Use of Control Flow Graphs with Edges Consideration for Fault Localization

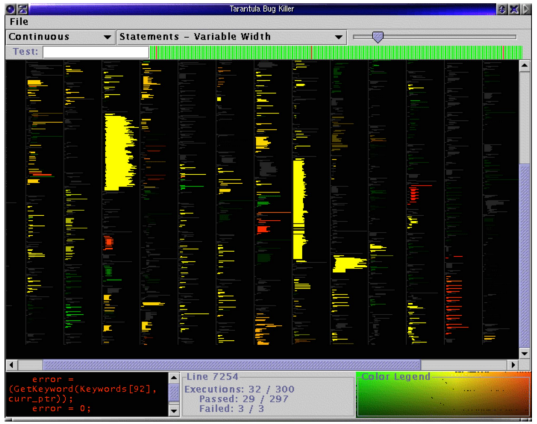
Zhuo (Cecilia) Chen - Bryn Mawr College
Chris Murphy - Swarthmore College

Background

- Locating faults to debug is a time-consuming and costly process
- Spectrum-Based Fault Localization (SBFL):
  - A technique to identify faulty code by analyzing program spectra (e.g., test coverage and execution results).
  - Combines passed/failed test outcomes with code coverage data to localize faults.
  - Ranks code line by their suspiciousness scores.

Tarantula Approach - Jones et al., 2002

1. Use the execution result with a test suite;
2. Compute the suspiciousness score;
3. Color mapping each code line.



		Test Cases					
		3,3,5	1,2,3	3,2,1	5,5,5	5,3,4	2,1,3
	mid() { int x,y,z,m;						
1:	read("Enter 3 numbers:",x,y,z);	●	●	●	●	●	●
2:	m = z;	●	●	●	●	●	●
3:	if (y<z)	●	●	●	●	●	●
4:	if (x<y)		●				
5:	m = y;		●				
6:	else if (x<z)	●				●	●
7:	m = y;	●					●
8:	else	●		●	●		
9:	if (x>y)			●			
10:	m = y;			●			
11:	else if (x>z)						
12:	m = x;						
13:	print("Middle number is:",m);	●	●	●	●	●	●
	} Pass/Fail Status	P	P	P	P	P	F

Ochiai Approach - Abreu et. al, 2006

- Introduced a different coefficient for computing the suspiciousness score
- Based on a similarity coefficient originally developed for biological taxonomy

Suspiciousness Score

Tarantula:

$$\text{suspiciousness\_T}(s) = \frac{\%\text{passed}(s)}{\%\text{passed}(s) + \%\text{failed}(s)}$$

Ochiai

$$\text{suspiciousness\_O}(s) = \frac{\text{failed}\ (s)}{\sqrt{\text{total failed} * (\text{failed}\ (s) + \text{passed}\ (s))}}$$

$$\text{color}(s) = \text{color}\ (\text{red}) + \text{suspiciousness\_T}(s) * \text{color range}$$

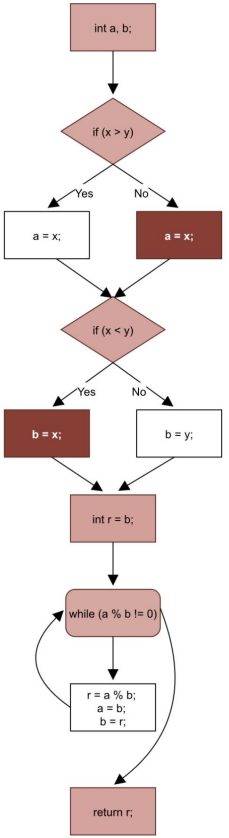
Contributions

- A new technique for providing users with a visualization for fault localization using a **Control Flow Graph**.
- The **+Edges** approach, i.e. Tarantula+Edges and Ochiai+Edges.
- A comparative real Java program analysis among three different techniques: Tarantula, Tarantula+Edges, and the Set-union technique.
- An empirical evaluation of Ochiai+Edges using 71 valid real word bugs from **Defects4J**.

