IEOR 266 — NETWORK FLOWS AND GRAPHS — October 14, 2008

Lecture 14

Last Class
1. D.B. Johnson's Algorithm
2. Introduction to Max-Flow
3. Residual graph definition

Today
1. Max flow/ Min cut
2. Augmenting path algorithm (Ford-Fulkerson)
3. Primal and Dual of max flow problem

1 Maximum Flow Problem (continued)

1.1 Introduction (rephrased)

See the handout titled *Travelers Example: Seat Availability.*
Question: What is the maximum number of people from SF to NYC? **Answer:** 9
–Total out of SF = Total into NYC
–Balance constraints for all other nodes: outflow - inflow = 0
–Start with the flow zero: feasible, satisfy all the constraints and is a lower bound

Definitions:
Assume the lower bounds are zero. A is the arc set of the graph and u_{ij} is the capacity of arc (i, j).
–A *feasible flow* f is a flow (i.e. obeys the flow balance constraints) that satisfies $0 \leq f_{ij} \leq u_{ij}\ \forall (i, j) \in A$
–If $f_{ij} = u_{ij}$, then the arc is *saturated*
–A *cut* is a partition $(S, \bar{S})$, $S \subseteq V$, $s \in S$, $\bar{S} = V \setminus S, t \in \bar{S}$
–*Cut capacity:* $U(S, \bar{S}) = \sum_{i \in S} \sum_{j \in \bar{S}} u_{ij}$

1.2 Certificate of optimality of flow

Statement: For a flow f to be feasible, $|f| \leq U(S, \bar{S})$ for any $(S, \bar{S})$ cut, where $|f|$ is the total amount of flow out of the source (equivalently into the sink).

Intuition: Any flow from s to t has to go through the cut arcs. The bottleneck is the cut with minimum capacity.

The following theorem, which we will establish algorithmically, can be viewed as a special case of the strong duality theorem in linear programming.

Theorem 1 (Max-Flow Min-Cut)**.** *The value of a maximum flow is equal to the capacity of a cut with minimum capacity among all cuts.*

The next sections introduce the terminology needed to prove the Max-flow/Min-cut duality and to give us an algorithm to find the max flow of any given graph.

1.3 Residual graph

The residual graph, $G_f = (V, A_f)$, with respect to a flow f, has the following arcs:

- forward arcs: $(i, j) \in A_f$ if $(i, j) \in A$ and $f_{ij} < u_{ij}$. The residual capacity is $u^f_{ij} = u_{ij} - f_{ij}$. The residual capacity on the forward arc tells us how much we can increase flow on the original arc $(i, j) \in A$.
- reverse arcs: $(j, i) \in A_f$ if $(i, j) \in A$ and $f_{ij} > 0$. The residual capacity is $u^f_{ji} = f_{ij}$. The residual capacity on the reverse arc tells us how much we can decrease flow on the original arc $(i, j) \in A$.

Generally, in the presence of lower bounds, $\ell_{ij} \leq f_{ij} \leq u_{ij}$, $(i, j) \in A$, the definition of forward arc remains, whereas for reverse arc, $(j, i) \in A_f$ if $(i, j) \in A$ and $f_{ij} > \ell_{ij}$. The residual capacity is $u^f_{ji} = f_{ij} - \ell_{ij}$.

Intuitively, residual graph tells us how much **additional** flow can be sent through the original graph with respect to the given flow. This brings us to the notion of an augmenting path.

1.4 Augmenting path

An **augmenting path** is a path from s to t in the residual graph. The *capacity of an augmenting path* is the minimum residual capacity of the arcs on the path – the bottleneck capacity.

If we can find an augmenting path with capacity δ in the residual graph, we can increase the flow in the original graph by adding δ units of flow on the arcs in the original graph which correspond to forward arcs in the augmenting path and subtracting δ units of flow on the arcs in the original graph which correspond to reverse arcs in the augmenting path.

Note that this operation does not violate the **capacity constraints** since δ is the smallest arc capacity in the augmenting path in the residual graph, which means that we can always add δ units of flow on the arcs in the original graph which correspond to forward arcs in the augmenting path and subtract δ units of flow on the arcs in the original graph which correspond to reverse arcs in the augmenting path without violating capacity constraints.

The **flow balance** constraints are not violated either, since for every node on the augmenting path in the original graph we either increment the flow by δ on one of the incoming arcs and increment the flow by δ on one of the outgoing arcs (this is the case when the incremental flow in the residual graph is along forward arcs) or we increment the flow by δ on one of the incoming arcs and decrease the flow by δ on some other incoming arc (this is the case when the incremental flow in the residual graph comes into the node through the forward arc and leaves the node through the reverse arc) or we decrease the flow on one of the incoming arcs and one of the outgoing arcs in the original graph (which corresponds to sending flow along the reverse arcs in the residual graph).

1.4.1 Ford-Fulkerson algorithm

The above discussion can be summarized in the following algorithm for finding the maximum flow on a give graph. This algorithm is is also known as the **augmenting path algorithm**. Below is a pseudocode-like description.

