An Investigation of Compression Techniques to Speed up Mutation Testing

Qianqian Zhu
Ph.D. student

Co-authors: Annibale Panichella and Andy Zaidman

Software Engineering Research Group,
Delft University of Technology, Netherlands

ICST 2018, April 9th, 2018

My experience with PiTest

Apache Commons Lang: 113 classes, 3869 tests

My experience with PiTest

Apache Commons Lang: 113 classes, 3869 tests

Test execution: 1 minute and 14 seconds

My experience with PiTest

Apache Commons Lang: 113 classes, 3869 tests

Test execution: 1 minute and 14 seconds

>> Generated 13021 mutations Killed 11113 (85%)

>> Ran 51176 tests (3.93 tests per mutation)

>> Total: 31 minutes and 38 seconds

My experience with PiTest

Apache Commons Lang: 113 classes, 3869 tests

Test execution: 1 minute and 14 seconds

>> Generated 13021 mutations Killed 11113 (85%)

>> Ran 51176 tests (3.93 tests per mutation)

>> Total: 31 minutes and 38 seconds

How to further speed-up mutation testing?

How to further speed-up mutation testing?

PiTest's optimisations:

- "selective mutation"
- coverage-based test selection
- bytecode translation

How to further speed-up mutation testing?

PiTest's optimisations:

- "selective mutation"
- coverage-based test selection
- bytecode translation

Others:

- mutant clustering
- mutant subsumption
- state infection
- test case prioritisation

How to further speed-up mutation testing?

PiTest's optimisations:

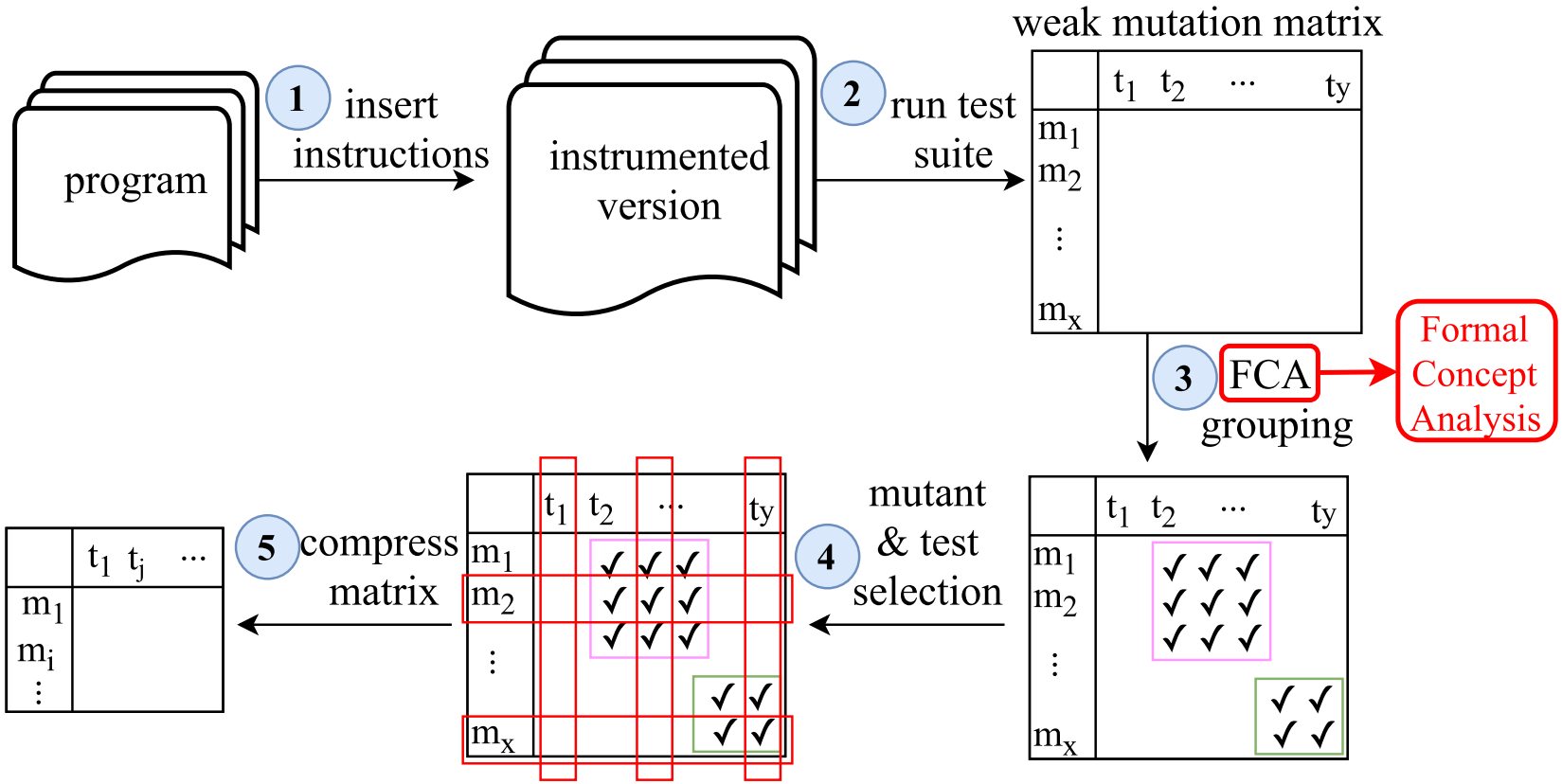
- "selective mutation"
- coverage-based test selection
- bytecode translation

