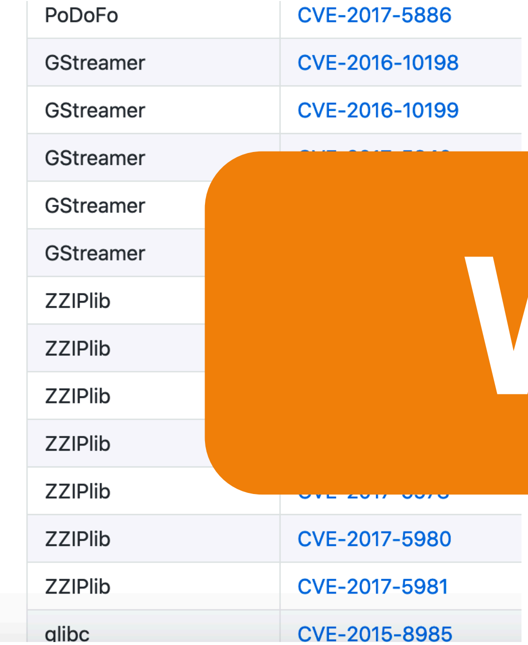
Ankou: Guiding Grey-box Fuzzing towards Combinatorial Difference

Valentin Manès[1], Soomin Kim[2], Sang Kil Cha[2]

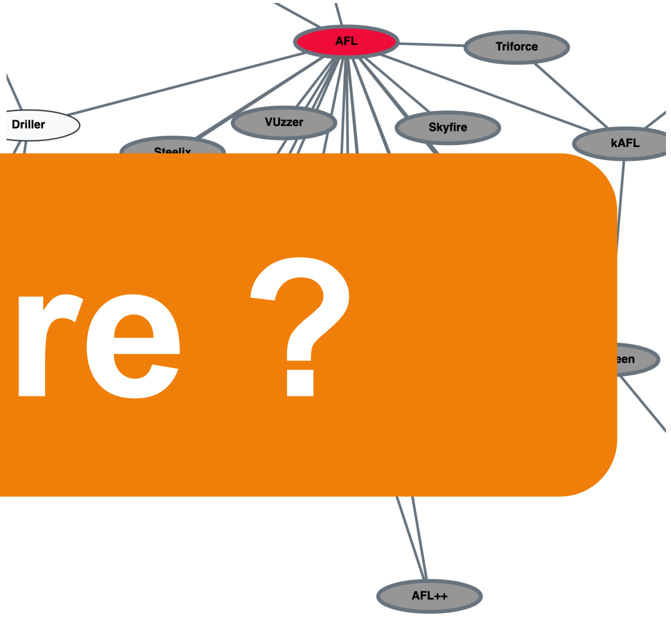
[1]CSRC, KAIST [2]KAIST

The Success of Grey-box Fuzzing

“**OSS-Fuzz** has found over 20,000 bugs in 300 open source projects.”

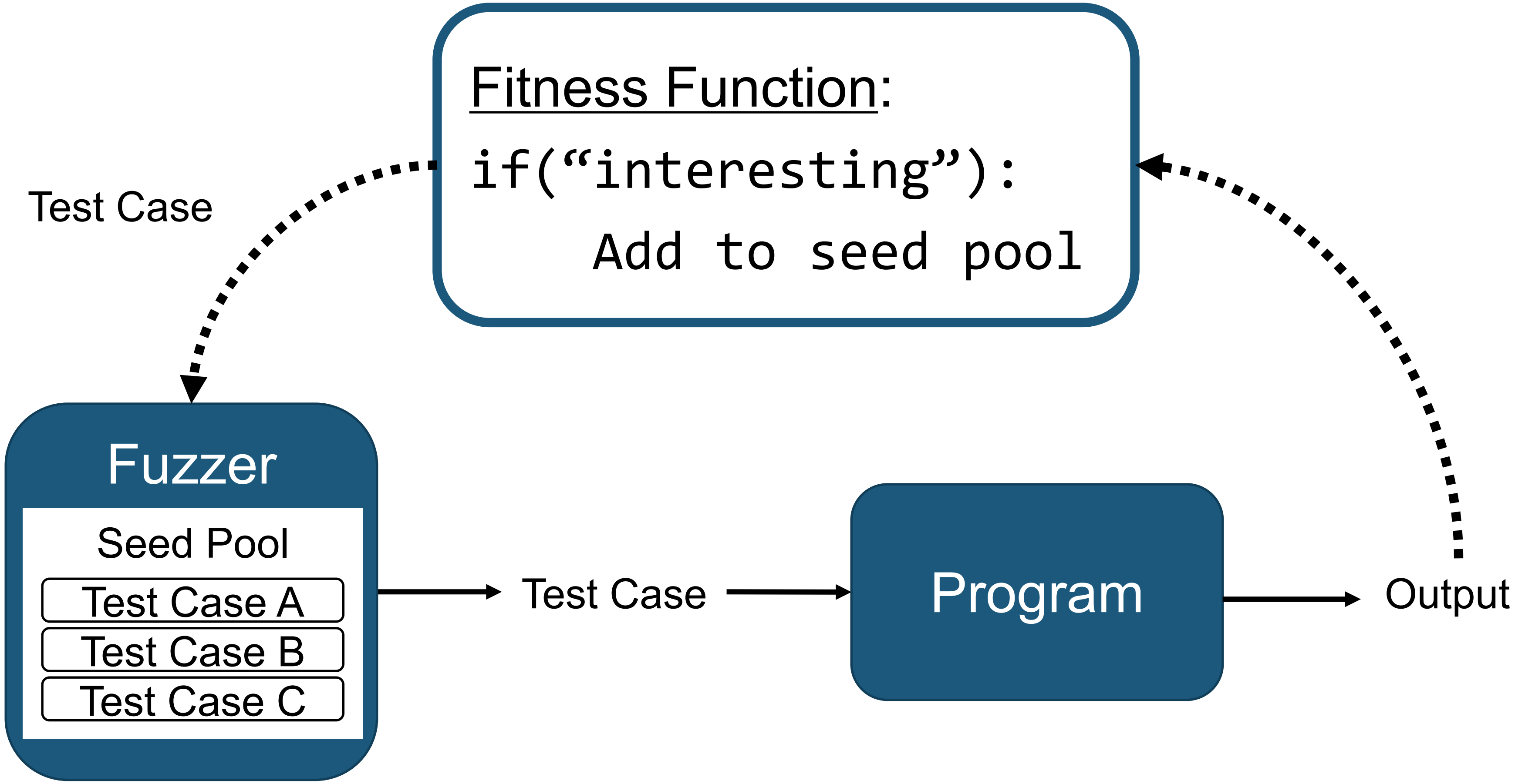


PoDoFo	CVE-2017-5886
GStreamer	CVE-2016-10198
GStreamer	CVE-2016-10199
GStreamer	
GStreamer	
GStreamer	
ZZIPlib	
ZZIPlib	
ZZIPlib	
ZZIPlib	
ZZIPlib	
ZZIPlib	CVE-2017-5980
ZZIPlib	CVE-2017-5981
glibc	CVE-2015-8985



Why one more ?