Pseudocode:
f: flow;
G_f: the residual graph with respect to the flow;

P_{st}: a path from s to t;
U_{ij}: capacity of arc from i to j.

```
Initialize $f = 0$
If $\exists P_{st} \in G_f$ do
    find $\delta = \min_{(i,j)\in P_{st}} U_{ij}$
    $f_{ij} = f_{ij} + \delta \quad \forall (i,j) \in P_{st}$
else stop $f$ is max flow.
```

Detailed description:

1. Start with a feasible flow (usually $f_{ij} = 0\ \forall (i,j) \in A$).
2. Construct the residual graph G_f with respect to the flow.
3. Search for augmenting path by doing breadth-first-search from s (we consider nodes to be adjacent if there is a positive capacity arc between them in the residual graph) and seeing whether the set of s-reachable nodes (call it S) contains t.
   If S contains t then there is an augmenting path (since we get from s to t by going through a series of adjacent nodes), and we can then increment the flow along the augmenting path by the value of the smallest arc capacity of all the arcs on the augmenting path.
   We then update the residual graph (by setting the capacities of the forward arcs on the augmenting path to the difference between the current capacities of the forward arcs on the augmenting path and the value of the flow on the augmenting path and setting the capacities of the reverse arcs on the augmenting path to the sum of the current capacities and the value of the flow on the augmenting path) and go back to the beginning of step 3.
   If S does not contain t then the flow is maximum.

Given the max flow (with all the flow values), a min cut can be found in by looking at the residual network. The set S, consists of s and the all nodes that s can reach in the final residual network and the set $\bar{S}$ consists of all the other nodes. Since s can't reach any of the nodes in $\bar{S}$, it follows that any arc going from a node in S to a node in $\bar{S}$ in the original network must be saturated which implies this is a minimum cut. This cut is known as the minimal source set minimum cut. Another way to find a minimum cut is to let $\bar{S}$ be t and all the nodes that can reach t in the final residual network. This a *maximal source set minimum cut* which is different from the minimal source set minimum cut when the minimum cut is not unique.

While finding a minimum cut given a maximum flow can be done in linear time, $O(m)$, we have yet to discover an efficient way of finding a maximum flow given a list of the edges in a minimum cut other than simply solving the maximum flow problem from scratch. Also, we have yet to discover a way of finding a minimum cut without first finding a maximum flow. Since the minimum cut problem is asking for less information than the maximum flow problem, it seems as if we should be able to solve the former problem more efficiently than the later one. More on this in Section 1.5.

1.5 Linear Programming duality and Max flow Min cut

Given $G = (N, A)$ and capacities u_{ij} for all $(i,j) \in A$:
Consider the formulation of the maximum flow problem on a general graph. Let x_{ij} be a variable denoting the flow on arc (i,j).

$$\begin{array}{ll} \text{Max} & x_{ts} \\ \text{subject to} & \sum_i x_{ki} - \sum_j x_{jk} = 0 \quad k \in V \\ & 0 \le x_{ij} \le u_{ij} \quad \forall (i,j) \in A. \end{array}$$

For dual variables we let $\{z_{ij}\}$ be the nonnegative dual variables associated with the capacity upper bounds constraints, and $\{\lambda_i\}$ the variables associated with the flow balance constraints.

$$\begin{array}{cl}
\text{Min} & \sum_{(i,j)\in A} u_{ij} z_{ij} \\
\text{subject to} & z_{ij} - \lambda_i + \lambda_j \geq 0 \quad \forall (i,j) \in A \\
 & \lambda_s - \lambda_t \geq 1 \\
 & z_{ij} \geq 0 \qquad \forall (i,j) \in A.
\end{array}$$

The dual problem has an infinite number of solutions: if (λ^*, z^*) is an optimal solution, then so is $(\lambda^* + C, z^*)$ for any constant δ. To avoid that we set $\lambda_t = 0$ (or to any other arbitrary value). Observe now that with this assignment there is an optimal solution with $\lambda_s = 1$ and a partition of the nodes into two sets: $S = \{i \in V | \lambda_i = 1\}$ and $\bar{S} = \{i \in V | \lambda_i = 0\}$.

The complementary slackness condition state that the primal and dual optimal solutions x^*, λ^*, z^* satisfy,

$$x^*_{ij} \cdot [z^*_{ij} - \lambda^*_i + \lambda^*_j] = 0$$

$$[u_{ij} - x^*_{ij}] \cdot z^*_{ij} = 0.$$

In an optimal solution $z^*_{ij} - \lambda^*_i + \lambda^*_j = 0$ so the first set of complementary slackness conditions do not provide any information on the primal variables $\{x_{ij}\}$. As for the second set, $z^*_{ij} = 0$ on all arcs other than the arcs in the cut $(S, \bar{S})$. So we can conclude that the cut arcs are saturated, but derive no further information on the flow on other arcs.

The only method known to date for solving the minimum cut problem requires finding a maximum flow first, and then recovering the cut partition by finding the set of nodes reachable from the source in the residual graph (or reachable from the sink in the reverse residual graph). That set is the source set of the cut, and the recovery can be done in linear time in the number of arcs, $O(m)$.

On the other hand, if we are given a minimum cut, there is no efficient way of recovering the flow values on each arc other than essentially solving from scratch. The only information given by the minimum cut, is the value of the maximum flow and the fact that the arcs on the cut are saturated. Beyond that, the flows have to be calculated with the same complexity as would be required without the knowledge of the minimum cut.

This asymmetry implies that it may be easier to solve the minimum cut problem than to solve the maximum flow problem. Still, no minimum cut algorithm has ever been discovered, in the sense that every so-called minimum cut algorithm known computes first the maximum flow.