EECS 290S: Network Information Flow **Fall 2008**

Lecture 2 — September 2

Lecturer: Anant Sahai and David Tse *Scribe: Naveen Goela*

2.1 Single Source Single Destination Network

Main Topic: Graph theory establishes a max-flow min-cut bound for single-source, single destination network coding.

2.1.1 Max Flow and Min Cut

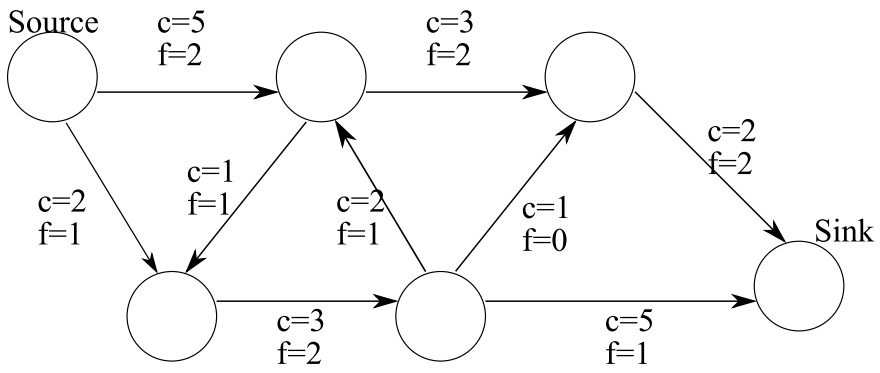

Figure 2.1. A valid flow on a graph.

A point-to-point communication network is represented by a finite *directed* graph $G = (V, E)$ where V is the set of nodes in G and E is the set of edges (point-to-point channels). Information is generated at the source s and absorbed at the sink t, similar to water flowing through pipes. Each point-to-point link has a maximum rate constraint or **capacity** $c(i, j) > 0$.

A **flow** is a valid assignment to all edges in the graph:

$$F = (f(i, j) : edge(i, j) \in E) \tag{2.1}$$

with the conditions:

$$f(i, j) \leq c(i, j) \tag{2.2}$$
$$f(i, j) = -f(j, i) \tag{2.3}$$

These conditions are met by the flow for the graph in Figure 2.1. Consequently, for any node that is not the source or sink, the total flow into the node is equal to the total flow out of

the node:

$$\sum_j f(i,j) = 0, \forall i \notin \{s,t\} \tag{2.4}$$

The next definition is the value of the flow F, the total flow from source s to t:

$$|F| = \sum_j f(s,j) \tag{2.5}$$

The value of the flow in Figure 2.1 is $|F| = 3$. A flow F is a *max-flow* if its value is greater than any other flow from s to t, subject to the capacities of the links.

A standard linear programming (LP) solution exists for finding the max-flow so the problem is tractable:

$$\begin{aligned} \max \quad & |F| && (2.6)\\ \textit{subject to}: \quad & f(i,j) \le c(i,j) && (2.7)\\ & f(i,j) + f(j,i) = 0 && (2.8) \end{aligned}$$

Towards a bound on the max-flow, a *cut* between s and t is a partitioning of V into two sets S and T such that $S \cap T = \phi$, $S \cup T = V$, $s \in S$, and $t \in T$. The capacity of the cut is defined as:

$$c(S,T) = \sum_{i \in S} \sum_{j \in T} c(i,j) \tag{2.9}$$

A cut is a *min-cut* if its capacity is less than or equal to the capacity of any other cut from s to t. The min-cut in Figure 2.1 is: $\min_{allcuts} c(S,T) = 5$. Since the flow $|F| = 3$, it seems that it is not the max-flow possible.

