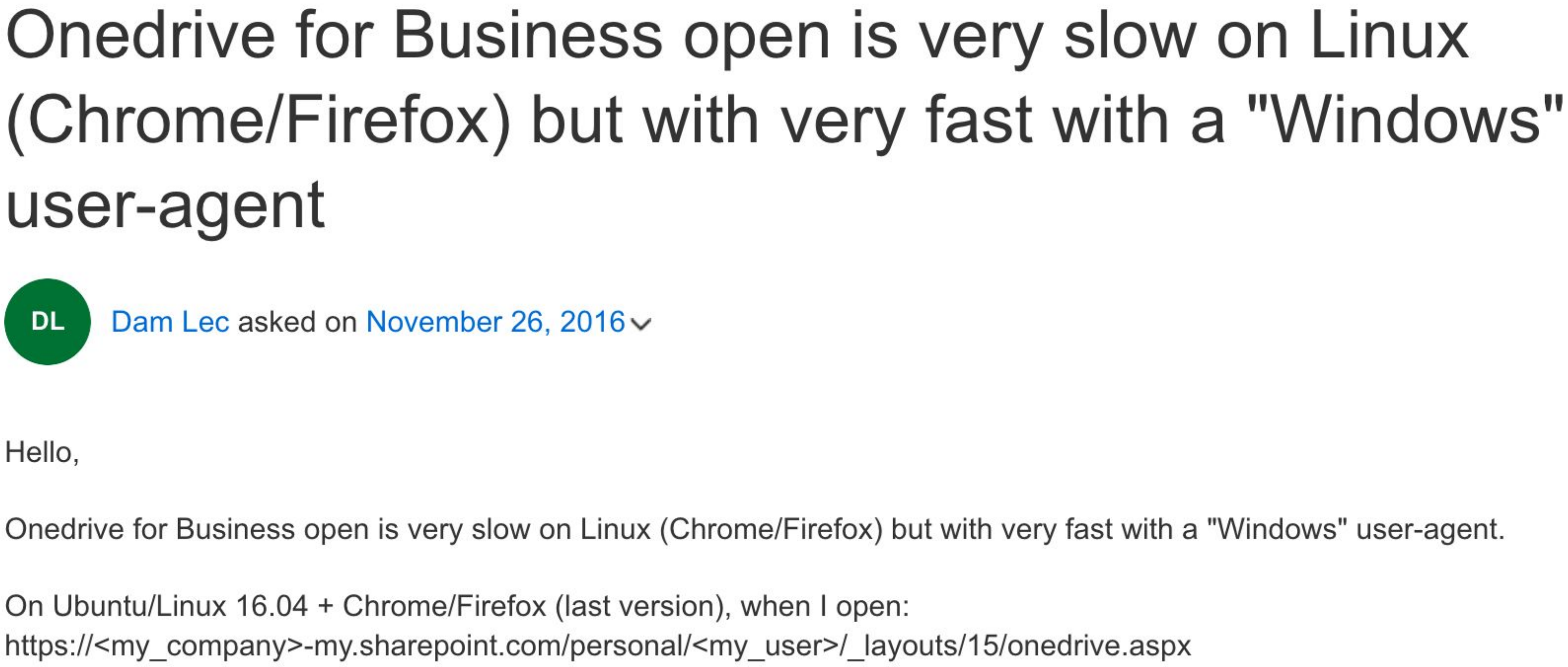
Automating Performance Bottleneck Detection using Search-Based Application Profiling

Sal Rahman
Grant Wallis
Saad Mahboob

Motivation



Onedrive for Business open is very slow on Linux (Chrome/Firefox) but with very fast with a "Windows" user-agent

