FONDAZIONE BRUNO KESSLER

SE SOFTWARE ENGINEERING

Test Generation as a Many-objective Optimization Problem

Fitsum KIFETEW
kifetew@fbk.eu
Fondazione Bruno Kessler

Joint work with:
Annibale Panichella and Paolo Tonella

GAUSS Kick-off
February, 2017

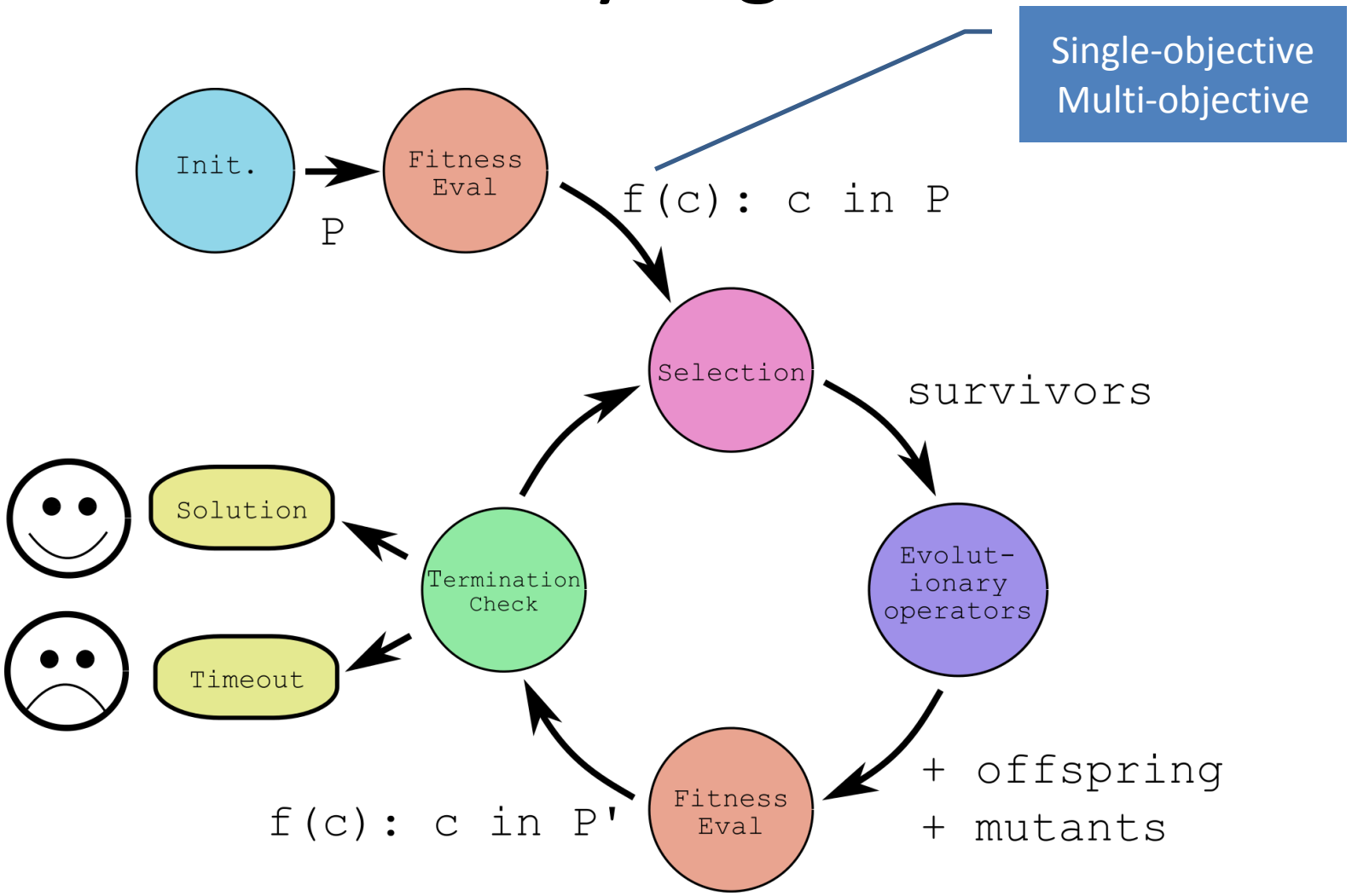
Context

- Automated test case generation
- *for* structural code coverage
- *of* object-oriented programs
- *using* search-based techniques (SBST)

Search-Based Software Testing (SBST)

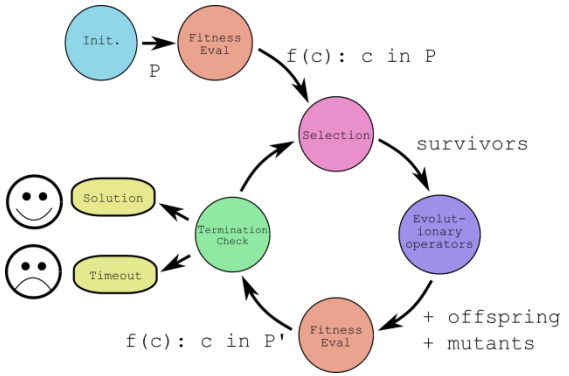
- Testing as optimization (search) problem
  - Minimize/maximize objective functions
- Metaheuristic (search) algorithms
  - e.g., Hill climbing, Evolutionary Algorithms, etc.
- Evolutionary algorithms
  - Genetic Algorithms