Theorem 2.1. *Let G be a graph with source s and sink t with link capacities c_{ij}. Then the value of a max-flow from s to t is equal to the capacity of a min-cut between s and t.*

Proof: In following exercises, it is shown that the value of any flow is less than the capacity of the min cut. The Ford-Fulkerson algorithm introduced in Sec. 2.2 shows that the max-flow is equal to the capacity of the min-cut. Refer to Raymond-Yeung Theorem 18.1 for further details. □

2.1.2 Basic Exercises

If X and Y are sets of vertices, a shorthand notation for flows on a graph is the following:

$$F(X,Y) \triangleq \sum_{i \in X} \sum_{j \in Y} f(i,j) \tag{2.10}$$

- Exercise: How is it possible to find the max-flow if there are two sources? One way is to modify the graph by inserting a new source node s with infinite capacity links to each of the two sources. Then a max-flow solution is the maximum combined flow from the two sources. How can we trade-off between different sources ? In other words, if we want a fraction of the flow from one source versus the other source? (We can set zero flow for one source, and maximize the flow from the other source, but how to trade-off flows from sources?)
- Exercise: Show $|F| = \sum_i f(i,t) = F(V-t,t)$. $F(V,V) = 0$ and $\sum_{\forall i \notin \{s,t\}} f(i,j) = 0$ implies that the flow out of the source s is equal to the flow coming into the sink t. This is true for the flow assignment in Figure 2.1.
- Exercise: Show $|F| \leq c(S,T)$ where (S,T) is a cut.

$$
\begin{aligned}
|F| &= F(s,V) && (2.11)\\
&= F(s,V) + 0 && (2.12)\\
&= F(s,V) + F(S-s,V) && (2.13)\\
&= F(S,V) && (2.14)\\
&= F(S,S) + F(S,T) && (2.15)\\
&= F(S,T) && (2.16)\\
&= \sum_{i \in S} \sum_{j \in T} f(i,j) && (2.17)\\
&\leq \sum_{i \in S} \sum_{j \in T} c(i,j) && (2.18)\\
&= c(S,T) && (2.19)
\end{aligned}
$$

Since the second result above holds for any cut, a bound on any flow from source to sink follows:

$$|F| \leq \min_{(S,T) cuts} c(S,T) \quad (2.20)$$

2.2 Ford-Fulkerson

Of polynomial time complexity, the Ford-Fulkerson algorithm finds the max-flow by selecting augmenting paths on the residual graph.

2.2.1 A Simple Greedy Algorithm

As a first step, to find max-flow assignments, a simple greedy algorithm might proceed as follows:

