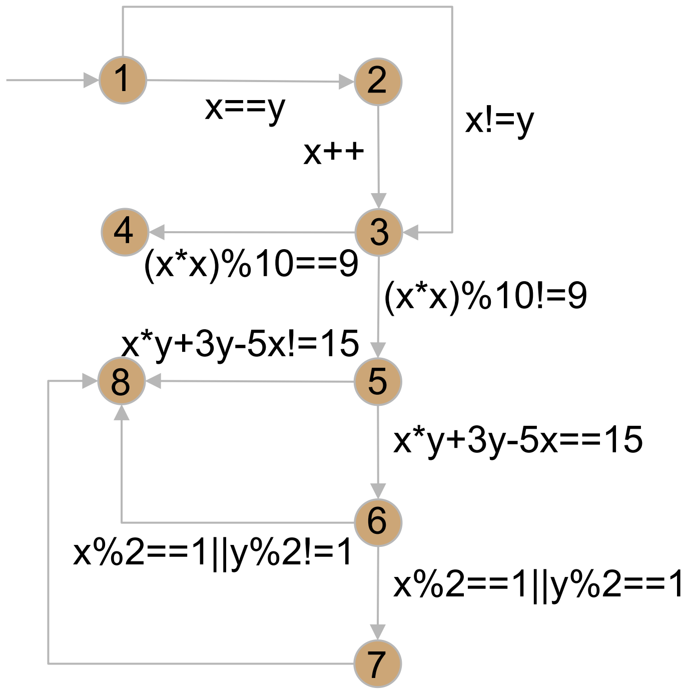
Towards Optimal Concolic Testing

Zhejiang University
Singapore U. Technology and Design
National U. Defence Technology
National U. Singapore

Example

```
void myfunc (int x, int y) {
1.      if (x==y) {
2.              x++;
        }
3.      if ((x*x)%10==9) {
4.                      return;
        }
5.      if (x*y+3*y-5*x==15) {
6.                      if (x%2==1 ||y%2==1) {
7.                                      x = x-y;
                }
        }
8.      return;
}
```

How to generate test cases to cover all statements?

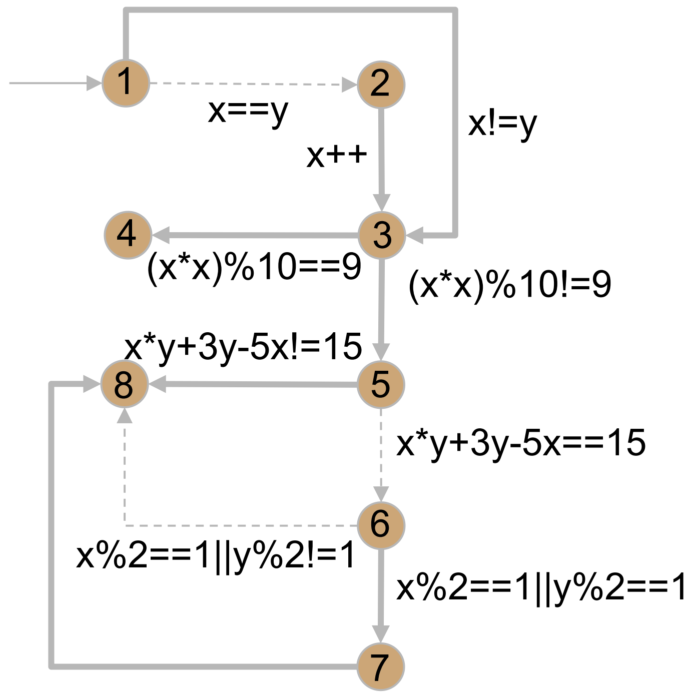


The Testing Problem

Option 1: Random testing

“Cheap” at covering high-probability program paths.

“Expensive” at covering low-probability program paths.



