The 35th ACM/SIGAPP Symposium On Applied Computing

Data-Flow-Based Evolutionary Fault Localization

Deuslirio Silva-Junior - Federal University of Goiás
Plinio S. Leitao-Junior - Federal University of Goiás
Altino Dantas - Federal University of Goiás
Celso G. Camilo-Junior - Federal University of Goiás
Rachel Harrison - Oxford Brookes University

INSTITUTO DE INFORMÁTICA UFG

CAPES

UFG UNIVERSIDADE FEDERAL DE GOIÁS

Introduction

- Software is adopted to manage critical tasks, however, it is hard to avoid the insertion of **software faults**;
- Fault localization (FL) is the activity of **precisely indicating** the faulty commands in a buggy program;
- It is known to be a **highly costly** and tedious process;
- Automating this process has showing to be a **challenging problem**.

Introduction

- A common approach is to associate **suspiciousness scores** to potential locations of a software fault;
- These scores build a **ranking of suspicious program elements** to guide the investigation of a fault;
- Most methods are **FL heuristics** that use a coverage spectrum to compute how suspicious each program element is.

Tarantula

The main approaches for fault localization using coverage analysis, are **heuristics** that address **suspiciousness score** for each command (node). Through this scores it is possible to make a ranking of suspicious elements.

$$Tarantula(node) = \frac{\frac{ef(node)}{ef(node)+nf(node)}}{\frac{ef(node)}{ef(node)+nf(node)} + \frac{es(node)}{es(node)+ns(node)}}$$

Search Based Fault Localization

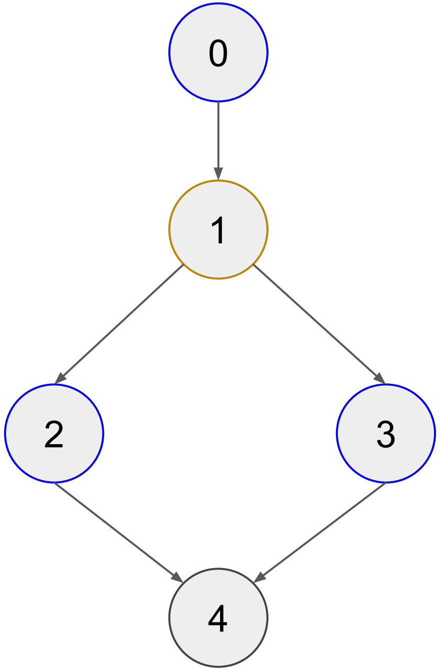
- *Wang et al.* Combine heuristics using search based optimization algorithms.

- Genetic algorithm (GA) is used to find weights *w* to be applied in each heuristic *H* value.

$$HC(node) = w_1 \times H_1(node) + w_2 \times H_2(node) + \ldots + w_n \times H_n(node)$$

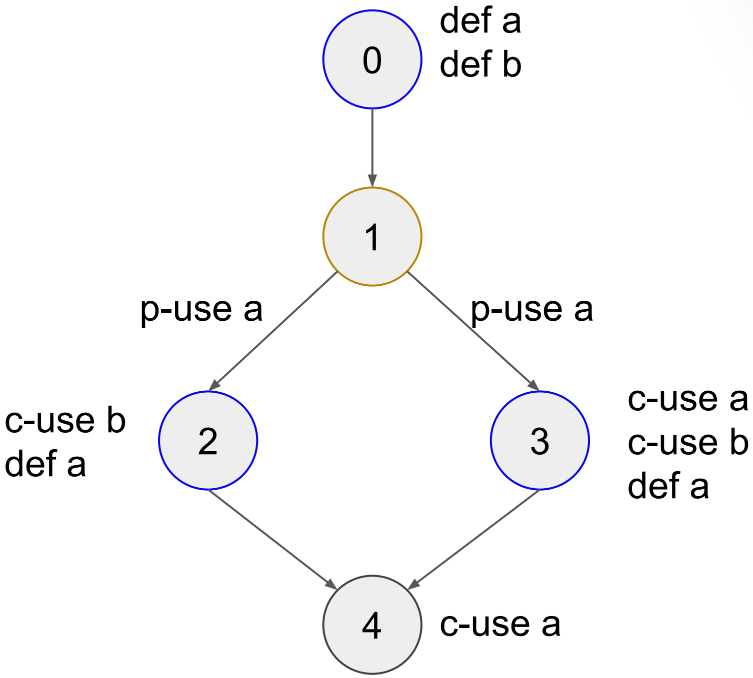
Control-flow

```
int a = 4; //0
int b = 6; //0
if (a < 5){ //1
     a = b; //2
}else{
     a = a + b; //3
}
return a; //4
```