Others:

- mutant clustering
- mutant subsumption
- state infection
- test case prioritisation

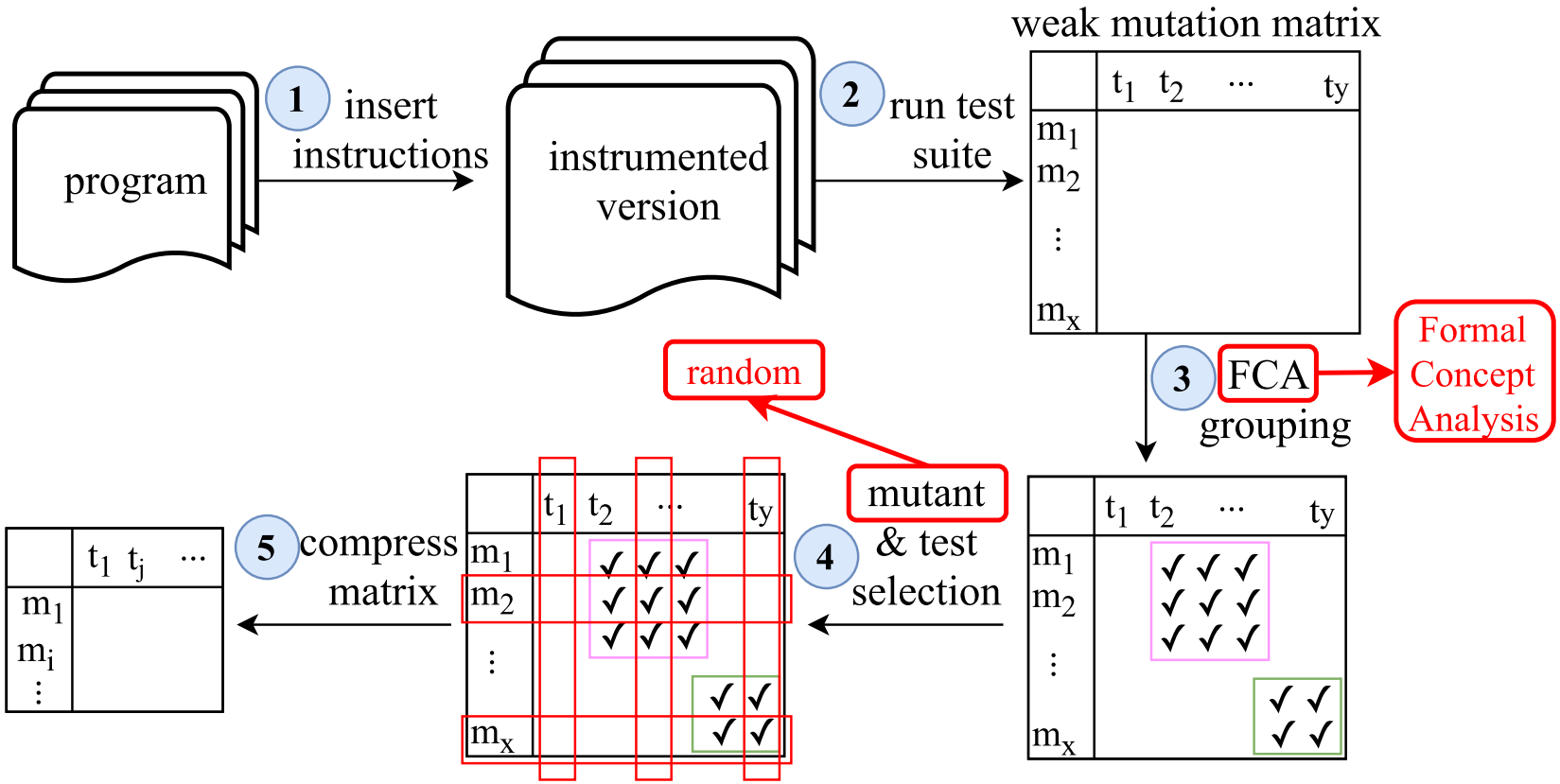
Mutation data compression

Assumption: mutants which have high similarity in weak mutation are very likely to have the same outcome in strong mutation.



