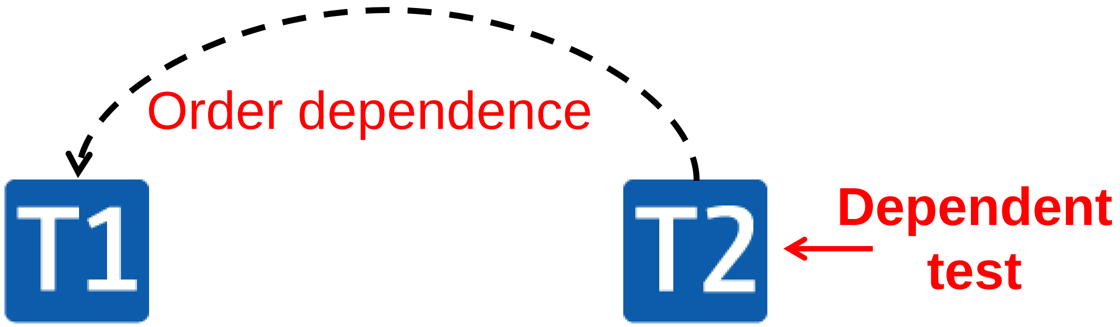
Empirically Revisiting the Test Independence Assumption

Sai Zhang, Darioush Jalali, Jochen Wuttke, Kıvanç Muşlu, Wing Lam, Michael D. Ernst, David Notkin

University of Washington

Computer Science & Engineering
UNIVERSITY *of* WASHINGTON

Two tests:



Order dependence

Dependent test

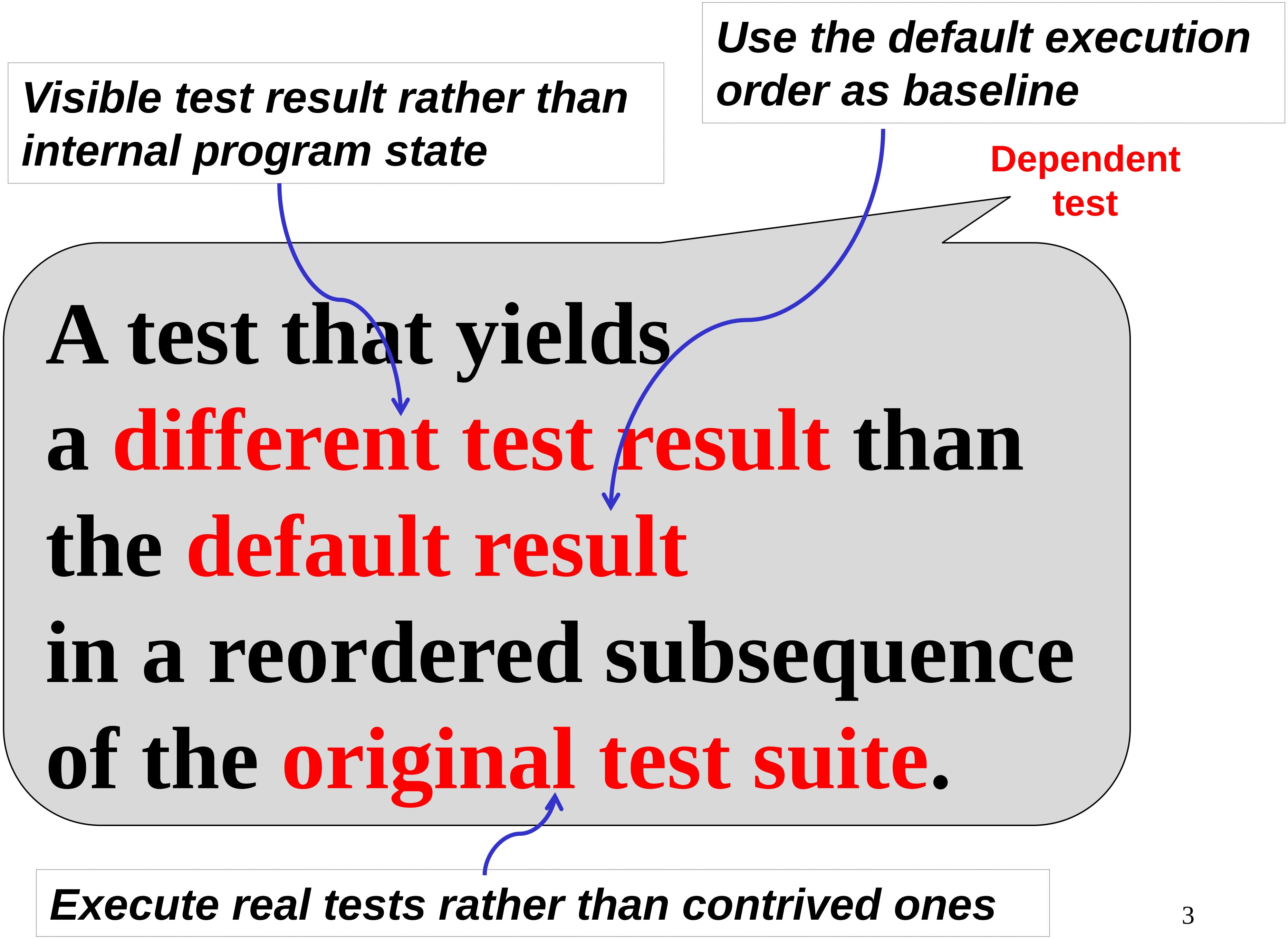
```
createFile("foo")
...
```

```
readFile("foo")
...
```

Executing them in **default** order: T1 ✓ T2 ✓

(the intended test results)

Executing them in a **different** order: T2 ✗ T1 ✓



Visible test result rather than internal program state

Use the default execution order as baseline

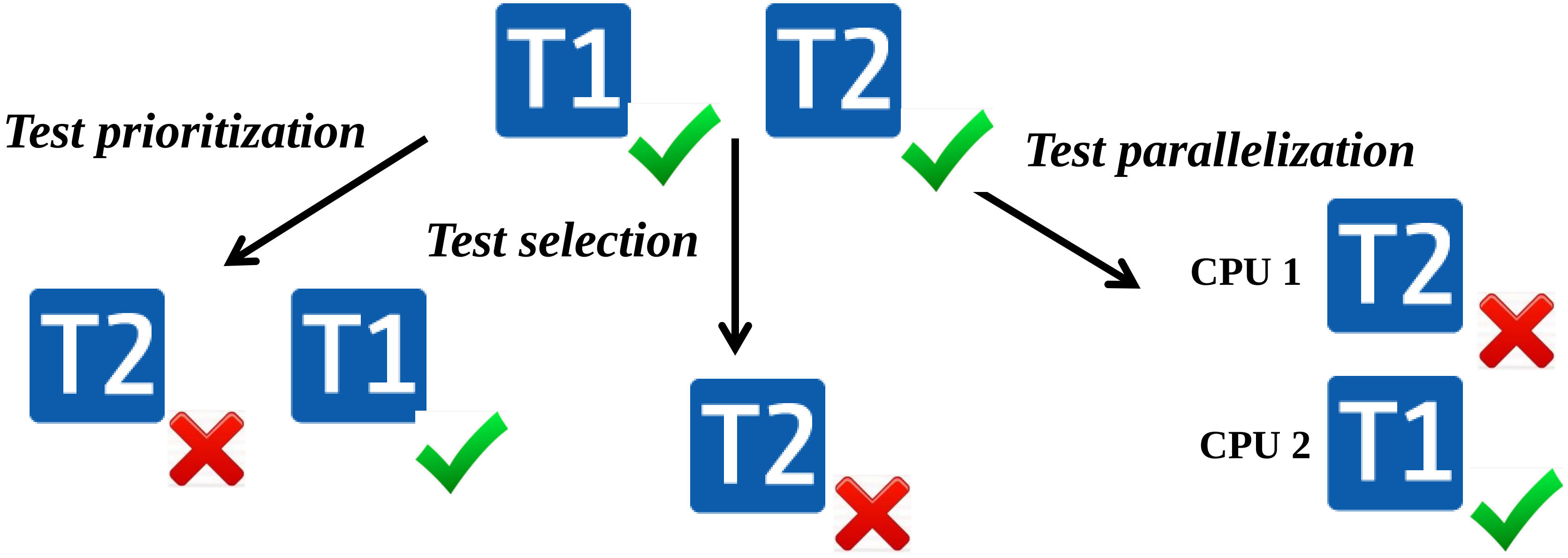
Dependent test

A test that yields a different test result than the default result in a reordered subsequence of the original test suite.

Execute real tests rather than contrived ones

Why should we care about test dependence?

