Evaluating and Improving Fault Localization

Spencer Pearson[†], José Campos[*], **René Just**, Gordon Fraser[*], Rui Abreu[‡], Michael D. Ernst[†], Deric Pang[†], Benjamin Keller[†]

University of Massachusetts
†University of Washington
*University of Sheffield
‡University of Lisbon

May 25, 2017

Fault localization: an important problem

Two use cases: developers and automated program repair

Many techniques and evaluations

Prior studies (winner > loser)
Ochiai > Tarantula [NLR11], [LTL], [WDGL14], [XM14], [LLT15]
Barinel > Ochiai [AZVG09]
Barinel > Tarantula [AZVG09]
Op2 > Ochiai [NLR11]
Op2 > Tarantula [NLR11], [MKKY14]
DStar > Ochiai [WDGL14], [LLT15]
DStar > Tarantula [WDGL14], [JJC+14], [LLT15]
Metallaxis > Ochiai [PLT15]
MUSE > Op2 [MKKY14]
MUSE > Tarantula [MKKY14]

Fault localization: an important problem

Two use cases: developers and automated program repair

Many techniques and evaluations

Prior studies (winner $>$ loser)
Ochiai $>$ Tarantula [NLR11], [LTL], [WDC
Barinel $>$ Ochiai [AZVG09]
Barinel $>$ Tarantula [AZVG09]
Op2 $>$ Ochiai [NLR11]
Op2 $>$ Tarantula [NLR11], [MKKY14]
DStar $>$ Ochiai [WDGL14], [LLT15]
DStar $>$ Tarantula [WDGL14], [JJC+14],
Metallaxis $>$ Ochiai [PLT15]
MUSE $>$ Op2 [MKKY14]
MUSE $>$ Tarantula [MKKY14]

Do these results hold for real world programs?

Fault localization: an important problem

Two use cases: developers and automated program repair

Many techniques and evaluations

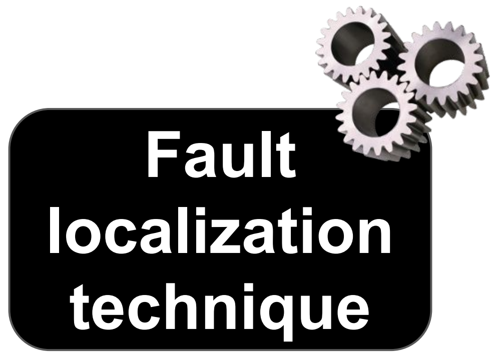
Prior studies (winner > loser)
Ochiai > Tarantula [NLR11], [LTL], [WDC
Barinel > Ochiai [AZVG09]
Barinel > Tarantula [AZVG09]
Op2 > Ochiai [NLR11]
Op2 > Tarantula [NLR11], [MKKY14]
DStar > Ochiai [WDGL14], [LLT15]
DStar > Tarantula [WDGL14], [JJC+14],
Metallaxis > Ochiai [PLT15]
MUSE > Op2 [MKKY14]
MUSE > Tarantula [MKKY14]

Do these results hold for real world programs? ***NO!***

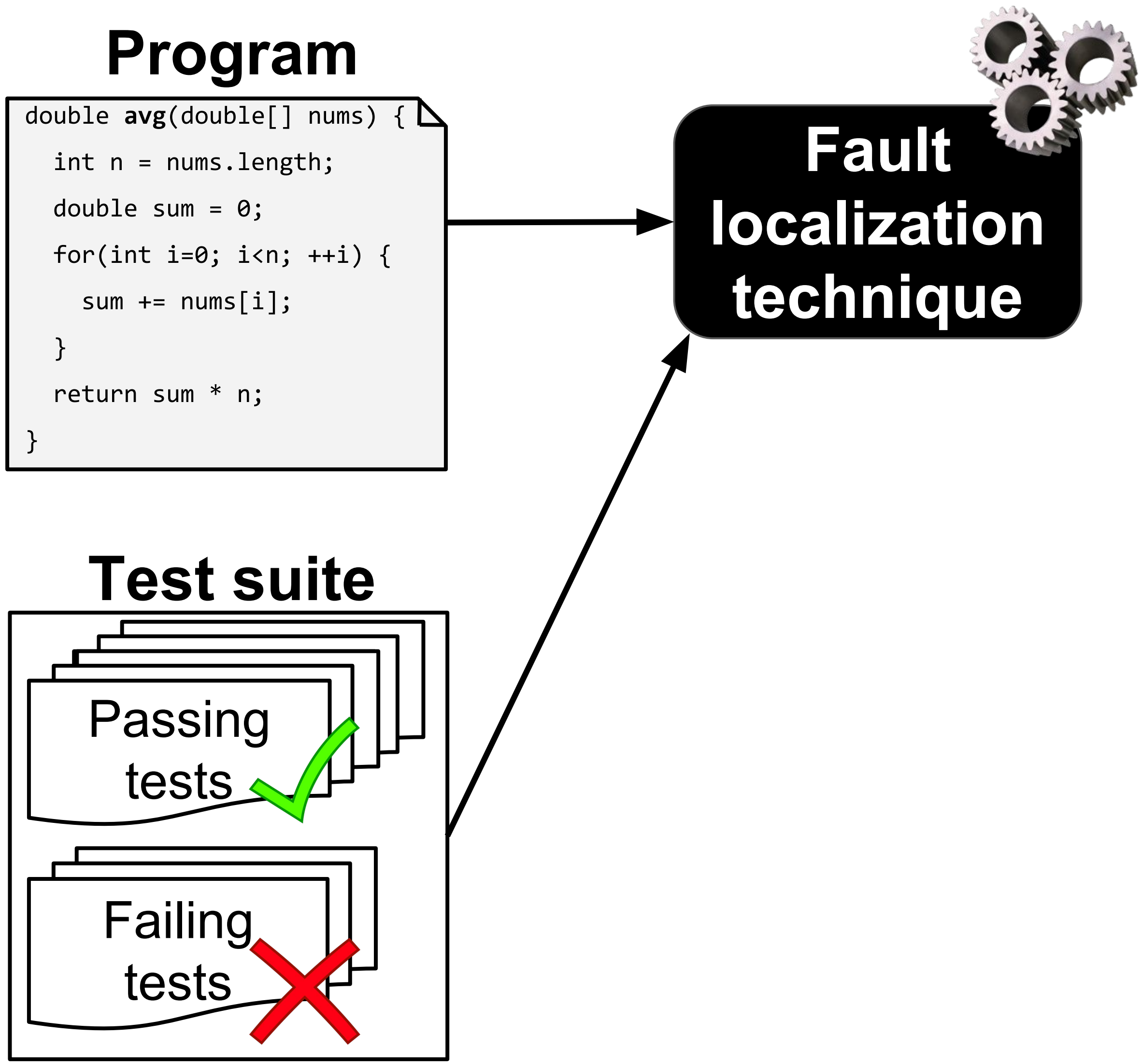
Why?

