Program Analysis

Path Profiling (Part 2)
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
Ball-Larus Algorithm

- Assign a **number to each path**
- Compute path number by **incrementing a counter at branching points**

**Properties of path encoding**
- Precise: A single **unique encoding for each path**
- Minimal: Instruments subset of edges with **minimal cost**
Example: Path Encoding

Instrumentation:
- state/counter: \( r \)
- array of counts: \( \text{counts} \)

<table>
<thead>
<tr>
<th>Path</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACDF</td>
<td>0</td>
</tr>
<tr>
<td>ACDEF</td>
<td>1</td>
</tr>
<tr>
<td>ABCDF</td>
<td>2</td>
</tr>
<tr>
<td>ABCDEF</td>
<td>3</td>
</tr>
<tr>
<td>ABDF</td>
<td>4</td>
</tr>
<tr>
<td>ABDEF</td>
<td>5</td>
</tr>
</tbody>
</table>
Algorithm for DAGs

Assumptions

- Control flow graph is a directed acyclic graph (DAG)
- $n$ paths (numbered 0 to $n - 1$)
- Graph has unique entry and exit nodes
- Artificial back edge from exit to entry
Assumptions

- unique entry node

Graph is acyclic

- artificial back edge

- unique exit node
Algorithm: Overview

- **Step 1: Assign integers to edges**
  - Goal: Sum along a path yields unique number for path
  - Enough to achieve "precise" goal

- **Step 2: Assign increment operations to edges**
  - Goal: Minimize additions along edges
  - Instrument subset of all edges
  - Assumes to know/estimate how frequent edges are executed
Representing Paths with Sums

- Associate with each node a value:
  \[ \text{NumPaths}(n) = \text{number of paths from } n \text{ to exit} \]

- Computing \textit{NumPaths}
  - Visit nodes in reverse topological order
  - If \( n \) is leaf node:
    \[ \text{NumPaths}(n) = 1 \]
  - Else:
    \[ \text{NumPaths}(n) = \text{sum of } \text{NumPaths} \text{ of destination of outgoing edges} \]
Representing Paths with Sums (2)

For each node in reverse topological order:

- If \( n \) is leaf node:
  \[ \text{NumPaths}(n) = 1 \]

- Else:
  \[ \text{NumPaths}(n) = 0 \]

  For each edge \( n \rightarrow m \):
  
  - \( \text{Val}(n \rightarrow m) = \text{NumPaths}(n) \)
  
  - \( \text{NumPaths}(n) += \text{NumPaths}(m) \)
Example: Num Paths & Edge Values

Reverse topological order:
Successor of n visited before n

<table>
<thead>
<tr>
<th>Node</th>
<th>Num Paths (n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>4</td>
</tr>
<tr>
<td>A</td>
<td>6</td>
</tr>
</tbody>
</table>

Values of edges
Quiz: Values for Edges

Values for edges

<table>
<thead>
<tr>
<th>n</th>
<th>NumPaths(n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Hint: \( \sum 5 \text{ NumPaths}(n) = 12 \)
Algorithm: Overview

- **Step 1:** Assign integers to edges
  - Goal: Sum along a path yields unique number for path
  - Enough to achieve "precise" goal

- **Step 2:** Assign increment operations to edges
  - Goal: Minimize additions along edges
  - Instrument subset of all edges
  - Assumes to know/estimate how frequent edges are executed
Spanning Tree

- Given: Graph $G$

- Spanning tree $T$: Undirected subgraph of $G$ that is a tree and that contains all nodes of $G$

- Chord edges: Edges in $G$ but not in $T$
Example: Spanning Tree

\[ G: \]

1. Which of these is a spanning tree of \( G \)?

2. \( \boxed{\text{Row 2}} \)

3. \( \boxed{\text{Row 3}} \)

4. \( \boxed{\text{Row 4}} \)

5. \( \boxed{\text{Row 5}} \)
Increments for Edges

Goal: Increment sum at subset of edges

- Choose **spanning tree with maximum edge cost**
  - Cost of individual edges is assumed to be known

- Compute **increments at the chords of the spanning tree**
  - Based on existing event counting algorithm
Example: Increments for Edges

- Edge cost
  - Most expensive spanning tree

- Non-minimal increments
  - Chordal edges

- Minimal increments
  - Path encoding
Instrumentation

- **Basic idea**
  - Initialize sum at entry: \( r = 0 \)
  - Increment at edges: \( r += \ldots \)
  - At exit, increment counter for path: \( \text{count}[r]++ \)

- **Optimization**
  - Initialize with incremented value, if first chord on path: \( r = \ldots \)
  - Increment sum and counter for path, if last chord on path: \( \text{count}[r+\ldots]++ \)
Regenerating the Path

Knowing the sum $r$, how to determine the path?

- Use edge values from step 1 ("non-minimal increments")
- Start at entry with $R = r$
- At branches, use edge with largest value $v$ that is smaller than or equal to $R$ and set $R = v$
Example: Regenerate the Path

A

B

C

D

E

F

r = 4:
ABDF

r = 1:
ACDEF

R = \lambda \geq 0

R = \lambda \geq 0