CS 4820, Summer 2010

Lecture Notes on Maximum Flow Problem

Instructor: Yogi Sharma

In these short lecture notes, we review the definitions given for network flows. We won't go into all the details here, but this should be a useful companion with your lecture notes.

1 Motivation

The bipartite matching problem in the one that motivated the study of maximum flows. In a bipartite graph, the node set can be partitioned into two sets, L and R, such that all edges are between nodes of L and R (and no edge is between vertices in L or R). A matching in a bipartite graph is a set of edges $E' \subseteq E$ such that edges in E' don't intersect, where e and f intersect if they share a common end point. The Maximum Bipartite Matching Problem seeks to find a bipartite matching of maximum cardinality.

Instead of solving this problem, we will look into Network Flows problem, which generalizes the bipartite matching problem. There are a lot of other application for the Maximum-Flow problem, see homepages.cwi.nl/~lex/files/histtrpclean.ps if you are interested in the history of the problem.

2 Definitions

We are going to think of transportation networks as (directed) graphs, in which vertices denote hubs/switches/highway-intersections, and the arcs represent traffic/network-data etc. The key ingredients to any network flow problem are (i) a source that generates the traffic, (ii) a sink that absorbs the traffic, and (iii) capacities on edges that actually carry the traffic.

We define a **flow network** as follows. It is a directed graph $G = (V, A)$, with capacities $c(a)$ for every arc $a \in A$. There is a source node $s \in V$ and a sink node $t \in V$. Refer to the textbook for examples of flow networks.

We assume that there is no arc entering s or leaving t (this is without loss of generality, or wlog), and that all capacities are integers (this does lose some generality, but maintains the essence of the problem and avoids some pathologies).

We define a flow to be a function $f : A \to \mathbb{R}_{\geq 0}$, where $f(a)$ denotes the amount of flow carried by arc $a \in A$. We can extend f to act on sets of arcs: $f(B) = \sum_{a \in B} f(a)$ for $B \subseteq A$. This function must satisfy two condition to be called a flow, which are as follows.

1. Capacity constraints: $0 \leq f(a) \leq c(a)$ for all $a \in A$. It says that no edge can carry an amount more than its capacity (and cannot carry negative amount).

2. Conservation constraints: For all vertices $v \neq s, t$

$$\sum_{u:(u,v)\in A} f(u, v) = \sum_{w:(v,w)\in A} f(v, w). \tag{2.1}$$

   This says that the amount of flow entering a vertex is equal to the amount of flow leaving that vertex (unless that vertex is s or t).

Let us make some definitions in order to talk about these quantities more succinctly.

1. $\delta^{\text{in}}(S) = \{(u,v) : u \notin S, v \in S\}$. This is the set of arcs *entering* $S \subseteq V$.
2. $\delta^{\text{out}}(S) = \{(u,v) : u \in S, v \notin S\}$. This is the set of arcs *leaving* $S \subseteq V$.
3. $\delta^{\text{in}}(v) = \delta^{\text{in}}(\{v\})$ for $v \in V$.
4. $\delta^{\text{out}}(v) = \delta^{\text{out}}(\{v\})$ for $v \in V$.

With these notations, and the notation $f(B)$ for set of arcs B, the conservation constraints can be written very succinctly:

$$f(\delta^{\text{in}}(v)) = f(\delta^{\text{out}}(v)) \quad \text{for all } v \in V - \{s,t\}.$$

We are now left with defining how good a flow is. This is captured by the value of flow f, which is just a fancy way of saying how much flow is leaving the source node s. So, $\text{value}(f) = f(\delta^{\text{out}}(s))$. Note that we don't have any arc entering s, so $\delta^{\text{in}}(s) = \emptyset$.

Let us make one more notation in order to succinctly write the mathematical expressions that will follows.

1. $\text{excess}_f(U) = f(\delta^{\text{in}}(U)) - f(\delta^{\text{out}}(U))$ for $U \subseteq V$. This is just how much *excess* flow is entering the set U than is leaving.
2. $\text{excess}_f(v) = \text{excess}_f(\{v\})$ for all $v \in V$.