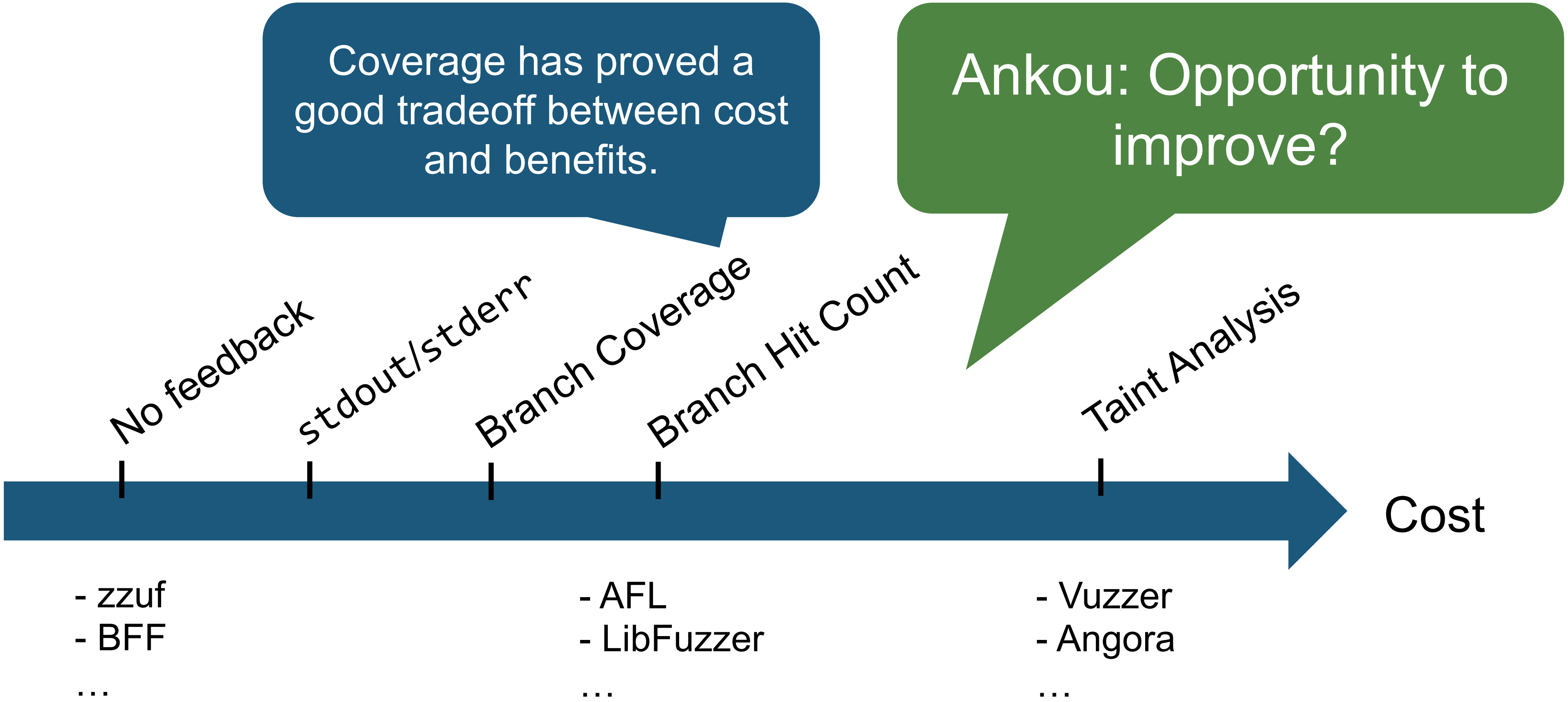
Grey-box, How?



Fitness Function:

```
if("interesting"):
    Add to seed pool
```

Which Feedback?



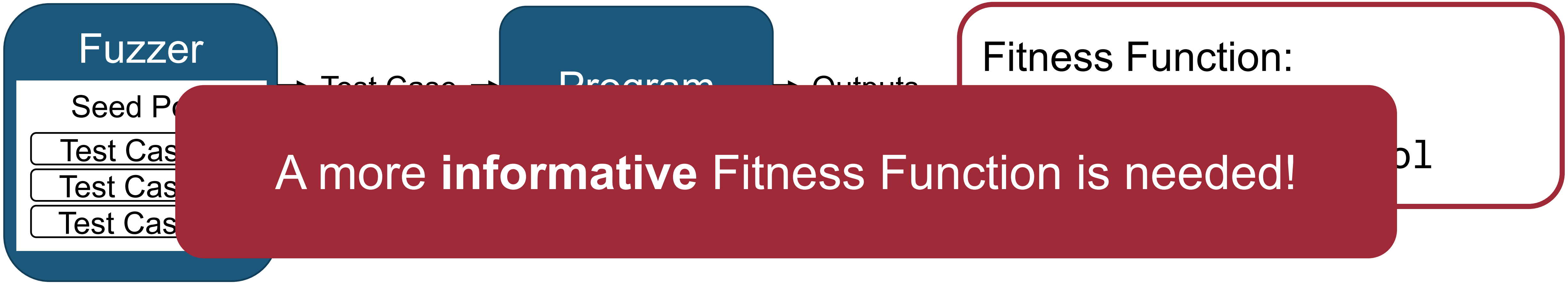
Coverage has proved a good tradeoff between cost and benefits.

Ankou: Opportunity to improve?

