Adaptive Bug Isolation

Piramanayagam Arumuga Nainar and Ben Liblit, ICSE 2010

Summary by Prof. Thomas LaToza

SWE 795, Spring 2017

Software Engineering Environments

Adaptive Bug Isolation

- Motivation:
  - Monitoring runtime behavior helps differentiate buggy runs from successful runs.
  - Instrumenting every branch execution on every run is expensive & impractical for deployed code.
- Key idea:
  - Work like an expert debugger
  - Sample subset of behavior, analyze cause, adaptively sample relevant program behavior

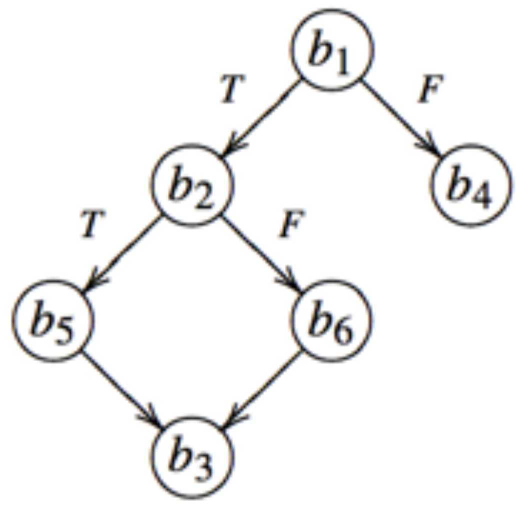
Instrumentation Approach

- Instrumentation predicates differentiate
  - taking true from false control flow branch
  - return value is negative, zero, positive
- Record only the *count* of hits rather than trace of hits
  - Saves space & (sometimes) time
- Send collected data from many program executions to server for analysis

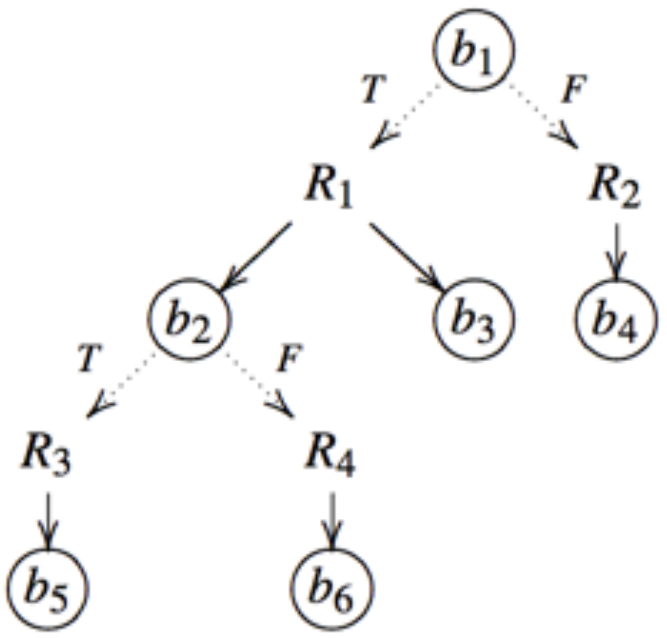
Adaptive analysis

Procedure 1 Pseudo code for Adaptive Analysis

```
1: $monitored = \emptyset$
2: $explored = \emptyset$
3: $plan = GetInitialSet()$
4: while debugging do
5:     Instrument and monitor sites in $plan$
6:     $WaitForSufficientData()$
7:     $monitored = monitored \cup plan$
8:     $best$ = branch predicate with highest $score$ in
              $monitored \setminus explored$
9:     $explored = explored \cup \{best\}$
10:    $plan = Vicinity(best) \setminus monitored$
11: end while
```

