Program Analysis

Path Profiling (Part 1)

Prof. Dr. Michael Pradel

Software Lab, University of Stuttgart

Winter 2020/2021

Outline

1. **Motivation and Challenges**
2. **Ball-Larus algorithm for DAGs**
3. **Generalization and Applications**

Mostly based on this paper:

- *Efficient path profiling*, Ball and Larus, MICRO 1996

Other reading material:

- *Whole program paths* , Larus, PLDI 1999
- *HOLMES: Effective statistical debugging via efficient path profiling*, Chilimbi et al., ICSE 2009

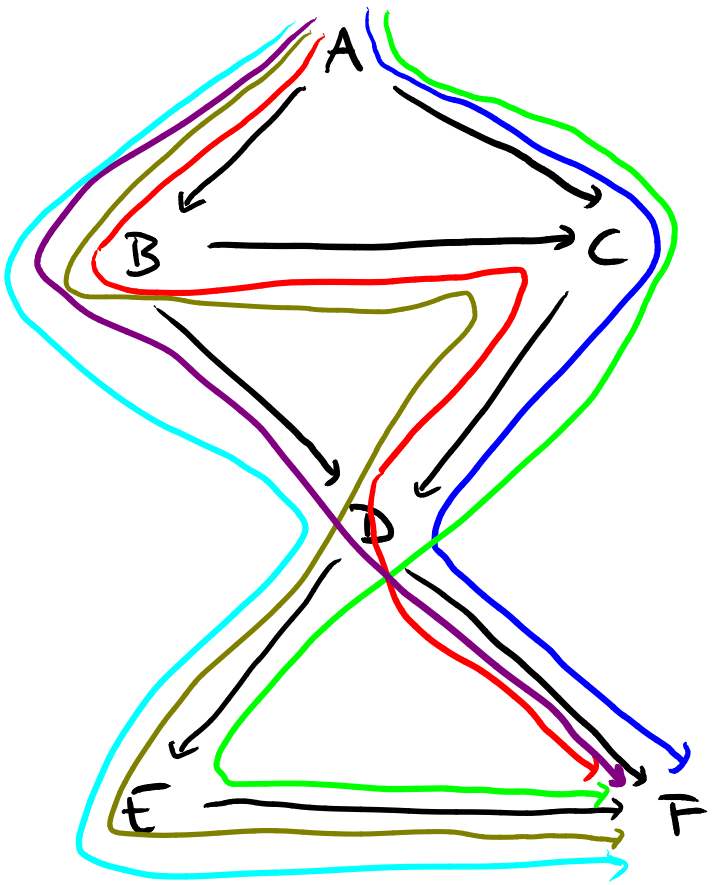
Path Profiling

- **Goal: Count how often a path through a function is executed**
- **Interesting for various applications**
  - Profile-directed compiler optimizations
  - Performance tuning: Which paths are worth optimizing?
  - Test coverage: Which paths are not yet tested?

Challenges

- **Runtime overhead**
  - Limit slowdown of program
- **Accuracy**
  - Ideally: Precise profiles (no heuristics, no approximations)
- **Infinitely many paths**
  - Cycles in control flow graph

Running Example



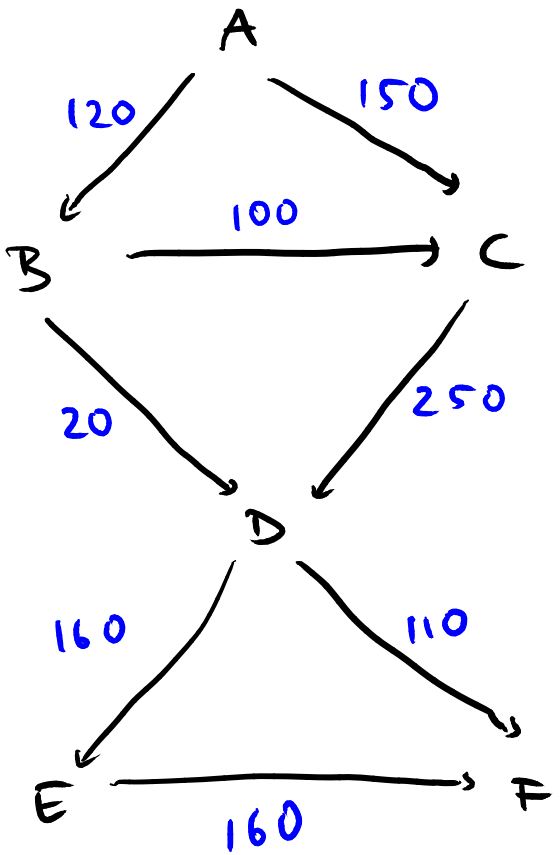
#	Path	Frequency
0	A C D F	
1	A C D E F	
2	A B C D F	
3	A B C D E F	
4	A B D F	
5	A B D E F	

Edge Profiling

Naive approach: Edge profiling

- **Instrument each branching point**
- **Count how often each CFG edge is executed**
- **Estimate most frequent path: Always follow most frequent edge**

Example: Edge Profiling



Frequency of execution

Most frequent path?

A C D E F

Really? Two possible path profiles:

Path	Profile 1	Profile 2
A C D F	90	110
A C D E F	60	40
A B C D F	0	0
A B C D E F	100	100
A B D F	20	0
A B D E F	0	20

Edge Profiling

Naive approach: Edge profiling

- **Instrument each branching point**
- **Count how often each CFG edge is executed**
- **Estimate most frequent path: Always follow most frequent edge**

Fails to uniquely identify most frequent path