Evolutionary Algorithms

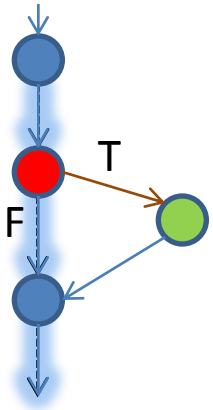


Test Case Generation

- Define a suitable
  - Encoding
    - Individuals as ***test cases***
  - fitness function (minimizing a distance)
    - Approach level + Branch distance




```
public class A {
  …
  void m1 (int x, int y){
     …
     if (x == y){
        // target
     } …
  }
}
```



tc1

```
A A1 = new A();
int x = 5, y = 2;
A1.m1(x, y);
```

al = 0
bd = |x-y| = 3
f(tc1) = 0 + 0.75 = 0.75

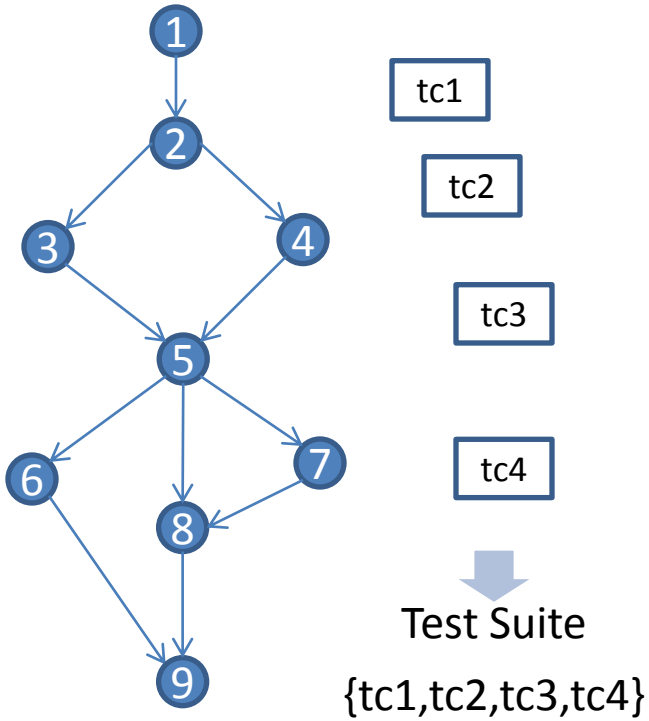
tc2

```
A A1 = new A();
int x = 3, y = 2;
A1.m1(x, y);
```

bd = |x-y| = 1
fitness = 0 + 0.5 = 0.5

Single Target

- Several targets in SUT
- Optimize for *one target at a time*



Single Target

- Search is more focused
  - Optimize only for a single branch
- Issues
  - Search budget (re)allocation
  - Accidental coverage
  - Infeasible branches
- Several works based on this approach [McMinn 2004]

WholeSuite

- Optimize towards *all targets simultaneously*
  - Individuals are ***Test Suites*** (set of test cases)
  - Test suite level operators
  - Fitness: sum of all branch distances

$$\min f_U(T) = \sum_{u \in U} d(u, T)$$

- Not affected by infeasible branches
- Search is less focused (aggregates all distances)

Problem Formulation

Let U = {u_1, …, u_k} be the **uncovered targets** in SUT,

find a set of non-dominated test cases

T = {t_1, …, t_n} that minimizes the following ***k*** objectives:

$$\begin{cases} \min f_1(t) = d(u_1, t) \\ \vdots \\ \min f_k(t) = d(u_k, t) \end{cases}$$

A fitness vector <f_1, …, f_m> for a Test Case ***t***

Pareto Optimality

A test case **x** dominates a test case **y** ($x \prec y$) iff :

$\forall i \in \{1, \ldots, k\}\ f_i(x) \leq f_i(y)$

and

$\exists j \in \{1, \ldots, k\}$ such that $f_j(x) < f_j(y)$

A test case $\boldsymbol{x}^*$ is ***Pareto optimal*** if and only if it is not dominated by any other test case.

Existing Algorithms

- traditional multi-objective EA (eg. NSGA-II) are not effective with > 3 objectives
- improvements: ϵ-dominance relation (ϵ-MOEA), IBEA, GrEA, POGA, θ-NSGA-III
- investigated for optimization problems with < 15 objectives
- ***number of non-dominated solutions*** increases
- designed to produce ***a rich set of tread-offs***