Coverage-Based Fuzzing

```
int combinedBranches(char *data) {
    int bits = 0;
    if (data[0] == 'A') bits |= 1;
    if (data[1] == 'B') bits |= 2;
    if (data[2] == 'C') bits |= 4;
    if (bits == 7)
        printf("BINGO\n");
    return 0;
}
```

Test Case	A	B	C	D
Value	“A”	“BB”	“AB”	“ABC”
Branch 1	X		X	X
Branch 2		X	X	X
Branch 3				X



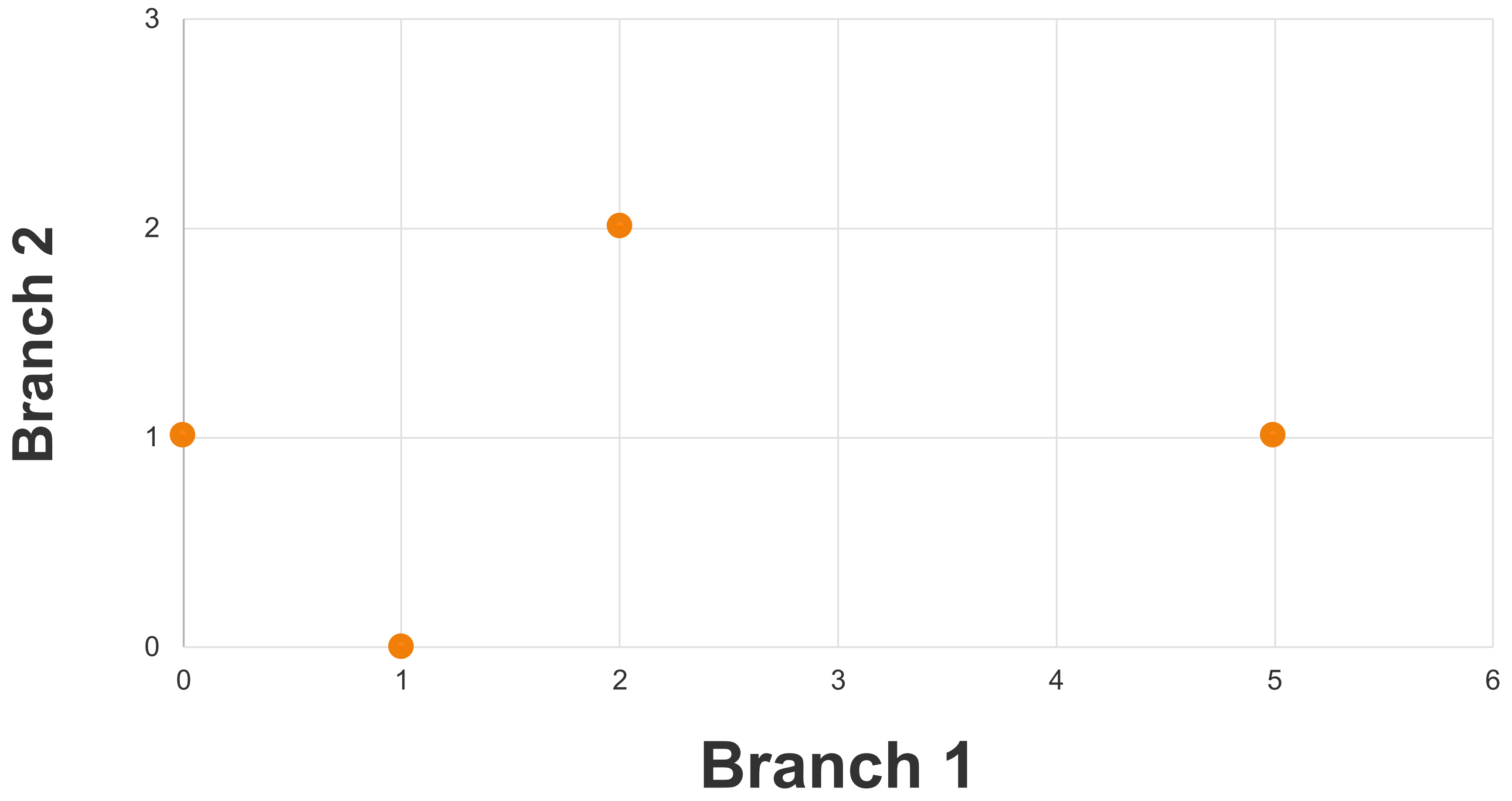
A more **informative** Fitness Function is needed!

Informative Fitness with Combination

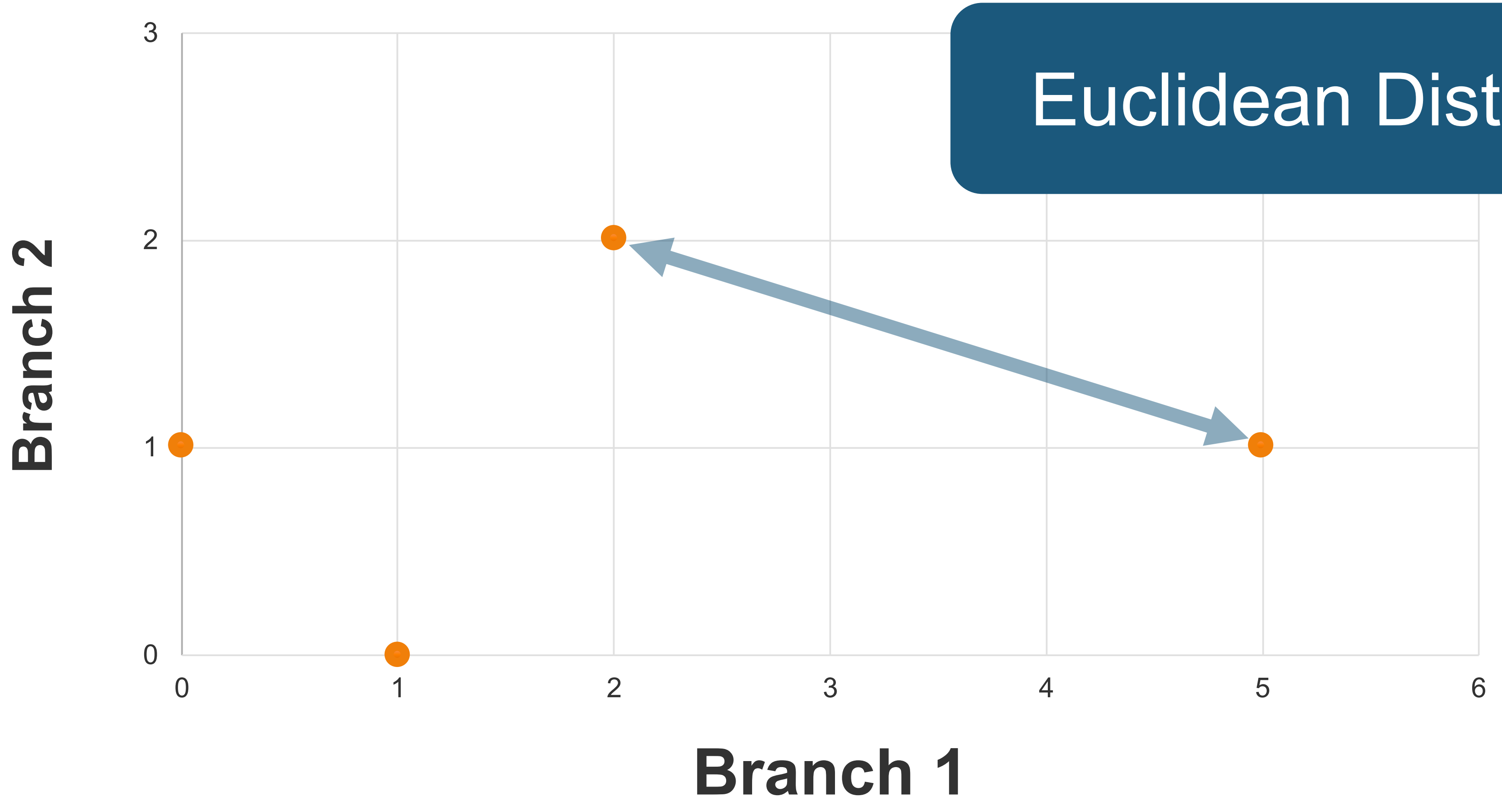
Ankou goal: developing a fitness function taking into account **combinations**.

