1 Introduction

The previous lecture introduced the concept of Link Reversal Algorithms. This lecture considers the first such algorithm, which appeared in: Distributed algorithms for generating loop-free routes in networks with frequently changing topology, by Gafni and Bertsekas, IEEE Transactions on Communication, 1981. This lecture proves correctness of the Full Reversal algorithm in that paper, and introduces the Partial Reversal algorithm, which will be discussed later.

1.1 Applications of link reversal algorithms

Applications of link reversal algorithms include: leader election, mutual exclusion, resource allocation problems. All these problems involve producing asymmetry among processors. Breaking symmetry is often difficult. In link reversal, we are not just breaking symmetry, but also coming up with a leader, or centralized processor for the whole network.

The original application of the Gafni-Bertsekas algorithm was for routing. They asked how to route all messages to a particular destination node.

1.2 High-level problem statement

We take as input an undirected graph $G$, with a distinguished node $D$. We then give an orientation to each edge of the graph, producing directed graph $\overrightarrow{G}$. Each processor, computing locally, can change the direction of the edges incident to it. The goal is to output an acyclic graph such that $D$ is the only sink of the graph; in other words, the graph has no cycles, and $D$ is the only node with no outgoing edges.

2 Preliminaries

2.1 Definitions

Definition 1 $\overrightarrow{G} = (V, \overrightarrow{E})$ is an orientation of the undirected graph $G$, if $V$ is the vertex set of $G$, and for each edge $\{u, v\} \in E$, there is exactly one of $(u, v)$ or $(v, u)$ in $\overrightarrow{E}$; and further that if $(u, v) \in \overrightarrow{E}$ then $\{u, v\} \in E$. 
Definition 2 A chain in $\overrightarrow{G}$ is a sequence $\langle v_1 v_2 \ldots v_k \rangle$ where, for all $1 \leq i \leq k - 1$, either $(v_i, v_{i+1}) \in \overrightarrow{E}$, or $(v_{i+1}, v_i) \in \overrightarrow{E}$.

So a chain does not take the direction of edges into account.

Definition 3 The pair $(v_i, v_{i+1})$ is right-way if $(v_i, v_{i+1}) \in \overrightarrow{E}$; it is wrong-way if $(v_{i+1}, v_i) \in \overrightarrow{E}$.

Definition 4 A path is a chain with no wrong-way links. A circuit is a chain that starts and ends at the same vertex. A cycle is a circuit with no wrong-way links.

Definition 5 Graph orientation $\overrightarrow{G}$ is destination-oriented, or $D$-oriented, if there is a path from every vertex to the distinguished vertex $D$.

2.2 Facts about directed graphs

Note that if a directed graph is acyclic, then all the paths (routes) in the graph will be loop-free. Therefore, our general objective, the production of an acyclic graph with a single sink, can be rephrased as: produce a loop-free graph with a sink. Then, if $D$ is a sink, every path from a vertex $v \in D$ will terminate at $D$.

Lemma 1 If $\overrightarrow{G}$ is acyclic, then every vertex in $V$ has a path to some sink.

Proof: Let $v$ be any vertex. Suppose for contradiction it doesn’t have a path to a sink. Then start at $v$ and traverse edges from $v$. Either the traversal will end somewhere that has no outgoing edges (which is a contradiction); or the traversal is an infinite path. But since $V$ is finite, vertices will repeat along that infinite path, so there is a cycle in the graph (which is also a contradiction). Hence, $v$ must have a path to a sink.

Note that this implies that every acyclic $\overrightarrow{G}$ contains at least one sink node. This comes in handy to prove the following lemma.

Lemma 2 Let $\overrightarrow{G}$ be acyclic. Then $\overrightarrow{G}$ is $D$-oriented iff $D$ is the unique sink of $\overrightarrow{G}$.

Proof: Suppose $\overrightarrow{G}$ is $D$-oriented. Then, by definition, every vertex has a path to $D$. So every vertex other than $D$ is not a sink. By Lemma 1, some vertex is a sink. Therefore, $D$ is a sink, and is the unique sink.