1. Unrealistic evaluations (artificial faults)
2. Negligible or small effect sizes
3. Unrealistic evaluation metrics

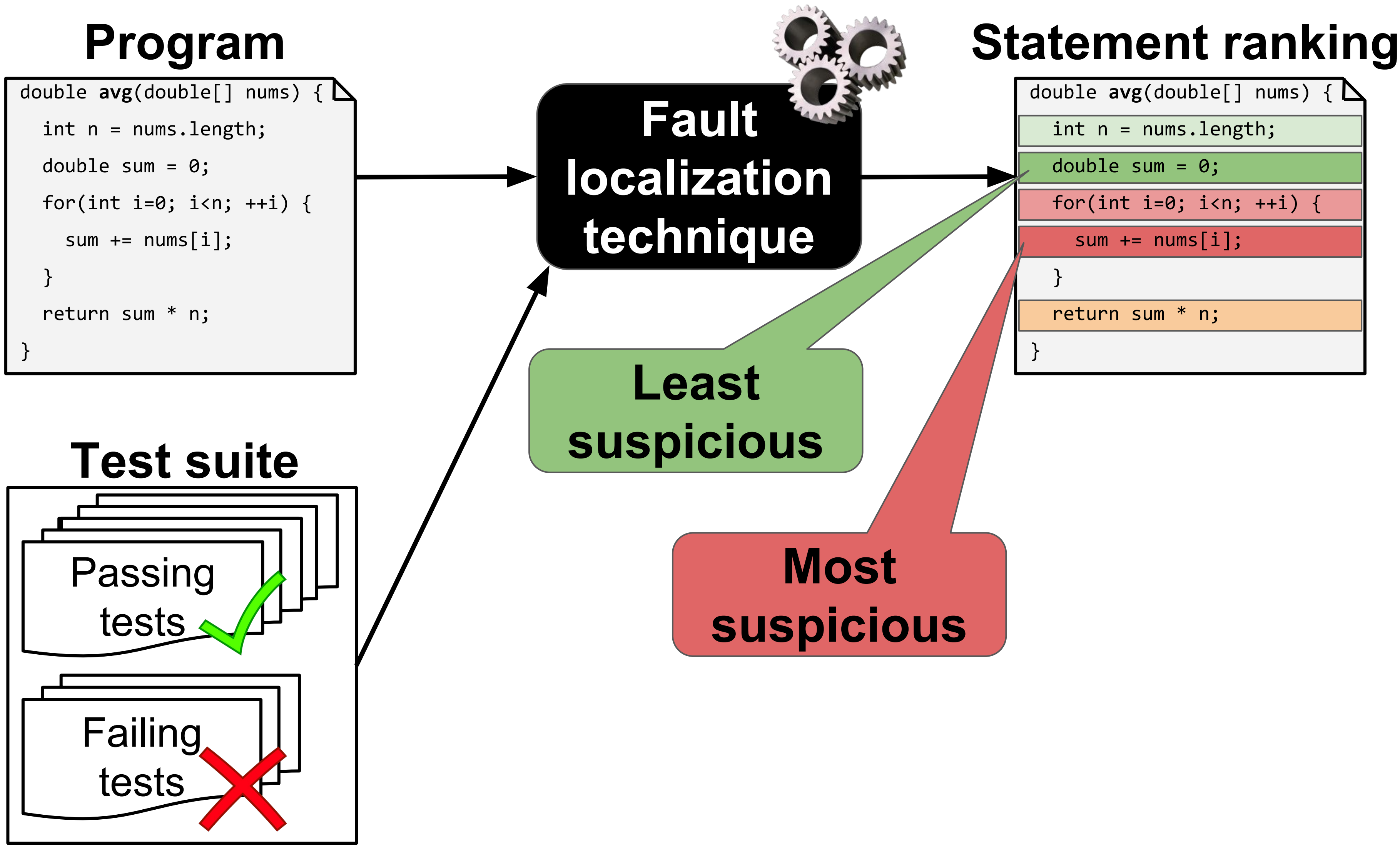
What is fault localization?



What is fault localization?



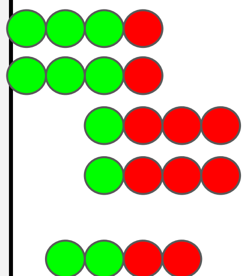
What is fault localization?



Fault localization: how it works

Program

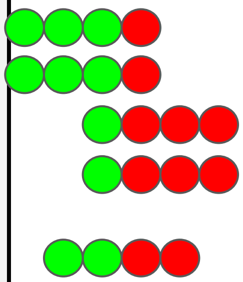
```
double avg(double[] nums) {
  int n = nums.length;
  double sum = 0;
  for(int i=0; i<n; ++i) {
    sum += nums[i];
  }
  return sum * n;
}
```



Spectrum-based fault localization

Program

```
double avg(double[] nums) {
  int n = nums.length;
  double sum = 0;
  for(int i=0; i<n; ++i) {
    sum += nums[i];
  }
  return sum * n;
}
```



Spectrum-based FL (SBFL)

- **Compute** suspiciousness **per statement**
- Example:

$$S(s) = \frac{failed(s)/totalfailed}{failed(s)/totalfailed + passed(s)/totalpassed}$$

● Statement **covered** by **failing test**

● Statement **covered** by **passing test**

More ● → statement is more suspicious!

Jones et al., *Visualization of test information to assist fault localization*, ICSE'02

Mutation-based fault localization

Program

```
double avg(double[] nums) {
  int n = nums.length;
  double sum = 0;
  for(int i=0; i<n; ++i) {
    sum += nums[i];
  }
  return sum * n;
}
```

Mutants

```
double avg(double[] nums) {
  int n = nums.length;
  double sum = 0;
  for(int i=0; i<n; ++i) {
    sum += nums[i];
  }
  return sum + n;
}
```

Mutation-based FL (MBFL)

- **Compute** suspiciousness **per mutant**
- **Aggregate** results **per statement**
- Example:

$$S(s) = \max_{m \in mut(s)} \frac{failed(m)}{\sqrt{totalfailed \cdot (failed(m) + passed(m))}}$$

Mutant **affects failing test** outcome

Mutant **breaks passing test**

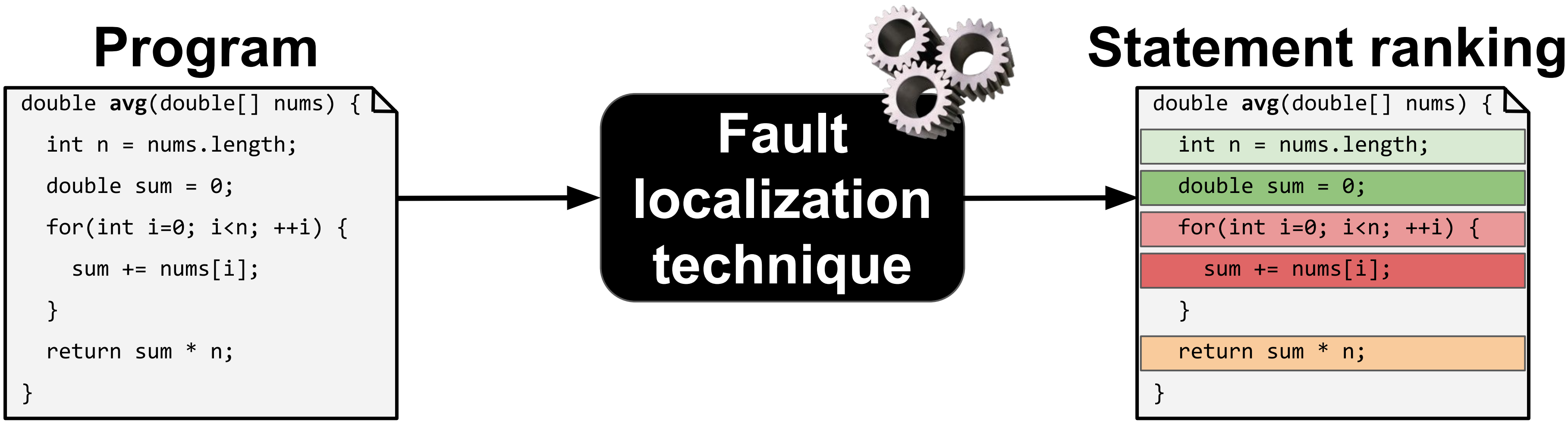
More → mutant is more suspicious!