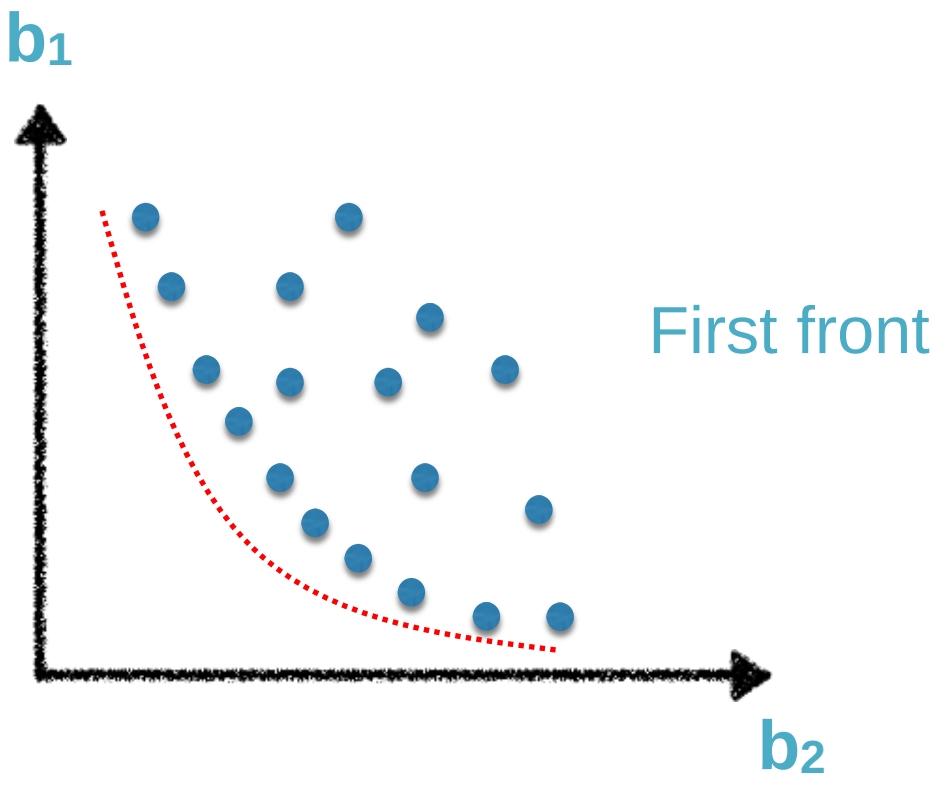
Many Objective Sorting Algorithm (MOSA)

- **scalability** to a large number of objectives is important (thousands)
- not all Pareto optimal test cases are useful, focus only on subset of the Pareto optimal set
  - focus on test cases that are closer to one or more uncovered branches
- for equal coverage, shorter test cases are preferred

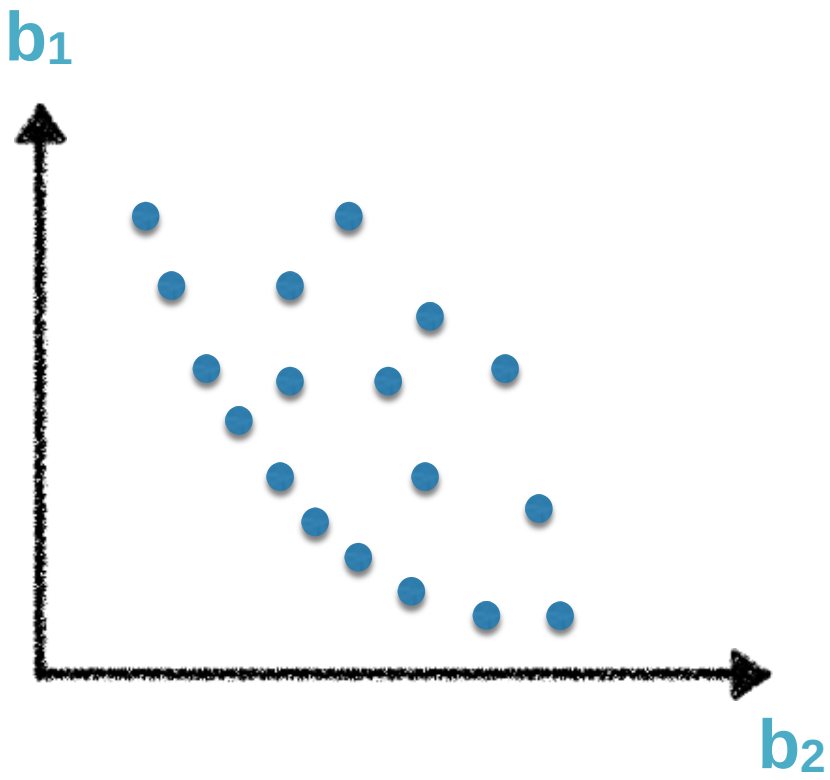
Example: Branch Coverage

```
int test (int a, int b, int c) {
    if (a == b)
        return 1; // uncovered: b1
    if (b == c)
        return -1; // uncovered: b2
    return 0;
}
```