1. **Quantify** the difference between program executions.
2. Make fitness computation **fast**.
3. Make the fitness **adaptive** to the program.

Point Representation

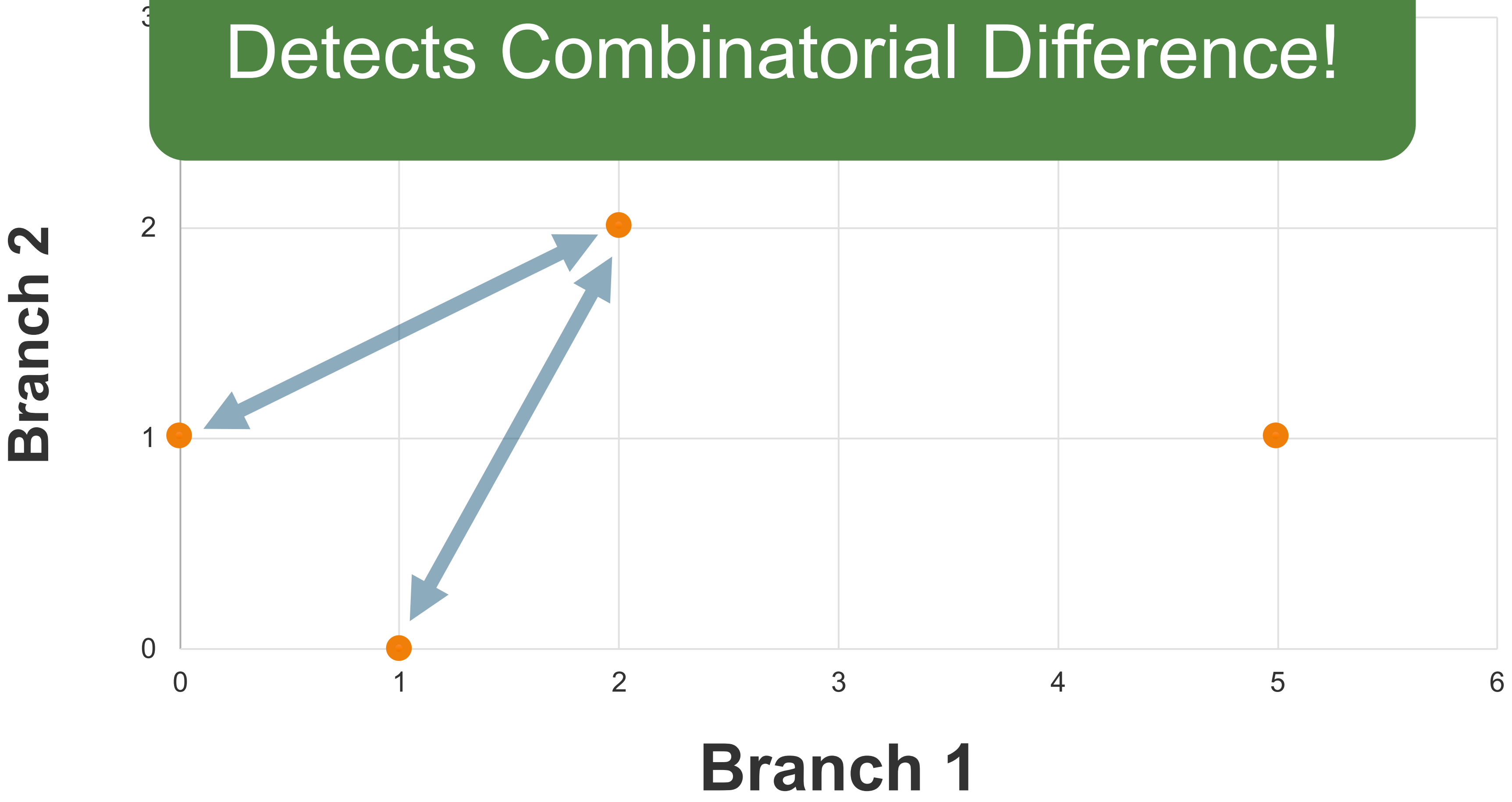