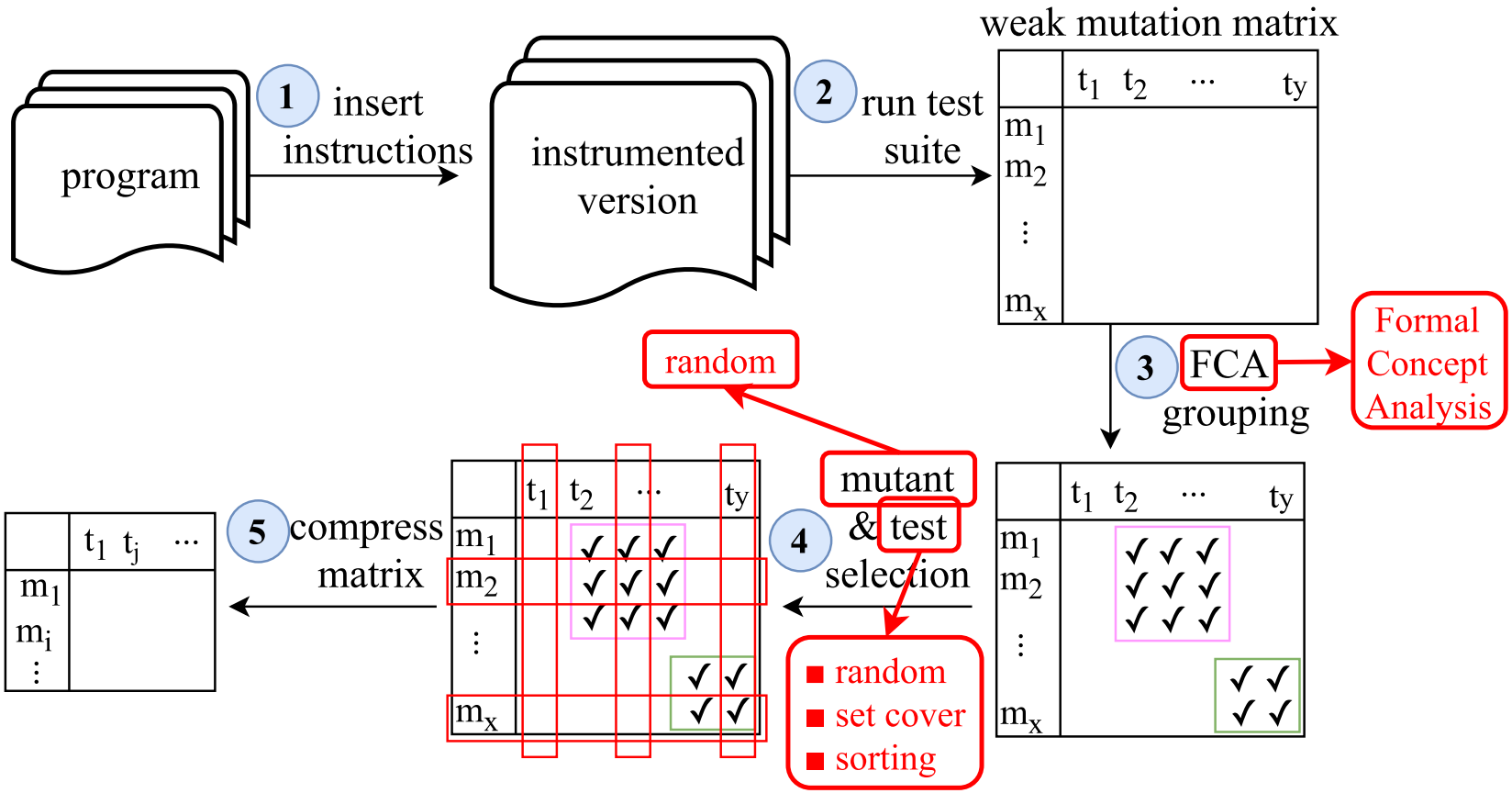
Mutation data compression

Assumption: mutants which have high similarity in weak mutation are very likely to have the same outcome in strong mutation.



Mutation data compression

Assumption: mutants which have high similarity in weak mutation are very likely to have the same outcome in strong mutation.



