Diversity Maximization Speedup for Fault Localization

Liang Gong[1], David Lo[2], Lingxiao Jiang[2], and Hongyu Zhang[1]

Authored by

清華大學 Tsinghua University [1] Tsinghua University

SMU SINGAPORE MANAGEMENT UNIVERSITY [2] Singapore Management University

Outline

Introduction & Framework

Tsinghua University

SMU SINGAPORE MANAGEMENT UNIVERSITY

Debugging

Problem

- Software errors cost the US economy 59.5 billion dollars (0.6% of 2002's GDP) [1]
- Testing and debugging activities are labor-intensive (30% to 90% of a Project) [2]

[1] National Institute of Standards and Technology (NIST). Software Errors Cost U.S. Economy $59.5 Billion Annually, June 28, 2002.

[2] B. Beizer. Software Testing Techniques. International Thomson Computer Press, Boston, 2nd edition, 1990.

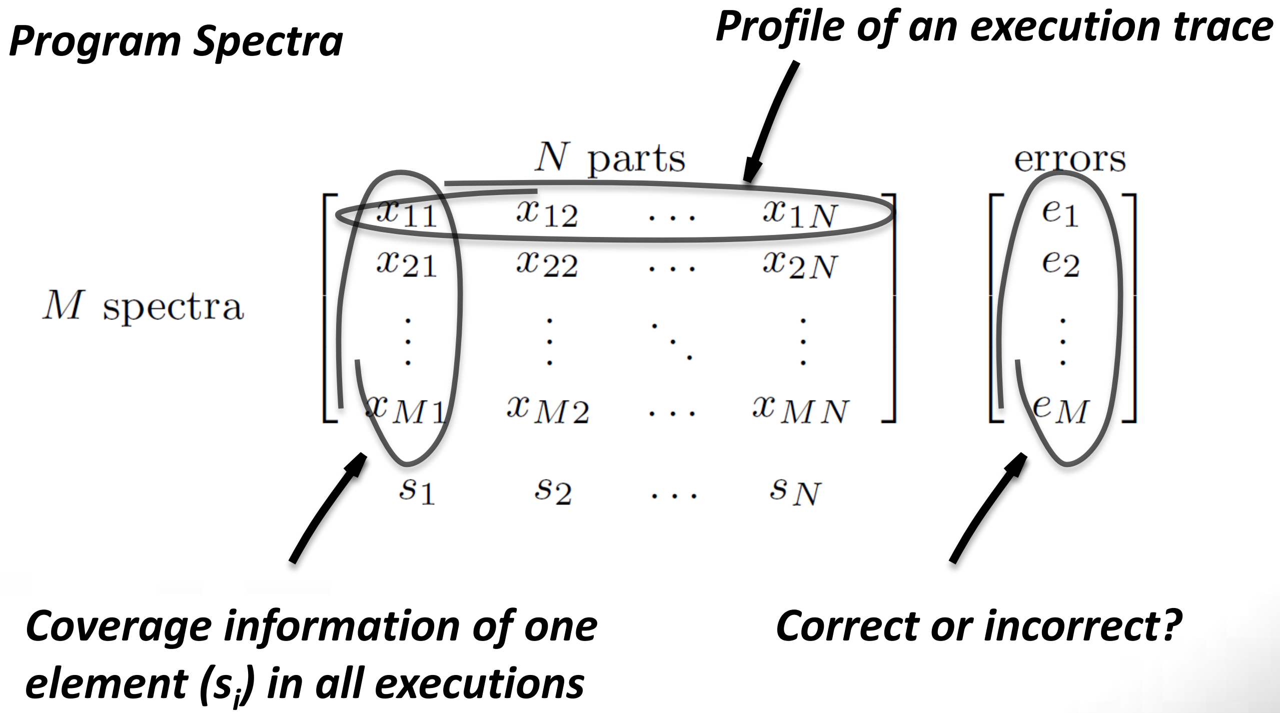
SBFL

Introduction

Spectrum-based Fault Localization(abbr. SBFL)

- Automatically recommend a list of suspicious program elements for inspection.

- **Program Spectra** consists of coverage information and execution labels.



Approaches

Fault Localization

For a given statement S

The formula calculates the suspiciousness of S.

Ochiai

$$\frac{a_{ef}}{\sqrt{(a_{ef}+a_{nf})(a_{ef}+a_{ep})}}$$

No. of failed traces covering S (a_{ef})