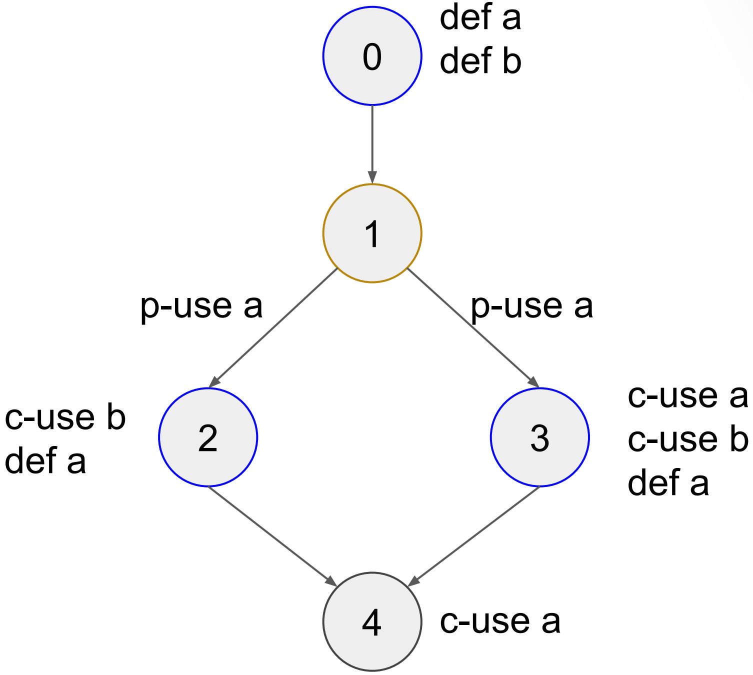
Data-flow

```
int a = 4; //0
int b = 6; //0
if (a < 5){ //1
      a = b; //2
}else{
      a = a + b; //3
}
return a; //4
```



def-use association (dua)

Variable	def node	use node
a	0	1 - 2
a	0	1 - 3
b	0	2
a	0	3
b	0	3
a	2	4
a	3	4



Contributions

- an approach to **compute the suspiciousness** score for each program element by using the **analysis of data-flow** information;
- an evolutionary method that **combines distinct sources of fault information** to improve the precision on locating faults;
- a new metric to investigate the **dependencies of tie-break strategies** in building the ranking of suspicious commands;
- the **evaluation and discussion** about the results of the proposed methods, comparing them with popular baselines, and using well-known evaluation metrics.

Approach

Approach

- First step: **Obtain** suspiciousness score using data-flow information for each command (node)
- Second step: **combine** different suspiciousness scores obtained from both, control- and data-flow using genetic algorithm.