- Makes test behaviors inconsistent
- Affects downstream testing techniques



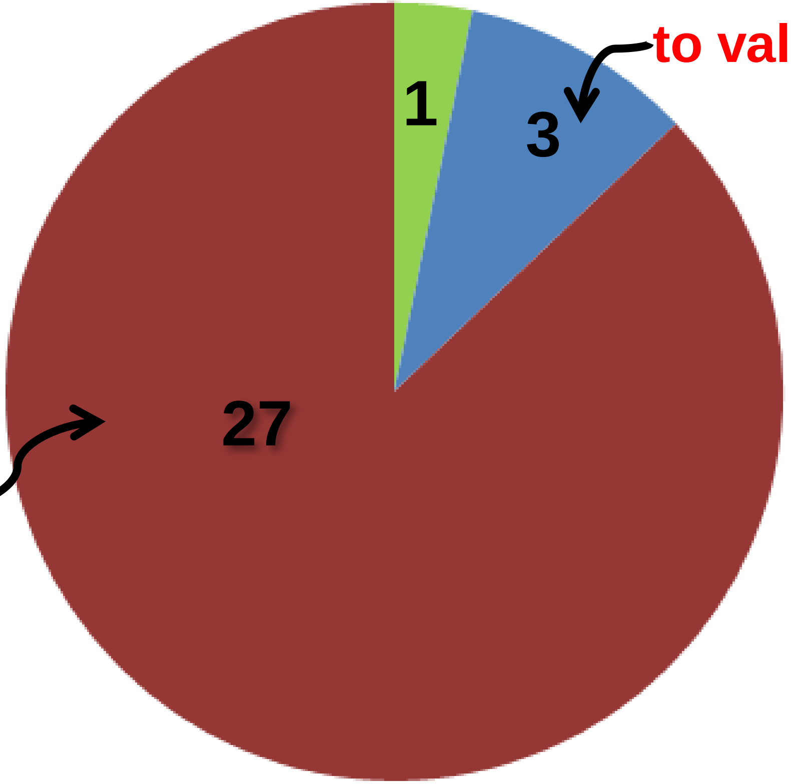
Conventional wisdom:
test dependence is not a significant issue

- Test independence is assumed by:
  - Test selection
  - Test prioritization
  - Test parallel execution
  - Test factoring
  - Test generation
  - …

31 papers in
ICSE, FSE, ISSTA, ASE,
ICST, TSE, and TOSEM
(2000 – 2013)

Conventional wisdom:
test dependence is not a significant issue

- Test independence is assumed by:
  - Test selection
  - Test prioritization
  - Test parallel execution
  - Test factoring
  - Test generation
  - …



Consider test dependence

As a threat to validity

Assume test independence without justification

Is the test independence assumption valid?

No!

- Does test dependence arise in practice?

  Yes, in both human-written and automatically-generated suites

- What repercussions does test dependence have?
  - *Inconsistent results: missed alarms and false alarms*
  - *Affecting downstream testing techniques*
- How to detect test dependence?
  - *Proof: the general problem is NP-complete*
  - *Approximate algorithms based on heuristics work well*

Implications:

Test independence should no longer be assumed

New challenges in designing testing techniques

Is the test independence assumption valid?

No!

- Does test dependence arise in practice?
  - *Yes, in both human-written and automatically-generated suites*
- What repercussions does test dependence have?
  - *Inconsistent results: missed alarms and false alarms*
  - *Affecting downstream testing techniques*
- How to detect test dependence?
  - *Proof: the general problem is NP-complete*
  - *Approximate algorithms based on heuristics work well*

Is the test independence assumption valid?

- Does test dependence arise in practice?

  Yes, in both human-written and automatically-generated suites

- What repercussion does test dependence have ?
  - *Inconsistent results: missed alarms and false alarms*
  - *Affecting downstream testing techniques*
- How to detect test dependence?
  - *The general problem is NP-complete*
  - *Approximate algorithms based on heuristics work well*

Methodology

Reported dependent tests

Apache

eclipse

JBoss™

HIBERNATE

Codehaus

5 issue tracking systems

New dependent tests

Joda-Time

The Apache xml-security

crystal

synoptic

4 real-world projects

Methodology

Reported dependent tests

5 issue tracking systems

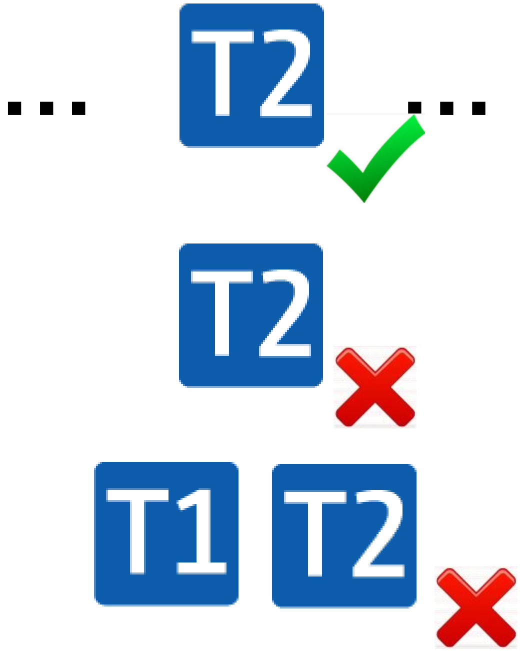
- Search for 4 key phrases:
  ("dependent test", "test dependence",
  "test execution order", "different test outcome")
- Manually inspect 450 matched bug reports
- Identify 96 distinct dependent tests

Characteristics:

– **Manifestation**
– **Root cause**
– **Developers' action**

Manifestation

Number of tests involved to yield a different result



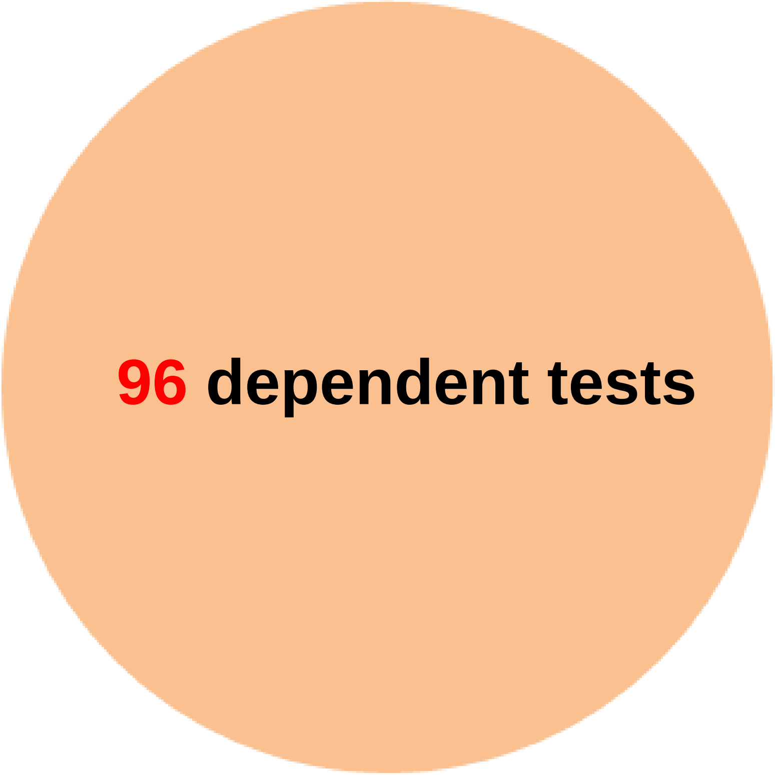
(default order)