Control Flow Graph

1. Use different opacity of the red color for coloring different blocks to represent their suspiciousness
2. Change the formats and colors of text

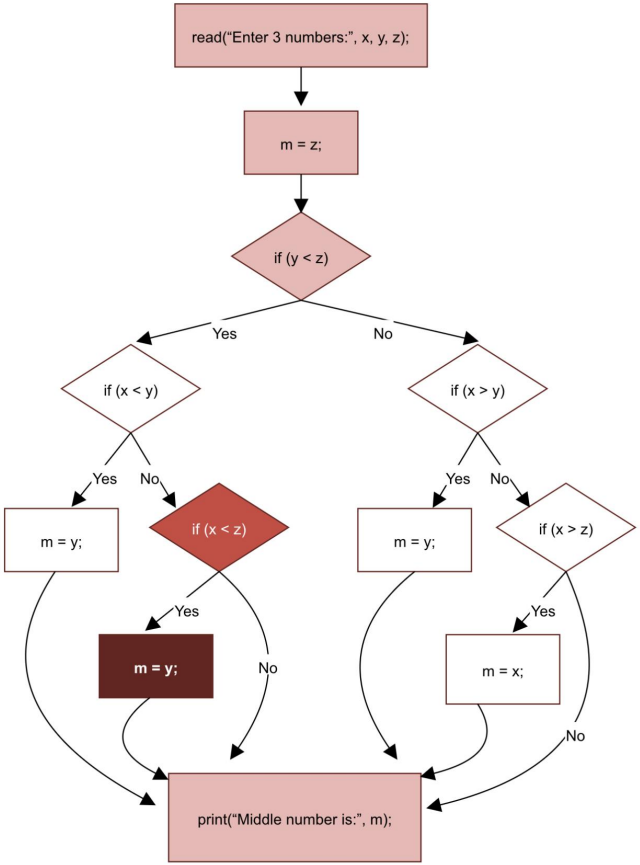
So that the user is more likely to notice the lines that are most suspicious



Control Flow Graph

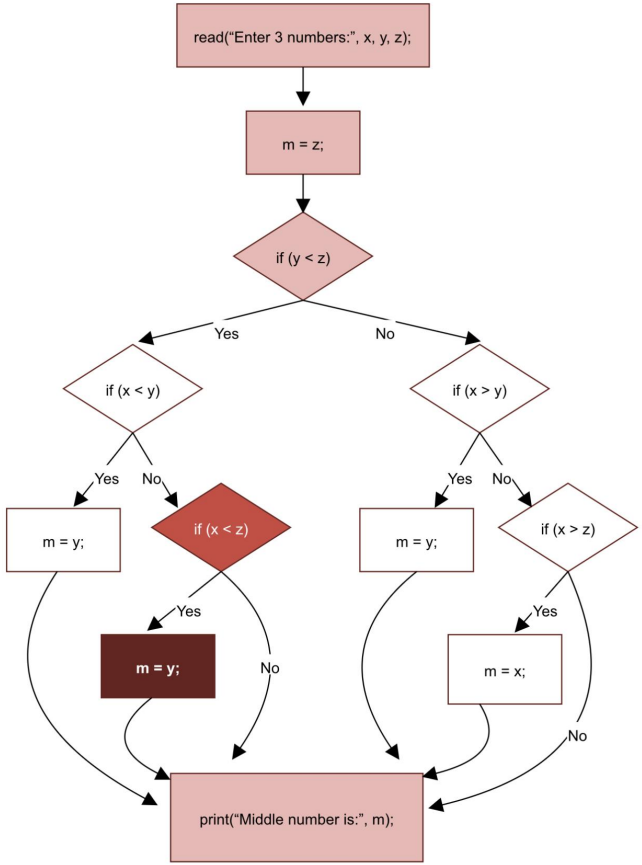
1. Use different opacity of the red color for coloring different blocks to represent their suspiciousness
2. Change the formats and colors of text

So that the user is more likely to notice the lines that are most suspicious



Control Flow Graph

		Test Cases					
		3,3,5	1,2,3	3,2,1	5,5,5	5,3,4	2,1,3
	mid() { int x,y,z,m;						
1:	read("Enter 3 numbers:",x,y,z);	●	●	●	●	●	●
2:	m = z;	●	●	●	●	●	●
3:	if (y<z)	●	●	●	●	●	●
4:	if (x<y)		●				
5:	m = y;		●				
6:	else if (x<z)	●				●	●
7:	m = y;	●					●
8:	else	●		●	●		
9:	if (x>y)			●			
10:	m = y;			●			
11:	else if (x>z)						
12:	m = x;						
13:	print("Middle number is:",m);	●	●	●	●	●	●
	} Pass/Fail Status	P	P	P	P	P	F



Jones et al., 2002

Edge Consideration

Given that $|a| < |b|$, returns the larger variable.

Line	Code
1	if (a $\geq$ 0)
2	if (b $>$ 0)
3	return b;
	else
4	return a;
	else
5	return b;

Edge Consideration

Given that $|a| < |b|$, returns the larger variable.

