

## 1 Introduction

Today's lecture was a continuation of the previous class, where we discussed link reversal algorithms. In the previous class, we discussed the *full reversal* technique. In this class, we took a brief look at the *partial reversal* technique.

## 2 Height of a Node

In the previous class, we discussed the concept of every node  $u$  in the graph  $G$  being associated with a certain *height*. For a quick recap, refer to the scribe notes: Part 18, Pages 2 and 3.

Reversals are implemented using heights. A reversal algorithm assigns a height to every node  $u$  in the network. The link between adjacent nodes is directed from the node of greater height to the node of lesser height. Recall that a node  $v$  is a sink, if all of  $vs$  adjacent links are pointing in, and  $v$  is not the destination. A sink performs a reversal by increasing its height by a suitable amount. This will reverse the direction of some or all of its incident links. The authors consider deterministic link reversal algorithms, in which a sink node increases its height according to some deterministic function of the heights of the adjacent nodes. The link reversal algorithms by Gafni and Bertsekas are deterministic. We say that a node is *bad* if there is no route from the node to the destination. Any other node, including the destination, is *good*. Note that a *bad* node is not necessarily a sink.

### 2.1 Reversal by *Height* Manipulation

There is a pair  $(\alpha_i, i)$  associated with each node where  $i$  is the unique ID of the node and  $\alpha_i$  is an integer. The pairs can then be totally ordered lexicographically (e.g.  $(\alpha_i, i) > (\alpha_j, j)$  if  $\alpha_i > \alpha_j$ , or if  $\alpha_i = \alpha_j$ , and  $i > j$ ). Let us refer to the value associated with each node  $i$  as its *height* and denote it  $h_i$ . Now, assume that we assign an initial height to each node in the destination-disoriented DAG such that node  $i$  is upstream from node  $j$  if and only if  $h_i > h_j$ . Then it is clear that node  $i$  has no downstream links when, measured by its height, it is a local minimum with respect to its neighbors,  $h_i < h_j$  for all  $j \in N_i$ . Recall that  $N_i$  is the neighbor list of node  $i$ . To achieve the desired behavior in the full reversal method, node  $i$  must select a new height such that it becomes a local maximum with respect to its neighbors,  $h_i > h_j$ , for all  $j \in N_i$ . Node  $i$  simply selects a new value  $\alpha_i = \max \alpha_j \mid j \in N_i + 1$  and broadcasts the value to all of its neighbors. The partial reversal method can neither be viewed conceptually nor explained as easily. Again, a node selects a new height only when it is a local minimum,

but it does not always become a local maximum. To reverse only some of its links (i.e. partial reversal), a node selects a new height which is higher than its own previous height and the height of some of its neighbors, but not higher than all of its neighbors.

### 3 Partial Reversal

As mentioned earlier, full reversal is when a sink node  $u$  reverses *all* the edges that are incident on it. In the worst case, this reversal algorithm performs the best. However in the average case, full reversal doesn't perform very well. Partial reversal is the method by which a sink node  $u$  reverses a subset of edges that are incident on it. Note that if the edges for reversal are randomly chosen, then oscillation could occur, rendering this protocol inefficient / useless. Therefore, the edges have to be carefully chosen for reversal.

#### 3.1 Abstract Partial Reversal

Each node  $u \neq d$  maintains a list  $L_u$  of the neighbors  $v$  that have fired and reversed edge  $(u, v)$  since the last firing of node  $u$ . When a node  $u$  detects that it has become a sink, it does the following:

- Recall that  $N_u$  is the neighbor list of node  $u$
- If  $|N_u - L_u| \neq 0$ , then it reverses the links to all nodes  $v$  such that  $v \in (N_u - L_u)$
- If  $N_u = L_u$ , then reverse all links

#### 3.2 Concrete Partial Reversal

The height of a node  $u$  is a triple  $(\alpha_u, \beta_u, u)$ . Consider what a node  $u$  does when it fires:

- It sets  $\alpha_u$  to be equal to  $\min_{v \in N_u} \alpha_v + 1$
- After the previous step, it could be that  $\alpha_u = \alpha_{\hat{v}}$  for some neighbor  $\hat{v} \neq v$ . Set  $\beta_u$  to be  $\min(\beta_v) - 1$ , for all  $v \in N_u$  such that  $\alpha_u = \alpha_v$ .
- By the previous step, we ensure that node  $u$  has its *height* triple different from all nodes  $v$  in its neighbor set  $L_u$ .

## 4 Conclusion

We have studied a lot about link reversal algorithms. We have looked at full reversal as well as partial reversal techniques. These algorithms have great practical application in mobile ad-hoc networks, such as sensor networks. Please refer to scribe notes 18 for a more detailed and basic explanation of link reversal techniques. The End.