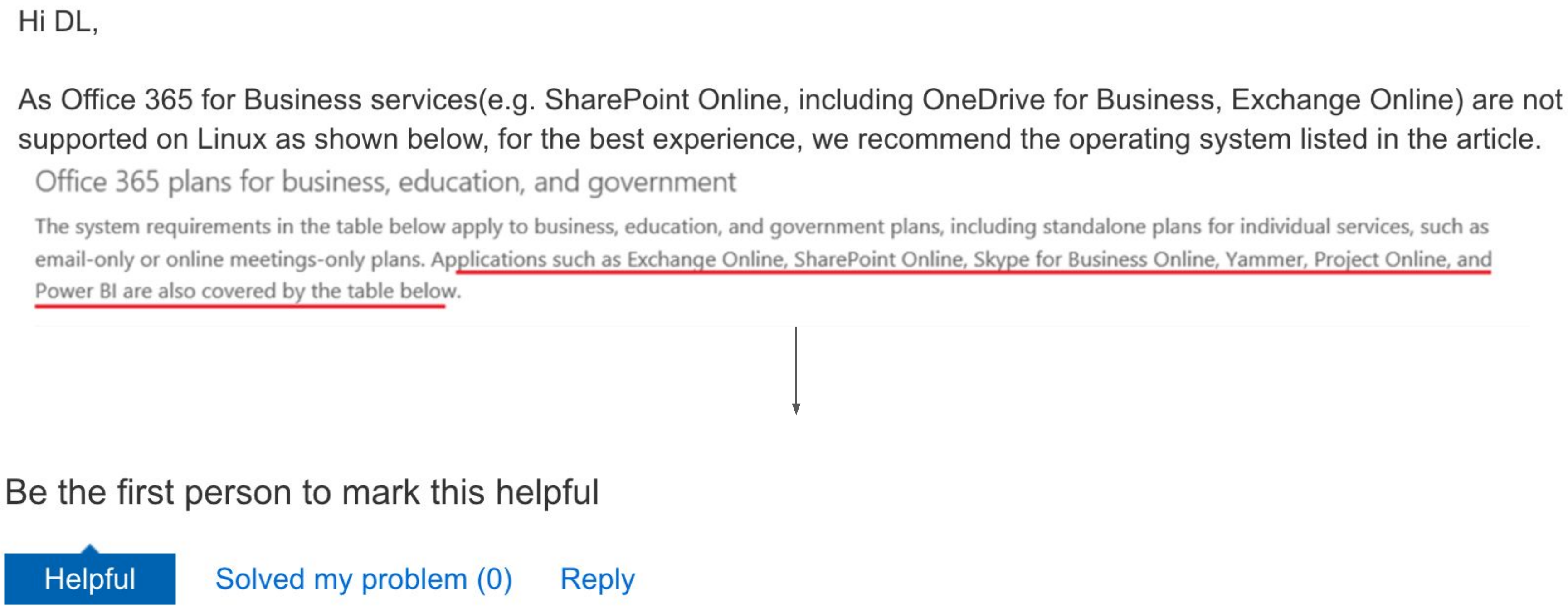
Dam Lec asked on November 26, 2016

Hello,

Onedrive for Business open is very slow on Linux (Chrome/Firefox) but with very fast with a "Windows" user-agent.

On Ubuntu/Linux 16.04 + Chrome/Firefox (last version), when I open:
https://<my_company>-my.sharepoint.com/personal/<my_user>/_layouts/15/onedrive.aspx

Motivation (Cont’d)