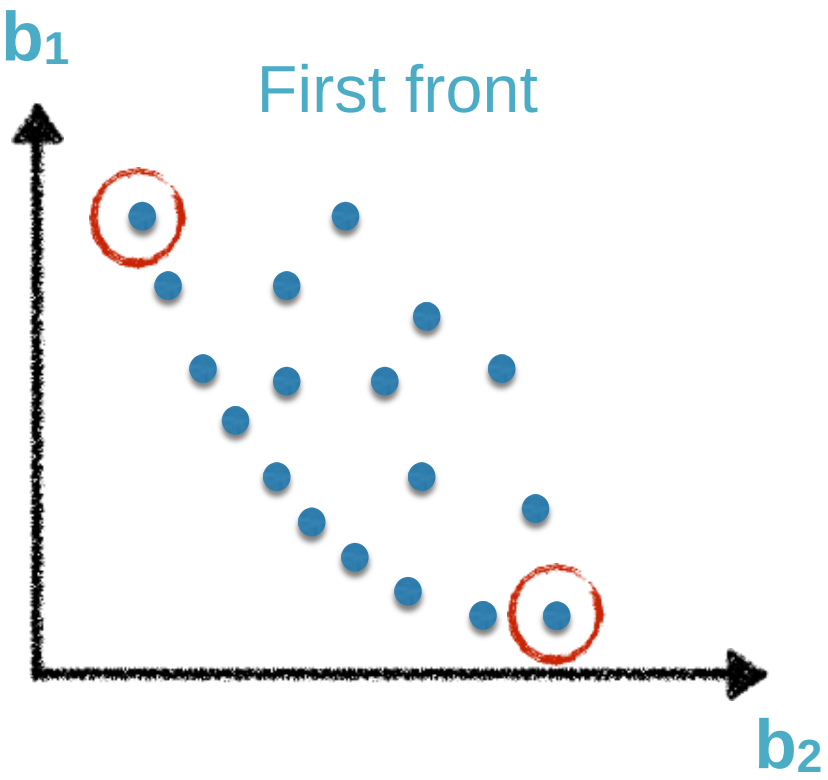
Traditional Ranking



MOSA

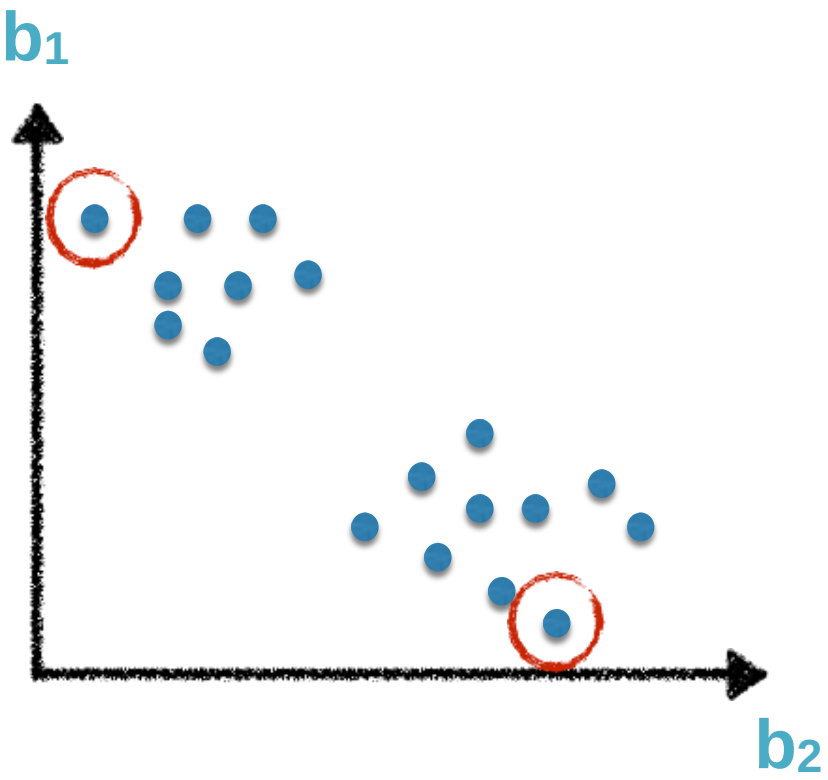


MOSA



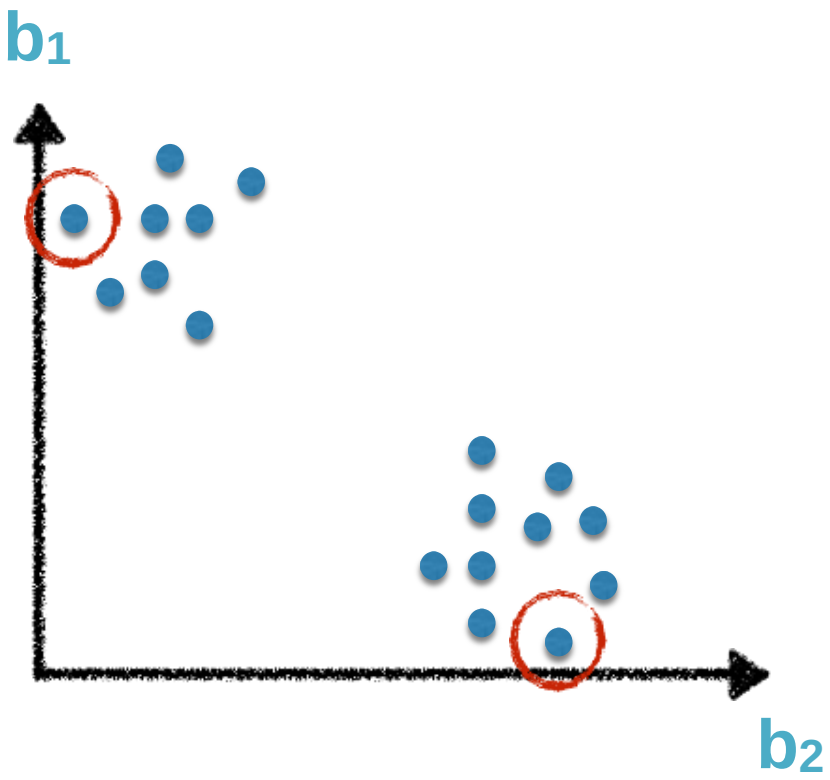
Selecting only a subset of the Pareto Front