Dua suspiciousness score (*H*) are obtained applying traditional heuristics in data-flow associations

$$Tarantula(dua) = \frac{\frac{ef(dua)}{ef(dua)+nf(dua)}}{\frac{ef(dua)}{ef(dua)+nf(dua)} + \frac{es(dua)}{es(dua)+ns(dua)}}$$

Approach

1 - **allduas(node) set**. contains all duas that a specific node is in.

$$alldu as(node) = \bigcup dua_i \quad | \; node \in dua_i$$

2 - **dua-to-node score (*Ĥ(node)*)**. This score denotes the suspiciousness of a node, obtained from the data-flow analysis.

$$\widehat{H}(node) = max\big(H(dua_i)\big) \quad | \; dua_i \in allduas(node)$$

Approach

With dua-to-node score is possible combine different heuristics using only control-flow as presented by Wang *et al.* but also use data-flow information as follows:

$$\widehat{HC}(node) = w_1 \times \widehat{H}_1(node) + w_2 \times \widehat{H}_2(node) + \ldots + w_n \times \widehat{H}_n(node)$$

$$HC_{hyb}(node) = w_1 \times H_1(node) + \ldots + w_n \times H_n(node) + w_{n+1} \times \widehat{H}_1(node) + \ldots + w_{n+m} \times \widehat{H}_n(node)$$

Experiments

Subject Programs

Siemens Suite - 112 versions of C programs; jsoup - 36 versions, JAVA

Program	LOC	Versions	Number of test cases
printtokens	472	6	4030
printtokens2	399	9	4415
replace	512	26	5542
schedule	292	8	2650
schedule2	301	7	2710
tcas	141	35	1608
tot_info	440	21	1051
jsoup	10K	36	468

Heuristics

H_1: Tarantula	H_7: GP13
H_2: Ochiai	H_8: OP2
H_3: DStar	H_9: Wong3
H_4: OP	H_{10}: Zoltar
H_5: Ample	H_{11}: Kulczynski2
H_6: Jaccard	H_{12}: Barinel

Genetic algorithm parameters

- Individual genotype is a binary string with seven bits for each heuristic suspiciousness score;
- Bitflip mutation operator (rate 0.01);
- Two-point crossover (rate 0.6);
- Tournament 3 selection strategy;
- 250 generations;
- 50 individuals as population size
- We used DEAP[1](Distributed Evolutionary Algorithms in Python) to implement the GA and the R Project[2] for statistical analysis.

1 deap.readthedocs.io
2 r-project.org

Experimentation process

- Genetic algorithm to search weights for a linear combination of suspiciousness values;
- Fitness function is the average proportion of code investigated before finding the faults;
- To deal with **Overfitting**, we conducted the experiments by applying a Three-Fold Cross Validation;
- To ease the GA **stochasticity** effects, we performed 30 executions of the cross-validation process;

RQ1: Is the proposed method competitive for locating software faults?

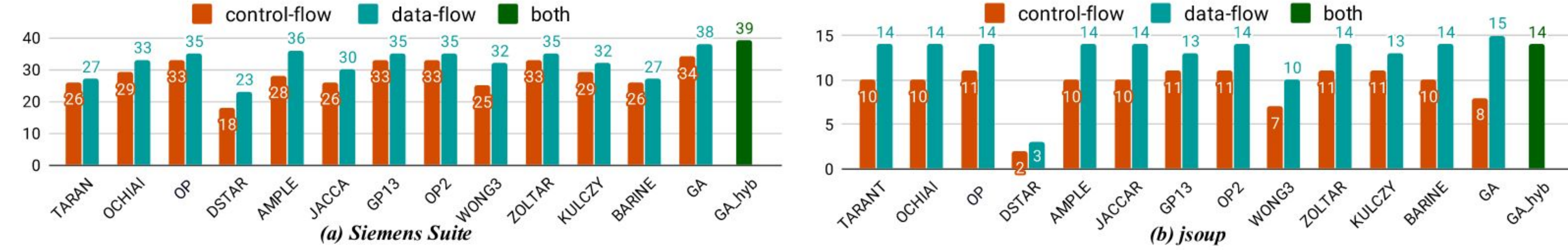


Figure 3: Accuracy (*acc@5*) results.