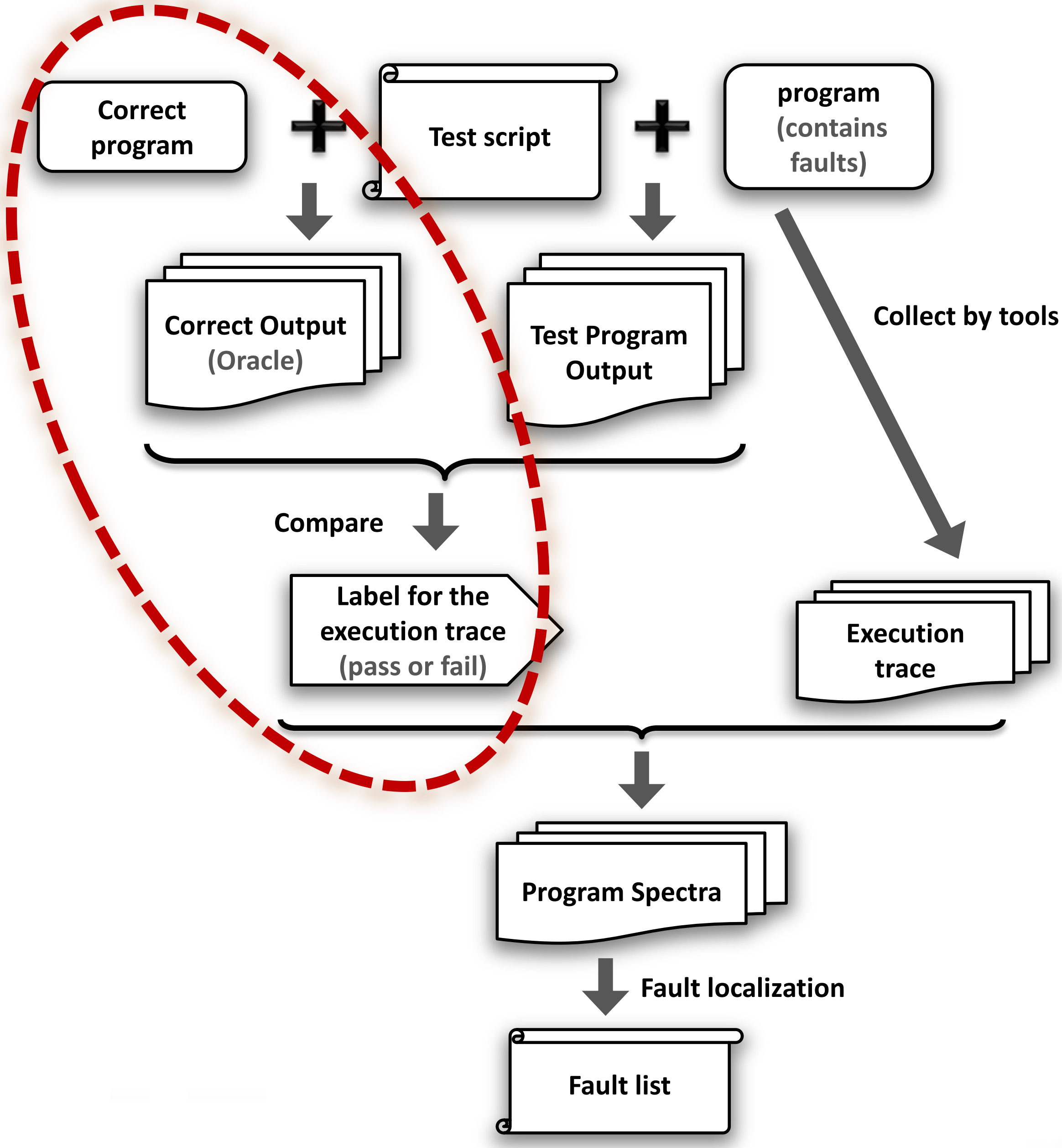
No. of failed traces ($a_{ef}+a_{nf}$)

No. of traces covering S ($a_{ef}+a_{ep}$)

Intuition: *If **S** is covered **more** in **failed** traces and **less** in **passed** traces, it is more likely to contain faults.*

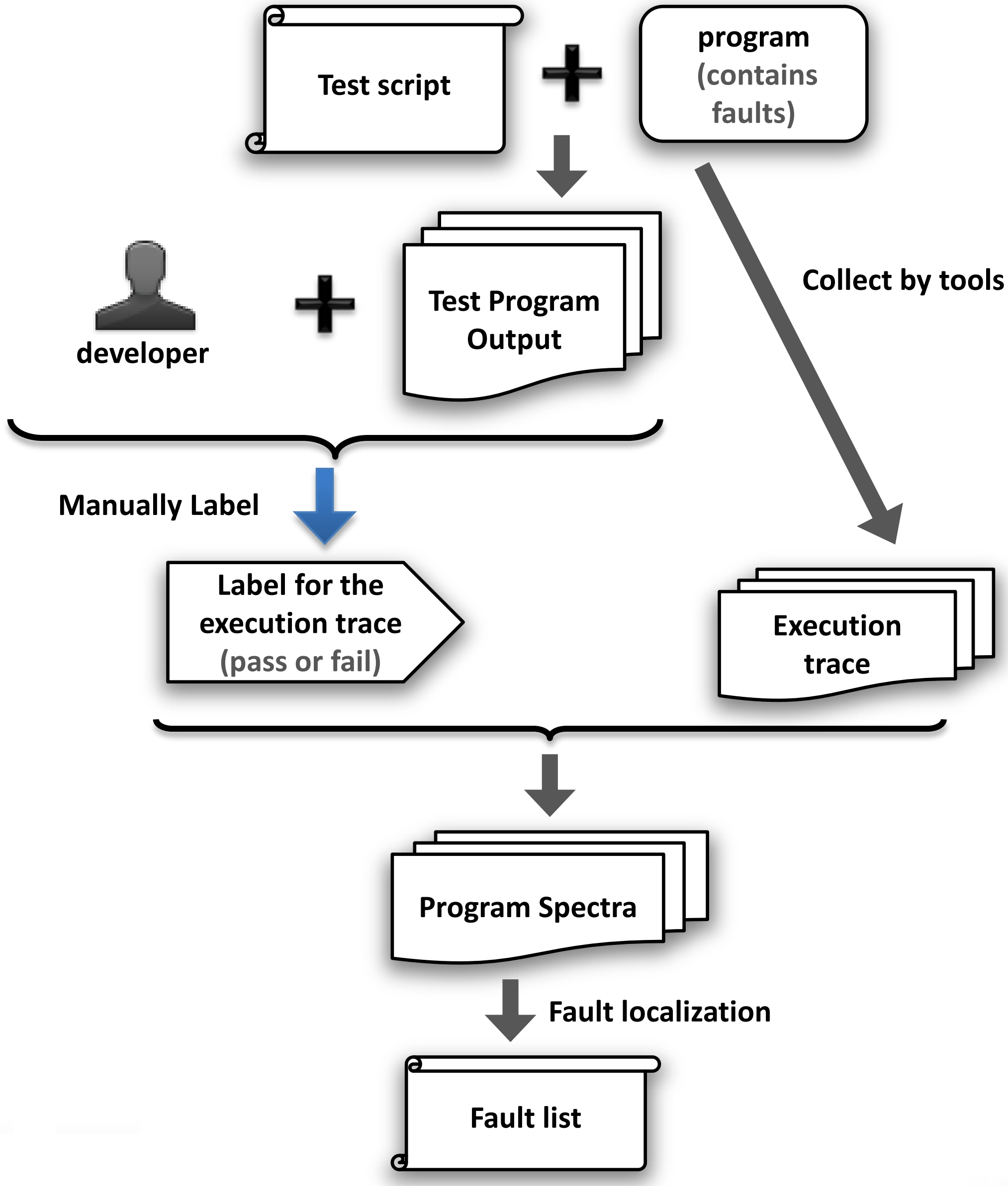
Process

In Experiments



Process

In Practice



Process

In Practice

Research Goal:

