ORIE 633 Network Flows August 23, 2007

Lecture 1

Lecturer: Jon M. Kleinberg *Scribe: Matthew Maxwell*

1 Course overview

The course will cover algorithms for network flow problems. Network flows is an important subfield of combinatorial optimization, and combinatorial optimization is all about how to make decisions that have discrete choices. It helps us answer natural questions involving networks of one sort or another, such as roads, railway lines, and computer networks. We can answer questions such as: how much stuff can be routed from point A to point B? What paths should it take? How can it be done most cost effectively? How quickly can it be done? What if some fraction of the stuff gets lost along the way?

Less obviously, network flow problems turn out to be useful in modelling problems that don't seem to have anything to do with networks. It can be used to answer questions such as: is my favorite baseball/basketball/hockey team eliminated from winning its division? What types of amino acids should be combined to give a protein that folds into a specified shape? How should prices of goods be set so that a marketplace operates most efficiently? We will see some of these applications during the semester.

In this class we will focus on several things:

- **Efficient algorithms in combinatorial optimization**. That is, algorithms that run in polynomial time. Fortunately for us, many network problems have efficient algorithms. We may look at some NP-hard network problems toward the end of the semester and there we will consider *approximation algorithms*; that is, efficient algorithms that return near-optimal solutions.
- **Central role of linear programming and duality.** LP is one of the fundamental tools in combinatorial algorithms and in thinking about these problems – both in modelling the problems and algorithms for them. Often for flow problems one can think about the problem in purely combinatorial terms, but we will sometimes step back and show LP-based explanations.
- **Problems/algorithms that are either a fundamental piece of background, a useful technique for solving other problems, or an interesting open research direction.**

2 The maximum s-t flow problem

We begin with the granddaddy of all network flow problems.

Maximum s-t Flow Problem

- **Input:**
  - Directed graph $G = (V, E)$
  - Capacities $u_{ij} \geq 0, \forall (ij) \in E, u_{ij}$ integer
  - Source node $s \in V$, sink node $t \in V, s \neq t$
- **Goal:** Find an *s-t* flow that maximizes the net flow out of the source node.

We need to start by defining what we mean by a flow.

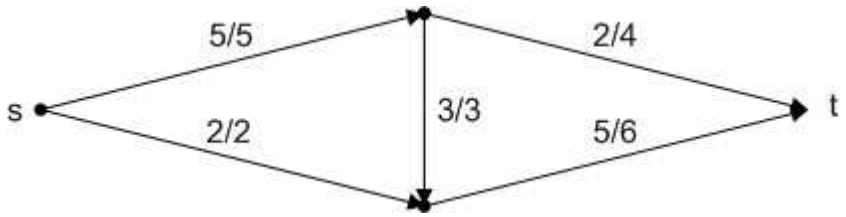
Definition 1 *An* s-t flow *is a function* $f : E \to \mathbb{R}^{\geq 0}$ *s.t.*

1. $f_{ij} \leq u_{ij} \qquad \forall (i,j) \in E$ *(capacity constraints)*
2. $\sum_{j:(j,i)\in E} f_{ji} = \sum_{k:(i,k)\in E} f_{ik}, \qquad \forall i \in V, i \neq s, t$ *(flow conservation constraints)*

For ease of notation we will define $f^{in}(i) \equiv \sum_{j:(j,i)\in E} f_{ji}$ and $f^{out}(i) \equiv \sum_{k:(i,k)\in E} f_{ik}$. Hence the second constraint, $f^{in}(i) = f^{out}(i)$, can be alternatively described as "flow in = flow out" for non-source, non-sink nodes.

Definition 2 *The value of a flow* f *is* $|f| \equiv \sum_{k:(s,k)} f_{sk} - \sum_{k:(k,s)} f_{ks} = f^{out}(s) - f^{in}(s)$

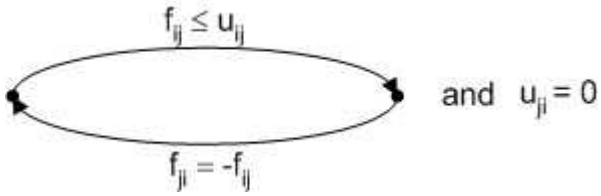
For the following network below, observe that both flow conditions are met and $|f| = 7$. (The first number above an arc is the flow along the arc, and the second number is the capacity of the arc.)