MOSA



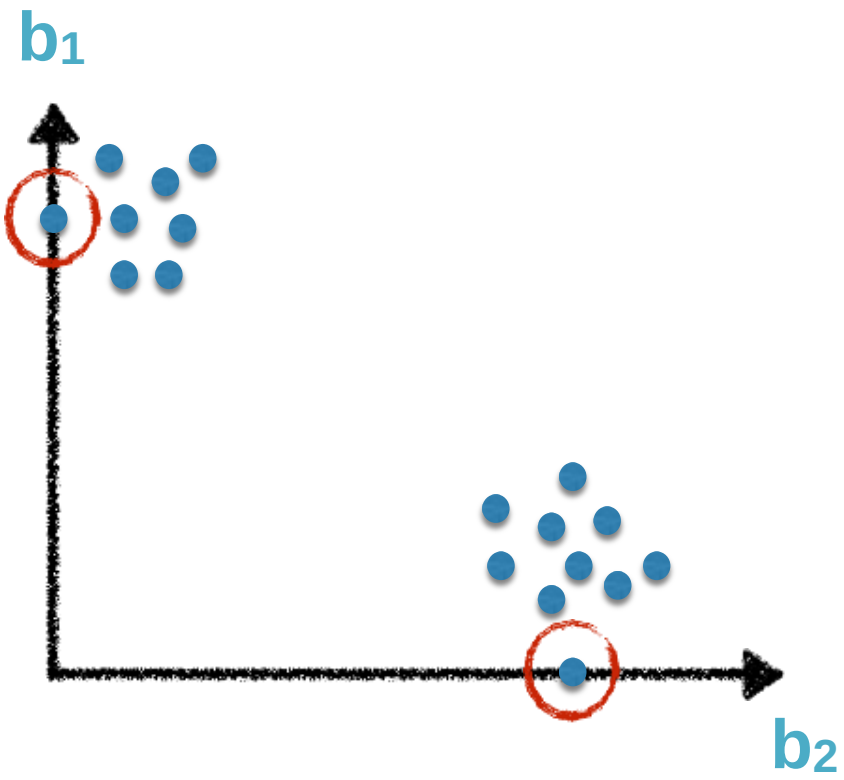
Selecting only a subset of the Pareto Front

MOSA



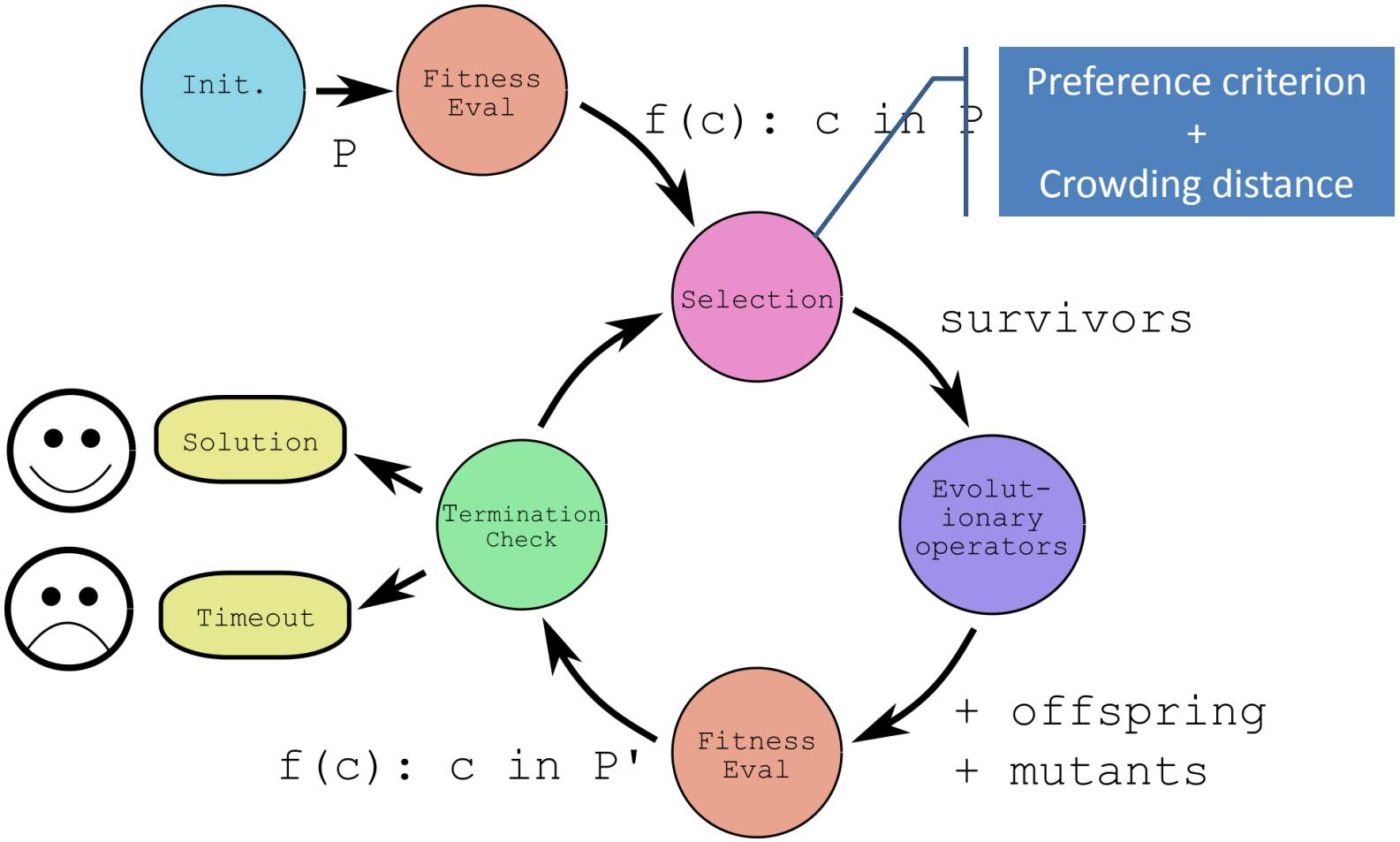
Selecting only a subset of the Pareto Front