- Minimize No. of executions to label
- Preserve fault localization effectiveness

Test Case Prioritization

Introduction

In [3], Rothermel *et al.* define the problem of test case prioritization as follows:

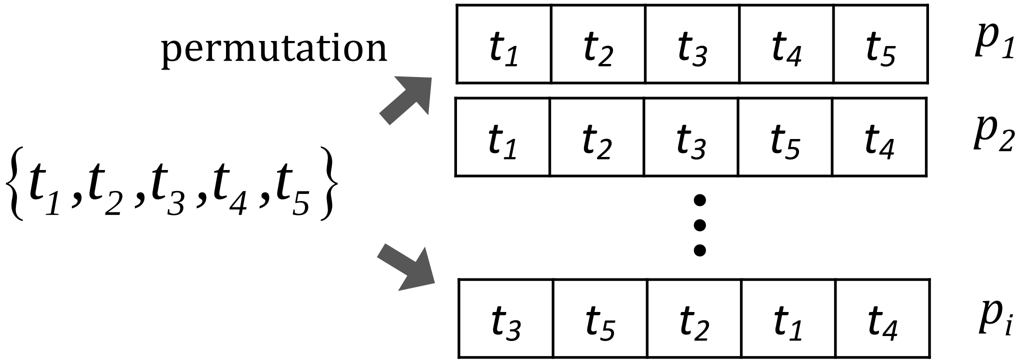
Definition 2.1 *(Test Case Prioritization). Given*

(1) T, a set of test cases,

(2) PT, the set of permutations of T

(3) f, a function mapping PT to real numbers,

the problem is to find a permutation p $\in$ *PT such that: for all p'* $\in$ *PT:* $f(p) \geq f(p')$.



$$\arg\max_{p \in PT} \{ f(p) \}$$

[3] G. Rothermel, R. H. Untch, C. Chu, and M. J. Harrold. Prioritizing test cases for regression testing. In IEEE TSE, pages 929-948, 2001.

Test Case Prioritization

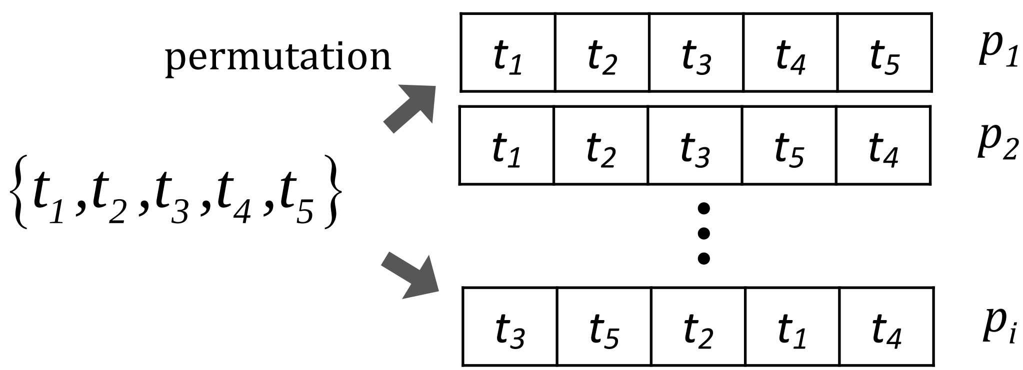
Introduction

In [3], Rothermel *et al.* define the problem of test case prioritization as follows:

Definition 2.1 (*Test Case Prioritization*). Given

for all p' ∈ PT: f (p) ≥ f(p').

f*(*p*)** is larger, when permutation ***p allows the **faulty** program elements to be **ranked higher** meanwhile a **shorter prefix** are considered.



$$\arg\max_{p \in PT} \{ f(p) \}$$

[3] G. Rothermel, R. H. Untch, C. Chu, and M. J. Harrold. Prioritizing test cases for regression testing. In IEEE TSE, pages 929-948, 2001.

Diversity **M**aximization **S**peedup (*abbr.* DMS)

- Greedy algorithm
- Use **diversity of suspiciousness** as the selecting criterion.
- **Speedup** suspiciousness rank changing process of **promising** program elements to further save labeling effort.

Diversity

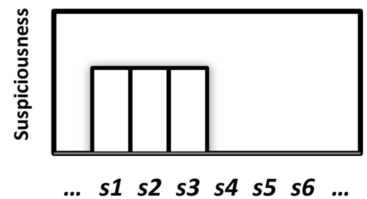
As Criterion

Diversity Maximization Speedup (*abbr.* DMS)

- t_0 is the initial failed trace that reveals the fault.
- t_1 and t_2 are candidates to be selected for labeling.

