2018 IEEE Congress on Evolutionary Computation

Mutation-based Evolutionary Fault Localization

Diogo M. de-Freitas (UFG), Plinio S. Leitao-Junior (UFG),
Celso G. Camilo-Junior (UFG) and Rachel Harrison (Oxford Brookes)

CAPES

UFG

INSTITUTO DE INFORMÁTICA UFG

OXFORD BROOKES UNIVERSITY

I4Soft Intelligence for Software

Context

Problem

- Software contain faults;
- Verification, testing and debugging takes from 50% to 75% of the project's budget;
- Average time to fix a security-critical error: 28 days;
- In 2005, a Mozilla developer reported that almost 300 bugs would appear everyday;
- The annual cost of software faults in the US was $59.5 billion by 2006.

Context

Problem

ID	Code	$S(e)$
1	`if (n % 2 == 0):`	0.5
2	`    x = n + 1 # bug`	1
3	`else:`	0
4	`    x = n - 1`	0

Context

Problem

- Fault localization is the process of identifying the location of software faults observed during testing;
- It takes a significant amount of time during development;
- Precedes program repair;
- Directly impact software cost and quality.

Context

SBFL Heuristics

- Spectrum-based Fault Localization;
- To determine the probability that a code element is faulty;
- c_{ep} number of **successful** executions that **cover** a certain element;
- c_{ef} number of **failed** executions that **cover** a certain element;
- c_{np} number of **successful** executions that **don't cover** a certain element;
- c_{nf} number of **failed** executions that **don't cover** a certain element.

Context

Example

ID	Code	n = 1	n = 2	n = 3	n = 4
1	`if (n % 2 == 0):`	✓	✓	✓	✓
2	`    x = n + 1 #  bug`		✓		✓
3	`else:`	✓		✓	
4	`    x = n - 1`	✓		✓	

Context

Example

$$S(e)_{tarantula} = \frac{\frac{c_{ef}}{c_{ef}+c_{nf}}}{\frac{c_{ef}}{c_{ef}+c_{nf}} + \frac{c_{ep}}{c_{ep}+c_{np}}}$$

ID	Code	c_{ep}	c_{ef}	c_{np}	c_{nf}	$S(e)$
1	`if (n % 2 == 0):`	2	2	0	0	0.5
2	`    x = n + 1 # bug`	0	2	2	0	1
3	`else:`	2	0	0	2	0
4	`    x = n - 1`	2	0	0	2	0

Context

Mutants Analysis

ID	Original Code	Mutant
1	`if (n % 2 == 0):`	`if (n % 4 == 0):`
2	`    x = n`	`    x = n`
3	`else:`	`else:`
4	`    x = n - 1`	`    x = n - 1`

Tests

n = 1 (live)

n = 2 (killed)

n = 3 (live)

n = 4 (live)

