IE170: Algorithms in Systems Engineering: Lecture 24

Jeff Linderoth

Department of Industrial and Systems Engineering
Lehigh University

March 28, 2007

Taking Stock

Last Time

- Transitive Closure (Fast)
- Flows in Networks

This Time

- Flows, Flows, Flows

Flows in Networks

- $G = (V, E)$ directed.
- Each edge $(u, v) \in E$ has a capacity $c(u, v) \geq 0$
- If $(u, b) \notin E \Rightarrow c(u, v) = 0$
- We will typically have a special source vertex $s \in V$, a sink vertex $t \in V$, and we will assume there exists paths from $s \rightsquigarrow v \rightsquigarrow t \quad \forall v \in V$
- The combination of all of these things (G, s, t, c) is known as a flow network.

Net Flows

- A net flow is a function $f : V \times V \to \mathbb{R}^{|V| \times |V|}$ that satisfies three conditions:

1. Capacity Constraints:
$$f(u, v) \leq c(u, v)$$
2. Skew Symmetry:
$$f(u, v) = -f(v, u) \ \forall u \in V, v \in V$$
3. Flow Conservation:
$$\sum_{v \in V} f(u, v) = 0 \ \forall u \in V \setminus \{s, t\}$$

More Flow

- An important value we will be worried about is the value of flow $f = |f| = \sum_{v \in V} f(s, v)$: The total flow out of the source.

The Maximum Flow Problem

Given $G = (V, E)$. source node $s \in V$, sink node $t \in V$, edge capacities c. Find a flow whose value is maximum.

Lemma, Lemma, Lemma

Recall Shorthand

$$f(X, Y) = \sum_{x \in X} \sum_{y \in Y} f(x, y).$$

1. $f(X, X) = 0 \ \forall X \subseteq V$
2. $f(X, Y) = -f(Y, X) \ \forall X, Y \subseteq V$
3. Let $X, Y, Z \subset V$ be such that $X \cap Y = \emptyset$, then

$$\begin{aligned} f(X \cup Y, Z) &= f(X, Z) + f(Y, Z) \\ f(Z, X \cup Y) &= f(Z, X) + f(Z, Y) \end{aligned}$$

4. $|f| = f(V, t)$

Cuts

- A cut of a (flow) network $G = (V, E)$ is a partition of V into S and $T = V \setminus S$ such that $s \in S$ and $t \in T$
- For flow f, net flow across a cut is $f(S, T)$ and the cuts capacity is $c(S, T) = \sum_{u \in S} \sum_{v \in T} c(u, v)$
- A minimum cut of G is a cut whose capacity is minimum

A Simple Upper Bound

Flow Across Cuts Lemma

- For any cut (S, T), $f(S, T) = |f|$

Coronary :-)

The value of any flow is no more than the capacity of any cut

$$|f| = f(S, T) = \sum_{u \in S} \sum_{v \in T} f(u, v) \leq \sum_{u \in S} \sum_{v \in T} c(u, v) = c(S, T).$$

Residual Capacity

- Given a flow f in a network $G = (V, E)$, we ask ourselves the question: How much more flow can I push from $u \in V$ to $v \in V$?
- The answer is simple: The residual capacity of the arc (u, v):

$$c_f(u, v) \stackrel{\text{def}}{=} c(u, v) - f(u, v) \geq 0.$$

- Note: $f(v, u)$ might be < 0. (like if $f(u, v) > 0$).
- So if "no" original arc (v, u), and flow from (u, v), $f(v, u) < 0 \Rightarrow$ residual capacity!
- We can "increase" the flow from (v, u) by reducing the from from (u, v)

Residual Network

- Give flow f, we can create a residual network from the flow. $G_f = (V, E_f)$, with

$$E_f \stackrel{\text{def}}{=} \{(u, v) \in V \times V \mid c_f(u, v) > 0\},$$

so that each edge in the residual network can admit a positive flow.
- So if there is a path from $s \rightsquigarrow t$ in G_f, then there must be a way to increase the flow without violating the capacity constraints on any of the edges

Augmenting Flow Lemma

- We define the flow sum of two flows f_1, f_2 as the sum of the individual flows

$$(f_1 + f_2)(u, v) = f_1(u, v) + f_2(u, v).$$

- Note that $f_1 + f_2$ is also a flow function
- Moreover, we have the following:

Flow Sum Lemma (26.2)

Given a flow network G, a flow f in G. Let f' be any flow in the residual network G_f. Then the flow sum $f + f'$ is a flow in G with value $|f| + |f'|$

Augmenting Paths

- Consider a path P_{st} from s to t in G_f.
- According to the lemma, we can increase the flow in G by increasing the flow along each edge in P_{st}
- Think of it as a sequence of pipes along which we can squirt more flow from s to t
- How much more? Simple: $c_f(P_{st}) = \min\{c_f(u, v) \mid (u, v) \text{ is on } P_{st}.$