t_0

…	s_1	s_2	s_3	s_4	s_5	s_6	…	*p/f*
…	●	●	●	○	○	○	…	*fail*



Initial Diagnostic Distribution

s_1, s_2, s_3
s_4, s_5, s_6

Initial Inspection List

t_1

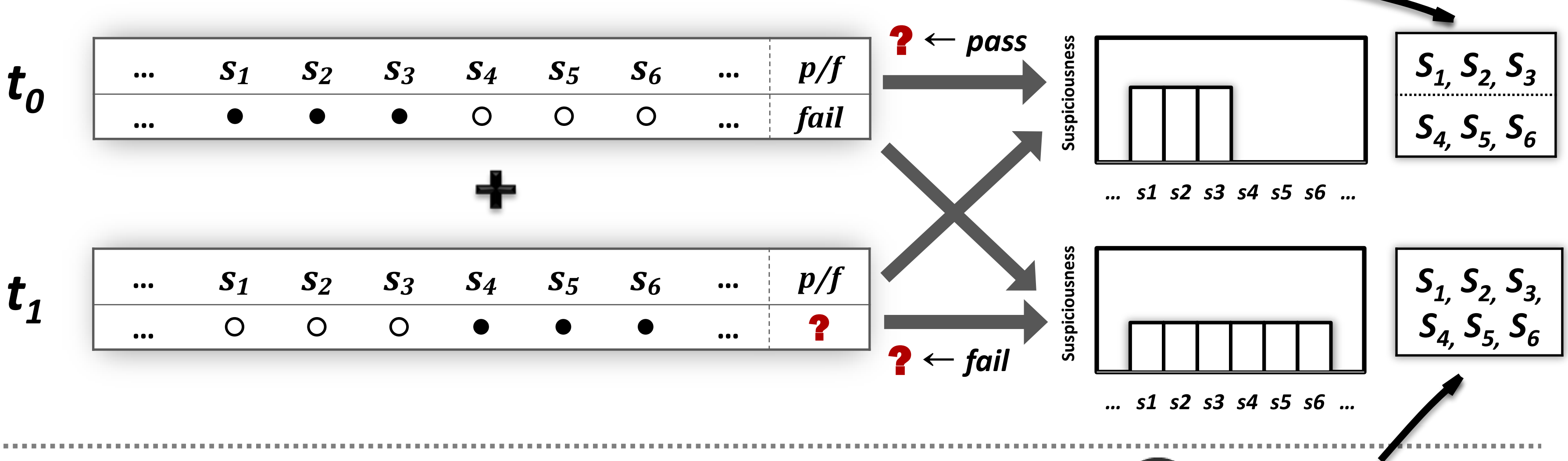
…	s_1	s_2	s_3	s_4	s_5	s_6	…	*p/f*
…	○	○	○	●	●	●	…	?

t_2

…	s_1	s_2	s_3	s_4	s_5	s_6	…	*p/f*
…	○	●	●	●	○	●	…	?

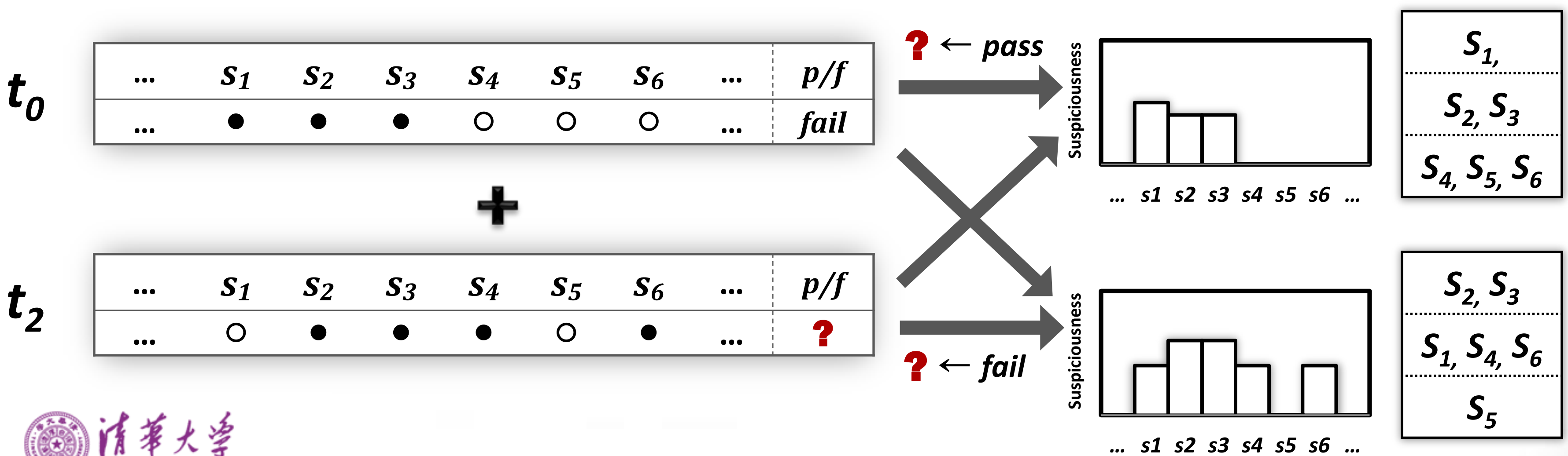
- t_1 is preferred by approaches aiming to detect faults

☹ ***Nothing Changed***



☹ ***No Suggestion***

- t_2 is selected according to Diversity Maximization Criterion



☺ ***More Diverse, Better Result***

Diversity Maximization Speedup (*abbr.* DMS)