With this notation, we have a compact notation for the value of flow:

$$\text{value}(f) = -\text{excess}_f(s).$$

Also, note that the conservation constraints can also be written much more compactly:

$$\text{excess}_f(v) = 0 \quad \text{for all } v \in V - \{s,t\}.$$

Let us prove a small lemma to exercise our definitions.

Lemma 1. *Let $U \subseteq V$. Then*

$$\text{excess}_f(U) = \sum_{v \in U} \text{excess}_f(v).$$

Proof. We have

$$\begin{aligned}
&\text{excess}_f(U) \\
&= \sum_{(u,v)\in A} f(u,v)\left(\mathbf{1}[u \in U \wedge v \notin U] - \mathbf{1}[u \notin U \wedge v \in U]\right) \quad \text{(definition of excess)} && (2.2)\\
&= \sum_{(u,v)\in A} f(u,v)\left(\mathbf{1}[u \in U \wedge v \notin U] - \mathbf{1}[u \notin U \wedge v \in U]\right) + \sum_{u,v\in U} f(u,v) - \sum_{u,v\in U} f(u,v), && (2.3)
\end{aligned}$$

where the second equality follows because we are adding and subtracting an equal amount.

Now, let us look at the right hand side.

$$\begin{aligned}
&\sum_{v\in U} \text{excess}_f(v) && (2.4)\\
&= \sum_{v\in U} \left(\sum_{u:(u,v)\in A} f(u,v) - \sum_{w:(v,w)\in A} f(v,w) \right) \quad \text{(definition of excess function)} && (2.5)
\end{aligned}$$

Now, we look at the arcs in the sum (2.3) and arcs in the sum (2.5). We claim that the same arcs appear in both sums, and they appear with the same sign (+ or −). There are four cases to consider:

1. $(u, v) \in A$ such that $u, v \notin U$.
2. $(u, v) \in A$ such that $u, v \in U$.
3. $(u, v) \in A$ such that $u \notin U, v \in U$.
4. $(u, v) \in A$ such that $u \in U, v \notin U$.

It is easy to see that all four kinds of arcs occur with same coefficient in both sums. Hence, both sums must be equal. □

Now, we can also claim an intuitive fact that the value of a flow is equal to the amount of flow entering t (which is also equal to the amount of leaving s).

Lemma 2. *Let f be a flow. Then* $\text{value}(f) = f(\delta^{\text{out}}(s)) = f(\delta^{\text{in}}(t))$.

Proof. We had by definition that $\text{value}(f) = f(\delta^{\text{out}}(s))$. This can be compactly presented as $\text{value}(f) = -\text{excess}_f(s)$ (note that there is no arc leaving s or entering t).

$$\begin{aligned}
\text{value}(f) &= -\text{excess}_f(s) \\
&= -\text{excess}_f(s) - \sum_{v \in V - \{s,t\}} \text{excess}_f(v) \\
&= -\text{excess}_f(V - \{t\}) \\
&= f(\delta^{\text{out}}(V - \{t\})) - f(\delta^{\text{in}}(V - \{t\})) \\
&= f(\delta^{\text{in}}(t)) - f(\delta^{\text{out}}(t)) \\
&= \text{excess}_f(t),
\end{aligned}$$

where the second last inequality follows from the fact that if an arc leaves $V - \{t\}$, it must enter t and if it enters $V - \{t\}$, then it must leave t. □

Now, our goal it so find a flow with maximum value. The basic obstacle to finding such a good flow are the cuts in the graph. Let us say we have a subset $S \subseteq V$ such that $s \in S$ and $t \notin S$. Then, every bit of flow from s to t must "cross" the edges in $\delta^{\text{out}}(S)$. If the capacity of $\delta^{\text{out}}(S)$ is small (that is $c(\delta^{\text{out}}(S))$ is small), then we cannot hope to find a flow of large value. Let us formalize this in the following lemma.