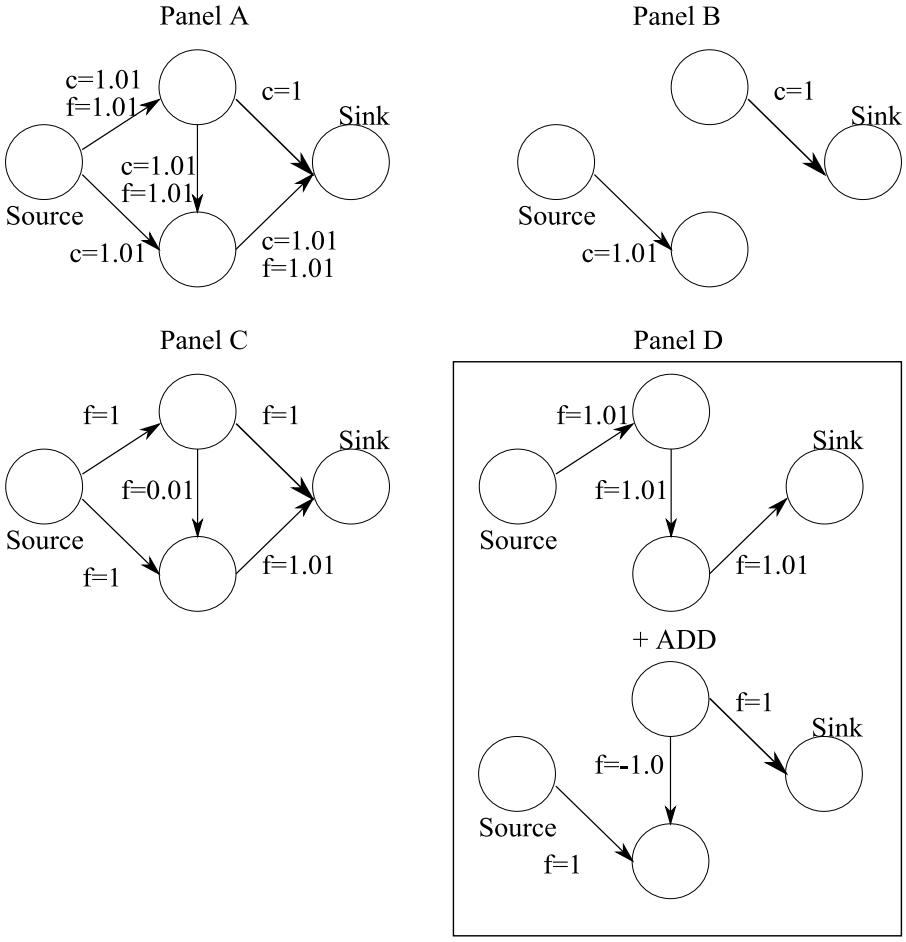

Figure 2.2. A simple greedy algorithm (Panel A and Panel B). The need for superposition of flows (Panel C and Panel D).

- Find the max-flow along one path in the graph.
- Remove all saturated links where the flow is equal to link capacity.
- Repeat if another path exists from s to t.

The simple greedy algorithm treats paths independently and is suboptimal by example, as shown in Panel A and Panel B of Figure 2.2. The flow value along the path in Panel A is 1.01 and after removing all saturated path links, there is no other path from s to t, so the algorithm terminates. However, it is possible to have a flow with $|F| = 2.01$.

A better approach is to have a method of handling the superposition of flows via the residual graph. Superposition of flows is shown in Panel C and Panel D of Figure 2.2.

2.2.2 Residual Graphs

A residual graph G_F is a graph defined with new link capacities based on the flow F on the original graph G:

$$c_F(i,j) \triangleq c(i,j) - f(i,j) \tag{2.21}$$

A residual graph example is shown in Figure 2.3. By applying the definition in eqn. 2.21, and noting that flows in the opposite direction are negative, the residual graph may have new positive link capacities in both directions from one node to another.

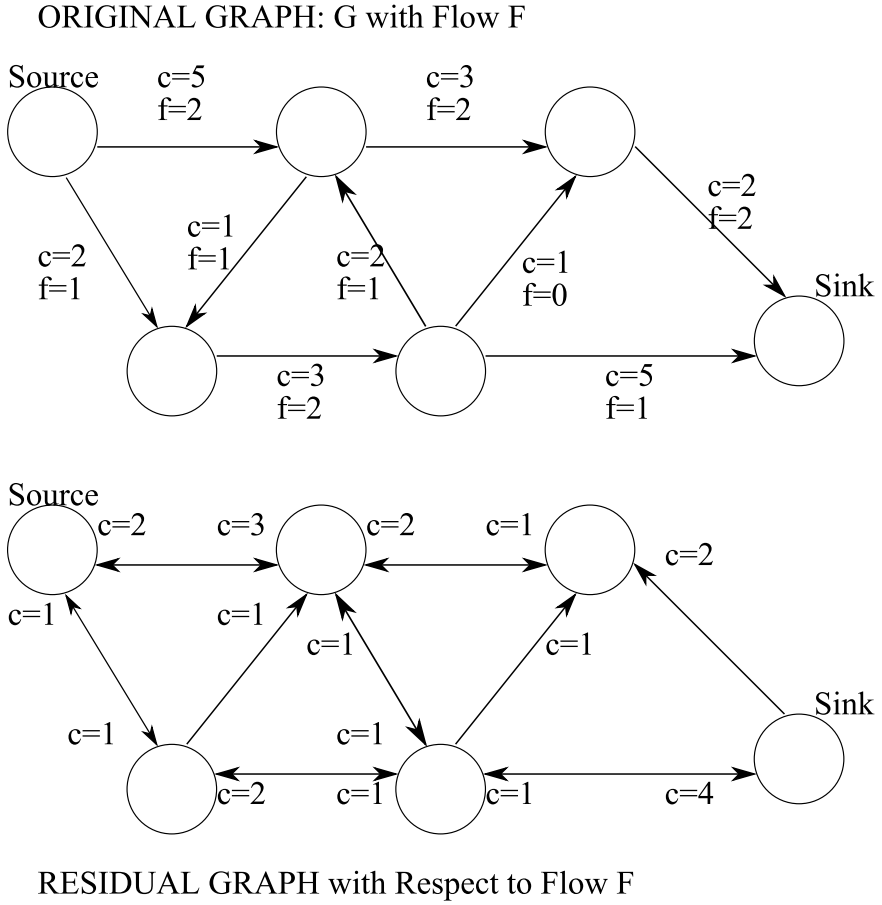

Figure 2.3. The residual graph G_F with respect to a flow on graph G.