⇢ hard to cover

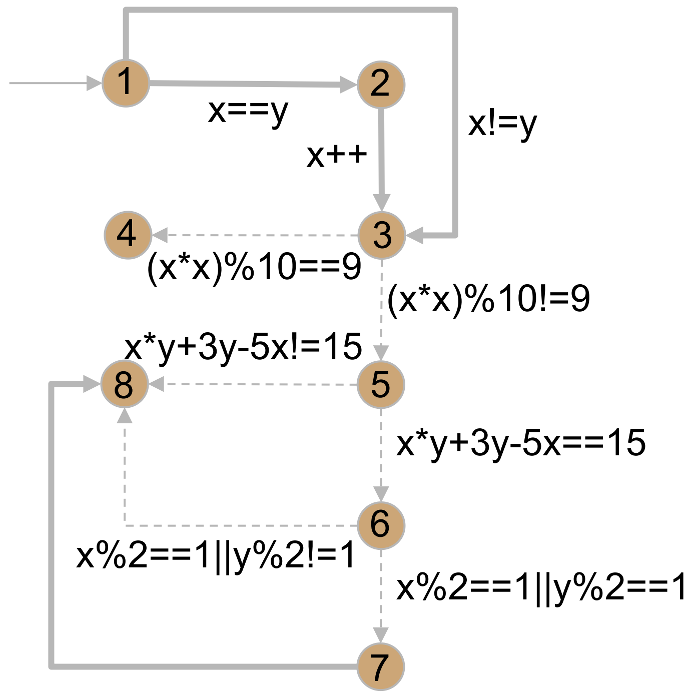
→ easy to cover

The Testing Problem

Option 2: Symbolic Execution

Relatively expensive.

Very expensive if the path condition is complicated.



⇢ hard to cover

→ easy to cover

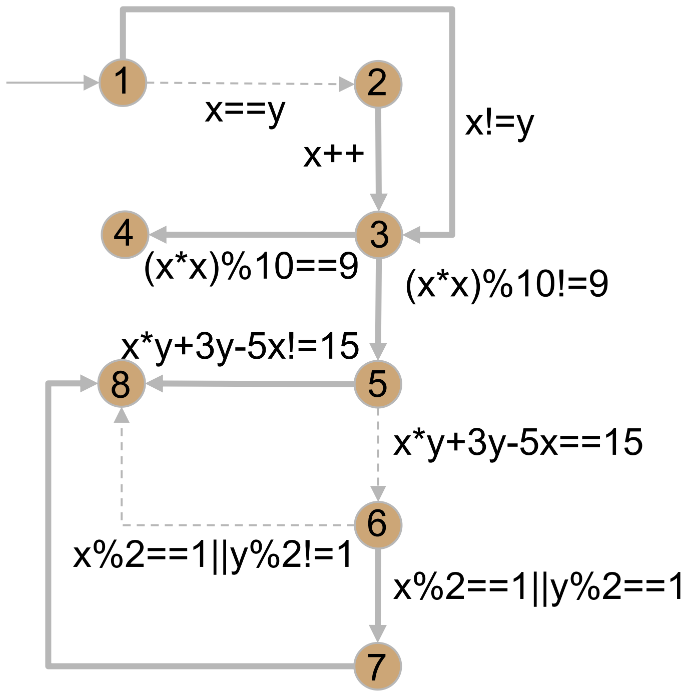
Concolic Testing = Symbolic + Random Testing

There are different strategies on combining symbolic testing and random testing.

Some are “better”, e.g.

- Solve path <1, 2> and path <1,3,5,6>
- Apply random testing to cover the rest.

How should we combine symbolic testing and random testing?