Context

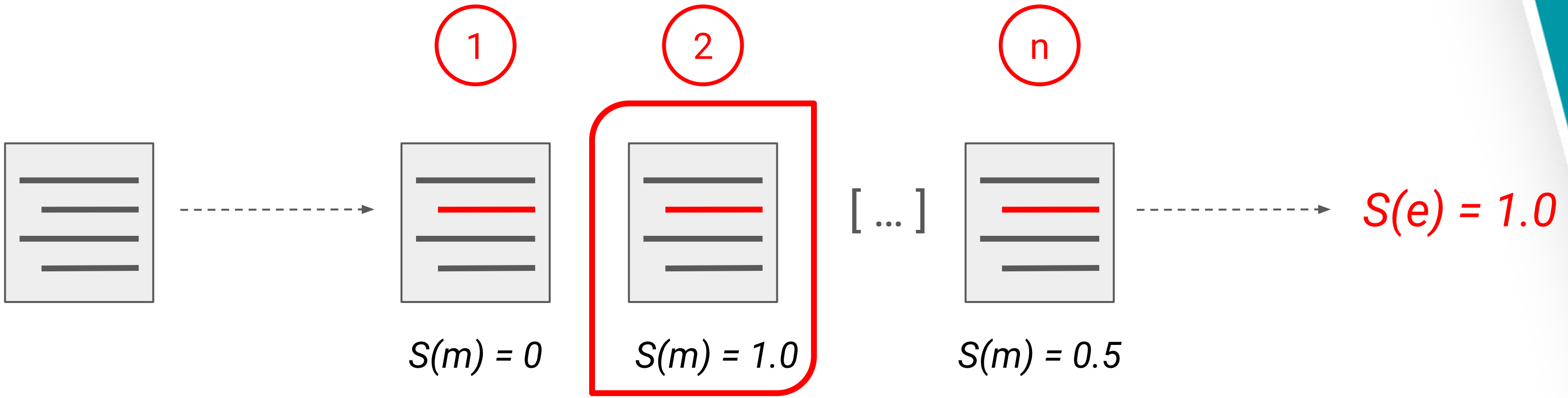
MBFL Heuristics

- m_{kf} number of **negative** test cases that **killed** the mutant;
- m_{kp} number of **positive** test cases that **killed** the mutant;
- m_{nf} number of **negative** test cases that **didn't kill** the mutant;
- m_{np} number of **positive** test cases that **didn't kill** the mutant.

$$S(e)_{mbfl-tarantula} = \frac{\frac{m_{kf}}{m_{kf}+m_{nf}}}{\frac{m_{kf}}{m_{kf}+m_{nf}} + \frac{m_{kp}}{m_{kp}+m_{np}}}$$

Context

Example



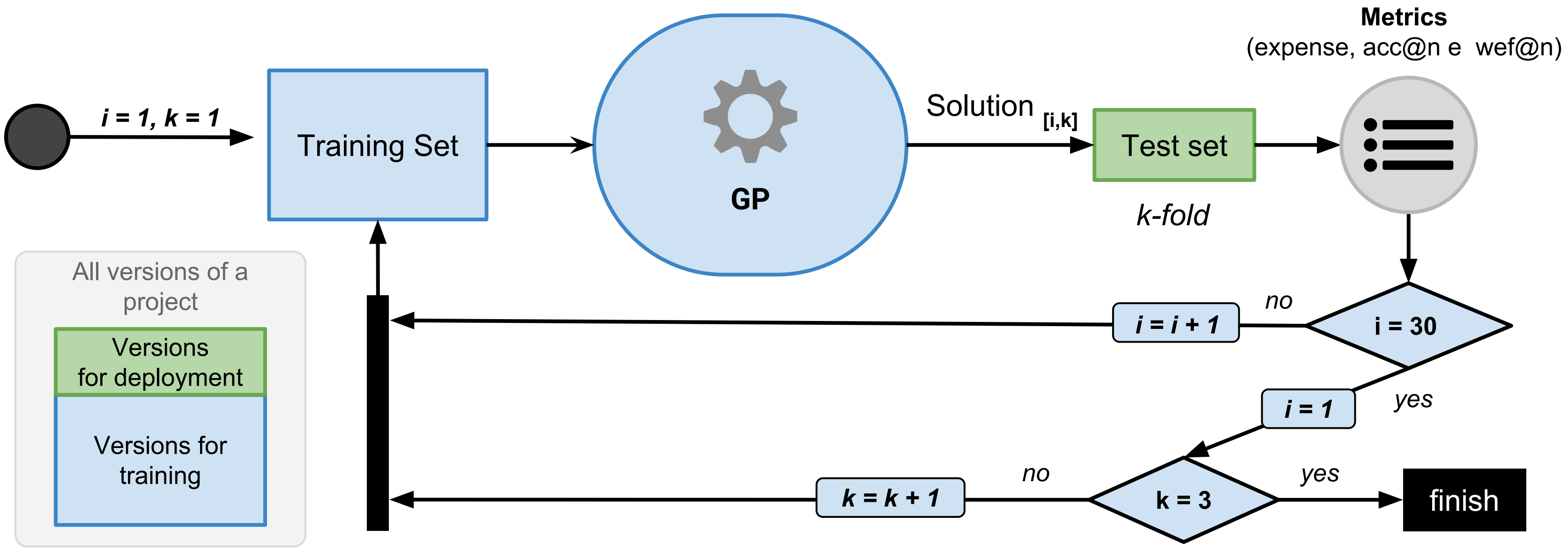
Proposal

GP Generated MBFL Heuristics

- An evolutionary approach to compose program-oriented MBFL heuristics automatically.
- Terminals: {m_{kf}, m_{kp}, m_{nf}, m_{np} e 1}
- Functions {+, -, •, ÷ and √ }
- Fitness function: the mean position of the faulty statement relative to the number of statements.