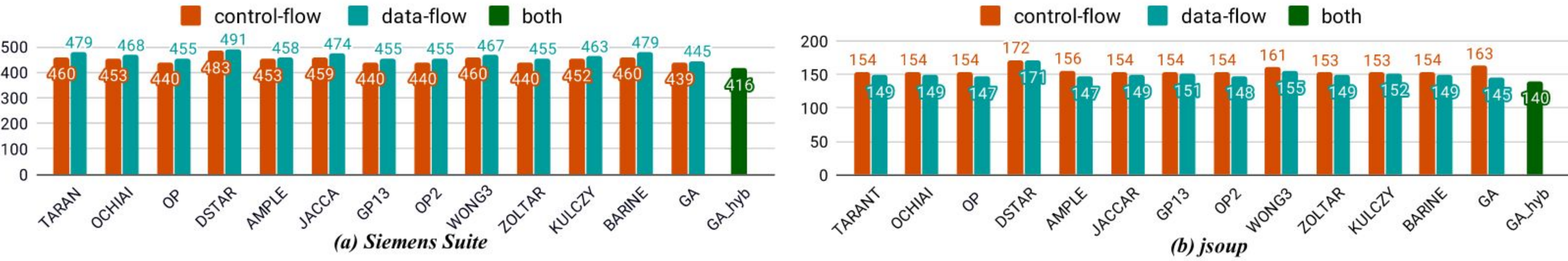


Figure 4: Wasted effort (*wef@5*) results.

Absolute Critical Tie

Xu *et al.* define *tie* as a set of statements, each of which has been assigned the same suspiciousness score, a *critical tie* is a tie that contains a faulty statement, the higher the number of ties that involve faulty statements, the **harder it is to precisely estimate** at what ranking position during the examination.

Absolute Critical Tie (*actie@n*): we define a new evaluation metric called Absolute Critical Tie (actie@n) that uses the **number of critical ties in the top-n elements of suspiciousness rankings.** One may interpret the metric as the potential effort wasted with critical ties, *i.e.*, the number of non-faulty program elements tied with faulty ones in the top-n ranked elements.

actie does not express the effectiveness of FL methods. **It is an auxiliary metric**, and it should be used together with the other metrics.

RQ2: Does the evolutionary combination of control-flow and data-flow coverage spectra improve fault localization ability?