Existing Heuristics

Depth-first search (PLDI'05)

Generational searching (CACM'12)

Control-flow Directed Strategy (ASE'08)

Steering Symbolic Execution to Less Traveled Paths (OOPSLA'13)

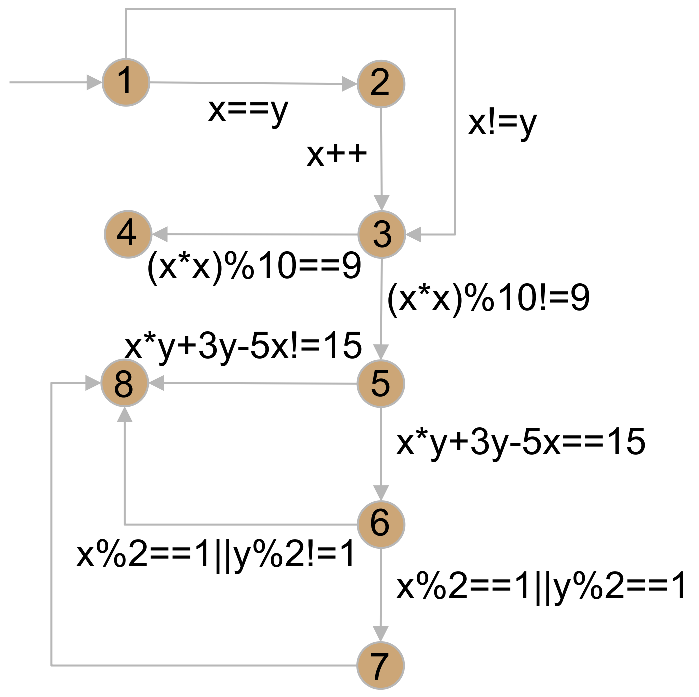
Random branch search (OSDI'07)

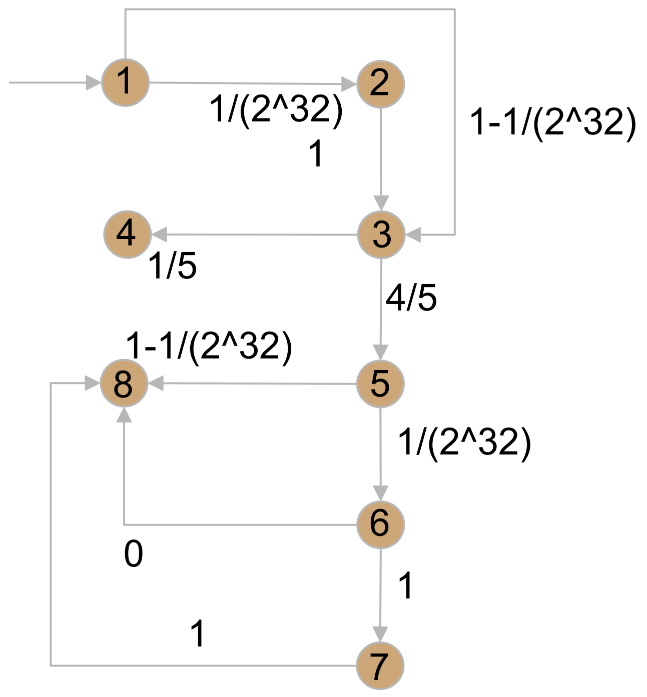
Automatically Generating Search Heuristics for Concolic Testing (ICSE'18)

…

These are all heuristics. Can we define what is the optimal strategy and propose better algorithms?