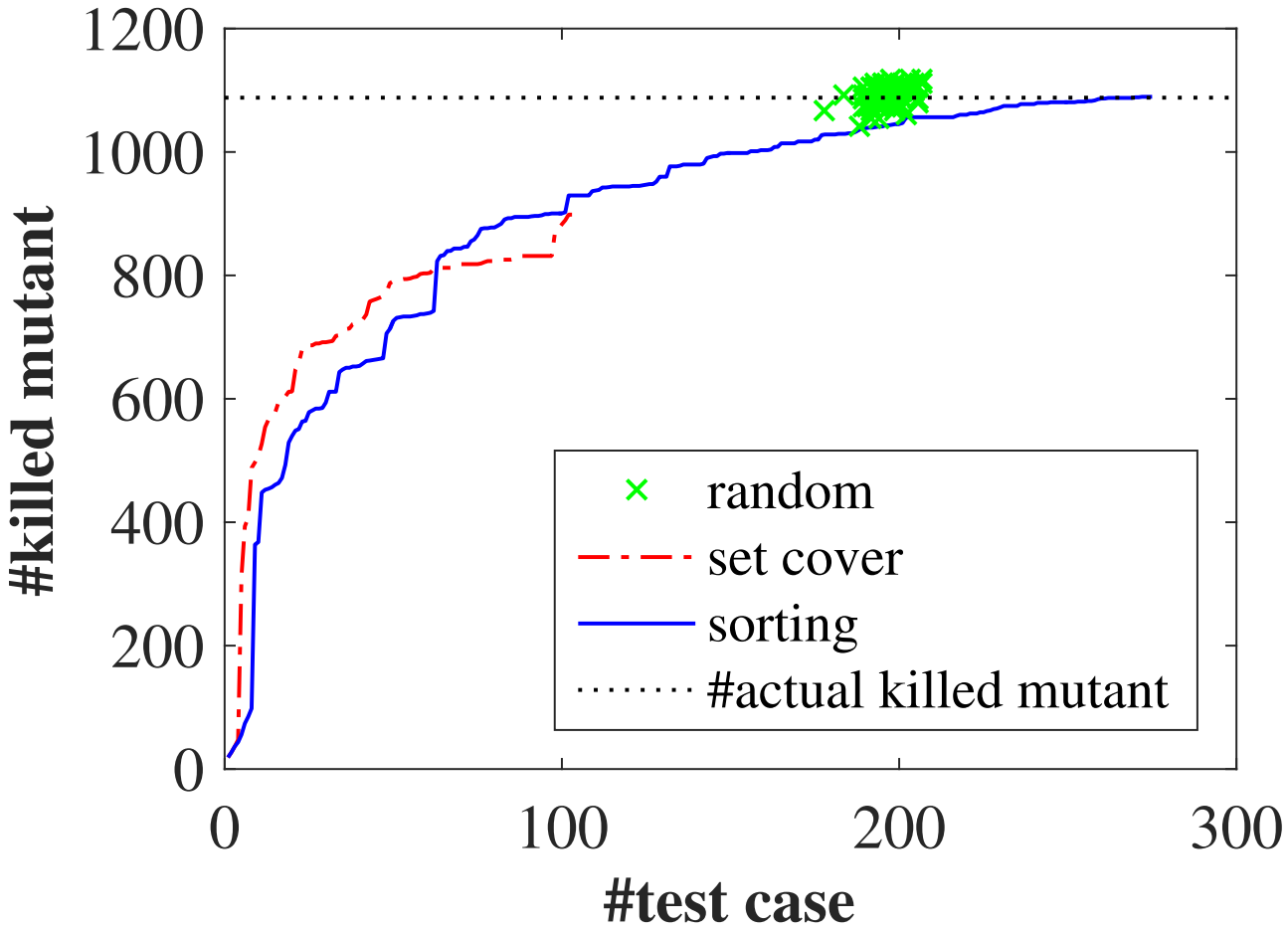
Experimental study

- **6 open-source projects**
- **automatically generated test suites (by Evosuite)**
- **comparison of test case selection**
  random, set cover, sorting
- **ComMT vs. other optimisations**
  coverage-based, infection-based