Papadakis and Traon, *Metallaxis-FL: mutation-based fault localization*, STVR'15

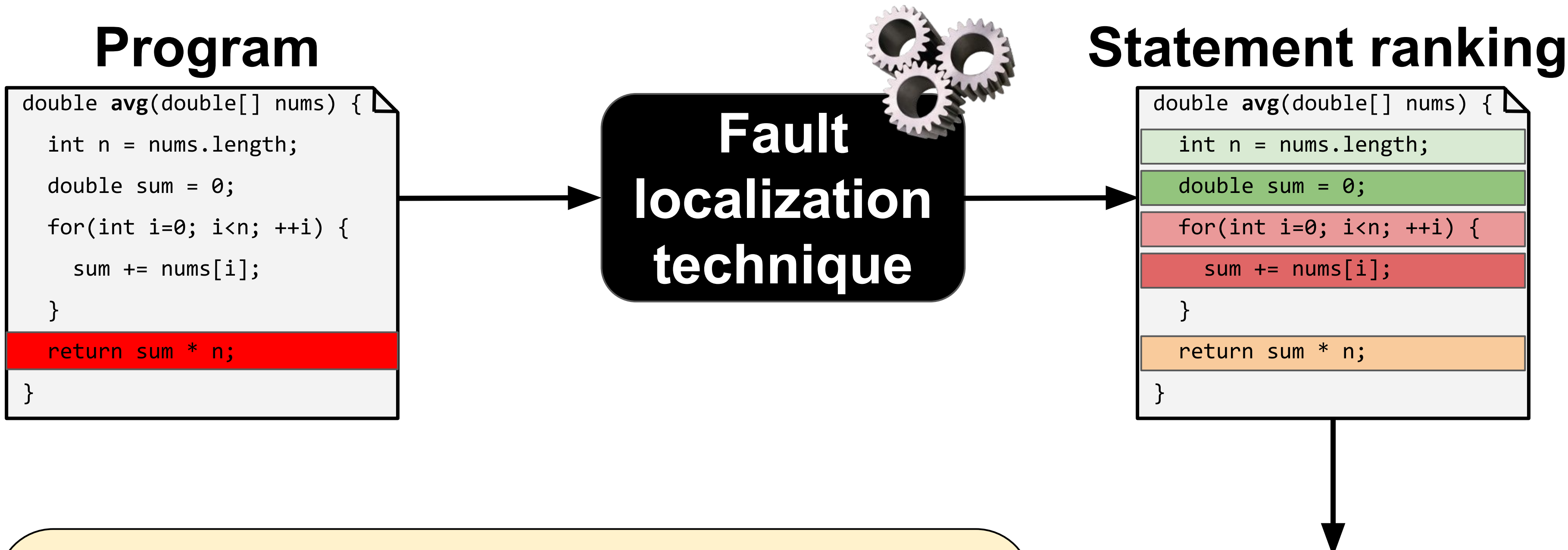
Outline and contributions

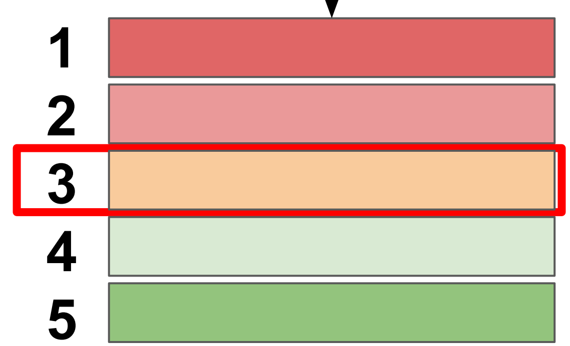
- **How to evaluate** fault localization techniques?
- **Empirical study** on artificial and real faults:
  - Do the results agree with prior work?
  - Do the results agree on artificial and real faults?
  - *No!* Explain why not.
- **What design decisions matter** (on real faults)?
- **How to improve** fault localization?

Evaluating fault localization techniques



Evaluating fault localization techniques





EXAM score: relative rank of the defective statement (e.g., **3/5 = 0.6**).

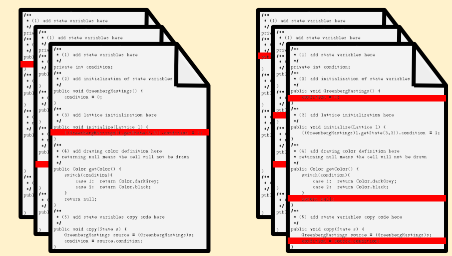
Smaller EXAM scores are better!

Evaluating fault localization techniques

Not straightforward for real faults:

- Multi-line defects (localize 1 or all lines?)
- Non-executable code (declarations)
- Fault of omission (>1 possible location)

Details in the paper



EXAM score: relative rank of the defective statement (e.g., **3/5 = 0.6**).

Smaller EXAM scores are better!

Empirical study on artificial and real faults

Experimental design

- **7** widely studied **FL techniques**
  - **SBFL**: Barinel, D*, Ochiai, Op2, and Tarantula
  - **MBFL**: Metallaxis and Muse
- **310 real faults** (5 times as many as prior studies combined)
- **2995 artificial faults** (more than prior studies combined)
- 100,000 CPU hours (MBFL is expensive)

http://www.defects4j.org *http://www.mutation-testing.org*

Results of prior studies

	Prior studies (winner > loser)
SBFL vs. SBFL	Ochiai > Tarantula
	Barinel > Ochiai
	Barinel > Tarantula
	Op2 > Ochiai
	Op2 > Tarantula
	DStar > Ochiai
	DStar > Tarantula
MBFL vs. SBFL	Metallaxis > Ochiai
	MUSE > Op2
	MUSE > Tarantula

Our results on artificial faults

	Prior studies (winner > loser)	Ours (artificial faults) Replicated	Effect
SBFL vs. SBFL	Ochiai > Tarantula	**yes**	small
	Barinel > Ochiai	**no**	small
	Barinel > Tarantula	**yes**	*negligible*
	Op2 > Ochiai	**yes**	*negligible*
	Op2 > Tarantula	**yes**	small
	DStar > Ochiai	**yes**	*negligible*
	DStar > Tarantula	**yes**	small
MBFL vs. SBFL	Metallaxis > Ochiai	yes	*negligible*
	MUSE > Op2	**no**	*negligible*
	MUSE > Tarantula	**no**	*negligible*