If F is a flow on G, and F' is a flow on G_F, then $F + F'$ is a flow on G. If $f'(i,j) \leq c(i,j) - f(i,j)$, then $f(i,j) + f'(i,j) \leq c(i,j)$, meaning it is possible to add a flow and a residual flow together. This is the basis of the Ford-Fulkerson algorithm.

2.2.3 Ford-Fulkerson Algorithm

The Ford-Fulkerson algorithm selects paths P which are augmenting paths. A path P is an augmenting path of G and F if it is a valid path from s to t in G_F (i.e. $c_F(i,j) \geq 0$ if

$edge(i,j) \in P$).

The basic algorithm follows:

- (0) Start with G and $F = \phi$.
- (1) Find an augmenting path P of G_F using breadth-first search for example.
- (2) If none exists, stop. Otherwise set $F' = P$ where the flow along the path is $\min_{(i,j)\in P} c_F(i,j)$. In other words, saturate the path.
- (3) Set $F = F + F'$.
- (4) GOTO Step (1).

Assuming that link capacities $c(i,j)$ are integers or rationals, and because the min-cut $\leq \infty$, the Ford-Fulkerson algorithm must terminate. Each iteration increases the value of the flow F.

The algorithm also finds the optimal solution as shown by the equivalence of the following statements:

- **(A)** F is a maximal flow.
- **(B)** G_F has no augmenting path.
- **(C)** $|F| = c(S,T)$ for some cut (S,T) of G.

The first implication is **(C)** $\Rightarrow$ **(A)**. In the exercises, it was shown than the value of any flow $|F|$ is less than or equal to the capacity of the min-cut. If $|F|$ equals the capacity of a cut in the graph, then that cut must be the min-cut, and the flow cannot be increased further, so it must be the max-flow.

A second implication is **(A)** $\Rightarrow$ **(B)**. By contraposition, if G_F has an augmenting path, then the Ford-Fulkerson algorithm is able to add a flow to the existing flow F, meaning F is not a max-flow assignment.

A third implication is **(B)** $\Rightarrow$ **(C)**. We must show that if G_F has no augmenting path, then $|F| = c(S,T)$ for some cut. Let S be all vertices that can be reached from s in the residual graph G_F on valid paths. Let $T = V - S$. Then (S,T) is a cut. Since G_F has no augmenting path, $c_F(i,j) = 0$. Then by definition of residual capacity, $c(i,j) - f(i,j) = 0$, and $c(i,j) = f(i,j)$. Finally, $|F| = F(S,T) = \sum_i \sum_j f(i,j) = \sum_i \sum_j c(i,j) = c(S,T)$. Mathematically, the proof is complete for **(B)** $\Rightarrow$ **(C)**.

The scenario is displayed in Figure 2.4. It is possible to characterize the flows across the cut. Consider $c(i,j)$ in G where $i \in S$ and $j \in T$. Suppose $c(i,j) > 0$. We know $c_F(i,j)$ must be zero because G_F has no augmenting path. Therefore $f(i,j) = c(i,j)$ implies saturation of the links from S to T. Now consider $c(j,i) > 0$, i.e. the capacity of links going back from T to S is positive. Intuitively, there should be zero flow on these links. Again $c_F(i,j) = 0$ and by definition $c(i,j) - f(i,j) = 0$. Assuming a backward link, $c(i,j) = 0$, so the flow $f(i,j) = 0 = f(j,i)$. Hence there is no flow backward.

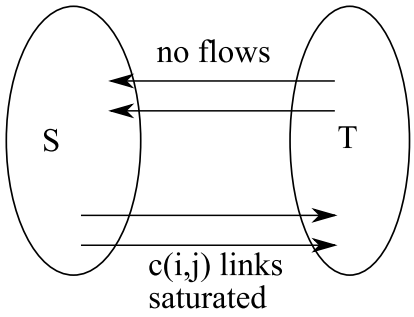

Figure 2.4. If G_F has no augmenting path, $|F|$ is equal to the capacity of a cut in G.

The Ford-Fulkerson algorithm establishes that the value of the max-flow is equal to the min-cut capacity.