For notational simplicity, we will use an alternative definition of flow. For this alternate definition, we will assume that if $(i,j) \in E$, then $(j,i) \in E$. In the alternate definition, if there is a flow f_{ij} on arc (i,j) then the flow $f_{ji} = -f_{ij}$ on the reverse arc (j,i). If (j,i) is not in the original graph, then we set $u_{ji} = 0$; note that since $f_{ji} = -f_{ij} \leq u_{ji} = 0$, this enforces that the flow on the "original" arc is nonnegative. Pictorially, for each "original" arc $(i,j) \in E$, we have

$$0 \leq f_{ij} \leq u_{ij}$$

$$\Leftrightarrow$$



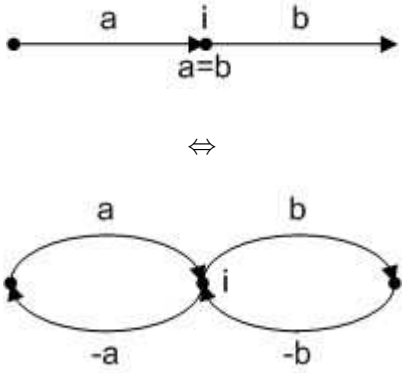
The original constraint for flow conservation,

$$f^{in}(i) = f^{out}(i), \qquad \forall i \in V, i \neq s, t$$

now becomes

$$f^{out}(i) = 0, \qquad \forall i \in V, i \neq s, t.$$

Visually,



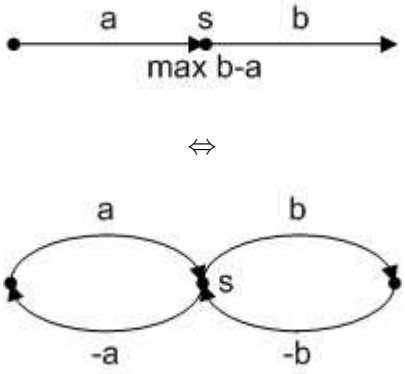
The original expression for the value of the flow of the network,

$$|f| = f^{out}(s) - f^{in}(s)$$

now becomes

$$|f| = f^{out}(s).$$

Visually,



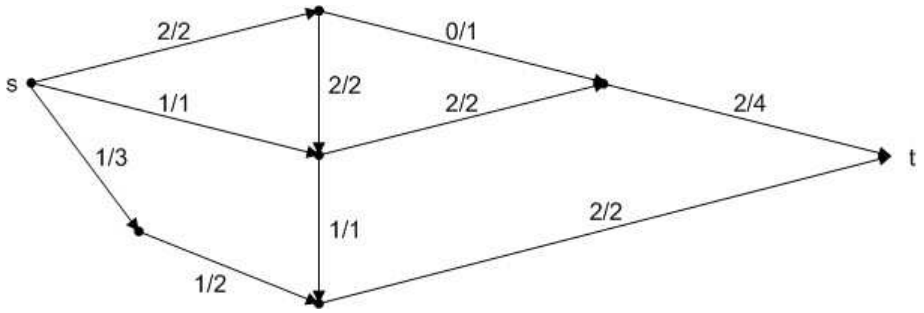
To summarize, the alternate definition of network flow is as follows.

Definition 3 *An* s-t flow $f : E \to \mathbb{R}$ *s.t.*

1. $f_{ij} \leq u_{ij}, \qquad \forall (i,j) \in E$ *(capacity constraints)*
2. $f_{ij} = -f_{ji}, \qquad \forall (i,j) \in E$ *(anti-symmetry)*
3. $f^{out}(i) = 0, \qquad \forall i \in V, i \neq s,t$ *(flow conservation constraints)*