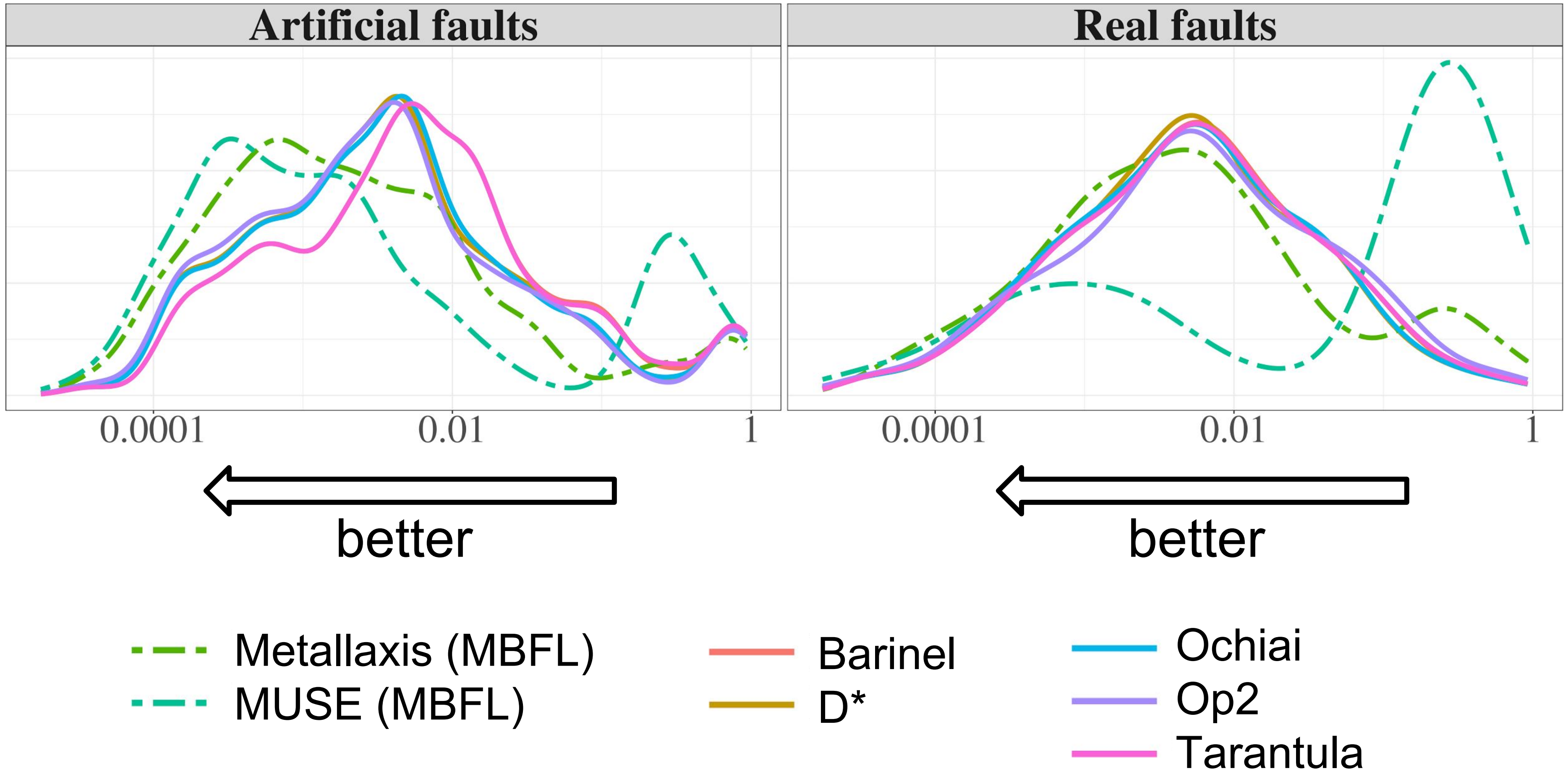
Results agree with most prior studies **on artificial faults** but **only 3 effect sizes** are **not negligible.**

Our results on real faults

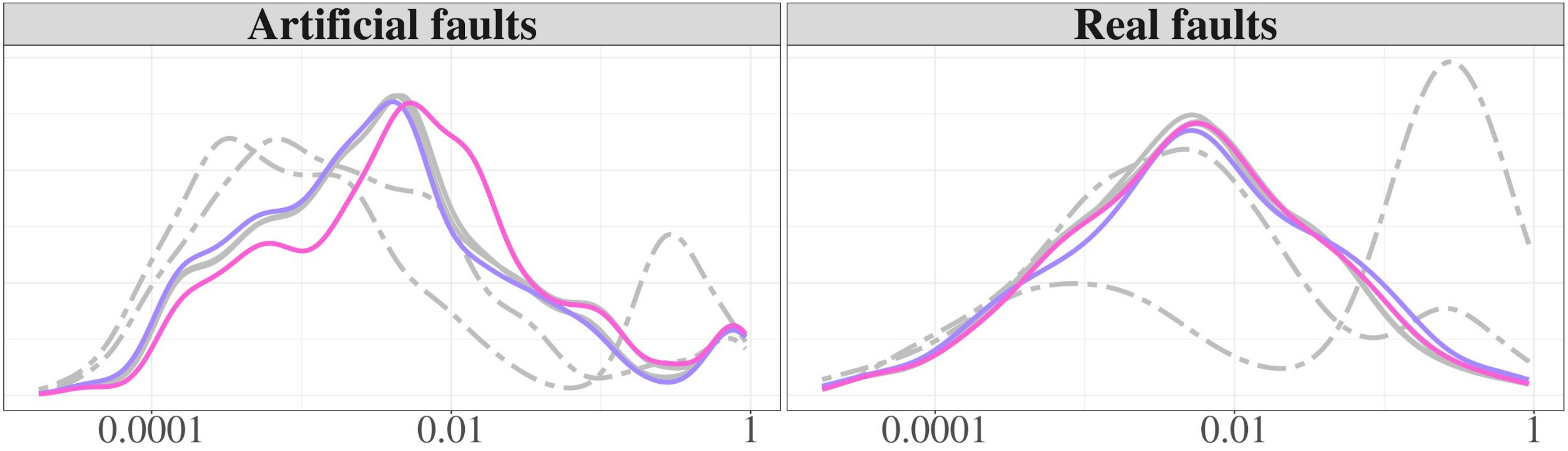
	Prior studies (winner > loser)	Ours (artificial faults) Replicated	Effect	Ours (real faults) Replicated	Effect
SBFL vs. SBFL	Ochiai > Tarantula	**yes**	small	*insignificant*	*negligible*
	Barinel > Ochiai	**no**	small	*insignificant*	*negligible*
	Barinel > Tarantula	**yes**	*negligible*	*insignificant*	*negligible*
	Op2 > Ochiai	**yes**	*negligible*	no	*negligible*
	Op2 > Tarantula	**yes**	small	*insignificant*	*negligible*
	DStar > Ochiai	**yes**	*negligible*	*insignificant*	*negligible*
	DStar > Tarantula	**yes**	small	*insignificant*	*negligible*
MBFL vs. SBFL	Metallaxis > Ochiai	yes	*negligible*	**no**	small
	MUSE > Op2	**no**	*negligible*	**no**	**large**
	MUSE > Tarantula	**no**	*negligible*	**no**	**large**

Results disagree with all prior studies **on real faults**.