(run in isolation) **#Tests = 1**

(run after another) **#Tests = 2**

Manifestation

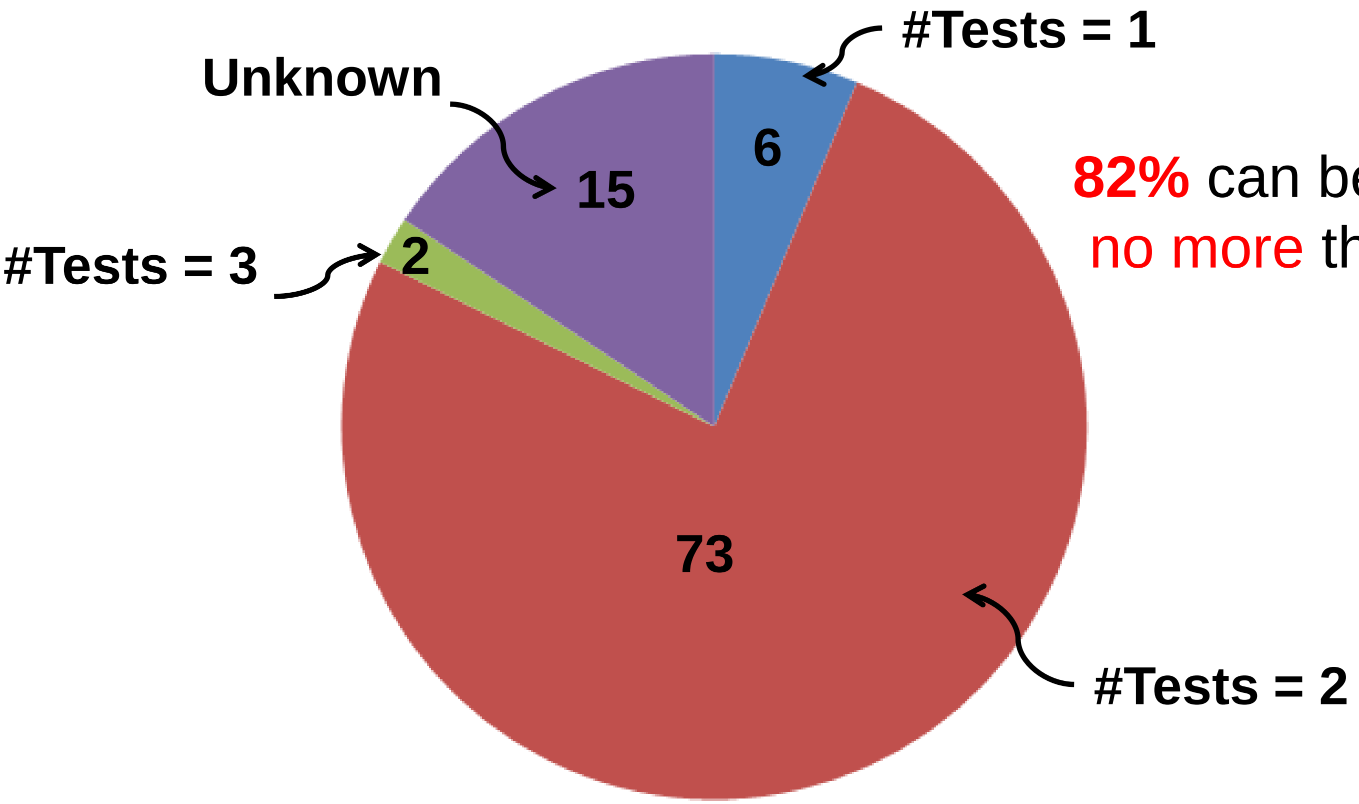
Number of tests involved to yield a different result



96 dependent tests

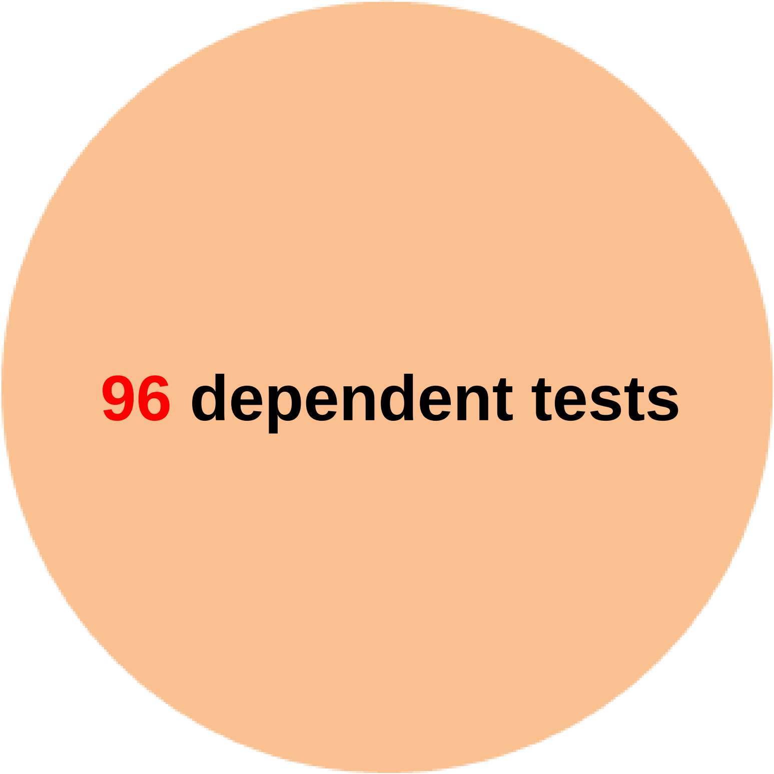
Manifestation

Number of tests involved to yield a different result



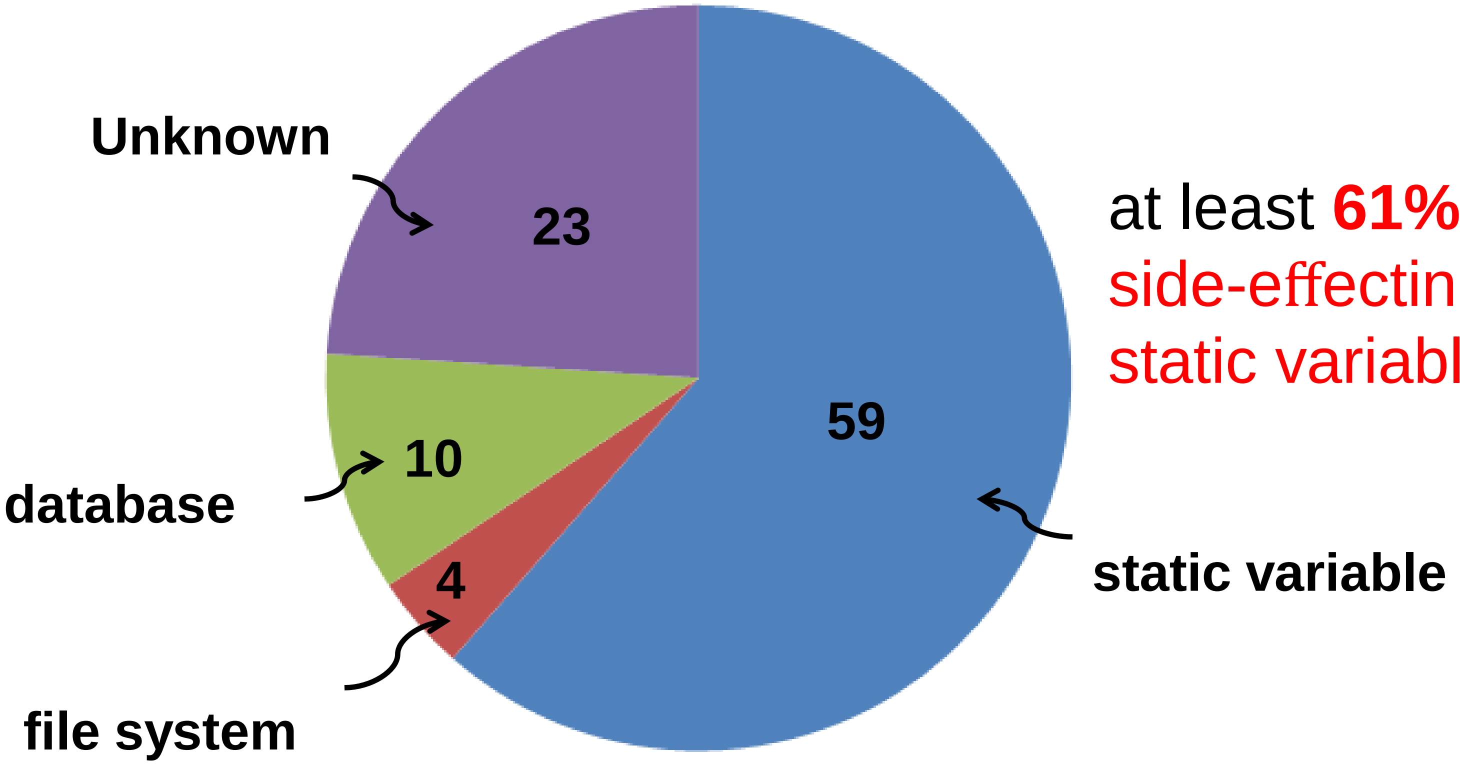
82% can be revealed by no more than 2 tests

Root cause



96 dependent tests

Root cause



at least **61%** are due to side-effecting access to static variables.