Understanding Random Testing



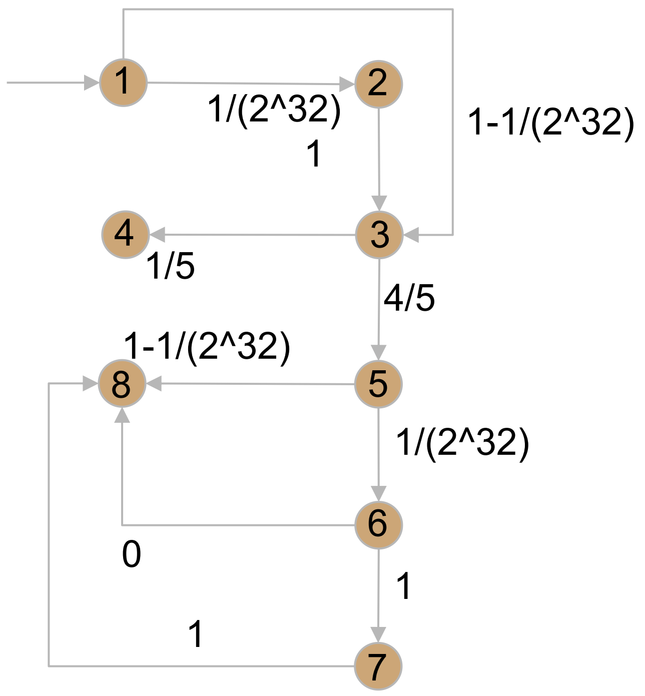


Understanding Random Testing

The effectiveness of random testing on covering a node depends on the probability of reaching the node.

Example:

The probability of covering node 4 is 1/5 which means on average 5 random test cases are needed to cover it.



Cost of Symbolic Execution

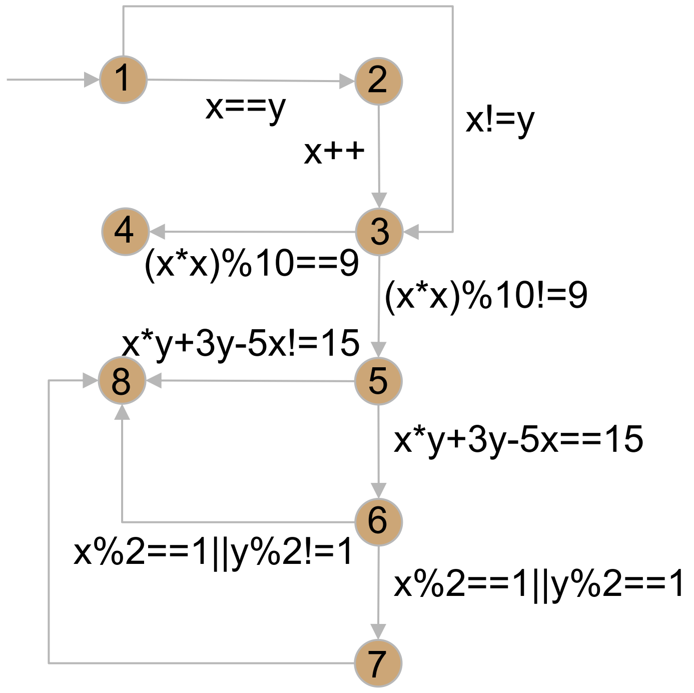
To measure the effectiveness of symbolic execution, we need to know the relative cost of symbolic execution.

Example

Generate a random test: cost 1 time unit.

Symbolically execute path <1,3>: cost 5 time units.

Symbolically execute path <1,3,5,6>: cost 50 time units.



Let’s Get Formal

The problem

A concolic testing strategy is a sequence of test case generation method <t1, t2, t3, ...>, each of which could be either random testing or applying symbolic execution to a certain program path.

Find an optimal strategy achieves maximum coverage and cost(t1) + cost(t2) + cost(t3) + ... is minimum.

The solution

The problem is reduced to a model checking problem of Markov Decision Processes with Costs.

If we know the program probability and the cost of symbolic execution (of each program path), the problem can be perfectly solved by model checking.

How Good Are Existing Heuristics?

Experiment Setting

Randomly generated DTMC models representing abstraction of programs.

Randomly generated costs representing the cost of symbolic execution for each path.

Existing approaches have a lot of room to improve!