Distance between Executions



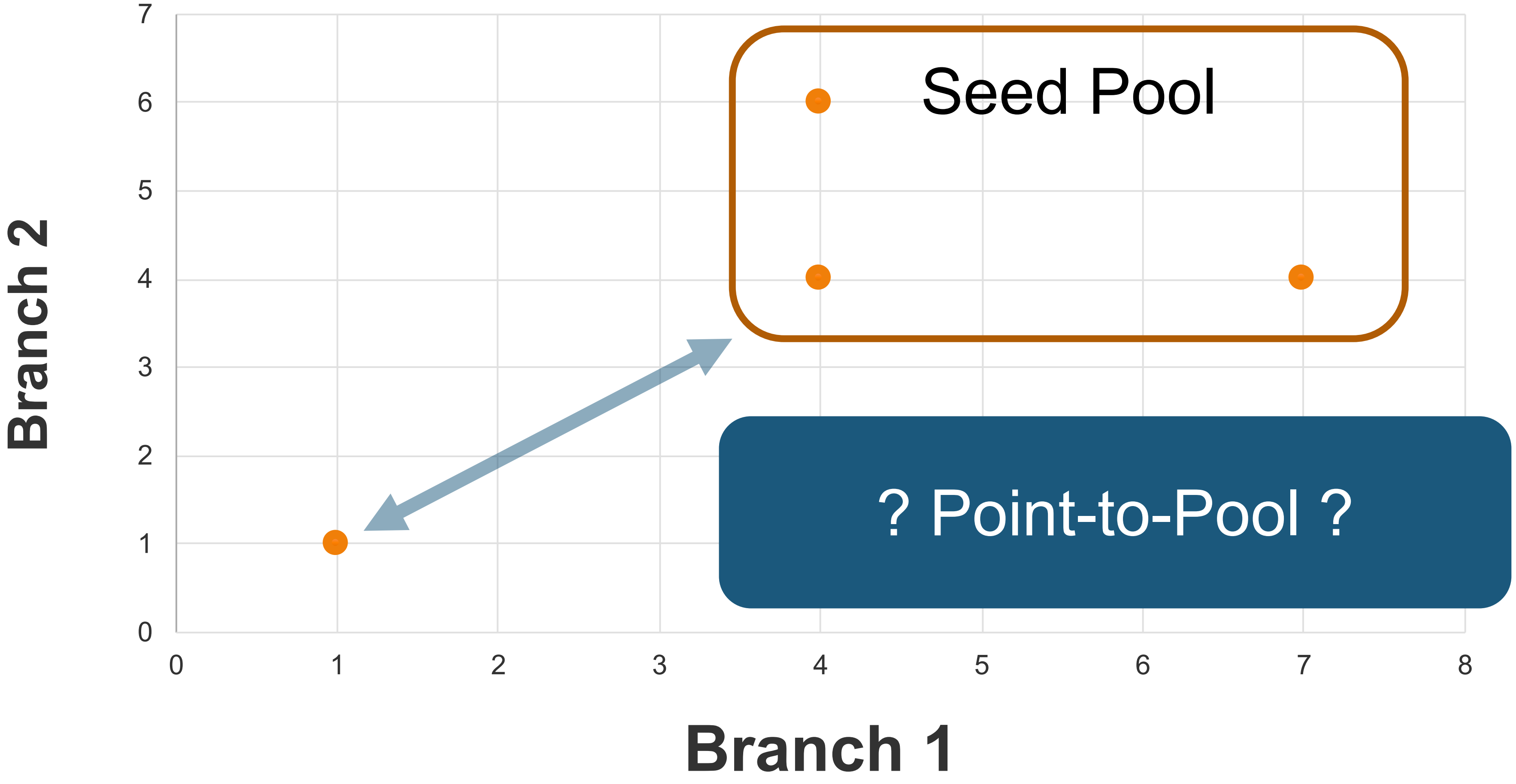
Euclidean Distance

Distance between Executions



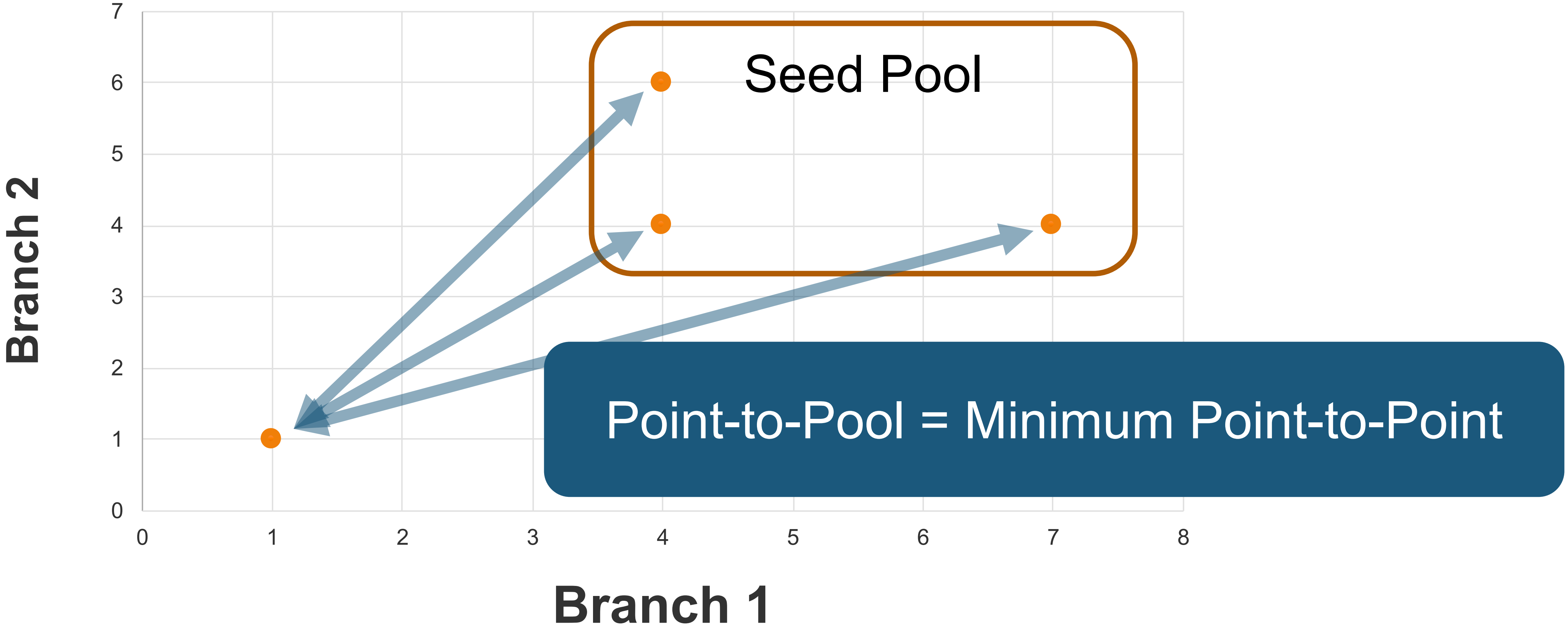
Detects Combinatorial Difference!