Developers' action

98% of the reported tests are marked as major or minor issues

91% of the dependence has been fixed

– Improving documents
– Fixing test code or source code

Methodology

- Human-written test suites
  - **4176** tests

    29 dependent tests
- Automatically-generated test suites
  - use Randoop [Pacheco'07]
  - **6330** tests

    354 dependent tests
- Ran dependent test detection algorithms (*details later*)

New dependent tests

Joda-Time

The Apache
xml-security

crystal

synoptic

4 real-world projects

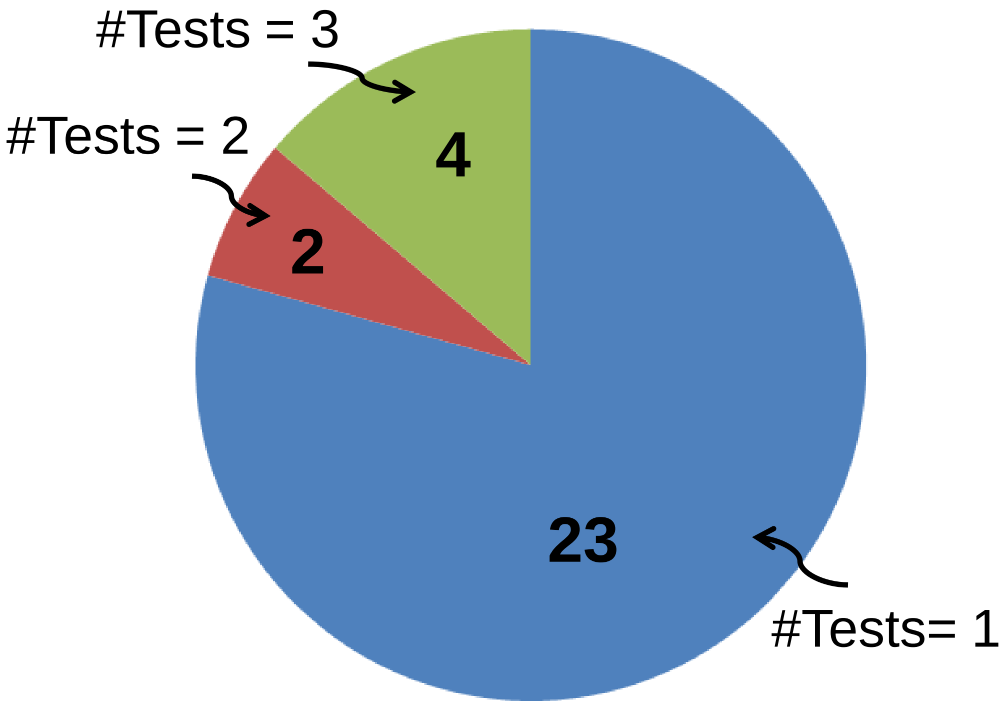
Characteristics

- Manifestation: number of tests to yield a different result

29 manual dependent tests

Characteristics

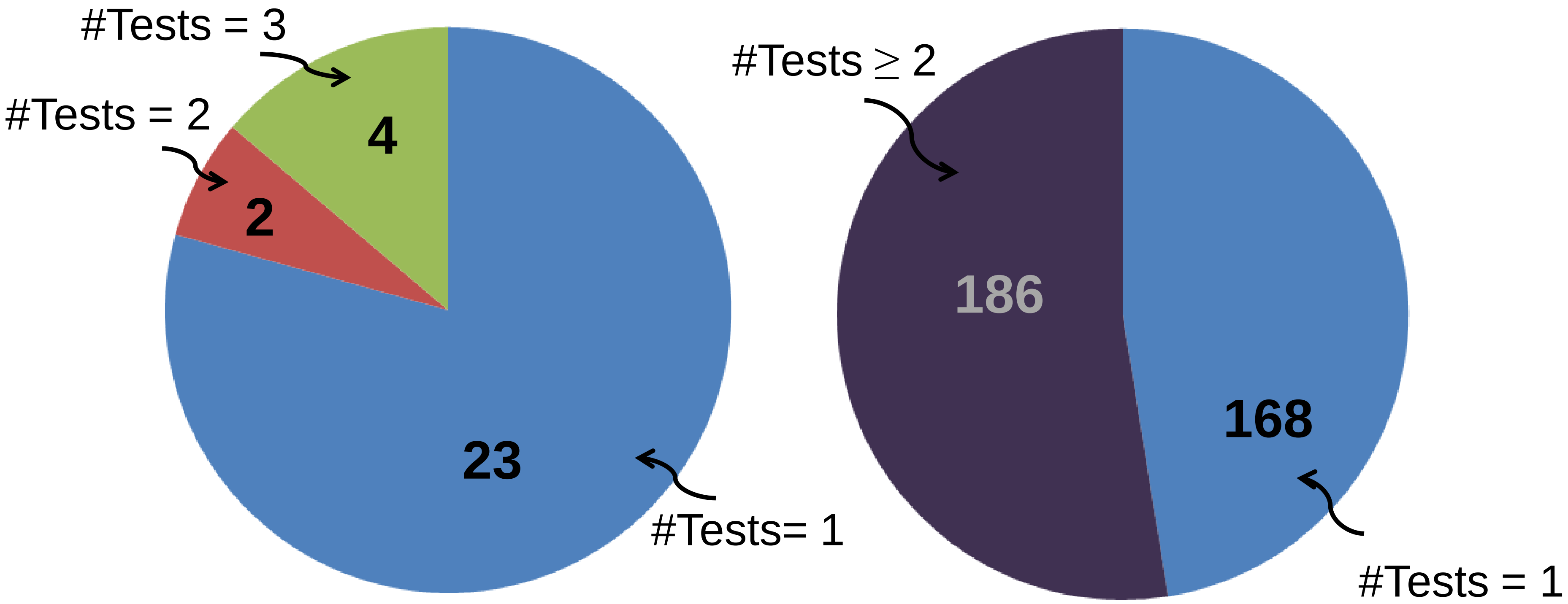
- Manifestation: number of tests to yield a different result



354 auto-generated dependent tests

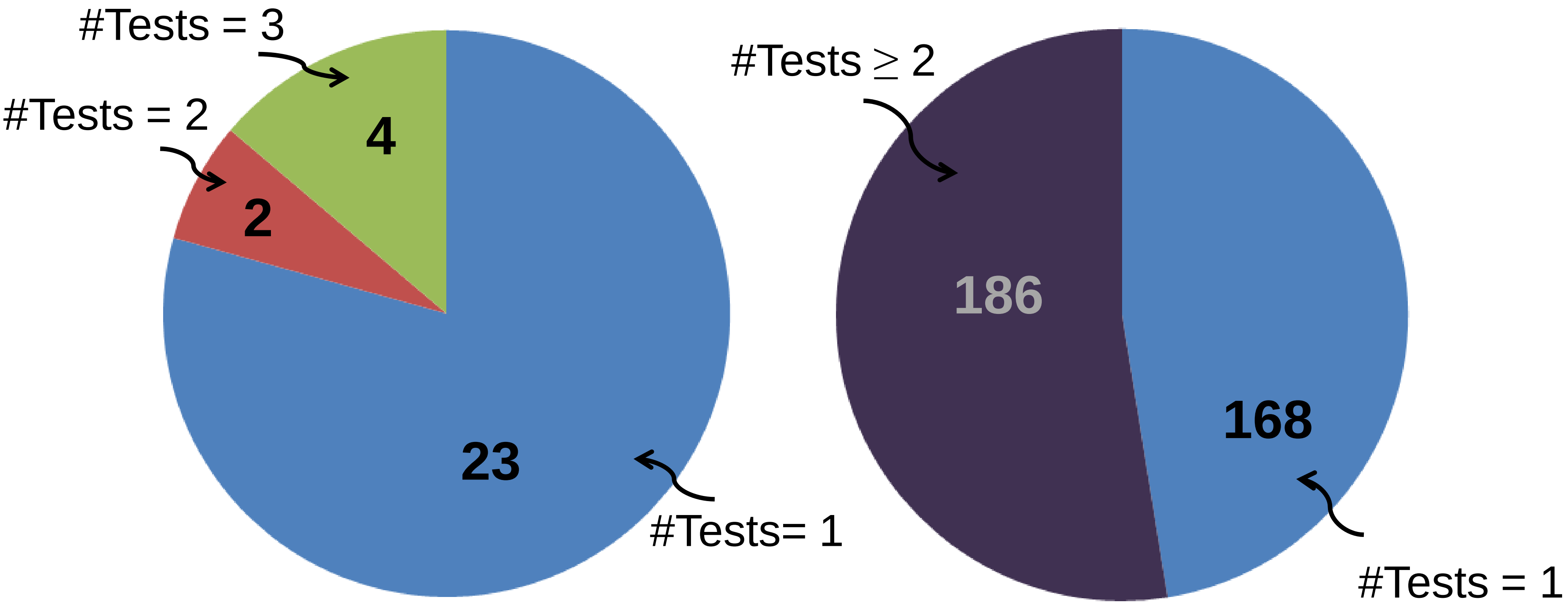
Characteristics

- Manifestation: number of tests to yield a different result



Characteristics

- Manifestation: number of tests to yield a different result



- Root cause
  - **All** because of **side-effecting** access of **static variables**

Developers' actions

- Confirm all manual dependent tests

The Apache
xml-security

  - tests should always "stand alone", that is "test engineering 101"

synoptic

  - Merged two tests to remove the dependence

crystal

  - Opened a bug report to fix the dependent test

Joda-Time

  - Wont fix the dependence, since it is due to the library design

Is the test independence assumption valid?