Statement		Test case: t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	t12	N_{ef}	N_{ep}	N_{nf}	N_{np}	Ochiai	Tarantula	Jaccard
`main(){`	s1	●	●	●	●	●	●	●	●	●	●	●	●	3	9	0	0	0.500	0.500	0.250
`int let, dig, c;`	s2																			
`let = dig = 0;`	s3																			
`while(c=getchar()){`	s4																			
`if('A'<=c && 'Z'>=c)`	s5	●	●	●	●	●	●	●		●	●	●	●	3	8	0	1	0.522	0.529	0.273
`let += 1;`	s6		●	●	●	●	●				●	●	●	2	6	1	3	0.408	0.500	0.222
`else if('a'<=c && 'z'>c) /*FAULT*/`	s7	●	●		●	●		●		●		●		3	4	0	5	0.655	0.692	0.429
`let += 1;`	s8	●	●		●	●		●						2	3	1	6	0.516	0.667	0.333
`else if('0'<=c && '9'>=c)`	s9	●	●			●		●		●		●		2	4	1	5	0.471	0.600	0.286
`dig += 1;`	s10	●	●			●				●		●		2	3	1	6	0.516	0.667	0.333
`printf("%d %d\n",let,dig);}`	s11	●	●	●	●	●	●	●	●	●	●	●	●	3	9	0	0	0.500	0.500	0.250
pass/fail		P	F	P	F	P	P	P	P	F	P	P	P							

(a) Fault Localization with All Test Cases

Ambiguity Group (the groups are ordered according to their suspiciousness)	Selected Test Case	Program Spectra: s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	p/f
{s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11}	t2	1	1	1	1	1	1	1	1	1	1	1	F
{s5,s6,s7,s8,s9,s10},{s1,s2,s3,s4,s11}	t8	1	1	1	1	0	0	0	0	0	0	1	P
{s7,s8,s9,s10},{s6,s5},{s1,s2,s3,s4,s11}	t6	1	1	1	1	1	1	0	0	0	0	1	P
{s7,s8},{s5,s6}{s1,s2,s3,s4,s11},{s9,s10}	t4	1	1	1	1	1	1	1	1	0	0	1	F
{s7,s8},{s6},{s5},{s10},{s1,s2,s3,s4,s11},{s9}	t7	1	1	1	1	1	0	1	1	1	0	1	P
{s7},{s10},{s5},{s1,s2,s3,s4,s11},{s6},{s8},{s9}	t9	1	1	1	1	1	0	1	0	1	1	1	F

Ambiguity Group	Selected Test Case	Normalized *Ochiai* Score: s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11
{s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11}	t2	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	**0.0909**	0.0909	0.0909	0.0909	0.0909
{s5,s6,s7,s8,s9,s10},{s1,s2,s3,s4,s11}	t8	0.0742	0.0742	0.0742	0.0742	0.1049	0.1049	**0.1049**	0.1049	0.1049	0.1049	0.0742
{s7,s8,s9,s10},{s6,s5},{s1,s2,s3,s4,s11}	t6	0.0696	0.0696	0.0696	0.0696	0.0852	0.0852	**0.1205**	0.1205	0.1205	0.1205	0.0696
{s7,s8},{s5,s6}{s1,s2,s3,s4,s11},{s9,s10}	t4	0.0824	0.0824	0.0824	0.0824	0.0951	0.0951	**0.1165**	0.1165	0.0824	0.0824	0.0824
{s7,s8},{s6},{s5},{s10},{s1,s2,s3,s4,s11},{s9}	t7	0.0840	0.0840	0.0840	0.0840	0.0940	0.1085	**0.1085**	0.1085	0.0664	0.0940	0.0840
{s7},{s10},{s5},{s1,s2,s3,s4,s11},{s6},{s8},{s9}	t9	0.0885	0.0885	0.0885	0.0885	0.0969	0.0834	**0.1084**	0.0834	0.0834	0.1022	0.0885

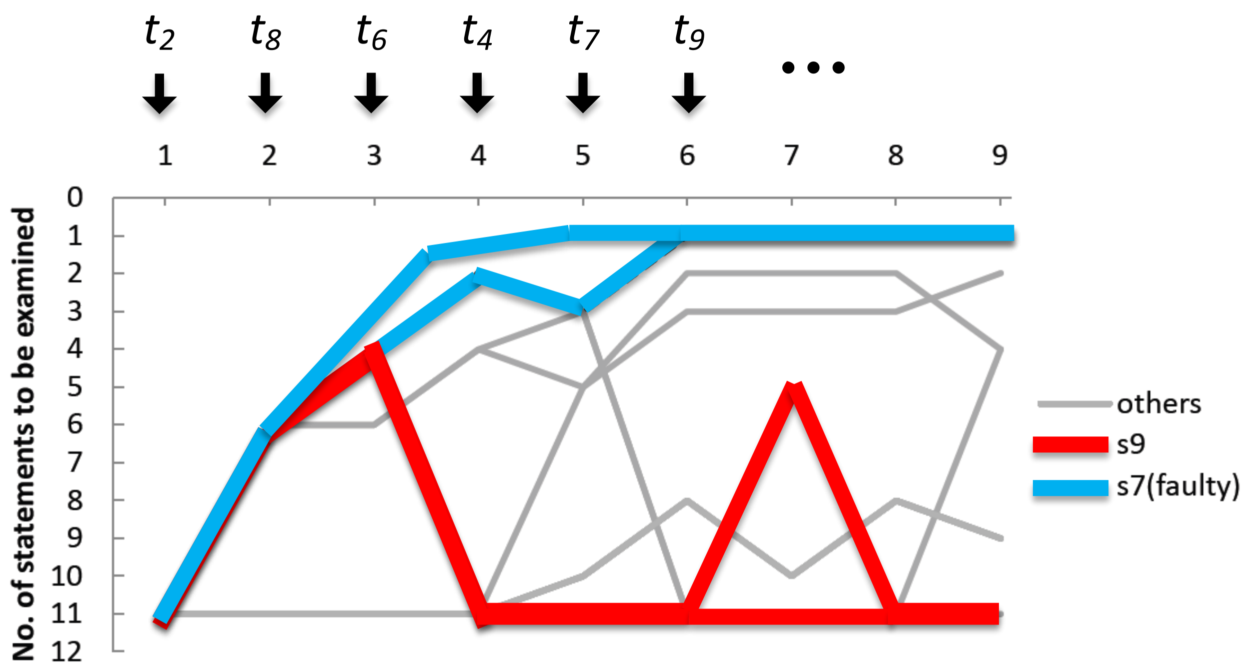
Figure 1: Running Example

清華大學 Tsinghua University

SMU SINGAPORE MANAGEMENT UNIVERSITY

Evolution Trend

Opportunities



Evolution Trend of
Statements' Suspiciousness.