Distance-based Fitness Function



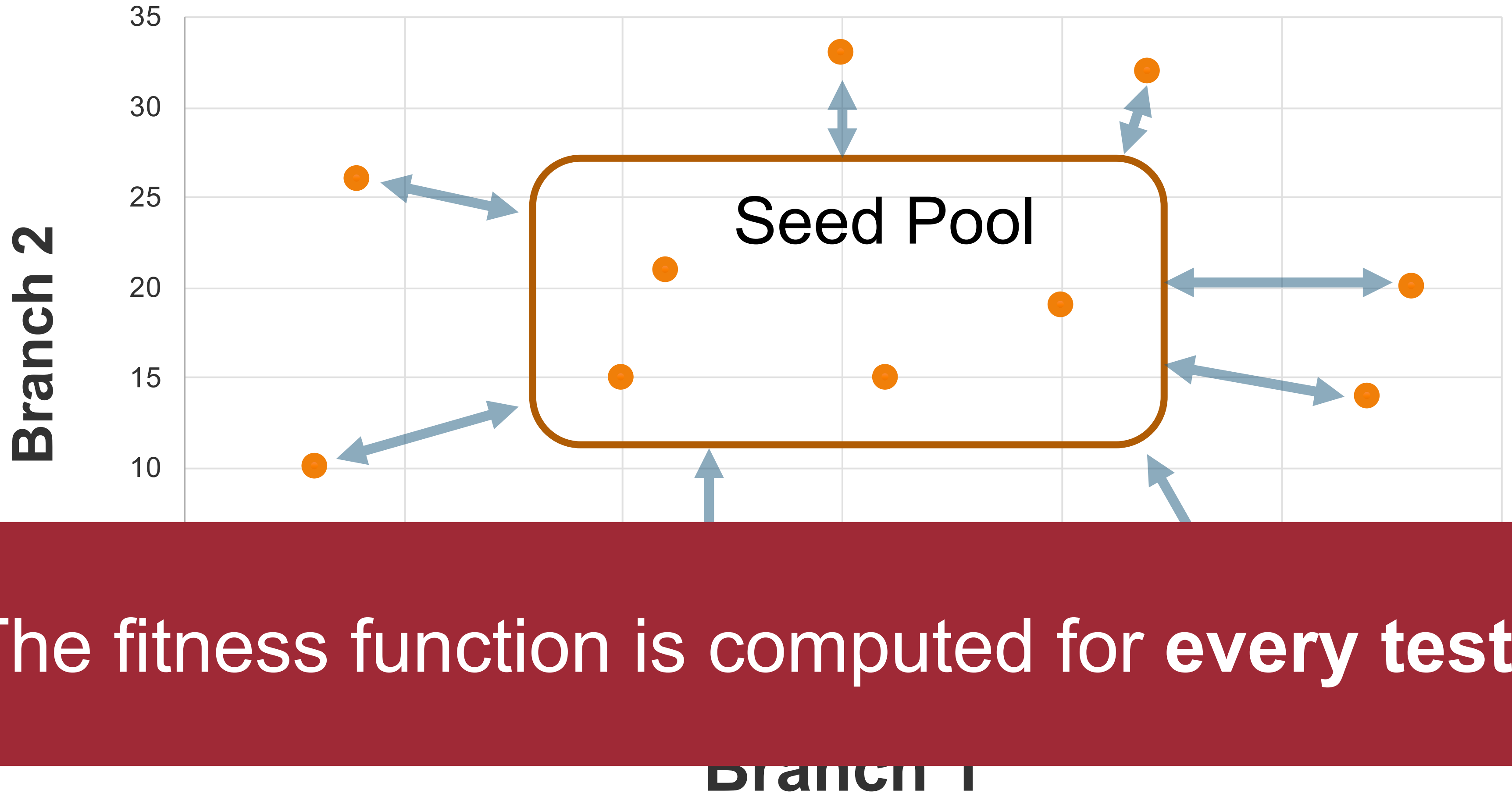
? Point-to-Pool ?

Distance-based Fitness Function



Point-to-Pool = Minimum Point-to-Point

Cost Sensitivity

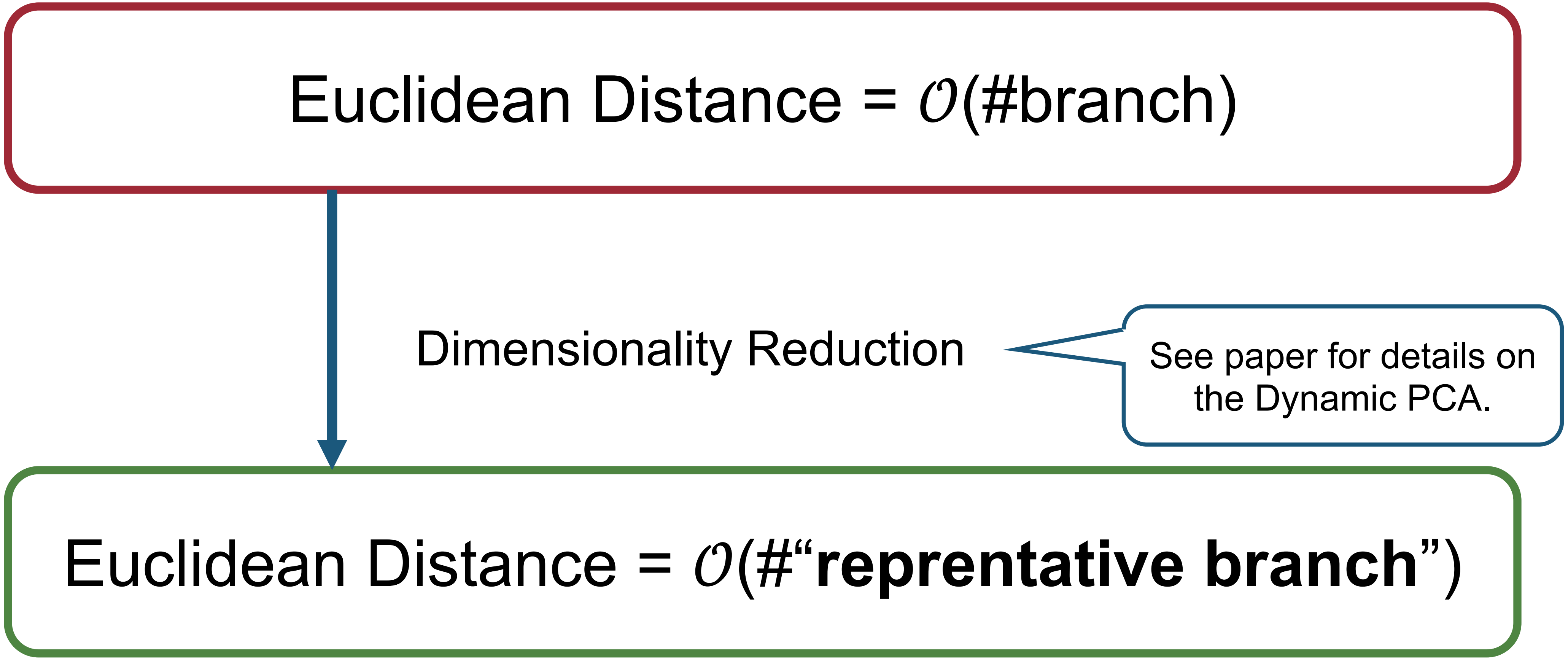


The fitness function is computed for **every test case.**

Problem: Slow Computation

Euclidean Distance = $\mathcal{O}$(#branch)