	5 states	10 states	15 states	20 states
Optimal	1	1	1	1
RT	138.9	11.3	44.2	114.7
RCN	1.7	14.4	15.1	12.7
RSS	12.8	50.7	64.0	68.1
RPS	12.8	50.6	63.9	68.5
DFS	7.1	27.4	21.8	18.6
DART	1.8	13.0	12.8	13.0
GS	1.9	13.5	13.9	13.3
CGS	1.8	12.6	13.6	13.8
SGS	11.2	32.4	29.4	25.5

Let’s Get Practical

Problem 1: How do we know the program probability?

Through estimation: Laplace or Good-Turing.

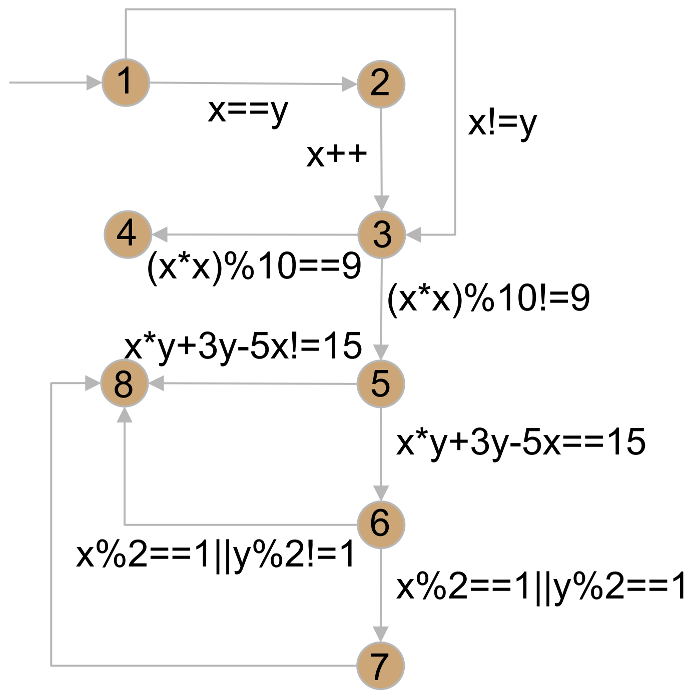
Problem 2: How do we know the cost of symbolic execution?

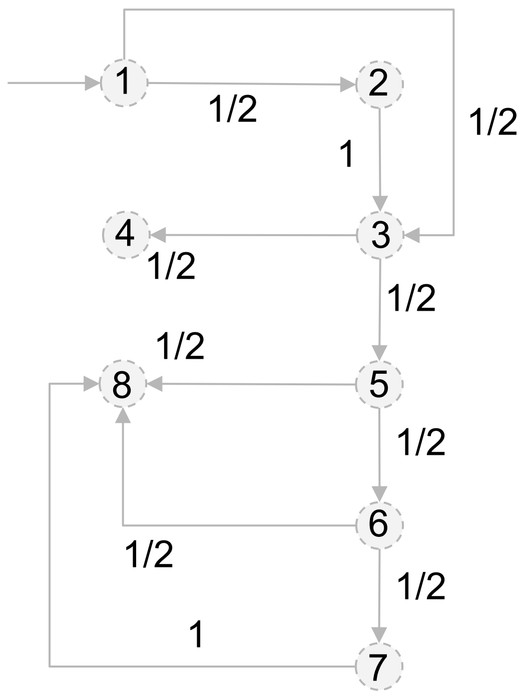
Empirically.

Problem 3: Model checking is too expensive!