Looking for test cases that could offer more **changing opportunities** to **"promising"** elements like s7 (with clear trend) instead of s9

Speed up

How to ?

Two questions prompt:

- How can we know which statements are **"promising"**?
- With "promising" statements, how can we **speed up** their suspiciousness changing process?

Promising

How to identify ?

Representative Time Series

- When the rank of the program element **decreases**, its time series **increases** by 1.

- When the rank of the program element **increases**, its time series **decreases** by 1.

- If the element's rank stays the same, its time series stays the same.

Evolution trend and time series(y_i) of S_8

Iteration (x_i)	1	2	3	4	5	6	7	...
Rank	11	6	4	2	3	11	5	...
Trend ($\mathcal{T}$)		[+]	[+]	[+]	[-]	[-]	[+]	...
y_i	0	1	2	3	2	1	2	...

Tsinghua University

SMU SINGAPORE MANAGEMENT UNIVERSITY

Promising

How to evaluate ?

Evolution trend of S_8

Iteration (x_i)	1	2	3	4	5	6	7	...
Rank	11	6	4	2	3	11	5	...
Trend ($\mathcal{T}$)		[+]	[+]	[+]	[-]	[-]	[+]	...
y_i	0	1	2	3	2	1	2	...

- Linear Regression Analysis:

$$y_i = \beta_1 \cdot x_i + \beta_0 + \epsilon_i$$

- Change-potential Score:

$$\mathcal{W}_{\mathcal{T}} = \left|\hat{\beta}_1\right| \cdot \frac{1}{\hat{\sigma}_{\beta_1} + 1}$$

Changing Rate ***Changing Stability***

Example trends and their potentials

$\mathcal{T}$		$\hat{\beta}_1$	$\hat{\sigma}_{\beta_1}$	$\mathcal{W}_{\mathcal{T}}$
[+]	[+]	1	0	1
[+]	[-]	0	0.577	0
[+]	[0]	0.5	0.289	0.388
[0]	[0]	0	0	0

Speed up

How to ?

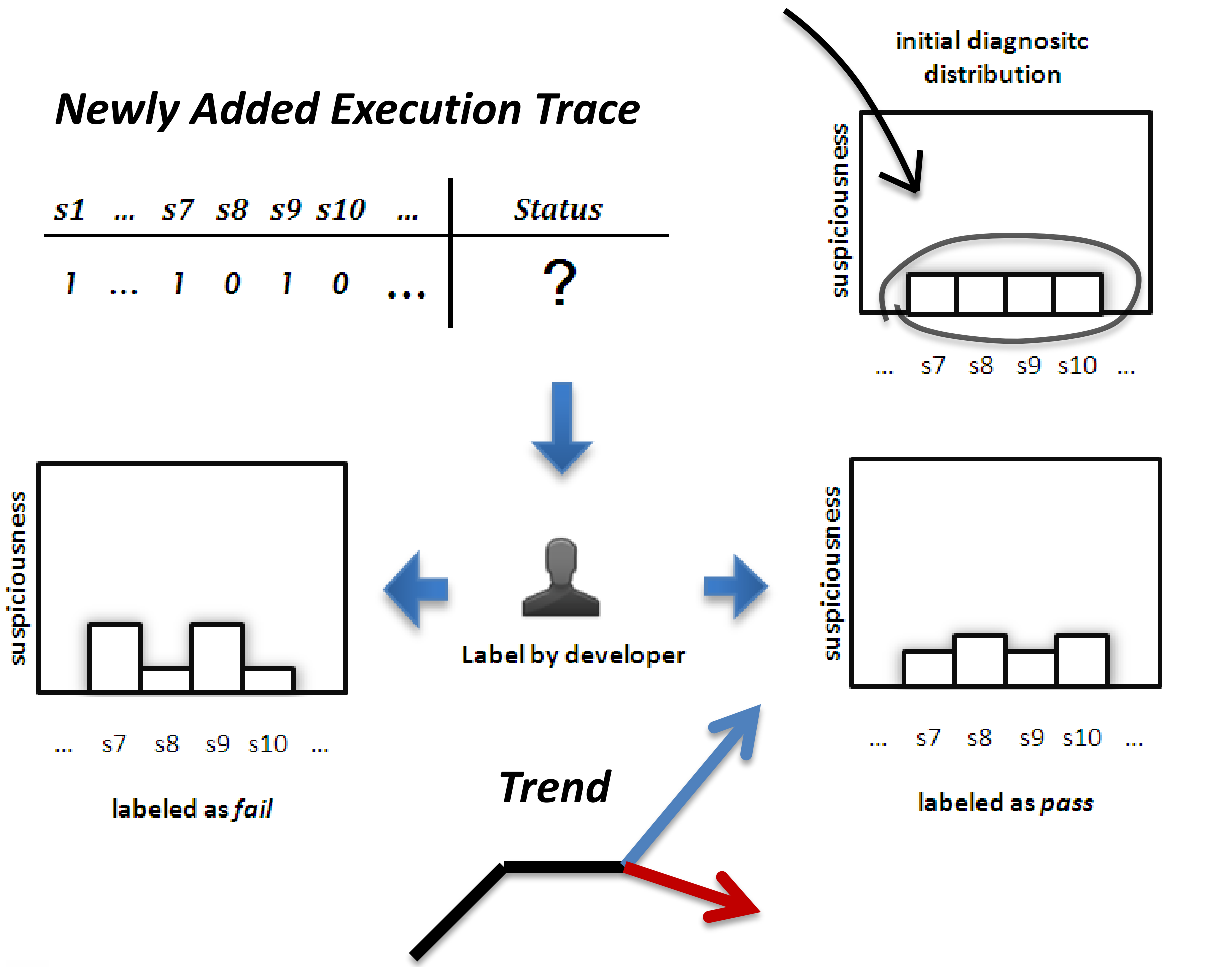
Two questions prompt:

- How can we know which statements are **"promising"**?
- With "promising" statements, how can we **speed up** their suspiciousness changing process?