Results on artificial vs. real faults

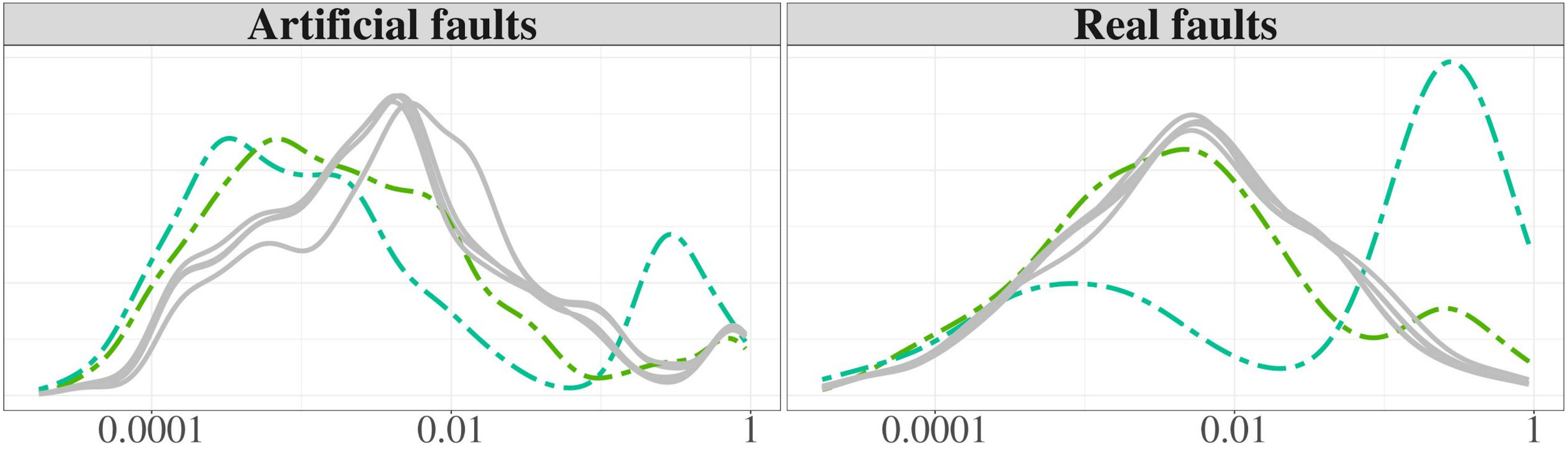


All SBFL techniques are equally good



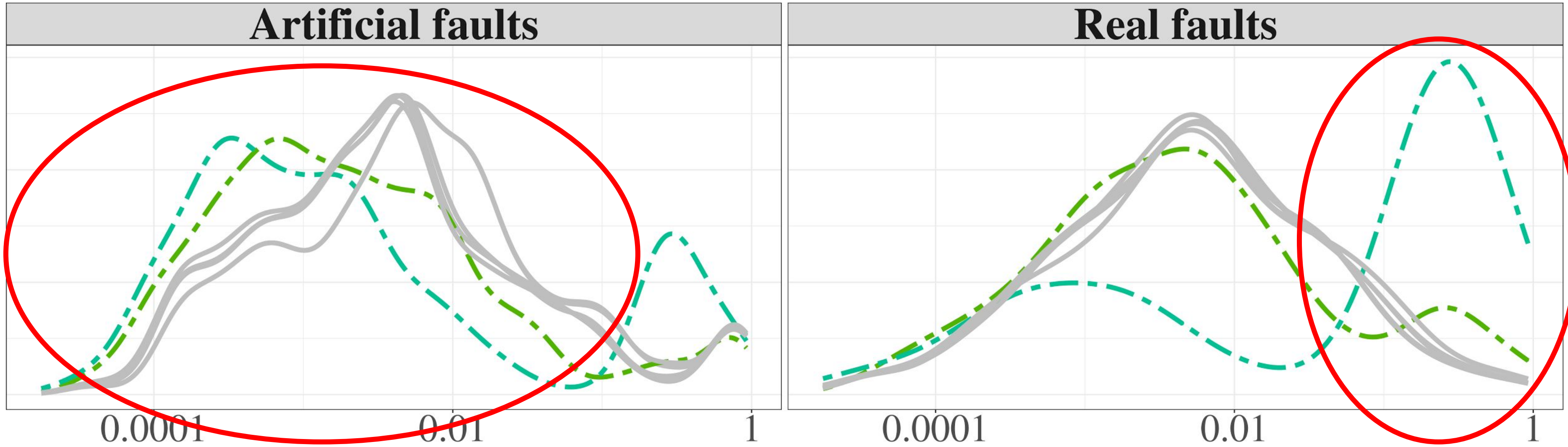
For SBFL, results on artificial faults
do not predict results on real faults!

MBFL is only better than SBFL on artificial faults

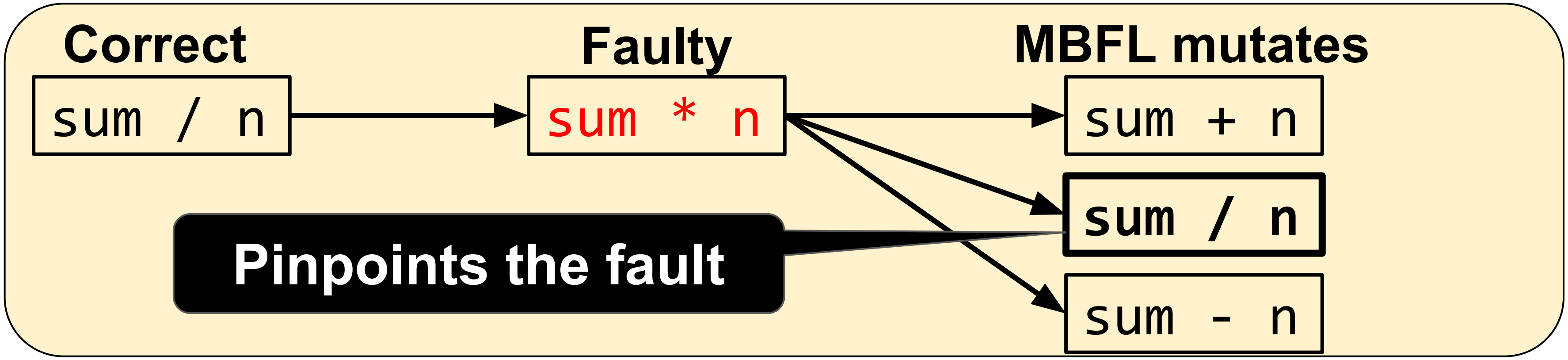


For MBFL, results on artificial faults
do not predict results on real faults!

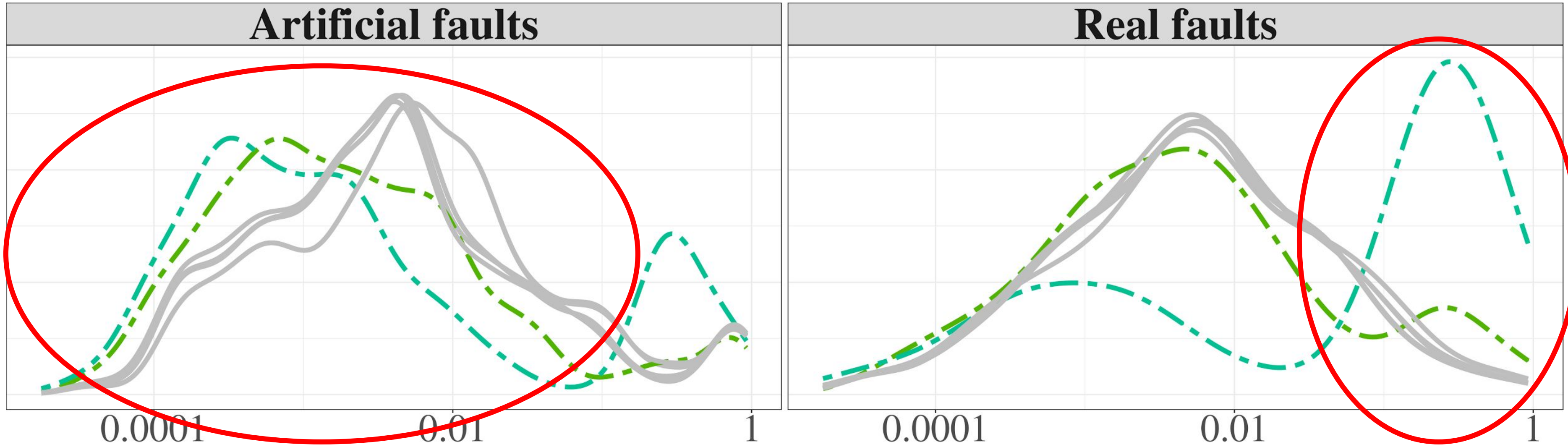
Why these differences?