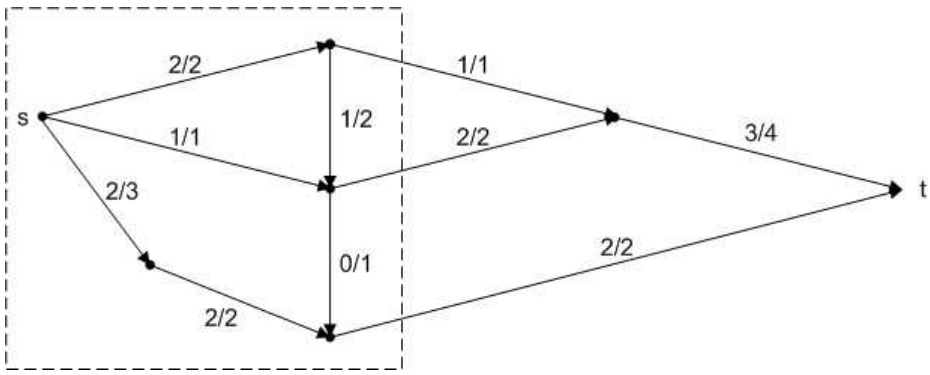
Definition 4 *The value of a flow* f *is* $|f| \equiv f^{out}(s)$

Example:



Question: Observe that $|f| = 4$. Is this a maximum flow? One reason for thinking that perhaps it is a maximum flow is because on every s-t path there is some arc that has reached its capacity, so we cannot augment flow along some s-t path.

However, it is not a maximum flow, because we can give a flow f where $|f| = 5$, as below.



Note that conceptually to get from the first flow to the second we performed a series of do/undo operations. First, we added one unit of flow from s through the two bottom-left edges. Then we "undid" one unit of flow from each of the two edges coming vertically down the graph in the middle. Last, we added one unit of flow along the two remaining edges leading to t. Hence this creates an s-t path where the edges involved in the "undoing" process are included as the reverse direction of the actual edges in the graph. We will formalize this later using our new definition of graph flow.

Now the question remains, is this a maximum flow? In this case we can show that this is a maximum flow by demonstrating an appropriate *s-t cut.* Intuitively, we can cut the network into one set of nodes containing the source node and another set of nodes containing the sink node. We want to send as much flow as possible across this cut. Intuitively, we can expect the amount of flow we can send is bounded by the total capacities of arcs that cross some cut. Specifically, the cut in the example has value 5, so the value of the maximum s-t flow can't be more than 5. Since we found a flow of value 5, we've found a maximum flow.

First, we will formalize the process of finding valid flows with larger values of $|f|$ as illustrated by the do/undo process above. Then we will formalize the above intuition of cut and the capacity of a cut bounding the value of a flow. Last, we will prove that max flow = min s-t cut.

Definition 5 *The residual graph G_f relative to a flow f is $G_f = (V, E_f)$ where E_f is the set of all edges for which $f_{ij} < u_{ij}$. Let E' be the union of all edges originally in the graph G and the edges added via our definition of flow. Then formally, $E_f = \{(i,j) \in E' : f_{ij} < u_{ij}\}$.*

Definition 6 *The residual capacity for each edge $(i,j) \in E_f$ is $u_{ij} - f_{ij}$.*

Definition 7 *An augmenting path P is an s-t path in the residual graph.*

Now to formalize our process of finding valid flows with greater $|f|$ values assume we have an augmenting path P in our residual graph. Let δ be the minimal residual capacity over all edges in P. Now we note that we can increase flow by δ on all edges (i,j) of P and reduced the associated (j,i) edges by δ without violating the capacity constraints, anti-symmetry constraints, or conservation constraints. Hence we have a valid flow and $|f|$ has increased by δ.

Next, we will formalize the intuition of a cut and the capacity of a cut which we used earlier to show that the flow we had found was maximal.

Definition 8 *An* s-t cut *is a set $A \subseteq V$ s.t. $s \in A, t \notin A$.*

Definition 9 *$\delta^+(A) = \{(i,j) \in E : i \in A, j \notin A\}$. Similarly, we will define $\delta^-(A) = \{(i,j) \in E : i \notin A, j \in A\}$. Note in particular that the set of edges for both these definitions only includes those edges in the original graph, not any added by our new definition of flow.*

Definition 10 *The* capacity *of an s-t cut A is $u(\delta^+(A)) \equiv \sum_{(i,j)\in\delta^+(A)} u_{ij}$.*

Using everything we have developed up to this point we can prove the following theorem.