Hi DL,

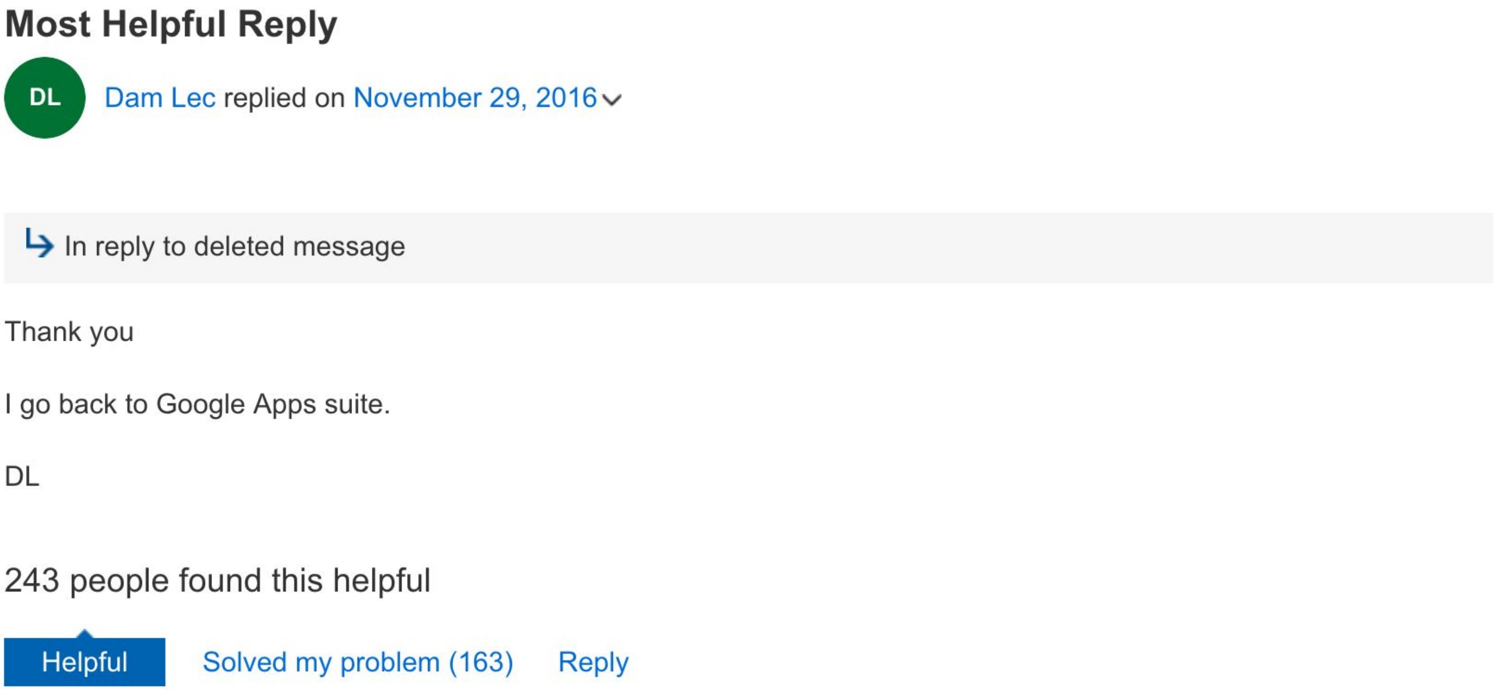
As Office 365 for Business services(e.g. SharePoint Online, including OneDrive for Business, Exchange Online) are not supported on Linux as shown below, for the best experience, we recommend the operating system listed in the article.

Office 365 plans for business, education, and government

The system requirements in the table below apply to business, education, and government plans, including standalone plans for individual services, such as email-only or online meetings-only plans. Applications such as Exchange Online, SharePoint Online, Skype for Business Online, Yammer, Project Online, and Power BI are also covered by the table below.

Be the first person to mark this helpful

Motivation (Cont’d)



Most Helpful Reply

Dam Lec replied on November 29, 2016

In reply to deleted message

Thank you

I go back to Google Apps suite.