- MBFL does exceptionally well on “reversible” faults

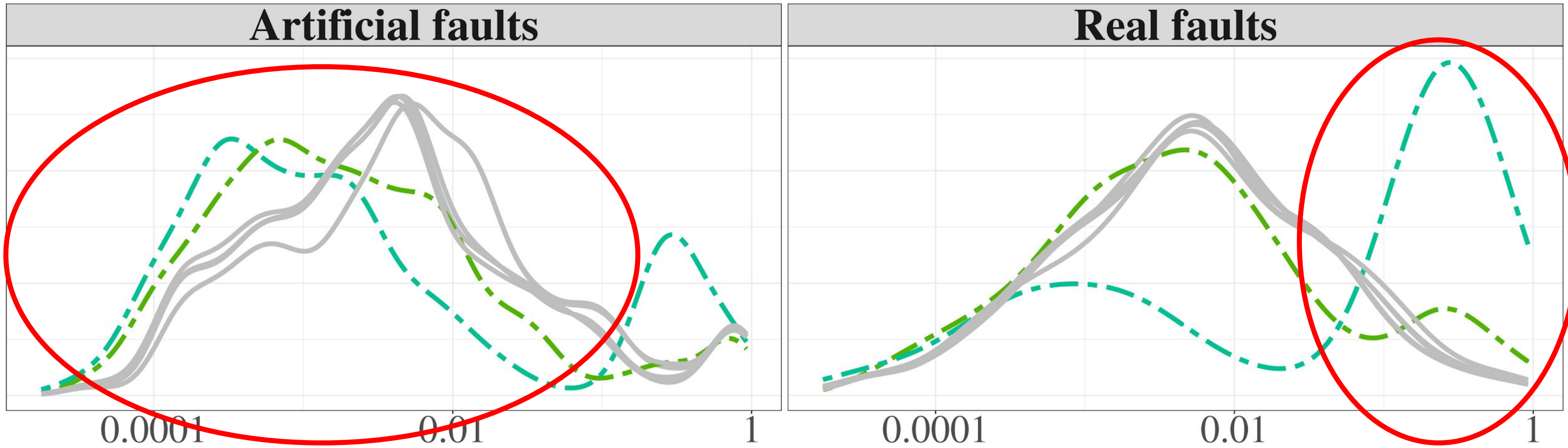


Why these differences?



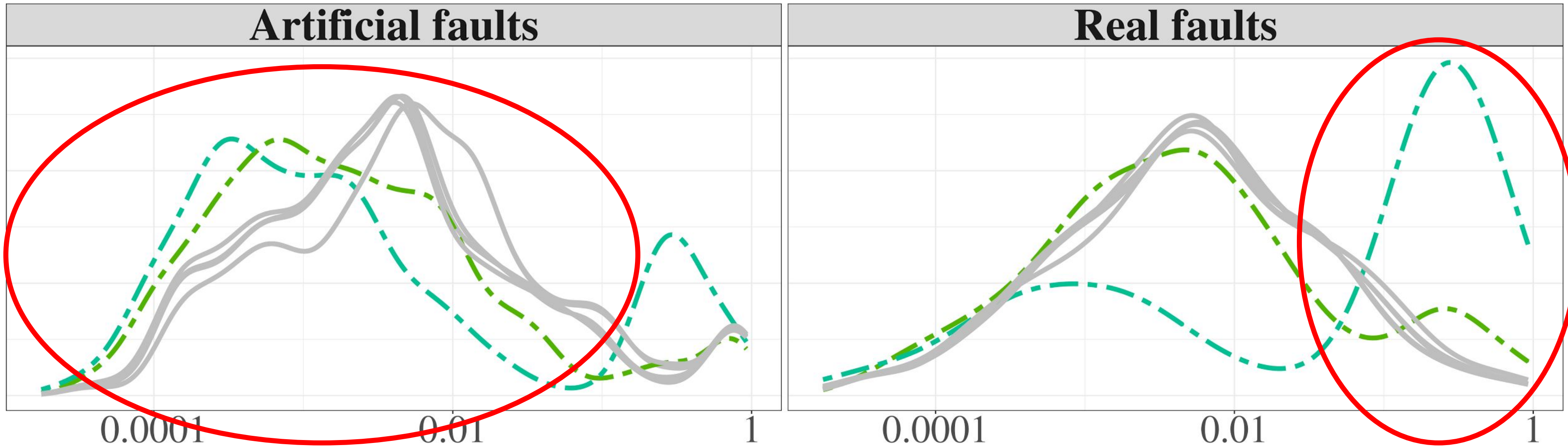
- MBFL does exceptionally well on “reversible” faults
- Most real faults are not reversible

Why these differences?



- MBFL does exceptionally well on “reversible” faults
- Most real faults are not reversible
- Real faults often involve unmutatable statements (e.g., `break`, `continue`, `return`)

Why these differences?



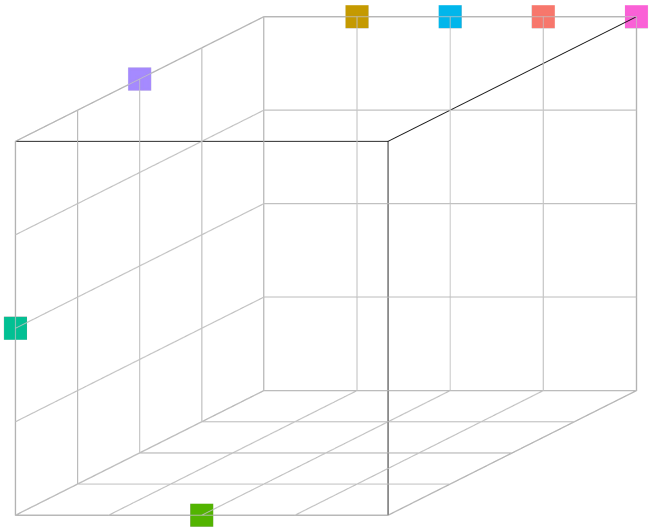
- MBFL does exceptionally well on “reversible” faults
- Most real faults are not reversible
- Real faults often involve unmutatable statements

MBFL has **pinpoint** accuracy on **artificial faults** but **poor performance on real faults**.

What design decisions matter on real faults?

Defined and explored a design space for SBFL and MBFL

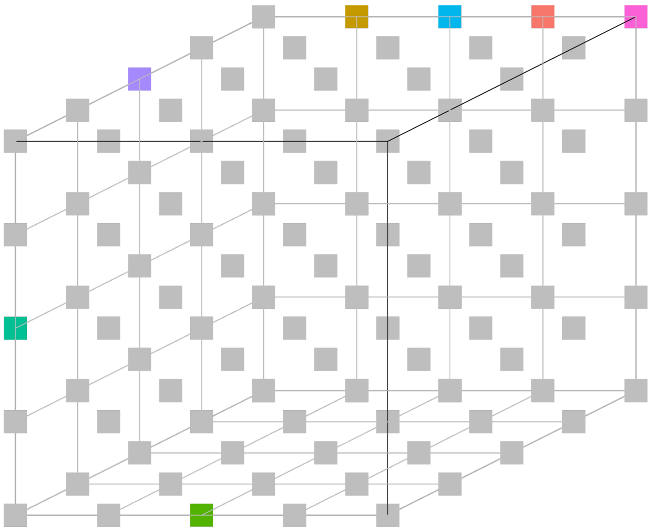
- 4 design factors (e.g., formula)



What design decisions matter on real faults?

Defined and explored a design space for SBFL and MBFL

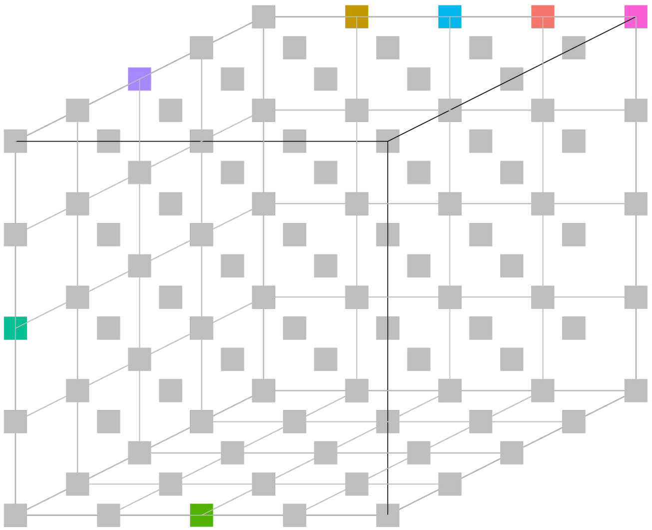
- 4 design factors (e.g., formula)
- 156 FL techniques



What design decisions matter on real faults?