Cost Reduction



Euclidean Distance = $\mathcal{O}$(#branch)

Dimensionality Reduction

See paper for details on the Dynamic PCA.

Euclidean Distance = $\mathcal{O}$(#“**reprentative branch**”)

Ankou Adaptive Fitness Function

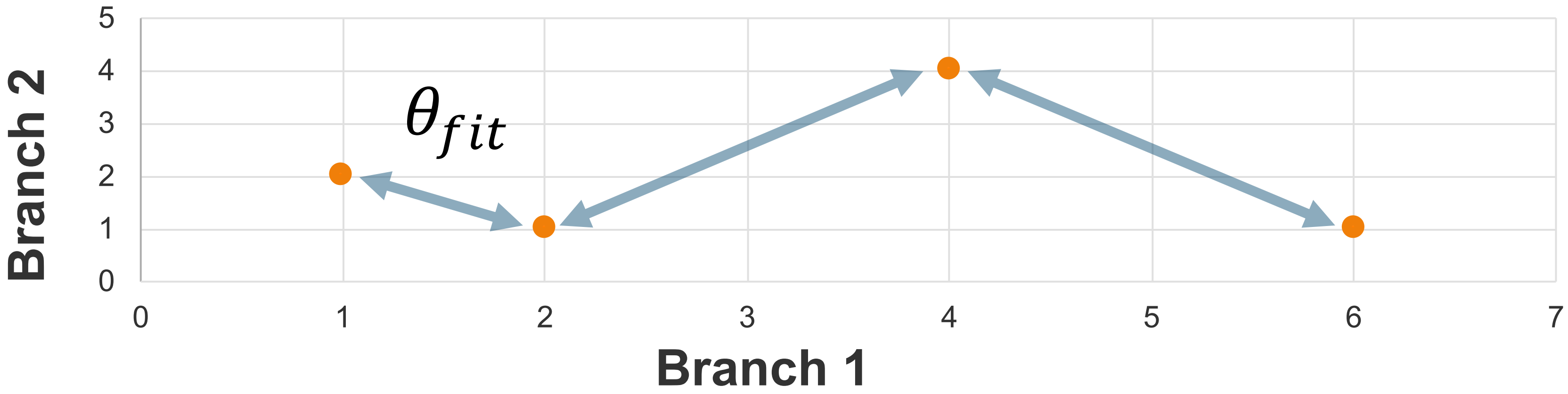
Ankou fitness function:

```
~~if(new branch):~~
if(Point-to-Pool distance ??):
     Add test to seed pool
```

Ankou Adaptive Fitness Function

Ankou fitness function:

```
~~if(new branch):~~
if(Point-to-Pool distance > $\theta_{fit}$):
    Add test to seed pool
    $\theta_{fit}$ ← Minimum inter-seed distance
```



Benchmark

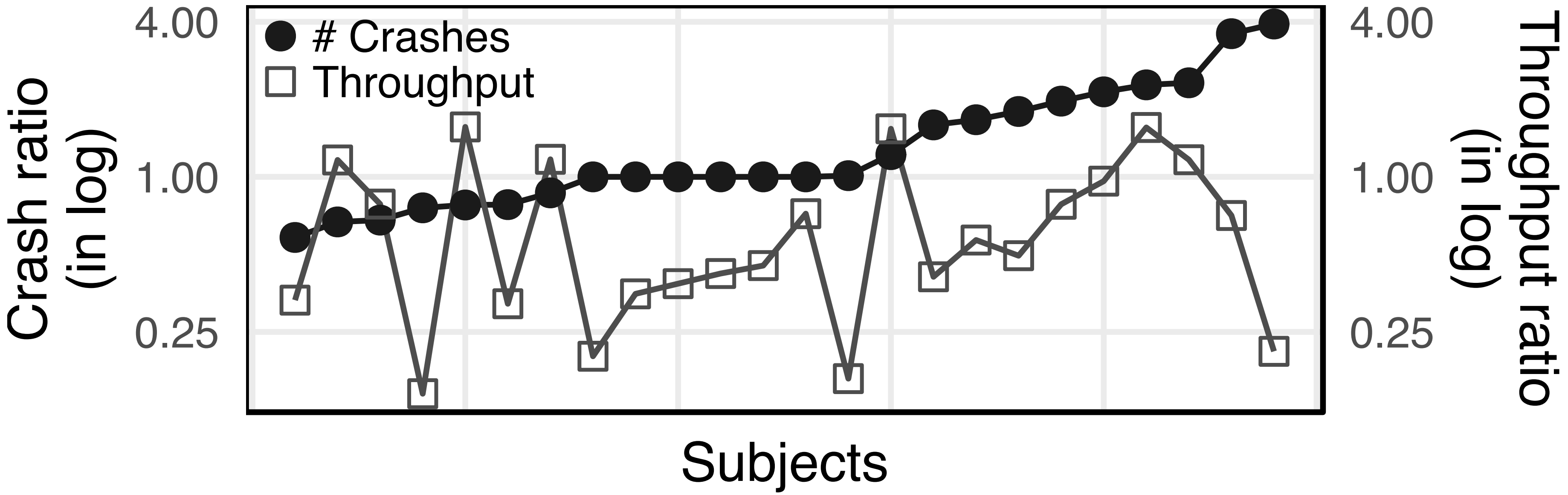
- Use 24 packages used by CollAFL[1].
- All experiments are 6x24 hours runs.
- In total: our experiments constitute 2,682 CPU days.

[1] S. Gan, C. Zhang, X. Qin, X. Tu, K. Li, Z. Pei, and Z. Chen, "CollAFL: Path sensitive fuzzing," in *Proceedings of the IEEE Symposium on Security and Privacy*, 2018, pp. 660–677.