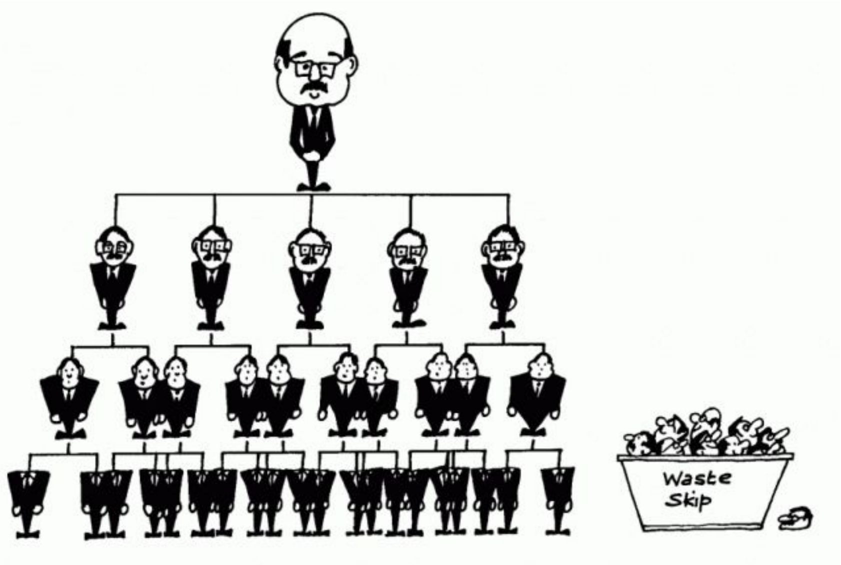
DL

243 people found this helpful

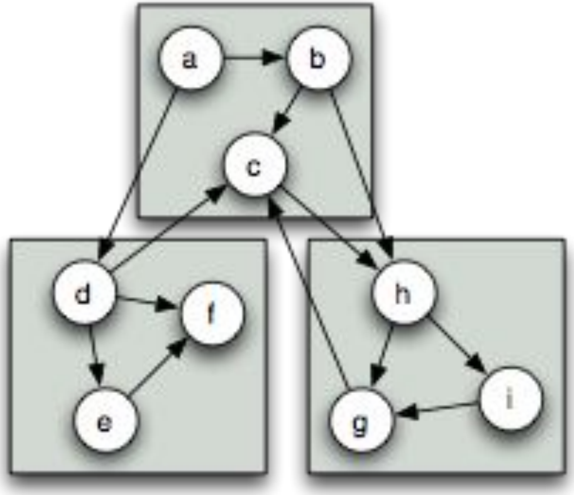
What could have caused the above bug?

Two approaches for performance analysis

Top-down



Granular



Let’s analyze this

```
let x, y, z, u
v = A.m( x, y )
if ( v > z ) {
  C.h(B.m(v))
} else {
  D.h(B.m(v))
}
```

Prior Work

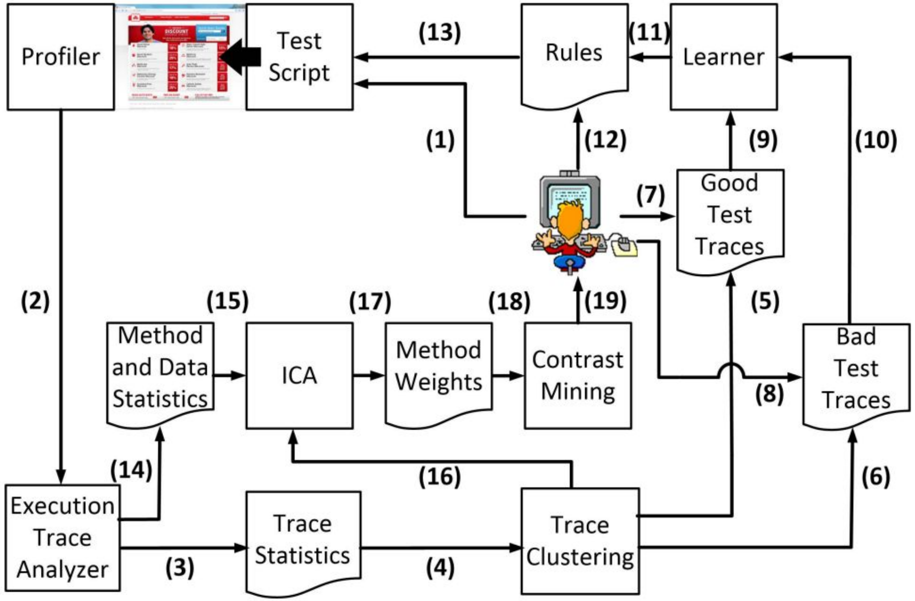
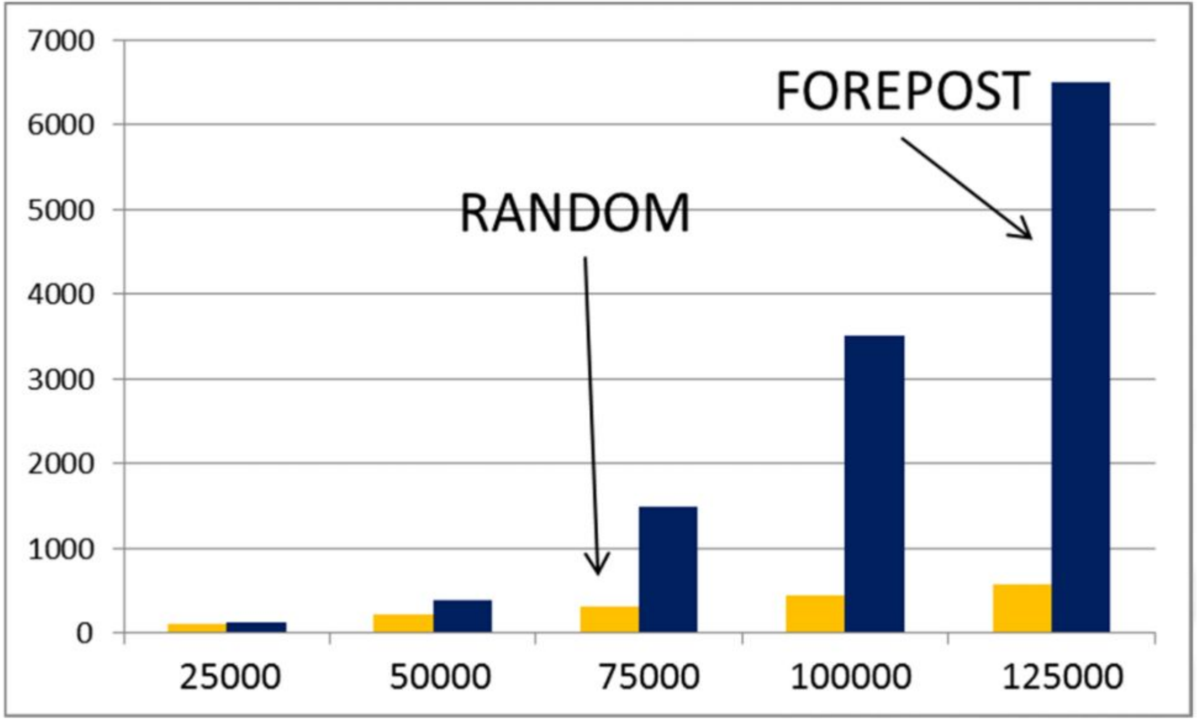