Defined and explored a design space for SBFL and MBFL

- 4 design factors (e.g., formula)
- 156 FL techniques



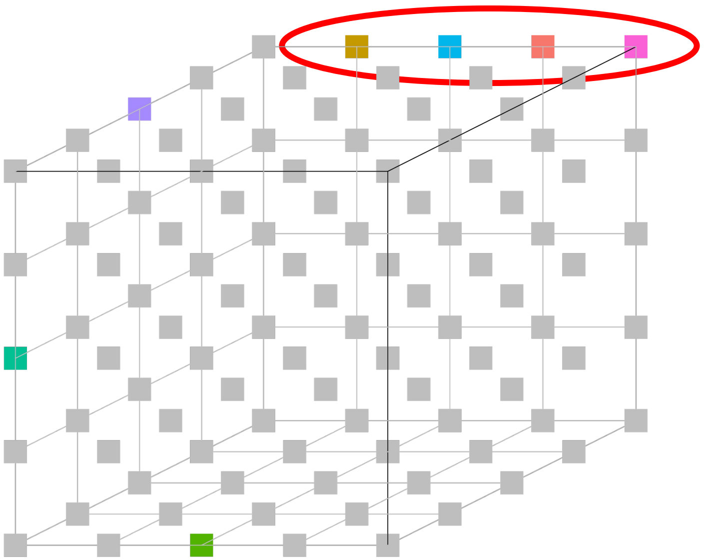
Results

- Most design decisions don’t matter (in particular for SBFL)
- Definition of test-mutant interaction matters for MBFL

What design decisions matter on real faults?

Defined and explored a design space for SBFL and MBFL

- 4 design factors (e.g., formula)
- 156 FL techniques

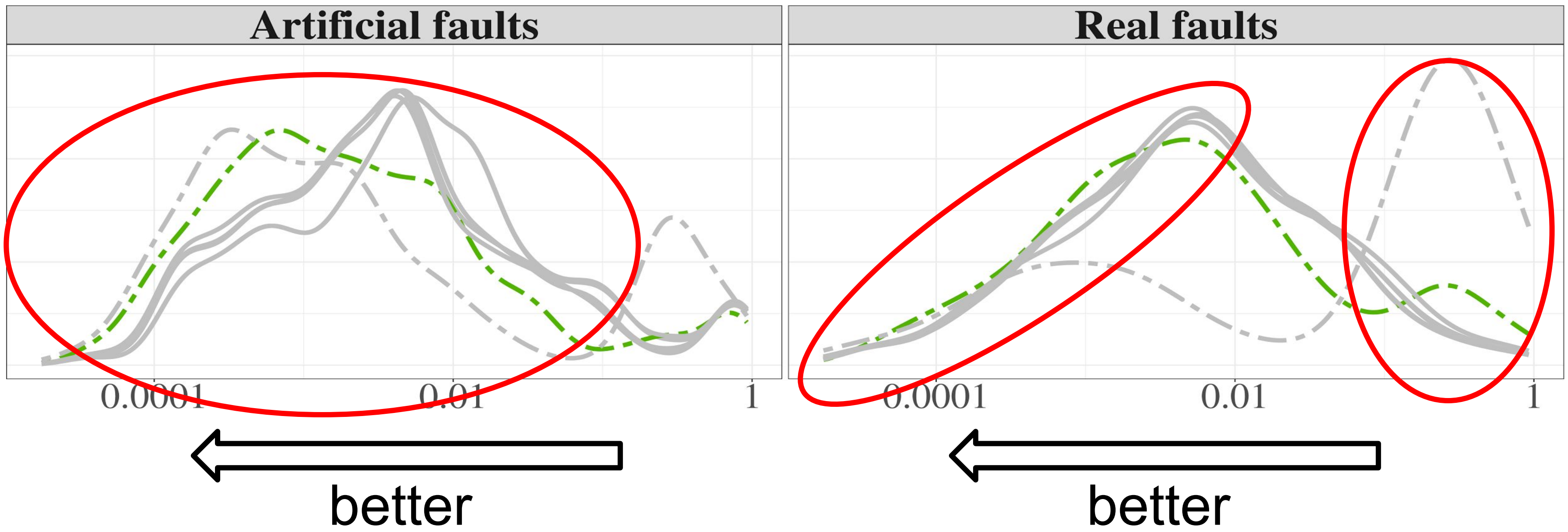


Results

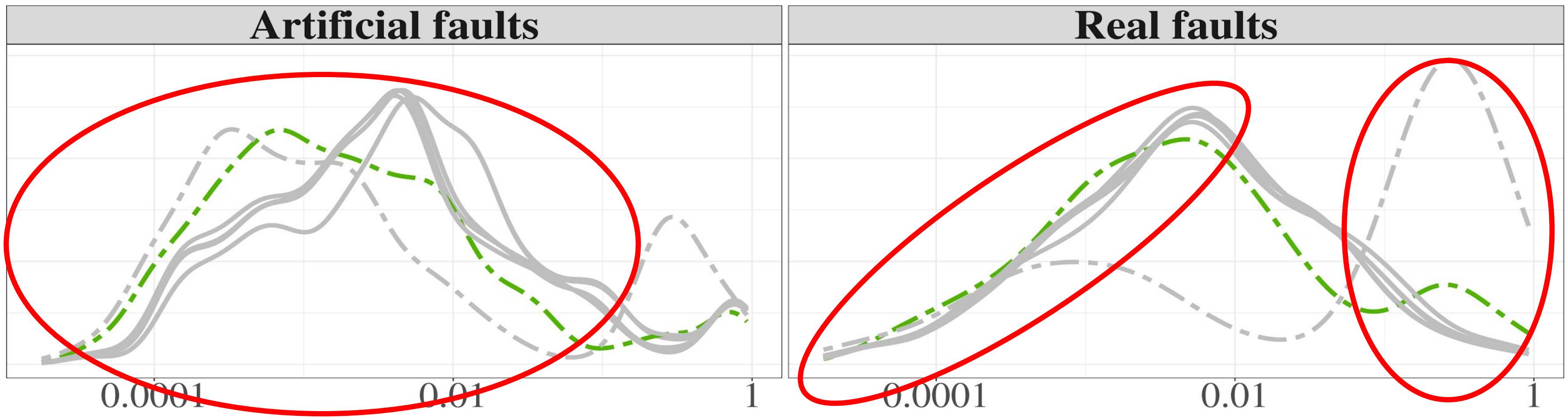
- Most design decisions don't matter (in particular for SBFL)
- Definition of test-mutant interaction matters for MBFL
- Barinel, D*, Ochiai, and Tarantula are indistinguishable

Existing **SBFL techniques** perform **best.**
No breakthroughs in the **MBFL/SBFL design space.**

How to improve fault localization?