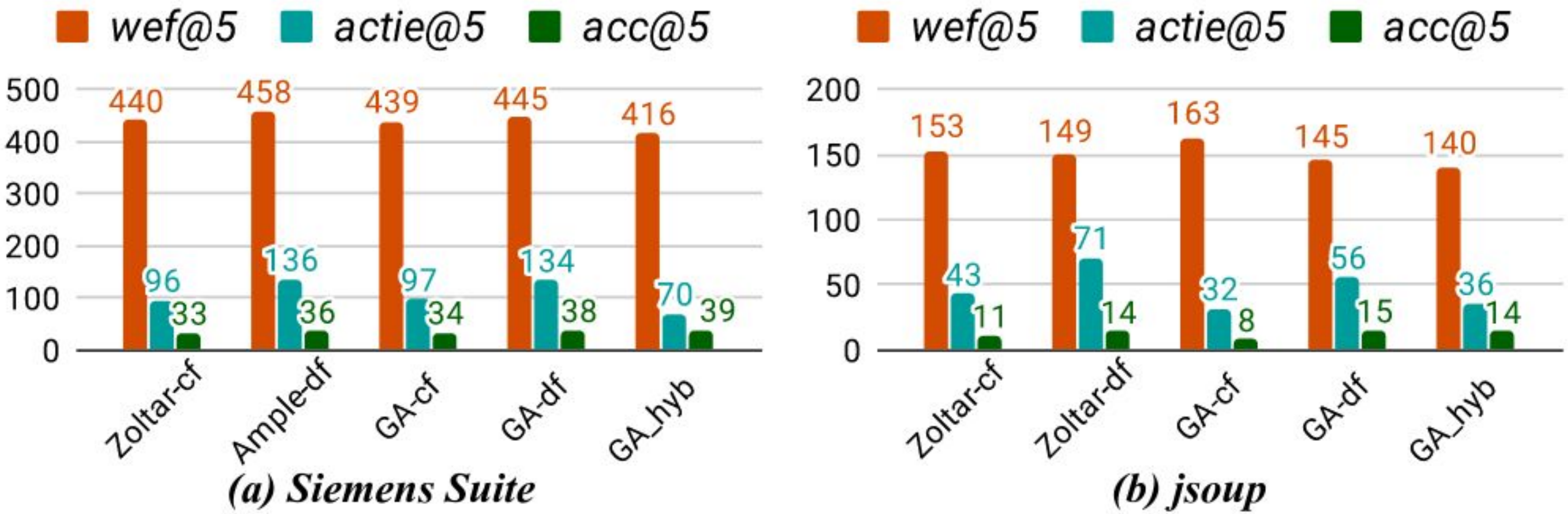


Figure 5: Results for Absolute critical tie (*actie@5*) in relation to Wasted effort (*wef@5*) and Accuracy (*acc@5*).

Statistical Tests Analysis

- It presented no significant statistical difference when compared to H-df.
- GA_hyb outperforms all the other methods in *acc@5* for Siemens Suite;
- In the case of jsoup, GA_hyb was statistically superior to GA-cf and H-cf, and inferior to GA-df, in relation to acc@5;
- Except against GA-cf in Jsoup, GA_hyb had lower critical ties (actie) than all the other methods

Table 3: Results from the Vargha & Delaney $\hat{A}_{12}$ test by considering all the FL methods and all metrics

		Siemens Suite			jsoup		
		GA_hyb	GA-cf	GA-df	GA_hyb	GA-cf	GA-df
GA-cf	acc@5	0.00			0.00		
	wef@5	1.00			1.00		
	actie@5	1.00			0.14		
GA-df	acc@5	0.15	1.00		0.82	1.00	
	wef@5	1.00	0.99		0.88	0.00	
	actie@5	1.00	1.00		1.00	1.00	
H-cf	acc@5	0.00	0.13	0.00	0.00	0.98	0.00
	wef@5	1.00	0.80	0.00	1.00	0.00	1.00
	actie@5	1.00	0.18	0.00	1.00	1.00	0.00
H-df	acc@5	0.00	1.00	0.00	**0.53**	1.00	0.03
	wef@5	1.00	1.00	1.00	1.00	0.00	0.83
	actie@5	1.00	1.00	1.00	1.00	1.00	1.00

Answers to RQs

RQ1: Is the proposed method competitive for locating software faults? the dua-to-node score performs better than control-flow heuristics in Siemens Suite and jsoup, but analyzing Wasted Effort, the same does not happen with the Siemens Suite results.

RQ2: Does the evolutionary combination of control-flow and data-flow coverage spectra improve fault localization ability? The evolutionary approach GA_hyb performed as accurately as the dua-to-node heuristics to locate faults looking at the top-5 elements of the suspiciousness ranking and demonstrated improvement in the wef and actie results.

Threats to validity

- GA approaches suffer from stochasticity. We conducted 30 executions of each GA setup, and applied statistical tests.

- We utilized a novel metric (actie) to investigate FL methods. But, it was used only to support the results of other metrics, which are well established in the literature.

- Large scale assessment is needed. The subject programs present different sizes, programming languages, seeded and real faults.

Final Remarks

- This paper reports an investigation about the use of data-flow based suspiciousness scores in FL methods, and also their behavior in evolutionary combinations of control and data-flow heuristics;
- The data-flow based approaches demonstrate superior results to their control-flow version;
- In future work, we will investigate attributes of the test that impact the occurrence of ties of the FL methods;
- We intend to analyze how the information concerning ties may also be used to guide the search in evolutionary FL approaches.

Thank you.

corresponding author at: deuslirio.junior@gmail.com

CAPES

INSTITUTO DE INFORMÁTICA UFG

UFG UNIVERSIDADE FEDERAL DE GOIÁS

I4Soft Intelligence for Software