Speed up

By Competing

- Speed up the suspiciousness ranking changing process by **competing in Suspicious Group.**



No matter what the label is, ties are broken anyway. Speedup happens anyway. It could keep "promising" → or become less "promising" →

Speed up

Our Method

- Change-potential Score of ***Suspicious Group***:

$$\mathcal{W}_g = \sum_{d \in g} \mathcal{W}_{T_d}$$

Change-potential Score of program element d

- Sums of Squares of Change-potential Score of all Groups(G)

$$\mathcal{H}_G = \sum_{g_i \in G} \mathcal{W}_{g_i}^2$$

- To choose the next trace ***t*** to label, we use the following formula:

$$\arg\max_{t \in T_{\mathcal{U}}} \{\mathcal{H}_G - \mathcal{H}_{(G \Leftarrow t)}\}$$

The Sum of Squares of change-potential of all groups

The Sum of Squares of change-potential of all groups when t is added

Speed up

Our Method

- Change-potential Score of ***Suspicious Group***:

$$\mathcal{W}_g = \sum_{d \in g} \mathcal{W}_{T_d}$$

Change-potential Score of program element d

- Sums of Squares of Change-potential Score of all Groups(G)

$$\mathcal{H}_G = \sum_{g_i \in G} \mathcal{W}_{g_i}^2$$

- To choose the next trace ***t*** to label,
 we use the following formula:

$$\arg\max_{t \in T_{\mathcal{U}}} \left\{ \mathcal{H}_G - \mathcal{H}_{(G \Leftarrow t)} \right\}$$

Intuition: ***When t breaks ties in more promising or larger Suspicious Groups, it is more likely to be selected.***

Test Case Prioritization

Existing Methods

- Coverage Based Prioritization
- STMT-TOTAL, STMT-ADDTL, and ART.

- Fault-Exposing Potential
- FEP-TOTAL and FEP-ADDTL.

- Diagnostic Prioritization
- SEQUIA and RAPTOR.

Tsinghua University
SMU SINGAPORE MANAGEMENT UNIVERSITY

Experiment

Dataset & Evaluation Metric

- **Benchmarks for Fault Localization**

Program	Description	LOC	Tests	Faults
tcas	Aircraft Control	173	1609	41
schedule2	Priority Scheduler	374	2710	8
schedule	Priority Scheduler	412	2651	8
replace	Pattern Matcher	564	5543	31
tot_info	Info Measure	565	1052	22
print_tokens2	Lexical Analyzer	570	4055	10
print_tokens	Lexical Analyzer	726	4070	7
space	ADL Compiler	9564	1343	30
flex	Lexical Parser	10124	567	43
sed	Text Processor	9289	371	22
grep	Text Processor	9089	809	17
gzip	Data Compressor	5159	217	15

1 (tcas – print_tokens) Siemens Suite

2 (space – gzip) UNIX Programs

- **Evaluation Metric for Fault Localization:**

$$cost = \frac{|\{j \mid f_{T_S}(d_j) \geq f_{T_S}(d_*)\}|}{|\mathcal{D}|}$$

Experiment

Settings & Design

Experiments comparing with existing methods:

- Effectiveness on reducing the number of test cases(*i.e.*, labeling effort) needed for a *target cost*
- Effectiveness on reducing cost for a given number of labeled test cases
- Defining ***target cost*** c_x:

Average fault localization cost when labeling all test cases

$$c_x = \frac{x}{100} \times c$$

Labeling Effort Needed When Setting c_{101} as Target Cost

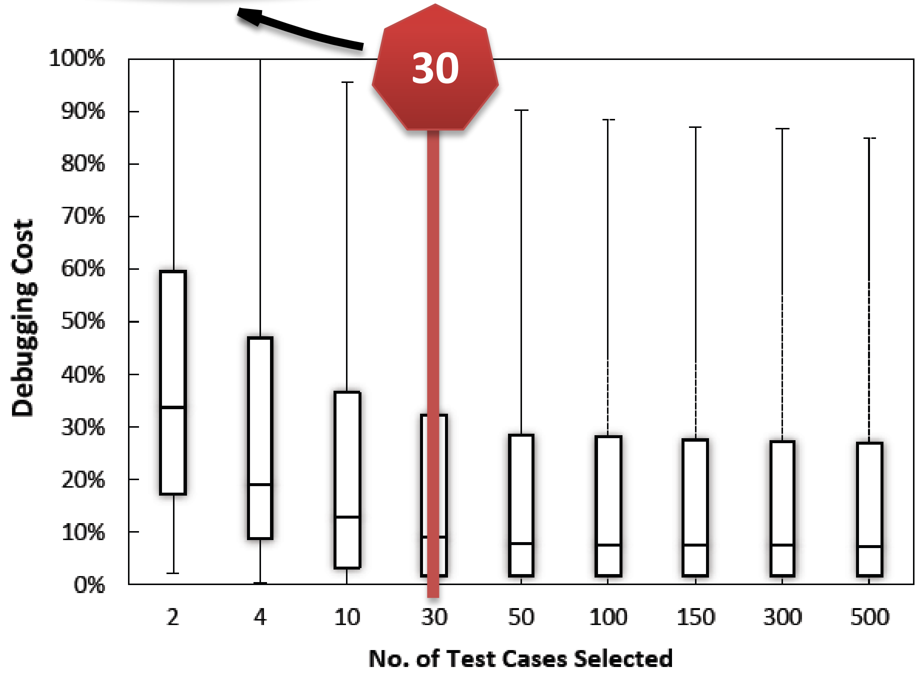
Subject Programs	DMS	RAP-TOR	SEQ-UOIA	STMT-ADDTL	STMT-TOTAL	FEP-ADDTL	ART-MIN
Siemens	**18**	20	500+	500+	500+	97	150
UNIX	**16**	48	176	150	500+	98	56