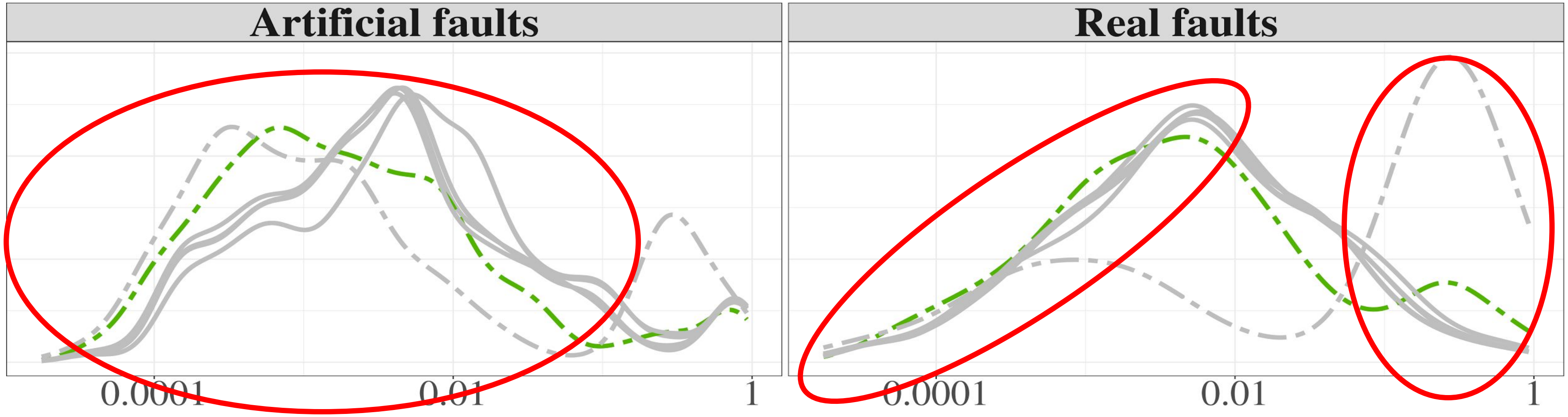
How to improve fault localization?



Explored two options:

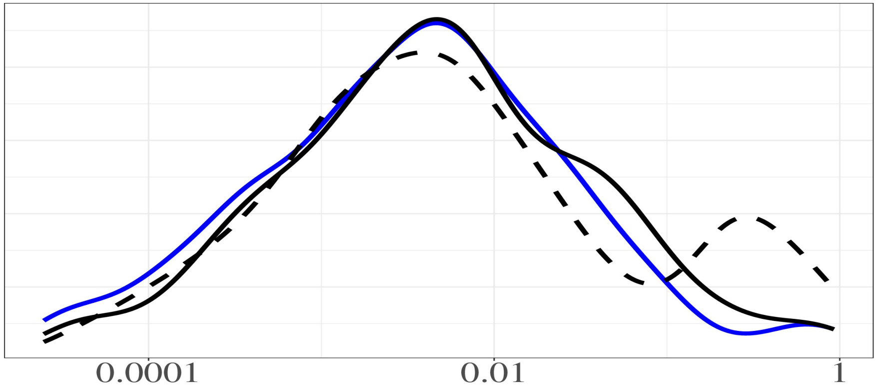
1. **Make MBFL great again**
2. **Hybrid: Stronger together**

How to improve fault localization?



Explored two options:

1. ~~Make MBFL great again~~
2. Hybrid: Stronger together



Hybrid technique is **significantly better than** all techniques in the MBFL/SBFL **design space (small effect size)**.

Only top-ranked results matter

- Top-10 useful for practitioners[1].
- Top-200 useful for automated program repair[2].

Technique	Top-5	Top-10	Top-200
Hybrid	36%	45%	85%
DStar (*best SBFL*)	30%	39%	82%
Metallaxis (*best MBFL*)	29%	39%	77%

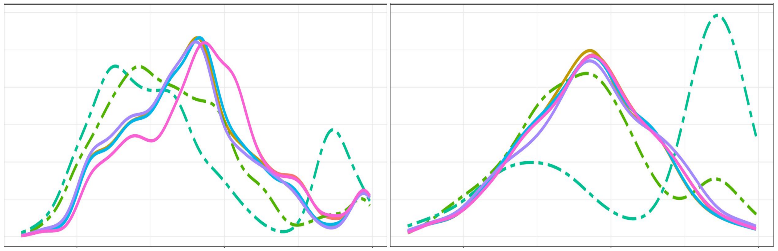
Hybrid technique **performs well** on **real use cases.**

[1]Kochhar et al., *Practitioners' Expectations on Automated Fault Localization*, ISSTA'16
[2]Long and Rinard, *An analysis of the search spaces for generate and validate patch generation systems*, ICSE'16

Evaluating and improving fault localization

FL performance on artificial faults is not predictive for real faults.

- MBFL only better on artificial faults
- All SBFL techniques are equally good

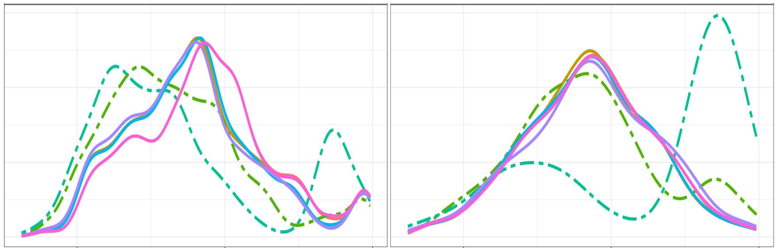


http://bitbucket.org/rjust/fault-localization-data   *http://www.defects4j.org*

Evaluating and improving fault localization

FL performance on artificial faults is not predictive for real faults.

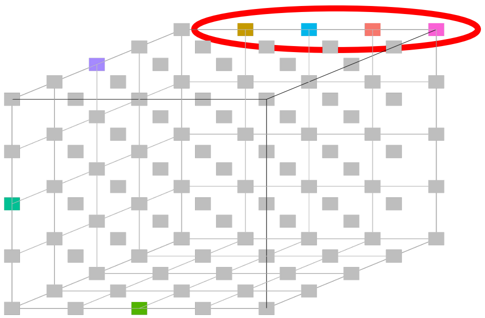
- MBFL only better on artificial faults
- All SBFL techniques are equally good



MBFL/SBFL design space exploration

- Most design decisions don't matter
- Existing SBFL techniques perform best
- No breakthroughs in the design space

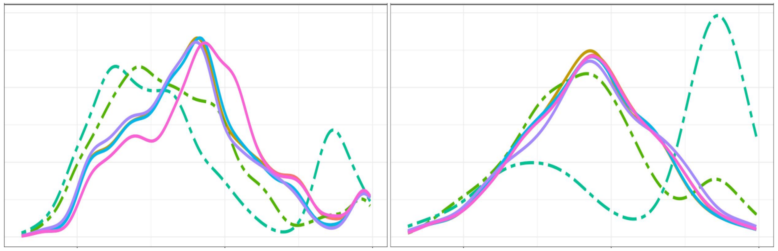
➡ *FL needs to employ more information*



http://bitbucket.org/rjust/fault-localization-data *http://www.defects4j.org*

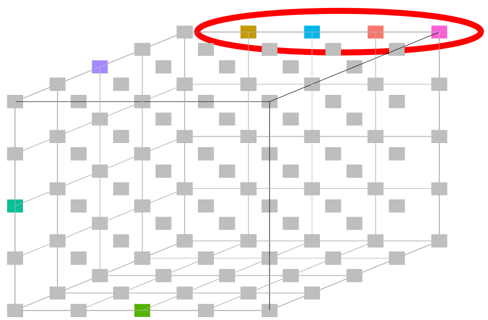
Evaluating and improving fault localization

FL performance on artificial faults is not predictive for real faults.



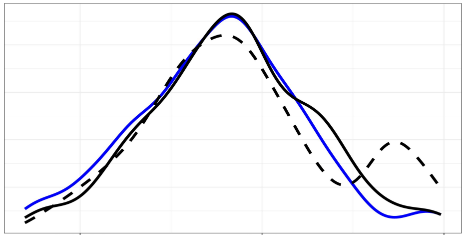
- MBFL only better on artificial faults
- All SBFL techniques are equally good

MBFL/SBFL design space exploration



- Most design decisions don't matter
- Existing SBFL techniques perform best
- No breakthroughs in the design space
  ➡ *FL needs to employ more information*

A new hybrid FL technique



- Combines MBFL and SBFL techniques
- Outperforms all existing FL techniques

http://bitbucket.org/rjust/fault-localization-data *http://www.defects4j.org*