Figure 1: The Architecture of FOREPOST.

Why not this?

Or even this?

According to the authors, probably this



Dynamic, non-deterministic approach; GA

Why genetic algorithm?

(Why not genetic algorithm?)

The GA process

```
let p = initializePopulation()

while (!terminate(p)) {
  p = crossover(p)
  p = mutation(p, i)

  f = p.map(el => ({ individual: el, fitness: fitness(el) }))
  f.sort((a, b) => b.fitness - a.fitness)

  p = f.first(floor(f.length / 2))
}
```

The process (Cont’d) (ICA/BSS)

$$||\mathbf{x}|| = ||\mathbf{A}|| \cdot ||\mathbf{s}||$$

S =

	Feature 1	Feature 2	Feature 3
Method 1	...	...	...
Method 2	...	...	...
Method 3	...	...	...

$$D_{e_g} = \sqrt{\sum_{i=0}^{N_{M_g}} \sum_{j=0}^{N_{R_g}} (S_{Good}^{ij} - S_{Bad}^{kl})^2}$$

`GA-Prof` at a glance

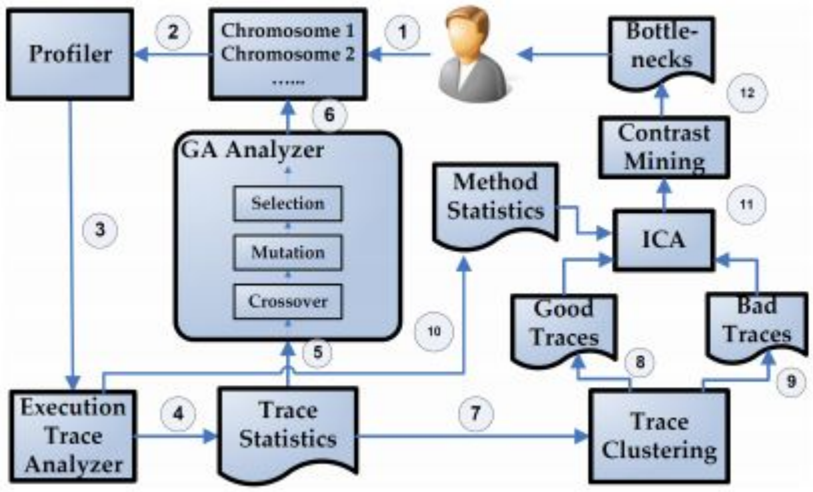


Figure 2: The architecture and workflow of `GA−Prof`.