Results of experimental study

Comparison of test case selection in each stage for `jsecurity`

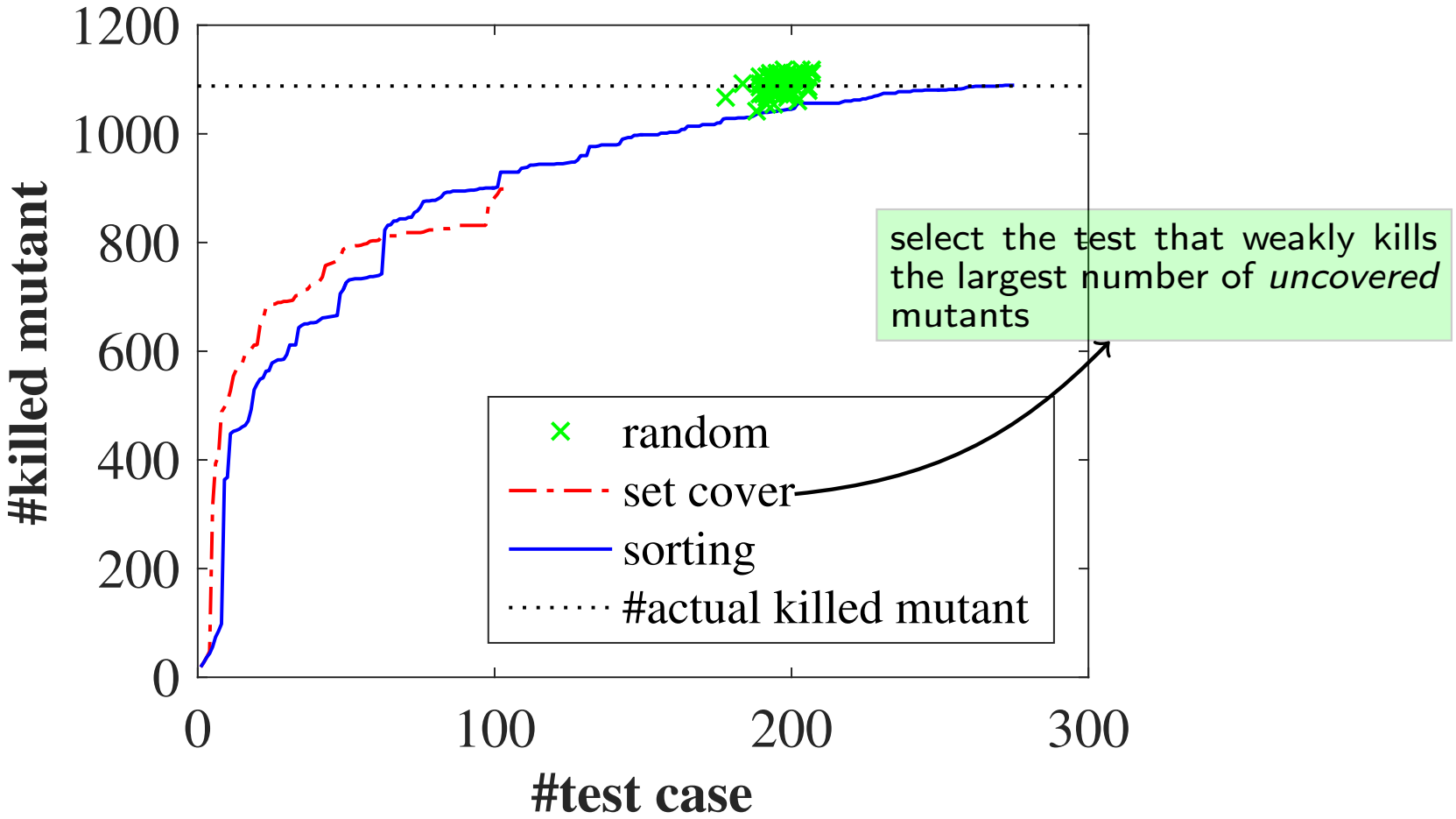
Based on random mutant selection



Results of experimental study

Comparison of test case selection in each stage for `jsecurity`

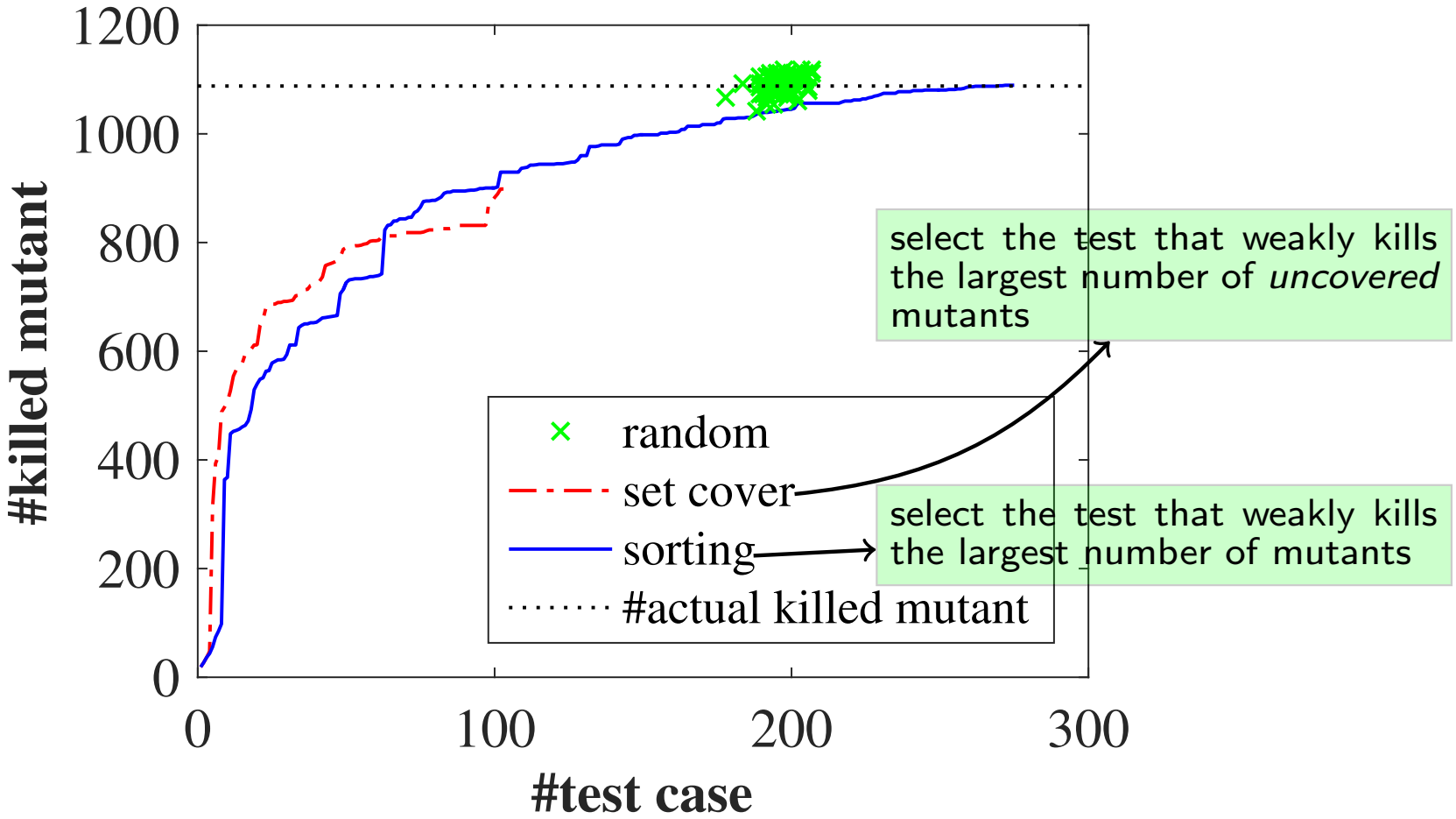
Based on random mutant selection



Results of experimental study

Comparison of test case selection in each stage for `jsecurity`

Based on random mutant selection



Results of experimental study

ComMT vs. other optimisations

Results of experimental study

ComMT vs. other optimisations

- **coverage-based:** baseline

Results of experimental study

ComMT vs. other optimisations

- **coverage-based:** baseline
- **infection-based:** 11.37% execution time reduction

Results of experimental study

ComMT vs. other optimisations

- **coverage-based:** baseline
- **infection-based:** 11.37% execution time reduction
- **ComMT:** trade-offs between execution time reduction and error rate (%)

mutant	test	red.	err.
random	no selection	83.93	0.257
	random	89.82	-19.36
	set cover	86.84	-4.76
	sorting	84.51	0.262

Results of experimental study

ComMT vs. other optimisations

- **coverage-based:** baseline
- **infection-based:** 11.37% execution time reduction
- **ComMT:** trade-offs between execution time reduction and error rate (%)

mutant	test	red.	err.
random	no selection	83.93	0.257
	random	89.82	-19.36
	set cover	86.84	-4.76
	sorting	84.51	0.262

Future work

- **Generalise to other test suites**
  Update: manually-written vs. generated

Future work

- **Generalise to other test suites**
  Update: manually-written vs. generated
- **Investigate other approximation methods**
  E.g. mutant subsumption, Principal Component Analysis (PCA)