- Does test dependence arise in practice?

  Yes, in both human-written and automatically-generated suites

- What repercussion does test dependence have ?
  - *Inconsistent results: missed alarms and false alarms*
  - *Affecting downstream testing techniques*
- How to detect test dependence?
  - *The general problem is NP-complete*
  - *Approximate algorithms based on heuristics work well*

Reported dependent tests

Apache

eclipse

JBoss™

HIBERNATE

Codehaus

5 issue tracking systems

96 dependent tests

Reported dependent tests

Apache

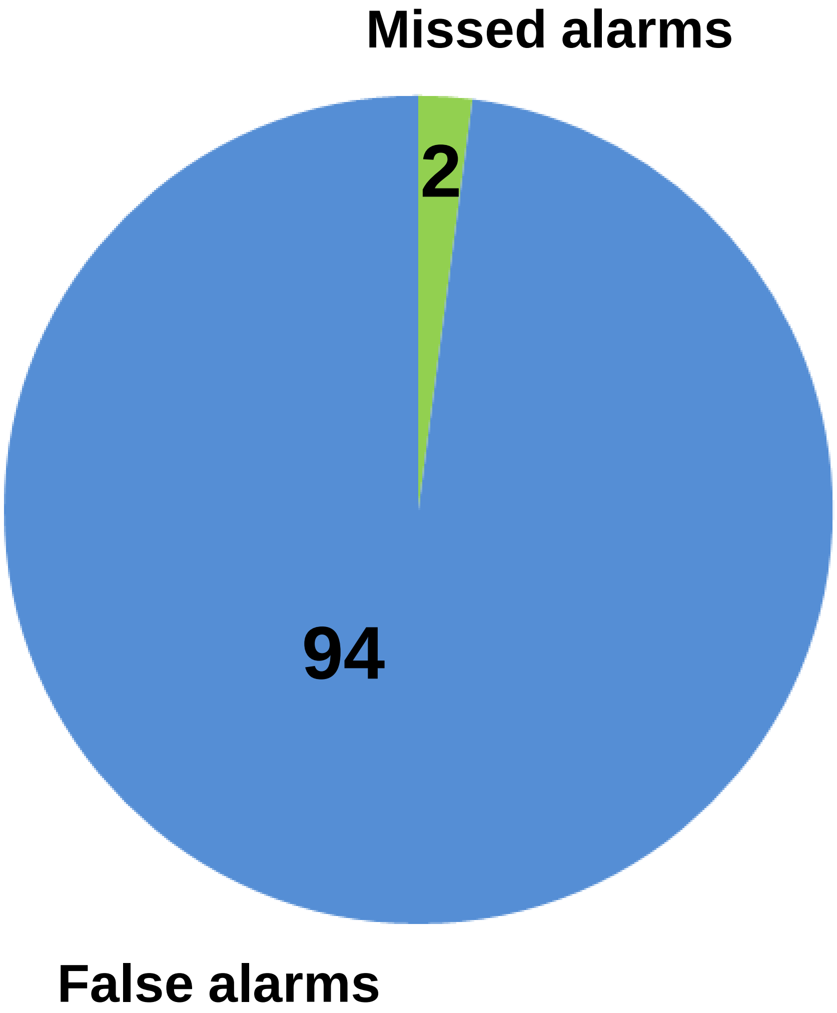
eclipse

JBoss

HIBERNATE

Codehaus

5 issue tracking systems



Example false alarm

```
void testDisplay() {
   //create a Display object
   …
   //dispose the Display object
}
```

```
void testShell() {
   //create a Display object
   …
}
```

In Eclipse, only one Display object is allowed.

In default order: testDisplay ✓ testShell ✓

In a non-default order: testShell ✓ testDisplay ✗

Led to a false bug report that took developers 3 months to resolve.

Example missed alarm

```
public final class OptionBuilder {
    static String argName = null;
    static void reset() {
        …
        argName = “arg”;
    }

}
```

Need to be set to “`arg`” before a client calls any method in the class.

`BugTest.test13666` validates correct behavior.

This test should fail,

but passes when running in the **default** order

- Another test calls `reset()` before this test

Hid a bug for 3 years.

Example missed alarm

```
public final class OptionBuilder {
    static String argName = null;
    static void reset() {
        …
        argName = "arg";
    }

}
```

Need to be set to "`arg`" before a client calls any method in the class.

`BugTest.test13666` validates correct behavior.

This test should fail,

but passes when running in the **default** order

- Another test calls `reset()` before this test

Hid a bug for 3 years.

Example missed alarm

commons
CLI™
Apache

```
public final class OptionBuilder {
     static String argName = null;
     static void reset() {
          ……
     }
     static {
         argName = “arg”;
     }
}
```

Need to be set to “`arg`” before a client calls any method in the class.

Bug fix

`BugTest.test13666` validates correct behavior.

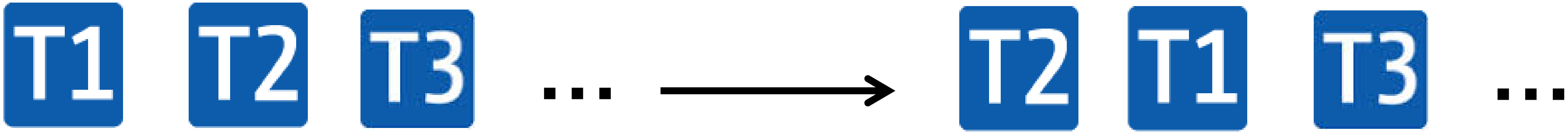
This test should fail,

but passes when running in the **default** order

- Another test calls `reset()` before this test

Hid a bug for 3 years.

Test prioritization



A test execution order

A new test execution order

Achieve coverage faster
Improve fault detection rate
…

Each test should yield the same result.

Five test prioritization techniques

[Elbaum et al. ISSTA 2000]

Test prioritization technique
Randomized ordering
Prioritize on coverage of statements
Prioritize on coverage of statements not yet covered
Prioritize on coverage of methods
Prioritize on coverage of methods not yet covered

Joda-Time

The Apache
xml-security

crystal

synoptic

4 real-world projects

Total: 4176 manual tests

- Record the number of tests yielding different results

Evaluating test prioritization techniques

Test prioritization technique	Number of tests that yield different results
Randomized ordering	**12**
Prioritize on coverage of statements	**11**
Prioritize on coverage of statements not yet covered	**17**
Prioritize on coverage of methods	**11**
Prioritize on coverage of methods not yet covered	**12**

Total: 4176 manual tests

- **Implication:**
  - Existing techniques are not aware of test dependence

Is the test independence assumption valid?

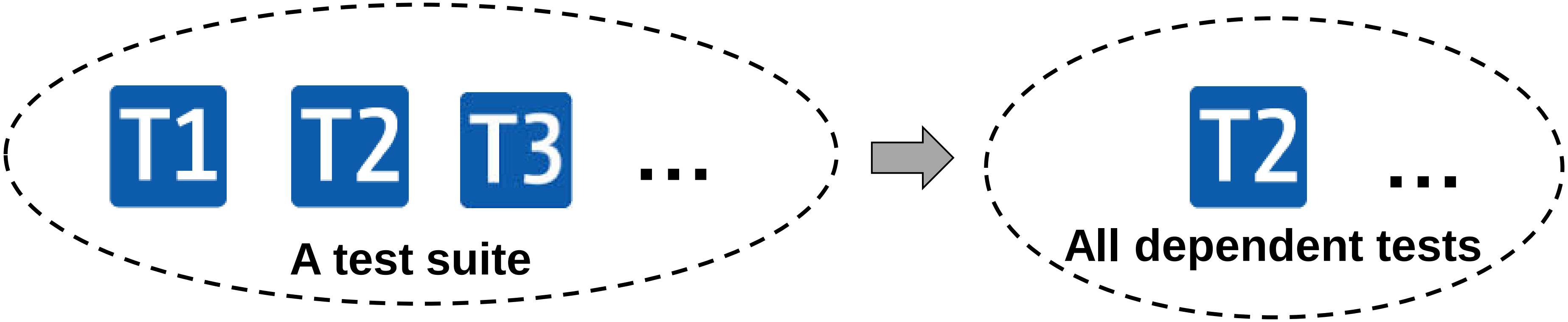
- Does test dependence arise in practice?