Empirical Results

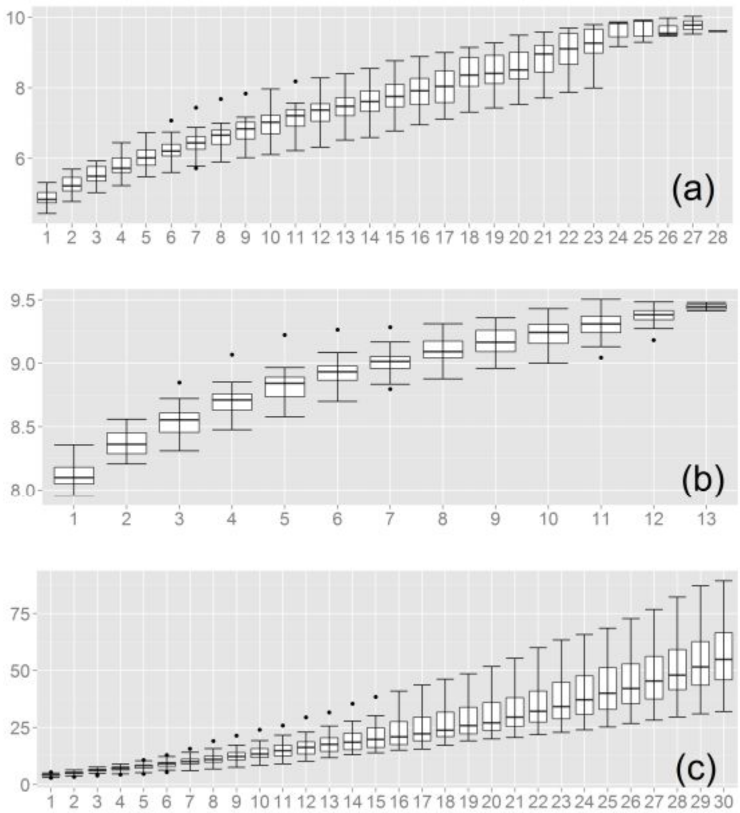
- Tested on 3 web applications:

(a) JPetStore - 69.4%

(b) DellDVDStore - 14.8%

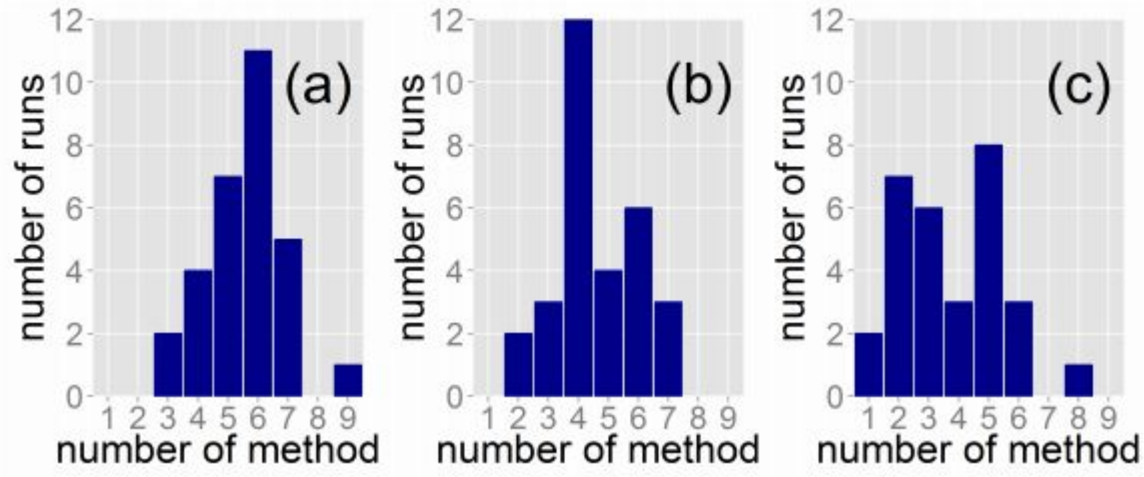
(c) Agilefant - 1409%

- Null hypothesis: do these results differ from random input?
  - Rejected Null hypothesis for all 3.
  - What do these results imply?