MOSA



Selecting only a subset of the Pareto Front

MOSA


Init.
Fitness Eval
P
f(c): c in P
Preference criterion
+
Crowding distance
Selection
survivors
Evolut-
ionary
operators
+ offspring
+ mutants
Fitness Eval
f(c): c in P'
Termination Check
Solution
Timeout

Challenges

- MOSA outperformed WholeSuite [ICST'15]

- However,
  - Large number of targets
  - Reduced search efficiency
  - E.g., mutation testing

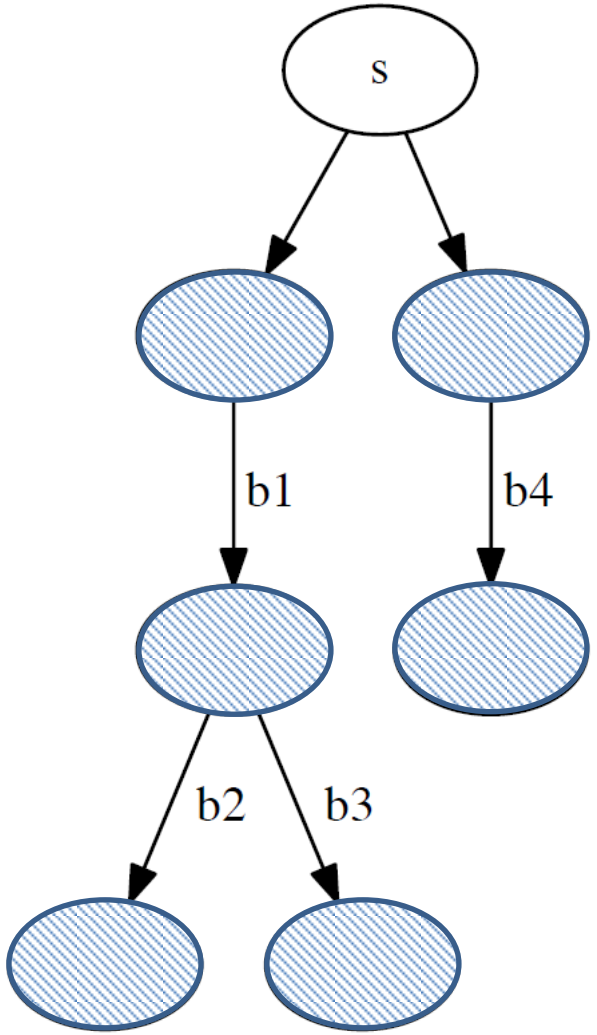
- ➔Reduce number of targets

DynaMOSA

- Select targets based on control dependency
  1. U ← { root, dependents }
  2. Run MOSA on U
  3. U ← U + dependents (u); u covered
  4. Goto step 2
- Search focuses on the important targets
- Equivalent to MOSA (theorem)
  - But more efficient

DynaMOSA

	Instructions
s	`int example(int a, int b, int c)`
	`{`
	`int x = 0;`
1	`if (a == b)` // b_1
2	`  if (a > c)` // b_2
3	`  x = 1;`
4	`  else` // b_3
5	`  x = 2;`
6	`if (b == c)` // b_4
7	`  x = -1;`
8	`return x;`
	`}`



Tool

- extended the  framework

- for unit testing of Java classes
  - Implementation of MOSA/DynaMOSA
  - Statement, branch, mutation