Theorem 1 *The following three statements are equivalent:*

1. *f is a max flow*
2. *There is no augmenting path in the residual graph G_f*
3. *There exists an s-t cut A such that $|f| = u(\delta^+(A))$.*

Proof: We will prove this by showing $1 \Rightarrow 2$, $3 \Rightarrow 1$, and then $2 \Rightarrow 3$.

$1 \Rightarrow 2$. We have already shown the contrapositive: if there is an augmenting path, then f is not maximal.

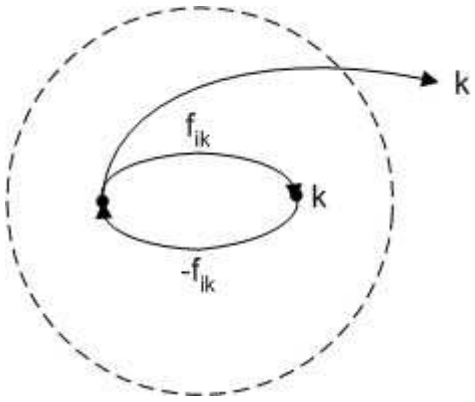
$3 \Rightarrow 1$. We will show that for all flows f and all s-t cuts A that $|f| \leq u(\delta^+(A))$.

$$\begin{aligned}
|f| &= f^{out}(s) + 0 \\
&= f^{out}(s) + \sum_{i \in A, i \neq s} f^{out}(i) \qquad (1) \\
&= \sum_{i \in A} f^{out}(i) \\
&= \sum_{i \in A, j \notin A} f_{ij} + \sum_{i,j \in A} f_{ij} \\
&= \sum_{i \in A, j \notin A} f_{ij} \qquad (2) \\
&\leq \sum_{i \in A, j \notin A} u_{ij} \\
&= u(\delta^+(A))
\end{aligned}$$

Equality (1) follows from the flow conservation constraints and the fact that $t \notin A$. Equality (2) follows from the fact that

$$\sum_{i,j \in A} f_{ij} = 0.$$

This follows from the flow conservation constraints and the fact that for $i, k \in A$; by anti-symmetry, f_{ik} will be canceled out by $f_{ki} = -f_{ik}$.



$2 \Rightarrow 3$. We will show that if there is no augmenting path in G_f then there exists an s-t cut A so that $|f| = u(\delta^+(A))$.

Let A be the set of all nodes reachable from s by a path in the residual graph. A is an s-t cut because by assumption there is not an augmenting path and hence t is not reachable

from s. Now we see that

$$\begin{aligned}
|f| &= f^{out}(s) + 0 \\
&= f^{out}(s) + \sum_{i \in A, i \neq s} f^{out}(i) \\
&= \sum_{i \in A} f^{out}(i) \\
&= \sum_{i \in A, j \notin A} f_{ij} + \sum_{i,j \in A} f_{i,j} \\
&= \sum_{i \in A, j \notin A} f_{ij} \\
&= \sum_{i \in A, j \notin A} u_{ij} \qquad (3) \\
&= u(\delta^+(A))
\end{aligned}$$

We note that the equality (3) holds because each of the f_{ij} in the summation must be saturated. If not, the respective node j for the non-saturated edge (i, j) would be reachable, but this contradicts the fact that $j \notin A$. □

Note that this proof suggests an algorithm for finding a maximal flow. Initialize with no flow on any edges. Find an augmenting path in the residual graph. If there is no augmenting path then our current flow is maximal by (1) and we can find a cut via (3) to prove optimality. Otherwise, we use the new flow from the augmenting path, compute the new residual graph, and repeat. Furthermore, we note that additional analysis of this algorithm would prove that if we started with integer capacities there would exist a maximal flow with only integer flow values.

The original version of this famous theorem was proven in 1955 by Ford and Fulkerson. Lex Schrijver has written a history of combinatorial optimization that shows that Ford and Fulkerson's initial interest in flows was motivated by looking at the total capacity of railway lines to ship goods between two points. After some digging, Lex discovered that their work arose from an Air Force application that was in fact interested in finding a minimum cut: an "interdiction" of railway lines in Eastern Europe that would cut shipments between the Soviet Union and its satellite states.