$$f(x) = \frac{\sum_{i=1}^{n} expense(i)}{n}$$

Experiments



Benchmarks

1. **Codeflaws:**
   a. 475-A;
2. **Siemens**:
   a. Printtokens2;
   b. Scheduleand;
   c. Tcas;
3. **Defects4J**:
   a. Math.

Baselines

1. GP-cov (SBFL);
2. Tarantula (cov & mut);
3. Ochiai (cov & mut);
4. OP^2 (cov & mut);
5. Barinel (cov & mut);
6. DStar (cov & mut).

11 Baselines

Results

RQ1: *How effective are the GP-evolved solutions based on mutation variables from a relative perspective?*

Technique	expense
TAR-cov	47.14%
TAR-mut	43.98%
OCH-cov	46.48%
OCH-mut	53.60%
OP2-cov	40.27%
OP2-mut	59.86%

Technique	expense
BAR-cov	48.37%
BAR-mut	43.70%
DST-cov	48.02%
DST-mut	46.78%
GP-cov	**30.00%**
GP-mut	**36.38%**

$$expense = \frac{rank(e)}{elements}$$

Results

RQ2: ***How effective are the GP-evolved solutions based on mutation variables from an absolute perspective?***

Technique	acc@1	acc@3	acc@5
TAR-cov	1	8	10
TAR-mut	2	8	14
OCH-cov	2	9	14
OCH-mut	3	6	10
OP2-cov	3	11	15
OP2-mut	0	2	7
GP-cov	0.33	4.63	0.17
GP-mut	**6.47**	**15.40**	**20.54**

Results

expense	acc@1	acc@3	acc@5
GP-cov: 2.75	**GP-mut: 4.5**	**GP-mut: 1.5**	**GP-mut: 3**
OP2-cov: 4	OP2-cov: 8.25	OP2-cov: 6.5	OP2-cov: 6
GP-mut: 4.5	GP-cov: 9.5	OCH-cov: 7.5	OCH-cov: 6.25
OCH-cov: 4.75	OCH-mut: 9.5	TAR-cov: 8.5	GP-cov: 7.5

Results

RQ3: How does mutation spectra quality impact FL ability?

- α: all original versions in the repository (77 versões);
- β: the versions whose minimum average number of mutants is at least 3 (37 versões);
- γ: set β without faults of omission faults (22 versões).

Results

RQ3: How does mutation spectra quality impact FL ability?

Set	Technique	expense	Avg. acc@1	Avg. acc@3	Avg. acc@5
α	GP-cov	4.82%	0.26	0.53	0.65
	GP-mut	14.41%	0.23	0.43	0.55
β	GP-cov	4.50%	0.25	0.48	0.61
	GP-mut	8.76%	0.37	0.67	0.81
γ	GP-cov	4.86%	0.10	0.27	0.40
	GP-mut	13.17%	0.27	0.48	0.57

Final Remarks

- The empirical analysis along with statistical analysis show that the proposal is competitive in both perspectives (relative and absolute);
- The proposal scored the faulty element on top more frequently;
- The minimum average number of mutants improved the performance of the evolved MBFL heuristics;

Final Remarks

- Specialisation of suspiciousness formulae for certain contexts;
- Application of mutation spectra to GP-evolved formulae;
- Coverage spectra versus mutation spectra in the context of evolutionary approaches;
- Mutation quantity impact MBFL performance.

Thanks!

Questions and Suggestions

celso@inf.ufg.br

CAPES

UFG

INSTITUTO DE INFORMÁTICA UFG

OXFORD BROOKES UNIVERSITY

I4Soft Intelligence for Software