  Yes, in both human-written and automatically-generated suites

- What repercussion does test dependence have ?
  - *Inconsistent results: missed alarms and false alarms*
  - *Affecting downstream testing techniques*

- How to detect test dependence?
  - *The general problem is NP-complete*
  - *Approximate algorithms based on heuristics work well*

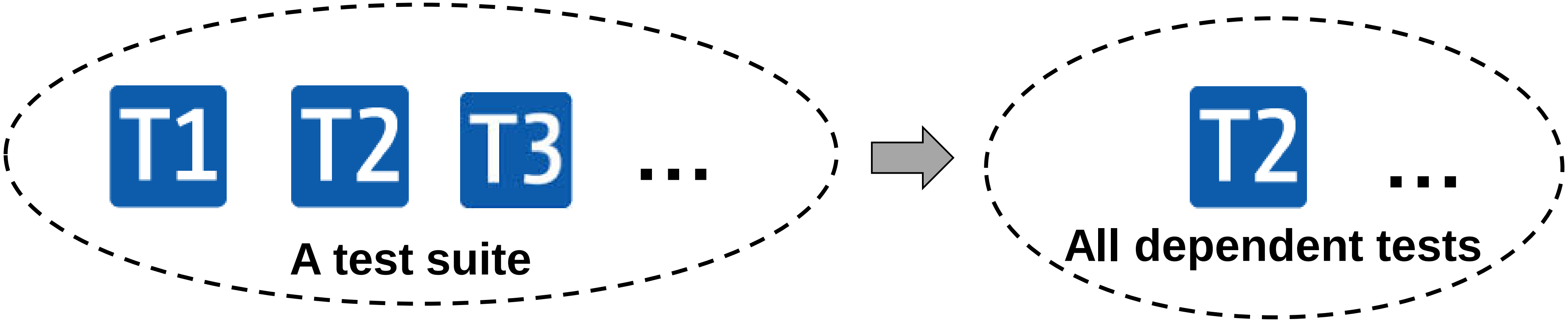
General problem of test dependence detection



NP-Complete

- Proof: reducing the Exact Cover problem to the dependent test detection problem

Detecting dependent tests in a test suite



A test suite

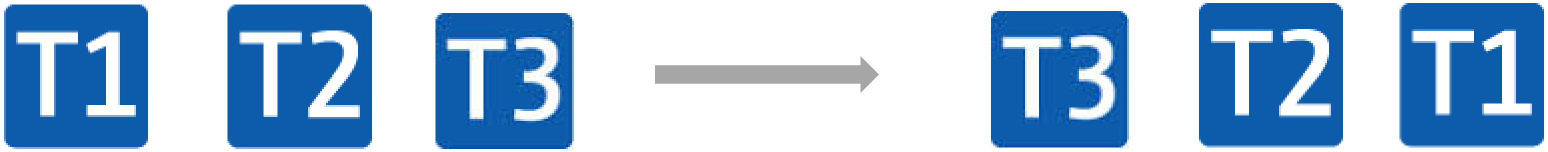
All dependent tests

- **Approximate** algorithms
  - Reversal algorithm
  - Randomized execution
  - Exhaustive bounded algorithm
  - Dependence-aware bounded algorithm

All algorithms are sound but incomplete

Approximate algorithms by heuristics

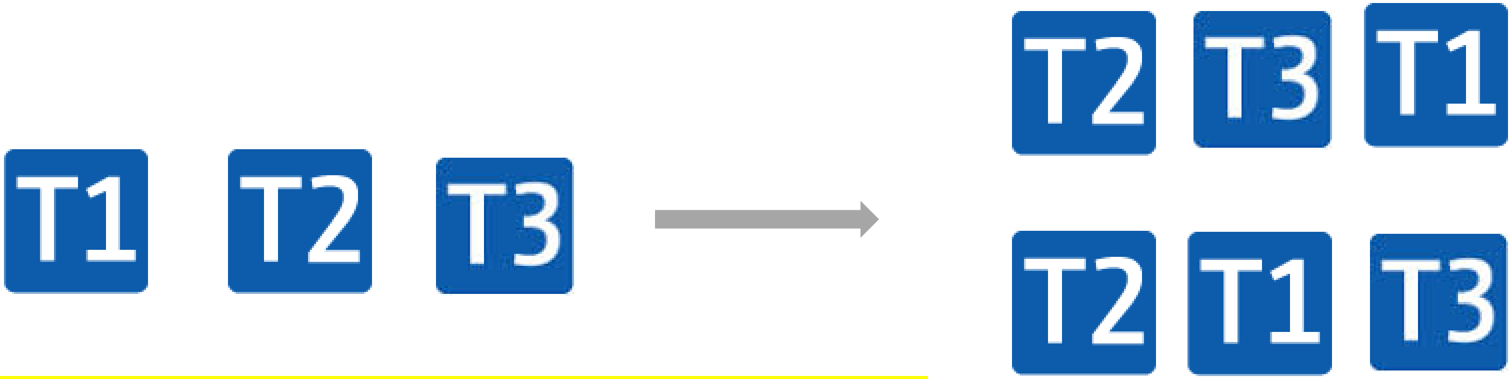
- Reversal algorithm
- Randomized execution
- Exhaustive bounded algorithm
- Dependence-aware bounded algorithm



Intuition: changing order of **each** pair may expose dependences

Approximate algorithms by heuristics

- Reversal algorithm
- Randomized execution
- Exhaustive bounded algorithm
- Dependence-aware bounded algorithm



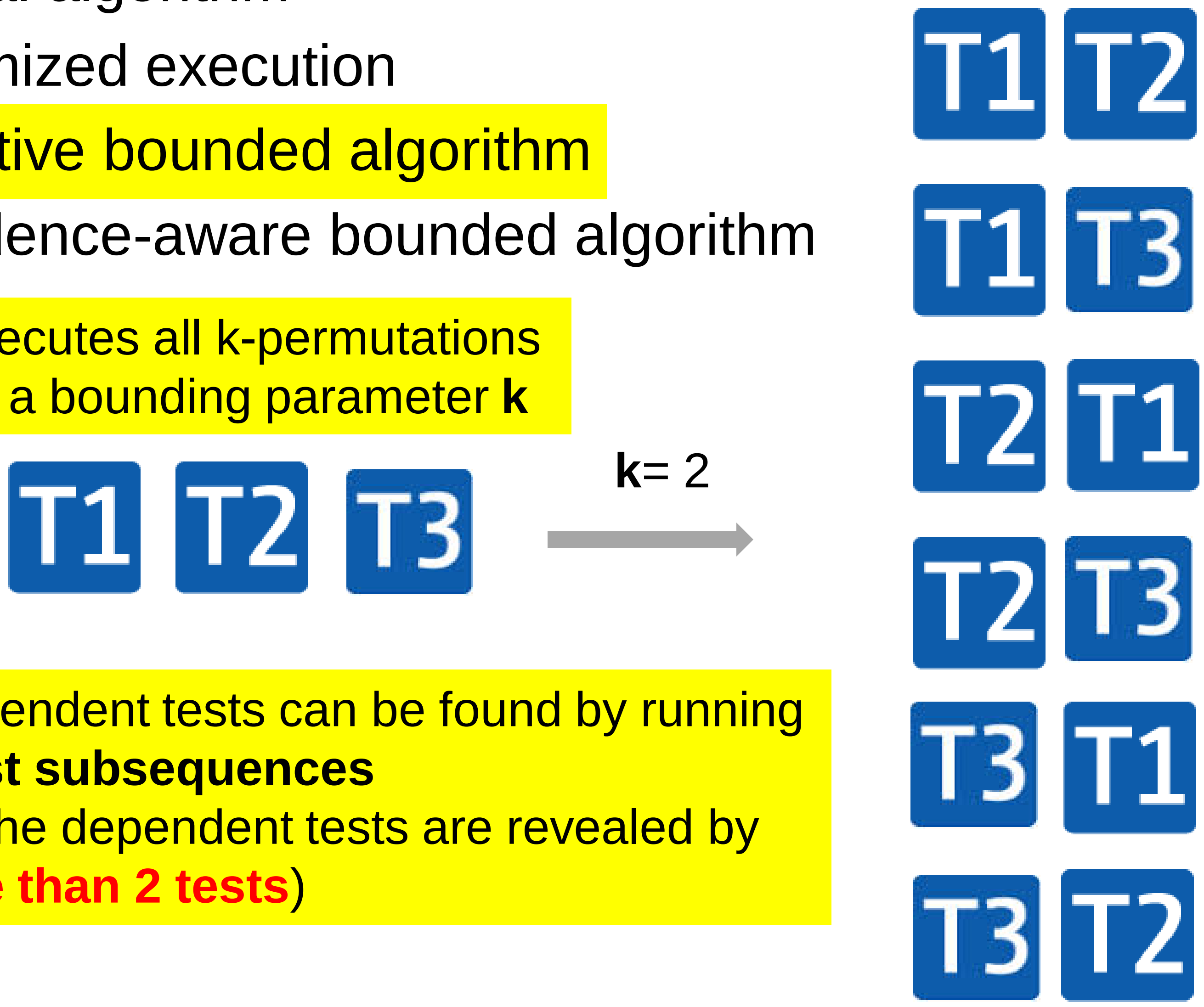
Shuffle the execution order multiple times