Lemma 3. *Let f be any flow in a flow network $G = (V, A)$ and let S be any set containing s and not containing t. Then,* $\text{value}(f) \leq c(\delta^{\text{out}}(S))$*, and the equality holds if and only if $f(a) = 0$ for all $a \in \delta^{\text{in}}(S)$ and $f(a) = c(a)$ for all $a \in \delta^{\text{out}}(S)$.*

Proof. We have that the value of flow is equal to $-\text{excess}_f(s)$.

$$\begin{aligned}
\text{value}(f) &= -\text{excess}_f(s) \\
&= -\text{excess}(S) \quad \begin{array}{l}\text{(because the excess of} \\ \text{vertices in } S - \{s\} \text{ is 0.)}\end{array} \\
&= f(\delta^{\text{out}}(S)) - f(\delta^{\text{in}}(S)) \\
&\leq c(\delta^{\text{out}}(S)) - 0,
\end{aligned}$$

where the inequality follows from the fact that $f(a) \leq c(a)$ and $f(b) \geq 0$ for all $a, b \in A$. Note that the equality will hold if and only if $f(a) = c(a)$ for all $a \in \delta^{\text{out}}(S)$ and $f(b) = 0$ for all $b \in \delta^{\text{in}}(S)$. □

This finishes the heavy duty notation part of the lecture. After this, we defined a residual graph and devised an algorithm for pushing the flows along arcs of the residual graph. I am including my notes for that part below, which are just some bullet points. They may or may not be so useful. Please read Section 6.1 and 6.2 from the textbook to get more familiar with these concepts.

3 Bullet point notes for algorithm

- ❶ Designing the algorithm.
  - ❷ We don't know of a dynamic programming or greedy algorithm.
  - ❷ Let us try a greedy solution.
    - ❸ We might get stuck if we made a wrong choice.
  - ❷ A general way of pushing flow.
    - ❸ Increase on arcs that are below capacity, decrease on arcs that are above zero.
    - ❸ Define the residual graph: $G_f = (V, A_f)$.
      - ❹ for all $(u, v) \in A$ s.t. $f(u, v) < c(u, v)$, include $(u, v) \in A_f$ with capacity $c_f(u, v) = c(u, v) - f(u, v)$. called forward arc.
      - ❹ for all $(u, v) \in A$ s.t. $0 < f(u, v)$, include $(u, v)^{-1} = (v, u) \in A_f$ with capacity $c_f(v, u) = f(u, v)$. called backward arc.
      - ❹ These are called **residual capacities**.
      - ❹ Note that we might have "parallel" arcs.
      - ❹ IMP Draw the picture for both kinds of arcs, and show the residual capacities.
  - ❷ Augmenting paths in residual graph.
    - ❸ G_f is residual graph.
    - ❸ Let P be a *simple* path in G_f. (called augmenting path)
    - ❸ $\text{bottleneck}_f(P) = \min_{a:a\in P} c_f(a)$. Denotes how much flow you can push along P.
      - ❹ If $(u, v) \in P$ and it is a forward arc, then increase $f(u, v)$ by $\text{bottleneck}_f(P)$.
      - ❹ If $(u, v) \in P$ and it is a backward arc, then decrease $f(v, u)$ by $\text{bottleneck}_f(P)$.
    - ❸ Let f' be a flow after pushing $\text{bottleneck}_f(P)$ amount of flow.
    - ❸ Lemma: If f was a flow, then so is f'.

      Proof: Just check the capacity constraints and conservation constraints.
    - ❸ This gives Ford-Fulkerson algorithm for maximum flow.
      - ❹ Written in lecture notes.
  - ❷ Analyzing the algorithm.
    - ❸ Lemma: At every intermediate stages, all residual capacities are integers. So, it the flow.

      Proof by induction.
    - ❸ Using this lemma, we want to prove termination.
    - ❸ Lemma: Let f be the flow, and P be an augmenting path in G_f, giving rise to f'. Then $\text{value}(f') = \text{value}(f) + \text{bottleneck}_f(P)$.

      Proof: Look at two different kinds of arcs out of s.
    - ❸ We have from an earlier lemma that $\text{value}(f) \le c(\delta^{\text{out}}(S))$ for any cut S. Take $S = \{s\}$, and we have that $\text{value}(f) \le \delta^{\text{out}}(s) \stackrel{\text{def}}{=} C$.
    - ❸ Lemma: Ford-Fulkerson terminates in at most C iterations of the while loop.
    - ❸ Lemma: Ford-Fulkerson runs in time $O(mC)$.