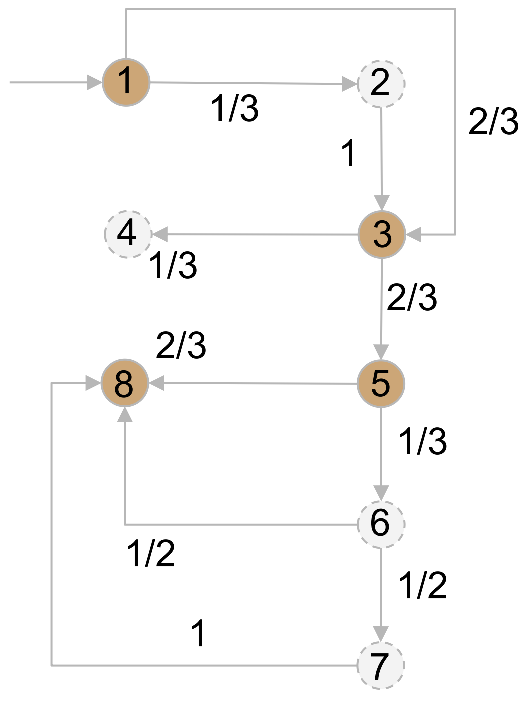
Be greedy. Look for local optimal.

The Example: Initial Estimation





An Example



Assume the random test case covers 1, 3, 5 and 8.

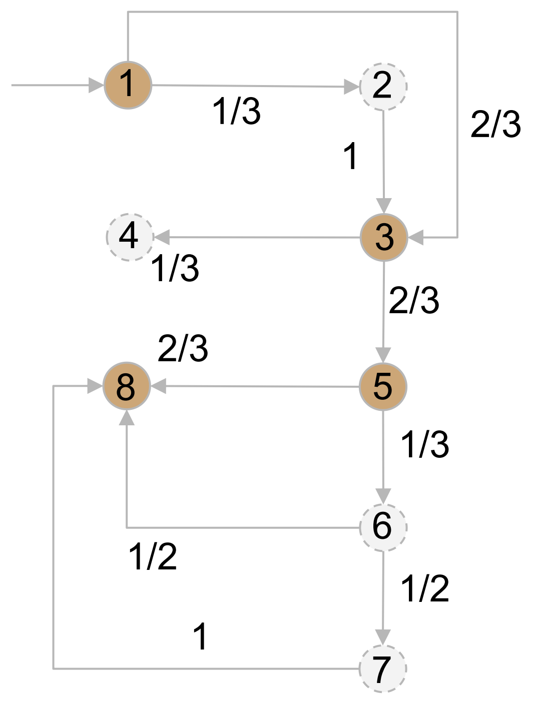
Apply Laplace estimation to update the DTMC.

Compute the reward of random testing by solving the following equation system:

$R_1 = 1/3 * R_2 + 2/3 * R_3$
$R_2 = R_3$
$R_3 = 1/3 * R_4 + 2/3 * R_5$
$R_4 = 1$
$R_5 = 2/3 * R_8 + 1/3 * R_6$
$R_6 = 1 + 1/2 * R_8 + 1/2 * R_7$
$R_7 = 1 + R_8$
$R_8 = 0$

Solution:
R1 = 1; R2 = 5/3;
R4=1; R6 = 3/2

An Example



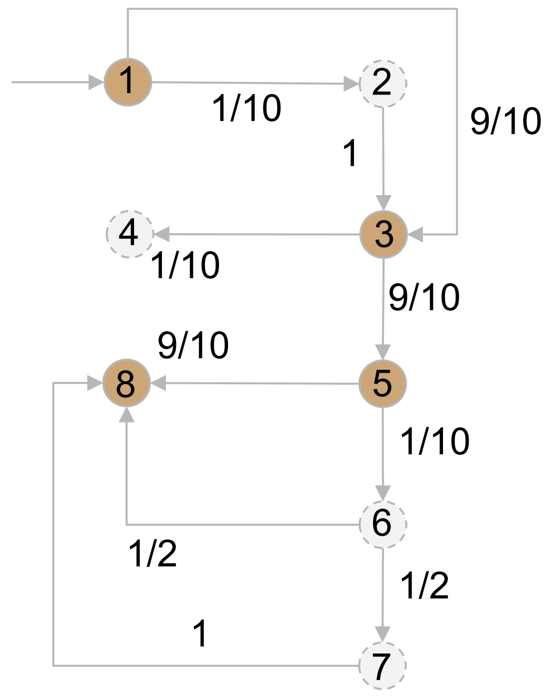
Compute the reward of applying symbolic execution:

Solving path <1, 2>: cost 4, reward 5/3
Solving path <1, 3, 4>: cost 50, reward 1
Solving path <1, 3, 5, 6>: cost 50, reward 3/2

Random testing: cost 1, reward 1

Thus, random testing.