Future work

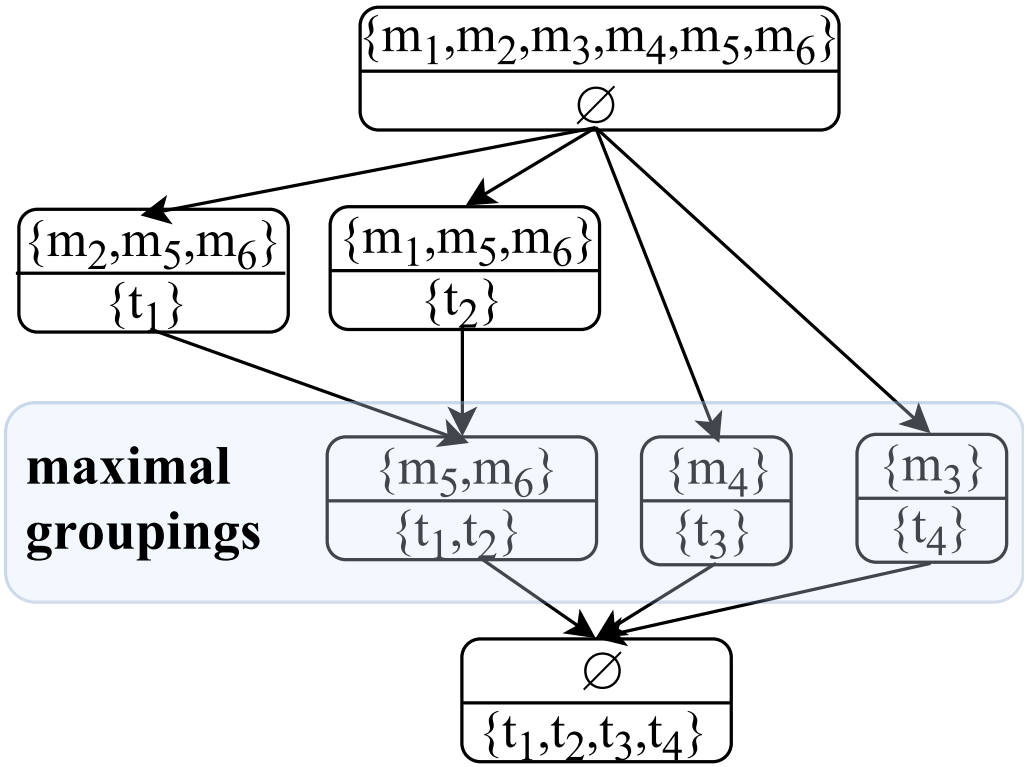
- **Generalise to other test suites**
  Update: manually-written vs. generated
- **Investigate other approximation methods**
  E.g. mutant subsumption, Principal Component Analysis (PCA)
- **Improve clustering accuracy**
  E.g. explore the relationships between clustering accuracy and other metrics

Future work

- **Generalise to other test suites**
  Update: manually-written vs. generated
- **Investigate other approximation methods**
  E.g. mutant subsumption, Principal Component Analysis (PCA)
- **Improve clustering accuracy**
  E.g. explore the relationships between clustering accuracy and other metrics
- **Implement on top of an existing mutation tool**

FCA grouping

	t_1	t_2	t_3	t_4
m_1	0	1	0	0
m_2	1	0	0	0
m_3	0	0	0	1
m_4	0	0	1	0
m_5	1	1	0	0
m_6	1	1	0	0



Results of experimental study

- **Trade-offs between execution time reduction and error rate (%):**

baseline

Project	Cov. based	Inf. based	ComMT							
			No selection		Set cover		Random		Sorting	
	Exec.	Red.	Red.	Err.	Red.	Err.	Red.	Err.	Red.	Err.
jsecurity	1.39 min	16.78	87.71	0.13	90.27	-17.42	88.55	-2.46	87.31	-0.04
summa	1.54 min	13.57	90.97	0.74	93.58	-14.1	92.18	-2.25	90.87	0.69
db-everywhere	0.02 min	3.65	59.29	-0.26	80.81	-18.83	70.43	-6.22	63.95	-0.26
noen	1.58 min	6.09	88.52	0.30	91.08	-30.69	89.52	-7.29	87.98	0.14
jtailgui	0.31 min	18.7	87.66	0.07	91.33	-13.11	89.86	-3.42	87.89	0.07
caloriecount	19.21 min	9.39	89.44	0.04	91.83	-21.99	90.5	-6.89	89.06	-0.38
Mean	-	11.37	83.93	0.257	89.82	-19.36	86.84	-4.76	84.51	0.262