...

Approximate algorithms by heuristics

- Reversal algorithm
- Randomized execution
- Exhaustive bounded algorithm
- Dependence-aware bounded algorithm

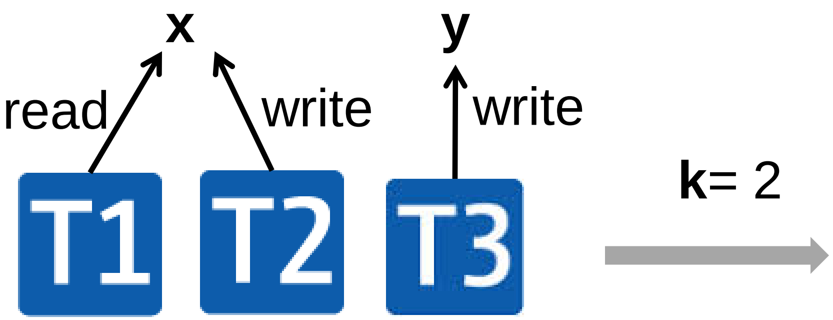
Executes all k-permutations for a bounding parameter **k**

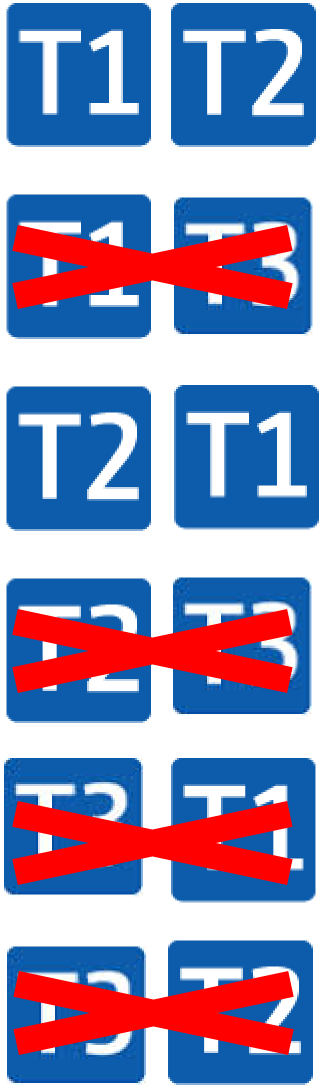


Most dependent tests can be found by running **short test subsequences**
(**82%** of the dependent tests are revealed by **no more than 2 tests**)

Approximate algorithms by heuristics

- Reversal algorithm
- Randomized execution
- Exhaustive bounded algorithm
- Dependence-aware bounded algorithm





Record read/write info for each test

Filter away unnecessary permutations

Evaluating approximate algorithms

- Human-written test suites
  - **4176** tests

    29 dependent tests

- Automatically-generated test suites
  - use Randoop [Pacheco'07]
  - **6330** tests