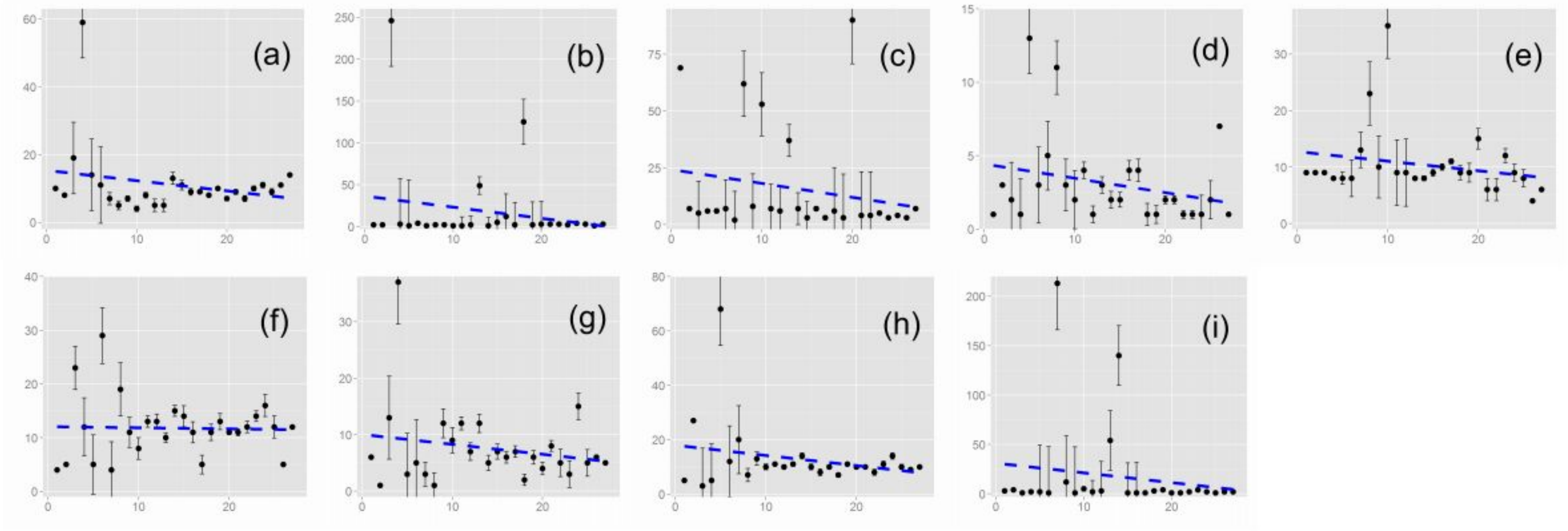
Performance Bottlenecks

- Bottlenecks are injected.
- 80% of the time five or more of the injected bottlenecks will be found (for a).
- Out of how many?



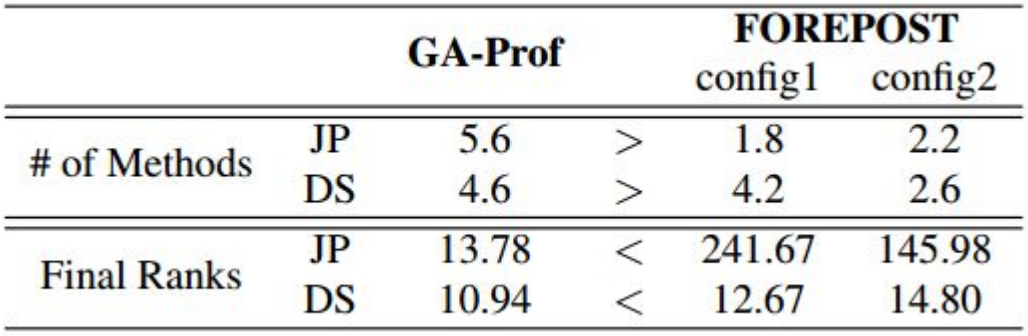
Ranking injected bottlenecks

Just look at that consistency!



Comparison with FOREPOST

- Ran 5 times.
- # Methods is the number of injected methods captured in a run.
- Final Ranks is the rank of the injected bottlenecks of the final ranked list.
- JP has 384 methods, DS is not specified.



		GA-Prof		FOREPOST	
				config1	config2
# of Methods	JP	5.6	>	1.8	2.2
	DS	4.6	>	4.2	2.6
Final Ranks	JP	13.78	<	241.67	145.98
	DS	10.94	<	12.67	14.80