    https://github.com/EvoSuite

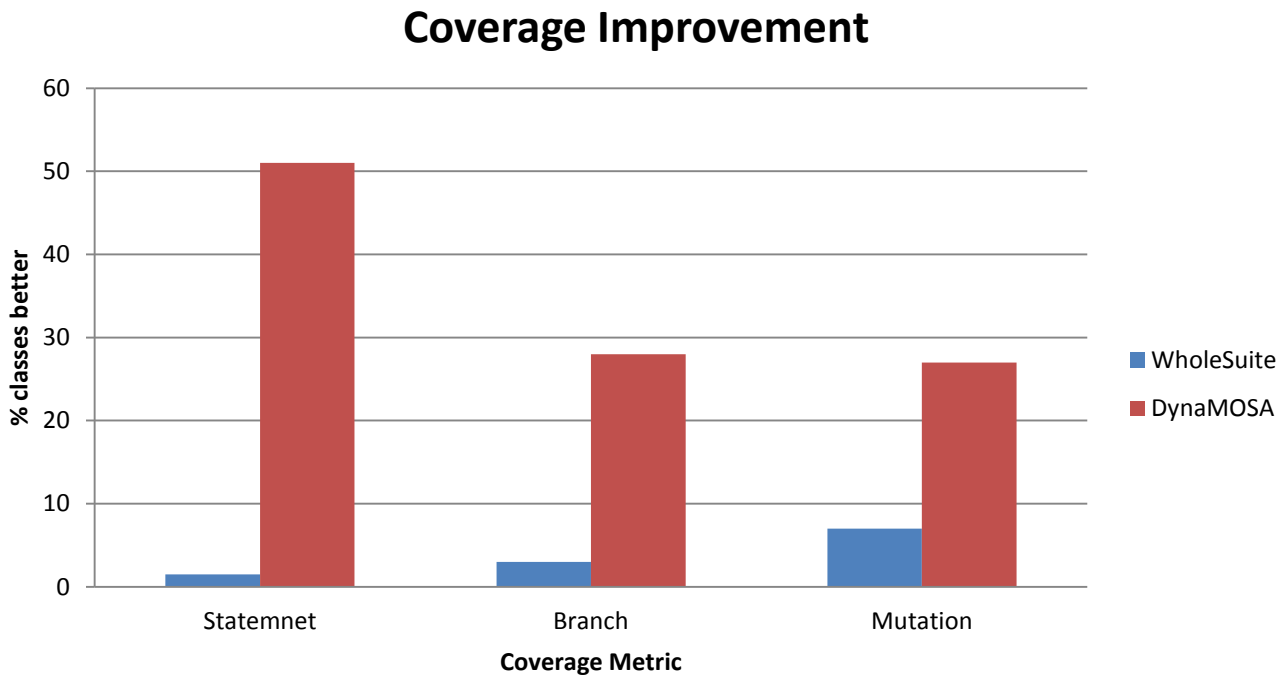
Evaluation

- Research Questions
  - **RQ1**: coverage compared to WholeSuite
  - **RQ2**: rate of convergence for equal coverage

- Subjects
  - **346** Java classes
  - 361K statements; 62K branches; 118K mutants

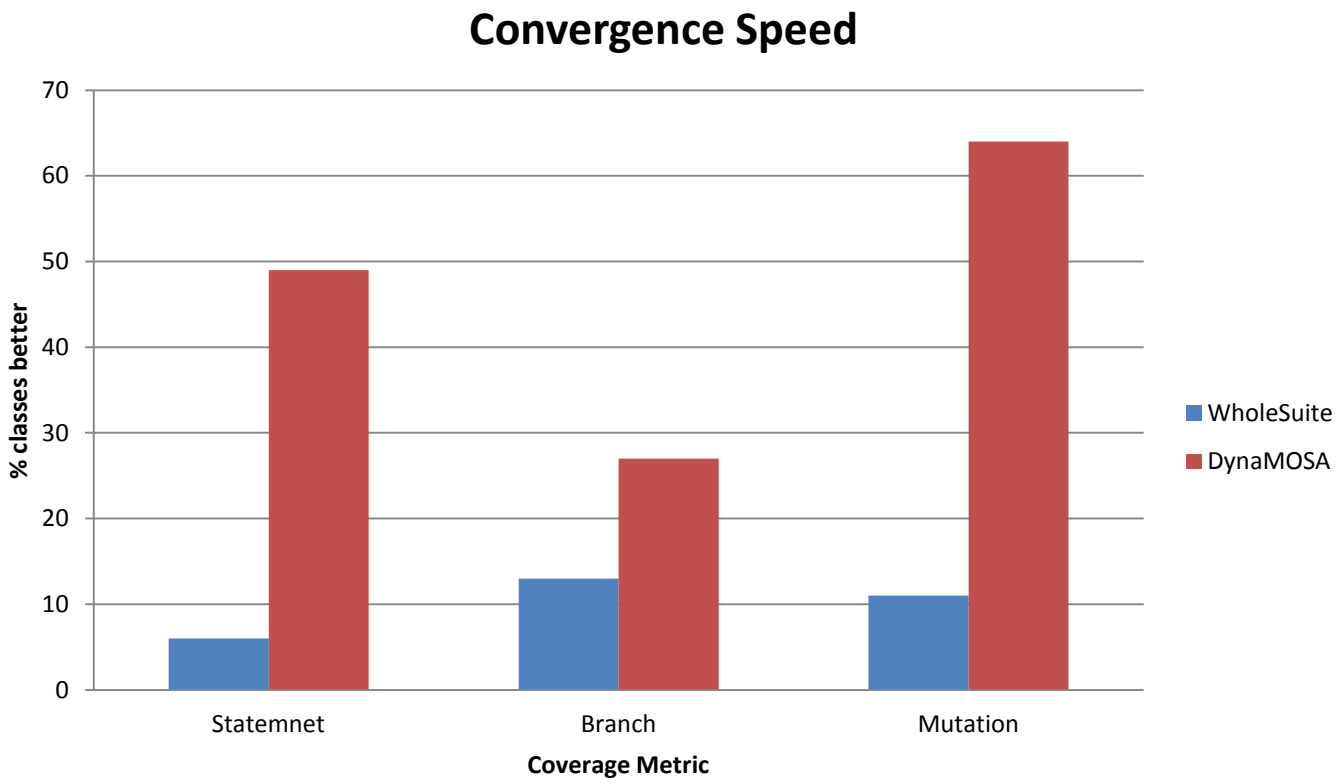
Results - RQ1 (effectiveness)