Now suppose $D$ is a unique sink. Then by Lemma 1, every vertex has a path to $D$, so $\overrightarrow{G}$ is $D$-oriented.
Algorithm 1 Generic Link Reversal Algorithm

1: while $\overrightarrow{G}$ has a sink other than $D$ do
2: choose subset $S$ of sinks in $\overrightarrow{G}$ (such that $D \notin S$)
3: reverse the direction of a subset of links incident on the vertices in $S$

3 Generic link reversal algorithm

We now present a “generic” link reversal algorithm, which will be the format of all link reversal algorithms discussed in this course, including the Full Reversal algorithm we will look at today. The generic algorithm appears formally as Algorithm 1.

While this is written as though it were a centralized algorithm, in practice each sink decides locally, to reverse a subset of the edges incident to it. If the subset of edges is nonempty, that vertex is a member of $S$. Part of the reason we only allow sinks to reverse edge directions is to help ensure termination. If both $u$ and $v$ could reverse $(u, v)$, the algorithm might fall into an infinite execution where the direction of the edge kept flipping back and forth. So we require that only destination nodes can flip an incident edge.

4 Full reversal algorithm of Gafni Bertsekas 1981

Algorithm 2 Full Reversal Algorithm (FR)

Input: $\overrightarrow{G} = (V, \overrightarrow{E})$ with distinguished $D \in V$

1: while $\overrightarrow{G}$ has a sink other than $D$ do
2: choose non-empty subset $S$ of sinks in $\overrightarrow{G}$ where $D \notin S$
3: for all $v \in S$, reverse direction of all links incident on $v$

The Full Reversal Algorithm of Gafni and Bertsekas appears as Algorithm 2. We will now show that it terminates, and that it is correct.

Lemma 3 The Full Reversal algorithm terminates.

Proof: Suppose not. Then there is some execution of Full Reversal that does not terminate. Let $W$ be the set of vertices that change their links infinitely often. Since $V$ is finite, $W$ is nonempty. Note that $V - W$ is nonempty, since it contains at least $D$.

Since $G$ is connected, there is an edge $(u, v) \in \overrightarrow{E}$ where $u \in W$ and $v \in V - W$. Let $t$ be the iteration of the WHILE loop in which $v$ takes its last step. Since $u \in W$, there is some iteration $t' > t$ at which $u$ takes a step, reversing the direction of $(u, v)$. Since $v$ never takes another step, it will never be a sink again, so $(v, u)$ will never reverse direction again, contradicting the fact that $u$ is a sink infinitely often. Therefore, Full Reversal must terminate.

Lemma 4 Full Reversal maintains acyclicity.
Proof: Suppose not. Because of the structure of the algorithm, a loop can only be created when some sink $v$ reverses the direction of all incident edges. Suppose $v$ is the first such sink to do this; that is to say, $\overrightarrow{G}$ is acyclic until $v$ fires. Since only edges incident to $v$ change direction, any cycle created must go through $v$. However, since all edges of the sink $v$ are flipped (“full reversal”) this means that all edges from $v$ are now outgoing edges, so there is no cycle that goes through $v$, contradicting our assumption that a cycle was created. Hence, if $\overrightarrow{G}$ starts acyclic, its acyclicity will be maintained.

The lemmas in this section do not require synchrony. For example, in Lemma 4, even if all nodes behave asynchronously, and do not reverse all their own edges in an atomic step, the proof still goes through. This is because only (a subset of) sinks can take steps in the algorithm, so if the reversal of edge $e$ incident to $v$ created a loop, no vertices in the loop would be sinks, so none of them could take steps until $v$ reversed another edge that was part of the loop (possibly creating a new sink) but eliminating the ability to have a loop that included $v$.

5 Conclusion

In the next lecture, we will look at Partial Reversal algorithms, where sinks can choose to reverse only some of their incident edges, not all of them.