Experiment

Settings & Design

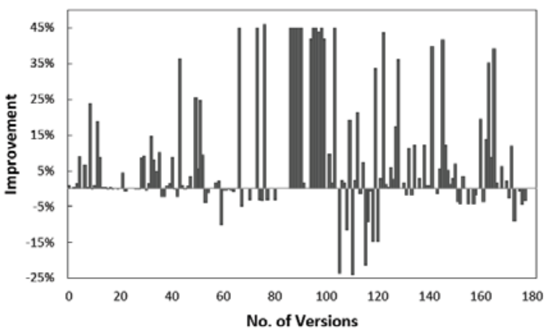
Experiments comparing with existing methods:

- Effectiveness on reducing the number of test cases(*i.e.*, labeling effort) needed for a *target cost*
- Effectiveness on reducing cost for a given number of labeled test cases

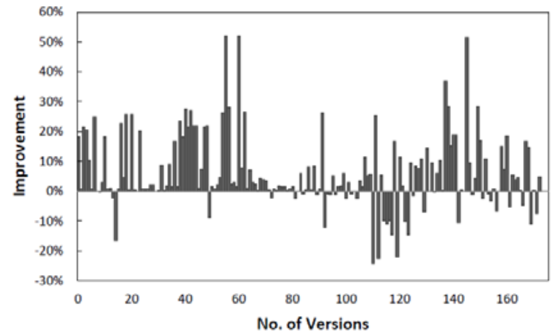


Average Cost of DMS when Selecting Different Number of Test Cases

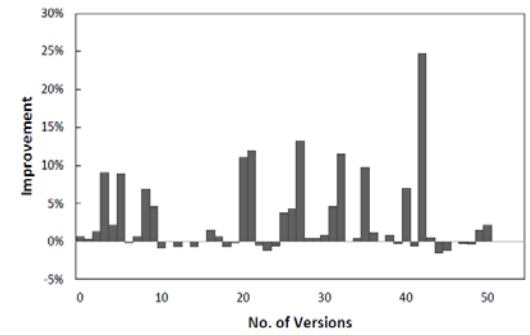
- Effectiveness on Reducing Cost for a Given Number of Labeled Test Cases



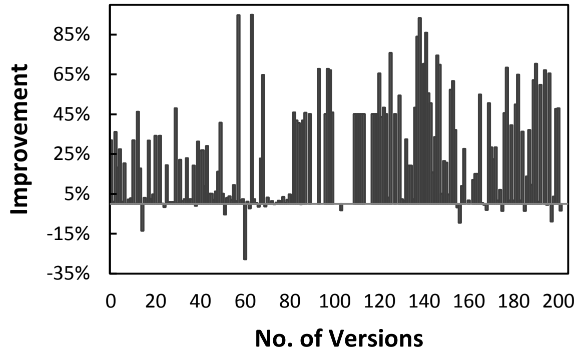
DMS VS FEP-ADDTL



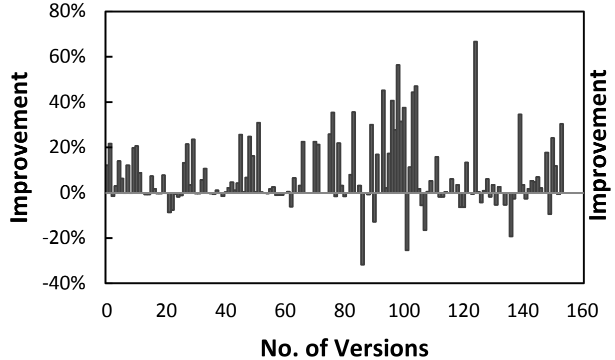
DMS VS ART-MIN



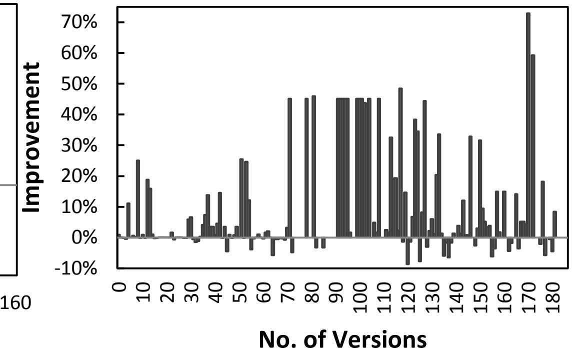
DMS VS RAPTOR



DMS VS STMT-TOTAL



DMS VS SEQUOIA



DMS VS STMT-ADDTL

清華大學 Tsinghua University

SMU SINGAPORE MANAGEMENT UNIVERSITY

Pair-wised T-test shows the improvements are statistically significant at 95% interval.

Conclusion

& Future Work

- # Conclusions

- We propose a new technique aiming to minimize the amount of effort in manual oracle construction, while still permitting effective fault localization.

✓ Given a target fault localization accuracy, our approach can significantly reduce the number of test cases needed to achieve it.

✓ Given a maximum number of test cases that a programmer can manually label, DMS can improve the accuracy of fault localization and thus helps reduce the debugging cost.

- # Future Work

- Evaluate on more subject programs.
- We will also explore the possibility of adopting more sophisticated trend analysis methods.

Tsinghua University

SMU SINGAPORE MANAGEMENT UNIVERSITY

GL

Thank you!

- Any questions?

• • •