Average over all classes
DynaMOSA: 87% branch, 93% statement, 23% mutation
WholeSuite: 85% branch, 81% statement, 21% mutation



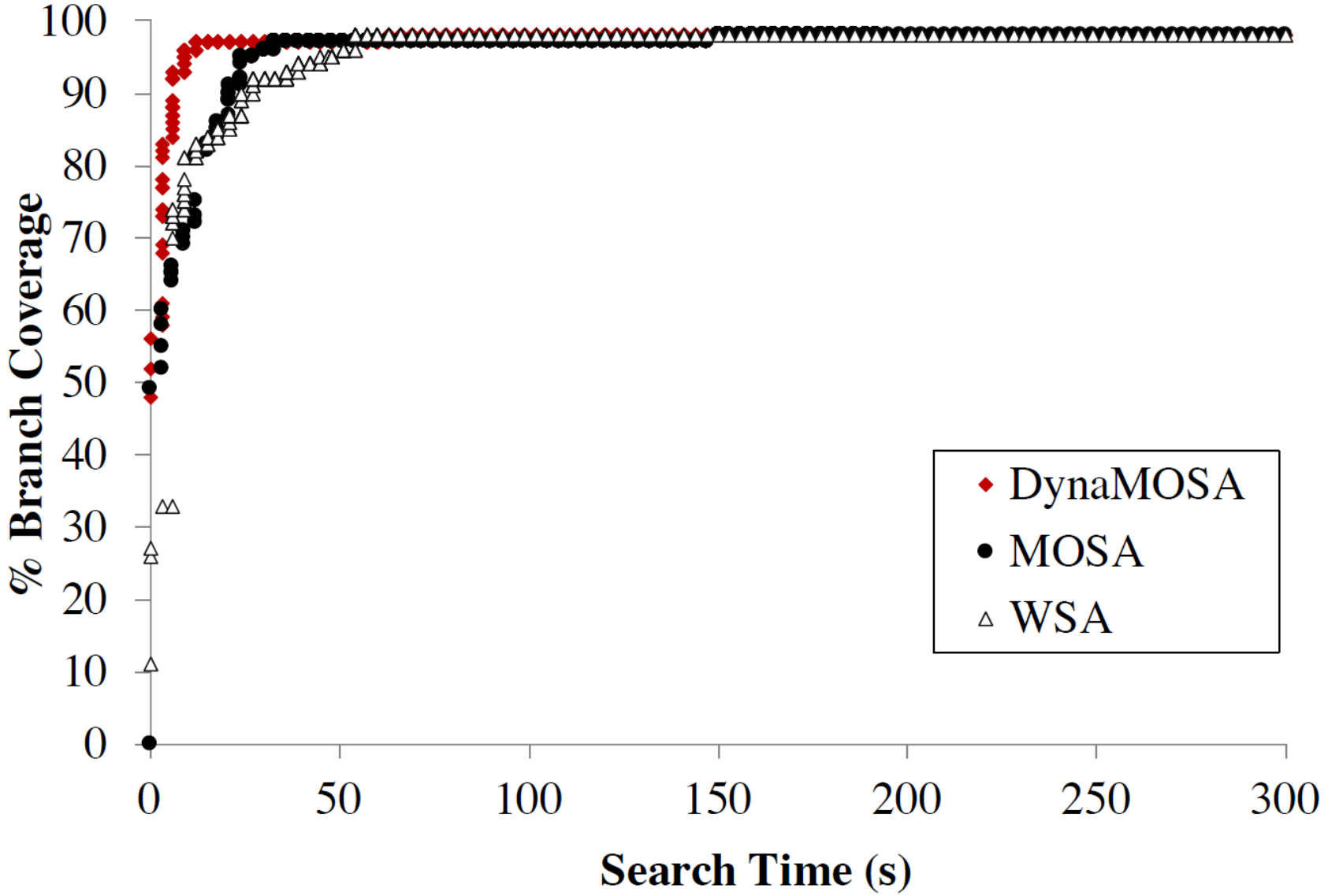
DynaMOSA achieves significantly better coverage than WholeSuite.

Results – RQ2 (efficiency)



DynaMOSA converges significantly faster than WholeSuite.

Convergence: Example



Summary

- Reformulated branch coverage as a many-objective optimization problem
- Introduced highly scalable MOSA
- Better performance than WholeSuite
  - 87% branch, 93% statement, 23% mutation
  - Quick convergence
- Next: Consider non-coverage objectives
  - Execution time, memory consumption, …

Referenecs

Annibale Panichella, Fitsum Meshesha Kifetew, and Paolo Tonella.

Automated Test Case Generation as a Many-Objective Optimisation Problem with Dynamic Selection of the Targets

In IEEE Transactions on Software Engineering (TSE). IEEE, 2017.

Annibale Panichella, Fitsum Meshesha Kifetew, and Paolo Tonella.
Reformulating Branch Coverage as a Many-Objective Optimization Problem

In International Conference on Software Testing, Verification and Validation (ICST). IEEE, 2015.

Annibale Panichella, Fitsum Meshesha Kifetew, and Paolo Tonella.

Results for EvoSuite-MOSA at the Third Unit Testing Tool Competition

In International Symposium on Search Based Software Testing (SBST). IEEE, 2015.

Thank you!