Link Reversal Algorithms

Jennifer L. Welch
Texas A&M University
What is Link Reversal?

- Distributed algorithm design technique
- Used in solutions for a variety of problems
  - routing, leader election, mutual exclusion, scheduling, resource allocation,…
- Model problem as a directed graph and reverse the direction of links appropriately
- Use **local** knowledge to decide which links to reverse
Outline

- Routing in a Graph: Correctness
- Routing in a Graph: Complexity
- Routing and Leader Election in a Distributed System
- Mutual Exclusion in a Distributed System
- Distributed Queueing
- Scheduling in a Graph
- Resource Allocation in a Distributed System
Routing

[Gafni & Bertsekas 81]

- Undirected connected graph represents communication topology of a system
- Unique destination node
- Assign virtual directions to the graph edges (links) s.t.
  - if nodes forward messages over the links, they reach the destination
- Directed version of the graph (orientation) must
  - be acyclic
  - have destination as only sink

Thus every node has path to destination.
Routing Example
Mending Routes

□ What happens if some edges go away?
  □ Might need to change the virtual directions on some remaining edges (reverse some links)

□ More generally, starting with an arbitrary directed graph, each node should decide independently which of its incident links to reverse
Mending Routes Example
Sinks

- A vertex with no outgoing links is a sink.
- The property of being a sink can be detected locally.
- A sink can then reverse some incident links
- Basis of several algorithms...
Full Reversal Routing Algorithm

- Input: directed graph $G$ with destination vertex $D$
- Let $S(G)$ be set of sinks in $G$ other than $D$
- while $S(G)$ is nonempty do
  - reverse every link incident on a vertex in $S(G)$
  - $G$ now refers to resulting directed graph
Full Reversal (FR) Routing Example
Why Does FR Terminate?

- Suppose it does not.
- Let W be vertices that take infinitely many steps.
- Let X be vertices that take finitely many steps; includes D.
- Consider neighboring nodes w in W, x in X.
- Consider first step by w after last step by x: link is $w \rightarrow x$ and stays that way forever.
- Then w cannot take any more steps, contradiction.
Why is FR Correct?

- Assume input graph is acyclic.
- Acyclicity is preserved at each iteration:
  - Any new cycle introduced must include a vertex that just did a reversal, but such a vertex is now a source (has no incoming links)
- When FR terminates, no vertex, except possibly D, is a sink.
- A DAG must have at least one sink:
  - if no sink, then a cycle can be constructed
- Thus output graph is acyclic and D is the unique sink.
Pair Algorithm

- Can implement FR by having each vertex v keep an ordered pair (c,v), the *height* (or *vertex label*) of vertex v
  - c is an integer counter that can be incremented
  - v is the id of vertex v
- View link between v and u as being directed from vertex with larger height to vertex with smaller height (compare pairs lexicographically)
- If v is a sink then v sets c to be 1 larger than maximum counter of all v’s neighbors
Pair Algorithm Example

Diagram:

- Node 0
- Node 1
- Node 2
- Node 3

Connections:
- (0,1) from Node 0 to Node 1
- (0,2) from Node 0 to Node 2
- (2,3) from Node 2 to Node 3
- (1,0) from Node 1 to Node 0
- (2,1) from Node 2 to Node 1
Pair Algorithm Example

```
(0,1)  (1,0)  (0,2)  (2,3)
(2,1)  0     (3,2)
(1,0)  (2,1)
```

1 2 3
Pair Algorithm Example

- Pair Algorithm Example

0

1

2

3

(0,1) (0,2) (2,3)

(2,1) (3,2)
Partial Reversal Routing Algorithm

- Try to avoid repeated reversals of the same link.
- Vertices keep track of which incident links have been reversed recently.
- Link \((u,v)\) is reversed by \(v\) iff the link has not been reversed by \(u\) since the last iteration in which \(v\) took a step.
Partial Reversal (PR) Routing Example
Why is PR Correct?

- Termination can be proved similarly as for FR: difference is that it might take two steps by $w$ after last step by $x$ until link is $w \rightarrow x$.

- Preservation of acyclicity is more involved, deferred to later.
Triple Algorithm

- Can implement PR by having each vertex v keep an ordered triple \((a,b,v)\), the height (or vertex label) of vertex v
  - a and b are integer counters
  - v is the id of node v
- View link between v and u as being directed from vertex with larger height to vertex with smaller height (compare pairs lexicographically)
- If v is a sink then v
  - sets a to be 1 greater than smallest a of all its neighbors
  - sets b to be 1 less than smallest b of all its neighbors with new value of a (if none, then leave b alone)
Triple Algorithm Example
Triple Algorithm Example
Triple Algorithm Example

Parasol

0

(0, 0, 2) (0, 1, 0) (0, 2, 3)

1
(0, 0, 1) (1, 0, 1)

2
(0, 0, 2) (1, -1, 2)

3
(0, 2, 3)
General Vertex Label Algorithm

- Generalization of Pair and Triple algorithms
- Assign a label to each vertex s.t.
  - labels are from a totally ordered, countably infinite set
  - new label for a sink depends only on old labels for the sink and its neighbors
  - sequence of labels taken on by a vertex increases without bound
- Can prove termination and acyclicity preservation, and thus correctness.
Binary Link Labels Routing
[Charron-Bost et al. 2009]

- Alternate way to implement and generalize FR and PR
- Instead of unbounded vertex labels, apply binary link labels to input DAG
  - Link directions are independent of labels (in contrast to algorithms using vertex labels)

- Algorithm for a sink:
  - If at least one incident link is labeled 0, then reverse all incident links labeled 0 and flip labels on all incident links
  - If no incident link is labeled 0, then reverse all incident links but change no labels
Binary Link Labels Example
Binary Link Labels Example
Binary Link Labels Example
Why is BLL Correct?

- Termination can be proved very similarly to termination for PR.
- What about acyclicity preservation?
  Depends on initial labeling:
Conditions on Initial Labeling

☐ All labels are the same
  ☐ all 1’s => Full Reversal
  ☐ all 0’s => Partial Reversal

☐ Every vertex has all incoming links labeled the same ("uniform" labeling)

☐ Both of the above are special cases of a more general condition that is necessary and sufficient for preserving acyclicity
References