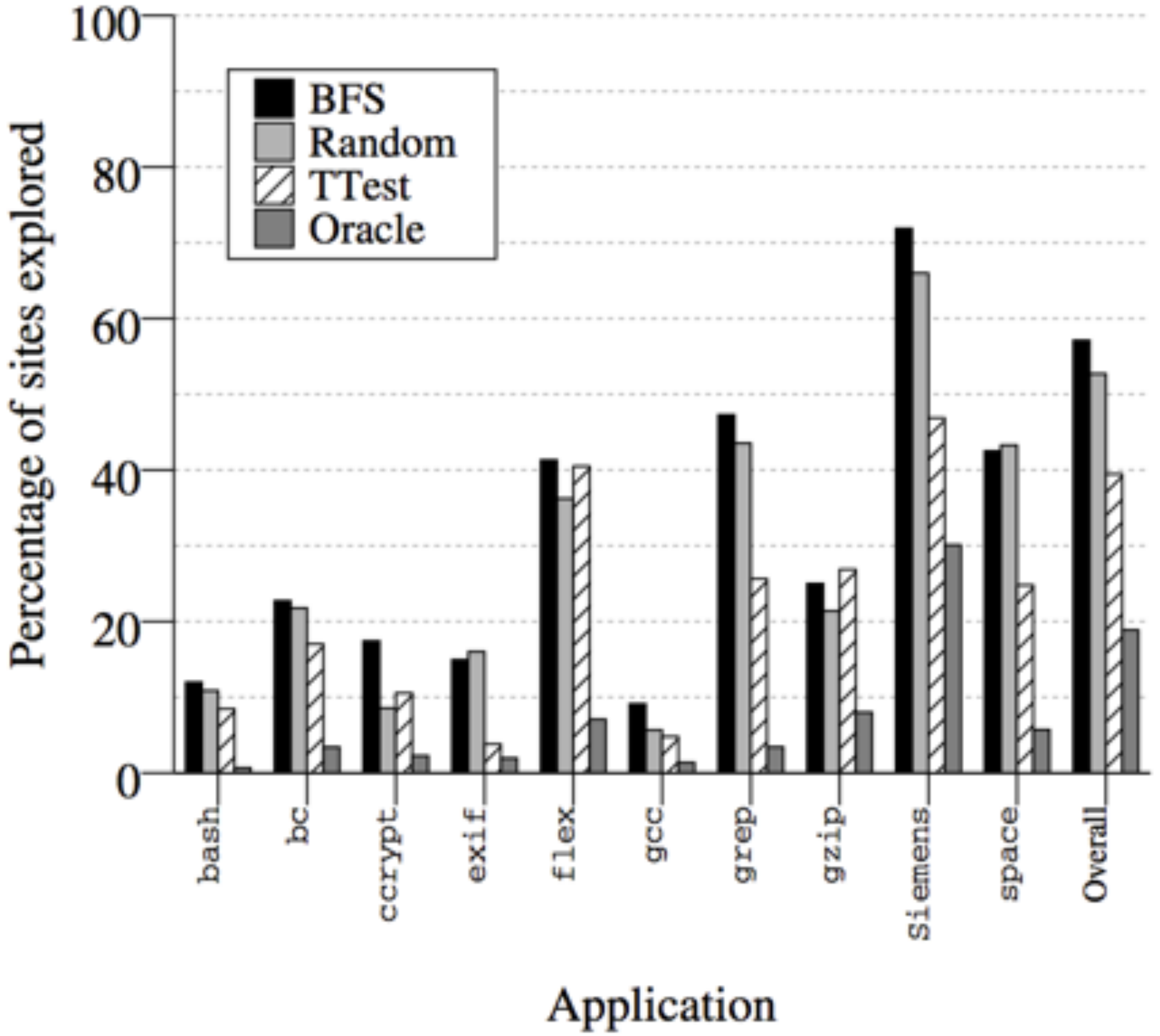


(a) Control-flow graph



(b) Control-dependence graph

Results


BFS
Random
TTest
Oracle
Percentage of sites explored
0
20
40
60
80
100
bash
bc
ccrypt
exif
flex
gcc
grep
gzip
siemens
space
Overall
Application

Questions for discussion

- Overall reactions
- Would you use this technique?
- What limitations does this have?
  - How might these limitations be overcome?
- In what ways might this technique be complimentary to other debugging techniques?