An Example



After 8 random test case, assuming all of them visit <1,3,5,8>, the DTMC is updated as shown on the left.

Random testing: cost 1, reward 0.24

Symbolic Execution on <1,2>: cost 4, reward 1.14
Symbolic Execution on <1,3,4>: cost 50, reward 1
Symbolic Execution on <1,3,4,6>: cost 50, reward 3/2

Thus, symbolic Execution on <1,2>.

repeat until we cover all reachable nodes or timeout.

Evaluation

Experiment 1

Randomly generated DTMC models representing abstraction of programs.

Randomly generated costs representing the cost of symbolic execution for each path.

	5 states	10 states	15 states	20 states
Optimal	1	1	1	1
RT	138.9	11.3	44.2	114.7
RCN	1.7	14.4	15.1	12.7
RSS	12.8	50.7	64.0	68.1
RPS	12.8	50.6	63.9	68.5
DFS	7.1	27.4	21.8	18.6
DART	1.8	13.0	12.8	13.0
GS	1.9	13.5	13.9	13.3
CGS	1.8	12.6	13.6	13.8
SGS	11.2	32.4	29.4	25.5
G	2.1	4.8	3.1	4.8

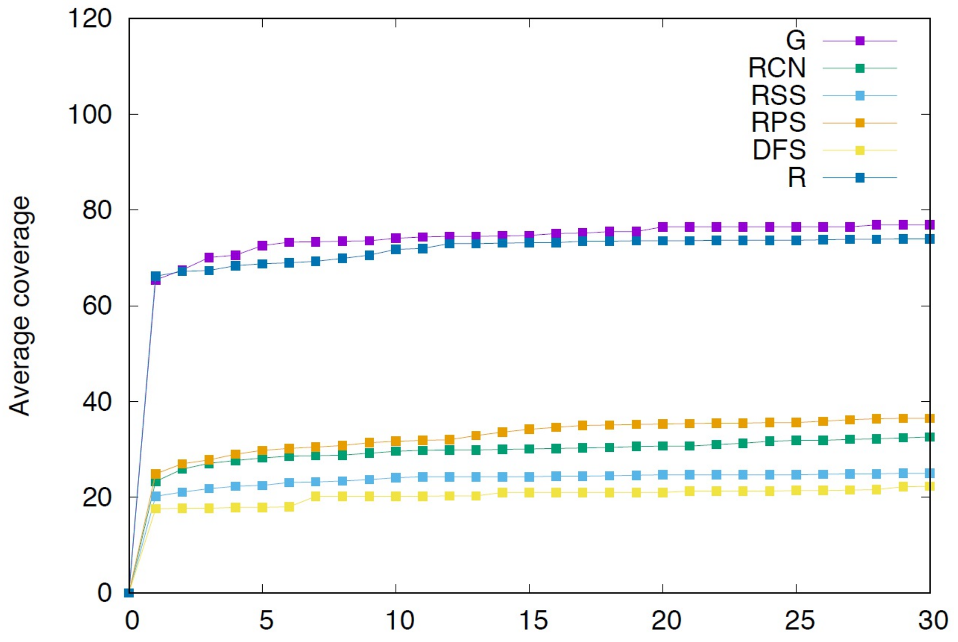
Implementation and Evaluation

Experiment 2

We extend KLEE with our strategy.

We use a set of scientific program as test subjects.

We measure the coverage achieved by different strategies over time.



Conclusion

We define and solve the problem of concolic testing.

We propose a practical algorithm.

There is room to improve.