    354 dependent tests

Finding New dependent tests

Joda-Time

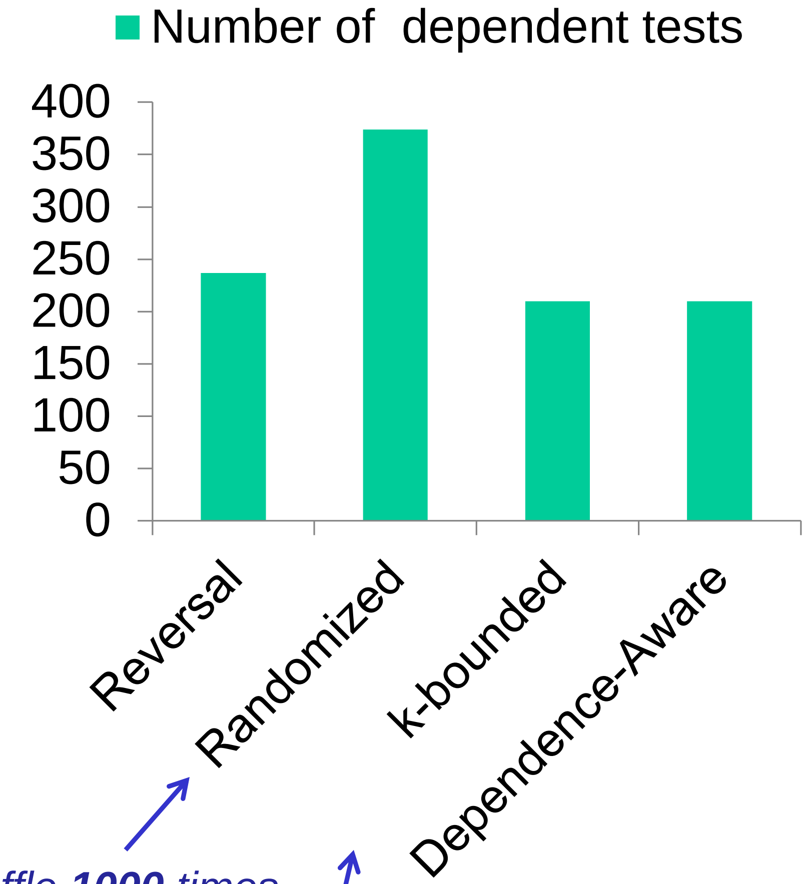
The Apache
xml-security

crystal

synoptic

4 real-world projects

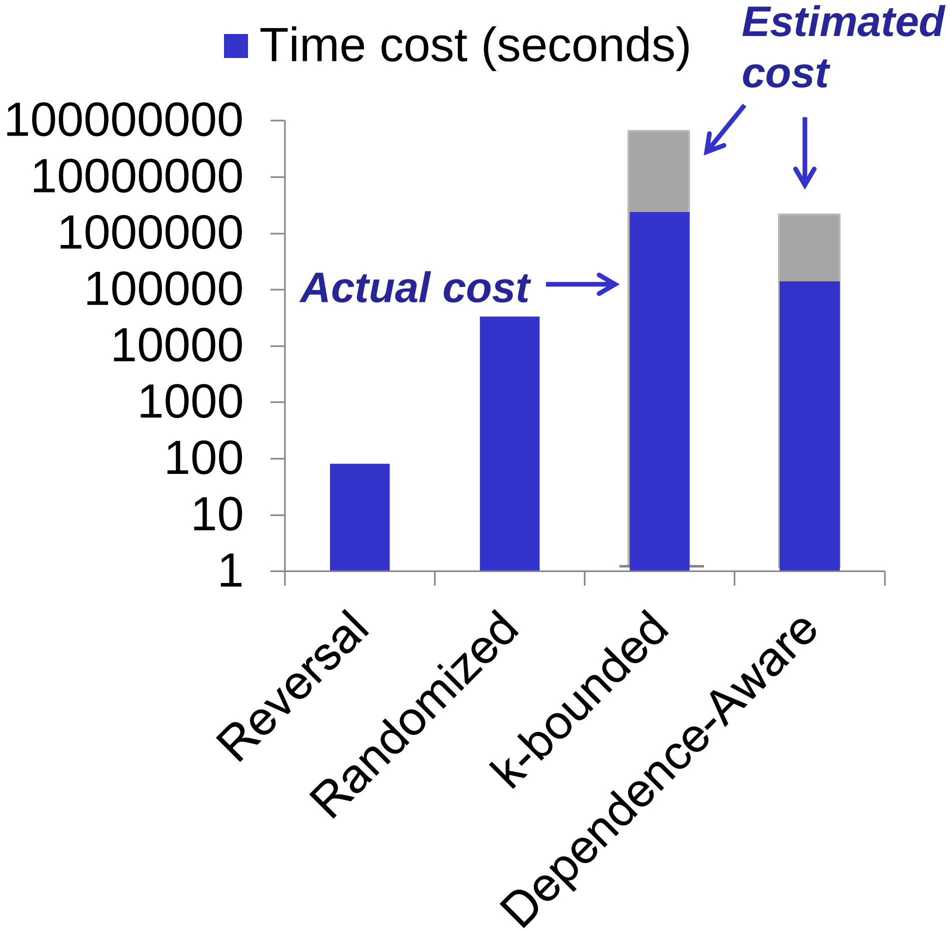
Evaluating approximate algorithms



*Shuffle **1000** times*

***k* = 2**

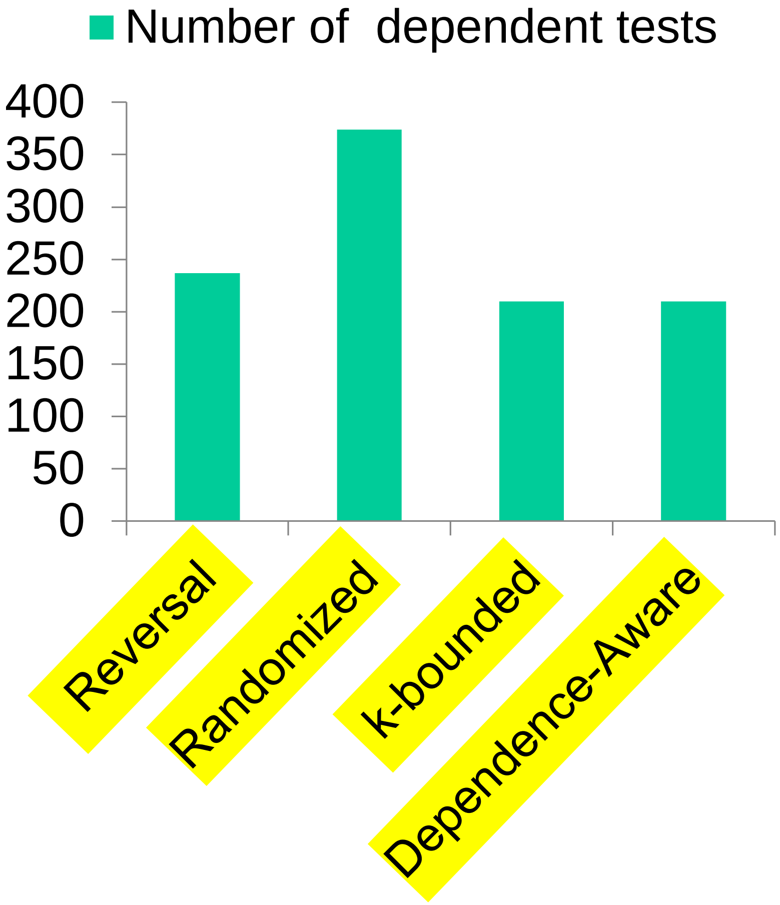
(did not finish for some programs)

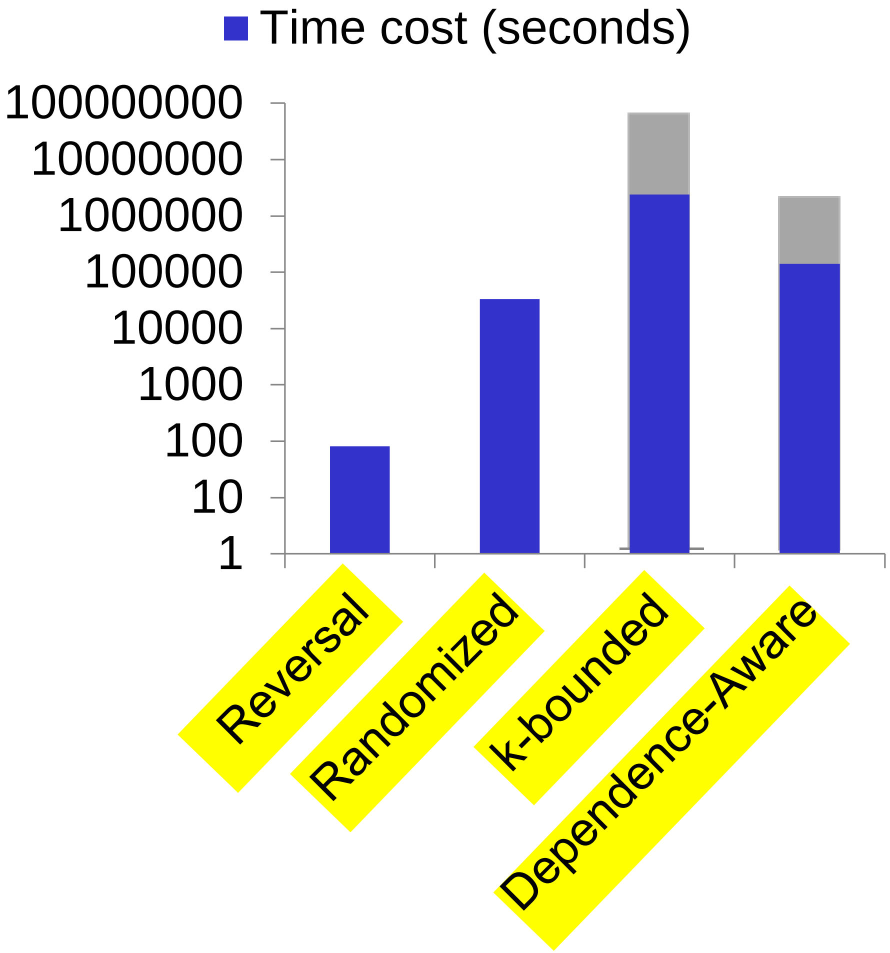


Estimated cost

Actual cost

Evaluating approximate algorithms





Chea Find all dependences within a bound, but computationally infeasible.

Related work

- Existing definitions of test dependence
  - Based on program state change [Kapfhammer’03]
  - Informal definitions [Bergelson’06]

  Our definition focuses on the concrete test execution result.

  Program state change may not affect test execution result.

- Flaky tests [Luo et al’14, Google testing blog]
  - Tests revealing inconsistent results

  Dependent test is a special type of flaky test.

- Tools supporting to execute tests in different orders
  - JUnit 4.1: executing tests in alphabetical order by name
  - DepUnit, TestNg: supporting specifying test execution order

  Do not support detecting test dependence.

Contributions

- Revisiting the test independence assumption
  - ✓ – Test dependence arises in practice
  - ✓ – Test dependence has non-trivial repercussions
  - ✓ – Test dependence detection is NP-complete
  - ✓ – Heuristic algorithms are effective in practice

Test independence should no longer be assumed!

- Our tool implementation

  http://testisolation.googlecode.com

[Backup slides]

Why not run each test in a separate process?

- Implemented in JCrasher
- Supported in Ant + JUnit

- Unacceptably high overhead
  - **10 – 138 X** slowdown

- Recent work merges tests running in separate processes into a single one [Bell & Kaiser, ICSE 2014]

Why more dependent tests in automatically-generated test suites?

- Manual test suites:
  - Developer’s understanding of the code and their testing goals help build well-structured tests
  - Developers often try to initialize and destroy the shared objects each unit test may use

- Auto test suites:
  - Most tools are **not** “state-aware”
  - The generated tests often “**misuse**” APIs, e.g., setting up the environment incorrectly
  - Most tools can **not** generate environment setup / destroy code

What is the default test execution order?

- The intended execution order as designed
  - Specified by developers
  - Such as, in make file, ant file, or **`TestAll.java`**
  - Lead to the intended results as developers want to see

Dependent tests vs. Nondeterministic tests

- Nondeterminism does not imply dependence
  - A program may execute non-deterministically, but its tests may deterministically succeed.

- Test dependence does not imply nondeterminism
  - A program may have no sources of nondeterminism, but its tests can still be dependent on each other

Controlled Regression Testing Assumption (CRTA) [Rothermel et al., TSE 1996]

- A stronger assumption than determinism, forbidding:
  - Porting to another system
  - Nondeterminism
  - Time-dependencies
  - Interaction with the external environment
  - (implicitly) test dependence

- The authors commented “CRTA is not necessarily impossible” to employ.

- Our paper has a more practical focus on the overlooked issue of test dependence