Line	Code
1	**if (a $\geq$ 5)**
2	if (b $>$ 0)
3	return b;
	else
4	return a;
	else
5	return b;

Edge Consideration

Given that $|a| < |b|$, returns the larger variable.

Line	Code	(5,4)	(-7,3)	(2,7)	(2,-5)
1	**if (a $\geq$ 5)**	•	•	•	•
2	if (b > 0)	•			
3	return b;	•			
	else				
4	return a;				
	else				
5	return b;		•	•	•
	Passed/Failed Status	Failed	Passed	Passed	Failed

Edge Consideration

Given that $|a| < |b|$, returns the larger variable.

Line	Code	(5,4)	(-7,3)	(2,7)	(2,-5)
1	**if (a ≥ 5)**	•	•	•	•
2	**if (b > 0)**	•			
3	**return b;**	•			
	else				
4	**return a;**				
	else				
5	**return b;**		•	•	•
	Passed/Failed Status	Failed	Passed	Passed	Failed

Line	Code	(5,4)	(-7,3)	(2,7)	(2,-5)
1a	**if (a ≥ 5)**	•			
2a	**if (b > 0)**	•			
3	**return b;**	•			
2b	**else**				
4	**return a;**				
1b	**else**		•	•	•
5	**return b;**		•	•	•
	Passed/Failed Status	Failed	Passed	Passed	Failed

Experiment

- Tarantula+Edges
  - Compare among Tarantula and Tarantula+Edges
  - based on randomly inserted faults in five methods from the Java Apache Commons Math library, and an iterative implementation of Euclid's GCD.
- Ochiai+Edges
  - Compare between Ochiai and Ochiai+Edges
  - based on 71 valid various real software faults from three different Java libraries from Defects4J.

Metrics

- Ranking of Faults (RF)
  - ranking information of the faults in the suspiciousness ranking.
  - Mean RF - the average of RF for all faults in the same faulty version.
  - Best RF - the highest ranking of faults among all faults in the same faulty version.
- Wasted Effort (WE)
  - # code lines investigated until locating the fault / # code line
  - Mean WE - average of WE for all faulty code lines in the same faulty version
  - Best WE - # of code line investigated before reaching **any of the faulty code lines** over the total # of code lines

Results - Tarantula

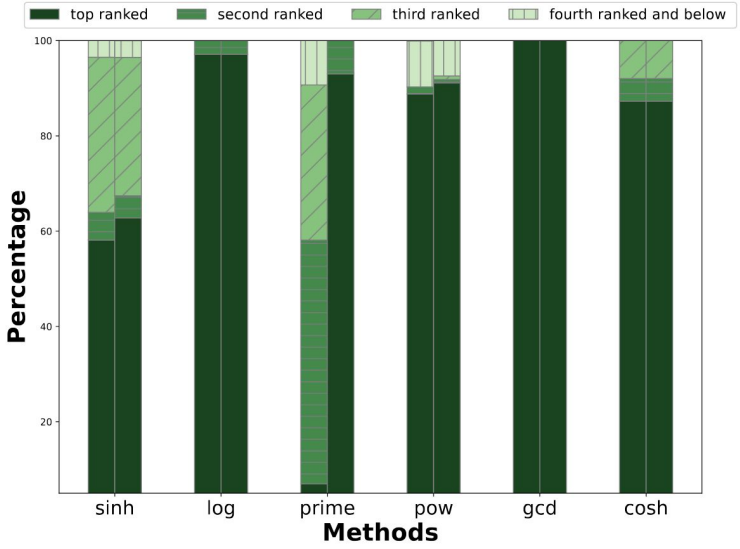


Figure 6: Ranking of Faults. For each program, rankings provided by Tarantula are shown on the left, and Tarantula+Edges is shown on the right.

Results - Ochiai

Among 25 buggy versions in Commons Math, 22 in JFreeChart, and 24 in Joda-Time.

- Improved Mean RF - 6 buggy versions
- Improved Best RF - 3 buggy versions
- Improved Mean WE - 12 buggy versions
- Improved Best WE - 14 buggy versions

Future work

- Apply +Edges modification for other SBFL approaches
- Compare performance with other state-of-the-art SBFL approaches, neural approaches, and Information-retrieval-based approaches.

Summary

- A new technique for providing users with a visualization for fault localization using a **Control Flow Graph**.
- The **+Edges** approach, i.e. Tarantula+Edges and Ochiai+Edges.
- A comparative real Java program analysis among three different techniques: Tarantula, Tarantula+Edges, and the Set-union technique.
- An empirical evaluation of Ochiai+Edges using 71 valid real word bugs from **Defects4J**.

Contact: Cecilia Chen zchen4@brynmawr.edu