Q: Is the New Fitness Function Effective?

Ankou with and without Distance-based

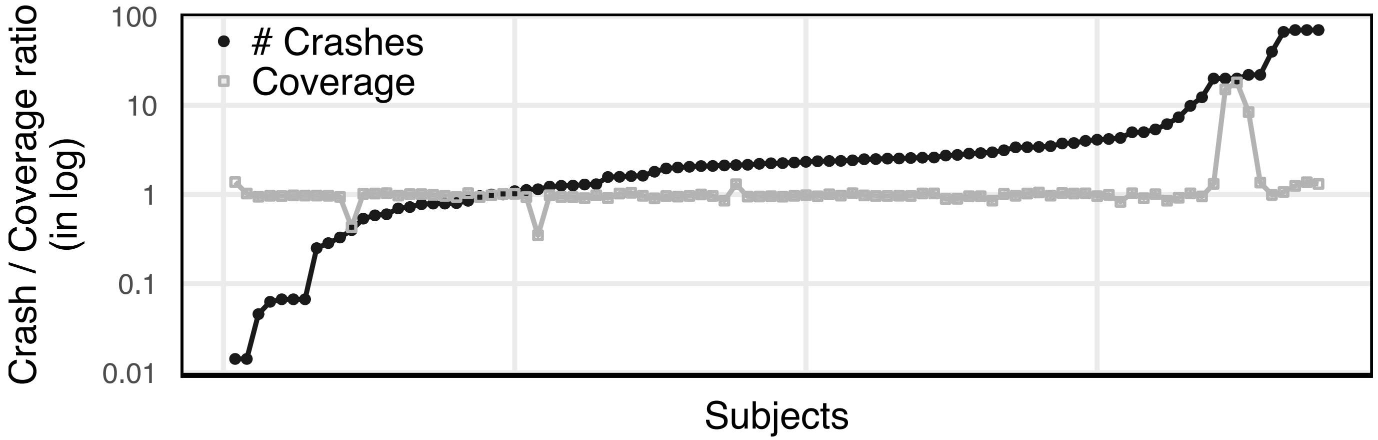
Distance-based finds 44% more crashes.

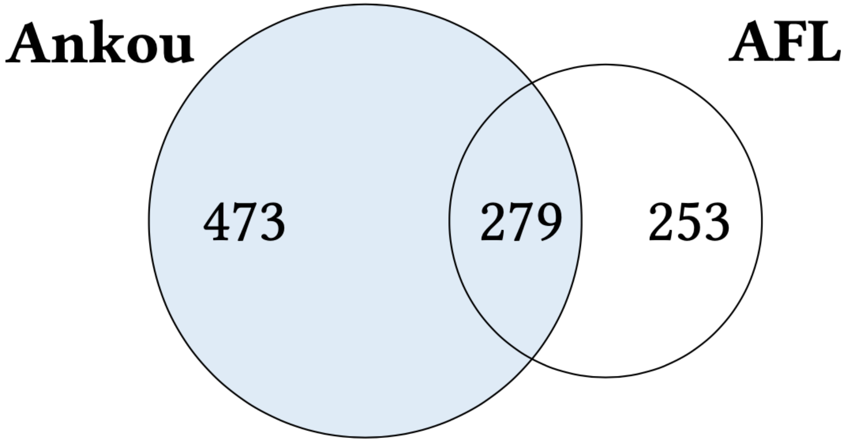


Q: How does Ankou compare to other grey-box fuzzers?

Ankou vs. AFL

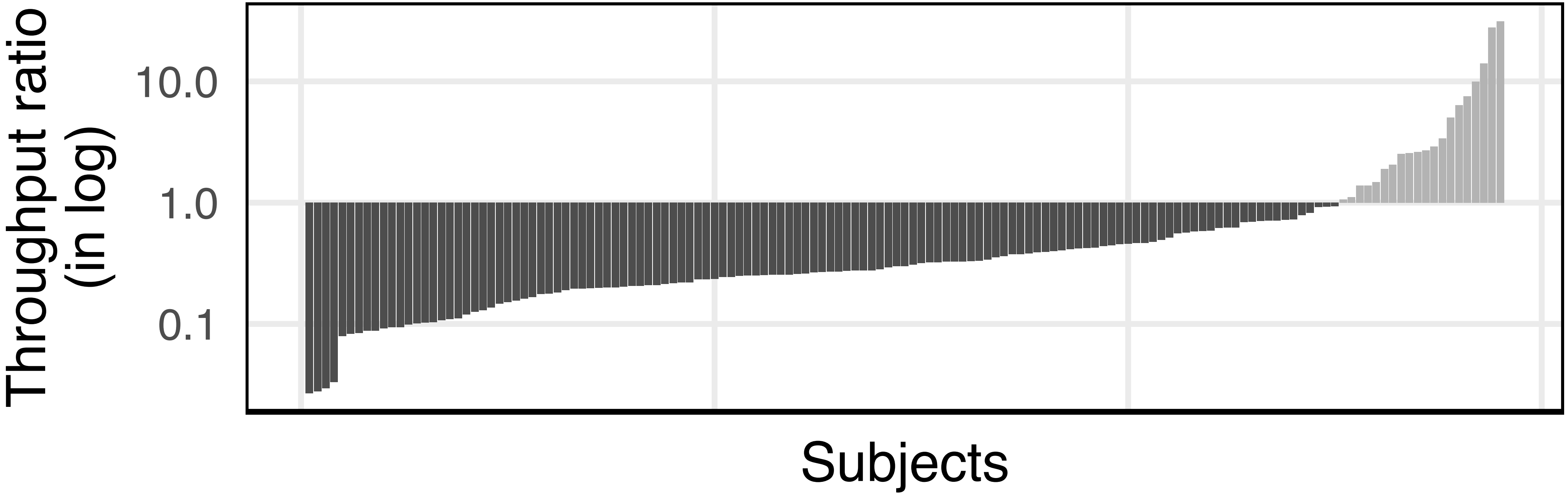
Ankou finds 41% more unique crashes.





Ankou vs. AFL: Speed

Ankou is 35% slower than AFL.



Conclusion

1. Coverage-based fuzzers ignore **combinations** of branches.
2. Ankou **distance-based** fitness function **quantify** combinatorial difference while being **fast** and **adaptive** to programs.
3. While being 35% slower than AFL, Ankou finds 41% more crashes.

Question?