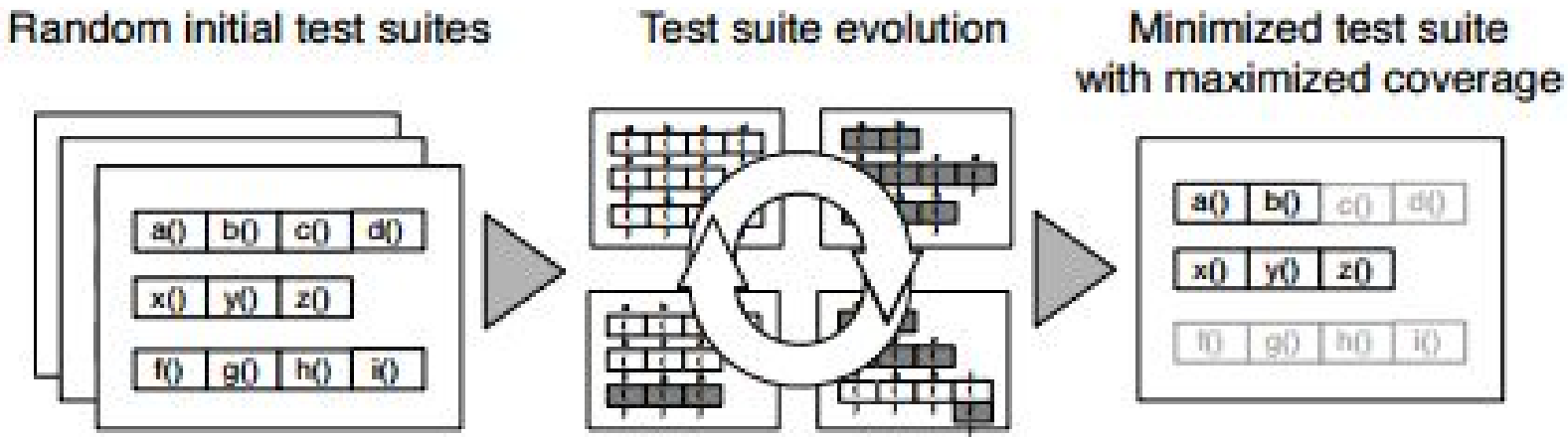



Fig. 2. The EvoSuite process: A set of randomly generated test suites is evolved to maximize coverage, and the best result is minimized.

# Search-based testing

- Handling dependencies among predicates
- Handling test case length dynamically without applying exploration impeding constraints
- Giving guidance towards reaching test goals in private function

# Test suites optimization

- Use search algorithm, namely a Genetic Algorithm(GA)
- GA is applied on a population of test suites

# Genetic Algorithm

Input: a random population

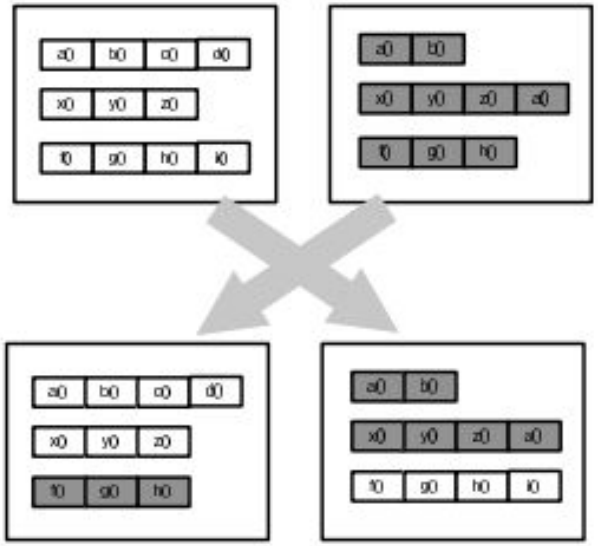
End: a solution is found that fulfills the coverage criterion or allocated resource (time, number of fitness evaluations)

- Rank selection
- Crossover
- Mutation (remove, insert, change)
- Fitness

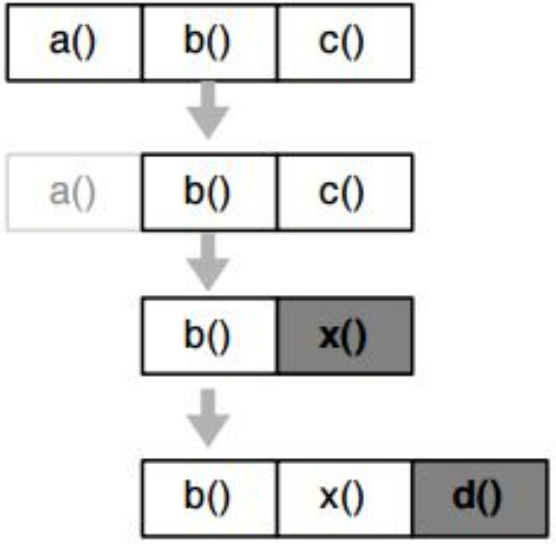
**Algorithm 1** The genetic algorithm applied in EvoSuite

```
1  current_population ← generate random population
2  repeat
3     Z ← elite of current_population
4     while |Z| ≠ |current_population| do
5        P1,P2 ← select two parents with rank selection
6        if crossover probability then
7           O1,O2 ← crossover P1,P2
8        else
9           O1,O2 ← P1,P2
10       mutate O1 and O2
11       fP = min(fitness(P1),fitness(P2))
12       fO = min(fitness(O1),fitness(O2))
13       lP = length(P1) + length(P2)
14       lO = length(O1) + length(O2)
15       TB = best individual of current_population
16       if fO < fP ∨ (fO = fP ∧ lO ≤ lP) then
17          for O in {O1,O2} do
18             if length(O) ≤ 2 × length(TB) then
19                Z ← Z ∪ {O}
20             else
21                Z ← Z ∪ {P1 or P2}
22       else
23          Z ← Z ∪ {P1,P2}
24    current_population ← Z
25 until solution found or maximum resources spent
```

# Genetic Algorithm



(a) Crossover




(b) Mutation

Fig. 4. Crossover and mutation are the basic operators for the search using a GA. Crossover is applied at test suite level, while mutation is applied to test cases and test suites.

# Demo

- Command line

- Eclipse