Augmenting Paths

- Augmenting flow: Let P be an augmenting path in G_f, define $f_P : V \times V \to \mathbb{R}^{|V| \times |V|}$:

$$f_P(u,v) = \begin{cases} c_f(p) & (u,v) \text{ on } P \\ -c_f(p) & (v,u) \text{ on } P \\ 0 & \text{otherwise} \end{cases}$$

  then f_P is a flow in G_f with value $|f_P| = c_f(P) > 0$
- corollary: $f' = f + f_P$ is a flow in G with value $|f'| = |f| + c_f(P) > |f|$

The Big Kahuna

Max-Flow Min-Cut Theorem

The following statements are equivalent

1. f is a maximum flow
2. f admits no augmenting path. (No (s,t) path in residual network)
3. $|f| = c(S,T)$ for some cut (S,T)

Proof of MFMC

- (1) $\Rightarrow$ (2). By contradiction. If f has an augmenting path, then the flow can't have been maximum (by previous corollary)
- (2) $\Rightarrow$ (3). Let

$$\begin{aligned} S &= \{v \in V \mid \exists \text{ path from } s \text{ to } v \text{ in } G_f\}. \\ T &= V \setminus S. \end{aligned}$$

  Note that $t \in T$ or else there was an augmenting path, so (S,T) is a cut. For each $u \in S, v \in T$, $f(u,v) = c(u,v)$ or otherwise $(u,v) \in E_f$ and we should have put $v \in S$. Therefore $|f| = f(S,T) = c(S,T)$ for the chosen cut (S,T)
- (3) $\Rightarrow$ (1). Since $|f| \leq c(S,T)$ (always), the fact that $|f| = c(S,T)$ for the chosen cut implies that f must be a maximum flow.

QUITE ENOUGH DONE

Ford-Fulkerson Algorithm

- This gave Lester Ford and Del Fulkerson an idea to find he maximum flow in a network:

```
FORD-FULKERSON(V, E, c, s, t)
for i ← 1 to n
do f[u,v] ← f[v,u] ← 0
while ∃ augmenting path P in G_f
do augment f by c_f(P)
```

- Assume all capacities are integers. If they are rational numbers, scale them to be integers.

Analysis

- If the maximum flow is $|f|^*$, then (since the augmenting path must raise the flow by at least 1 on each iteration), we will require $\leq |f|^*$ iterations.
- Augmenting the flow takes $O(|E|)$
- FORD-FULKERSON runs in $O(|f|^*|E|)$
- This is not polynomial in the size of the input.
- If you augment flow along the path with largest residual capacity, one can show that at most $O(|E| \lg U)$ iterations are needed.
  - $U = \max_{(u,v) \in V \times V} c(u,v)$
- The "greedy" (maximum capacity) aumenting path algorithm runs in $O(|E|^2 \lg U)$. This is polynomial in the size of the input, but not strongly polynomial (It still depends on the magnitude of the "numbers" in the instance, not on the size of the instance itself).

Can We Do Better!? – Edmonds-Karp

- Instead of augmenting on an *arbitrary* augmenting path, why don't we augment flow along the shortest augmenting path.
- Here shortest means simply number of edges taken, so all edges have weight 1.
- Therefore shortest pahs can be found just like you did in lab – with BFS
- With some heavy machinery (See book), one can show that if you only augment on shortest paths, then you have to do at most $O(|V||E|)$ augmentations of the flow
- Therefore Edmonds-Karp algorithm runs in $O(|V||E|^2)$ time.
- There are even faster algorithms, such as push-relabel, but we won't cover those.

Maximum Bipartite Matching

- A graph $G = (V, E)$ is bipartite if we can partition the vertices into $V = L \cup R$ such that all edges in E go between L and R
- A matching is a subset of edges $M \subseteq E$ such that for all $v \in V$, ≤ 1 edge of M is incident upon it.

Maximum Bipartite Matching

Given (undirected) bipartite graph $G = (L \cup R, E)$, find a matching M of G that contains the most edges

Applications

There are lots of applications of matching problems

- Airlines
  - L set of planes
  - R set of routes
  - $(u, v) \in E$ if plane u can fly route v
  - Maximize the number of routes served by planes

Solving It

- Bipartite matching is one of many problems that can be equivalently formulated (and solved) via maximum flows.
- Given $G = (L \cup R, E)$, create flow network $G' = (V', E')$
  - $V' = V \cup \{s, t\}$
  - $E' = \{(s, u) \mid u \in L\} \cup E \cup \{(v, t) \mid v \in R\}$
  - $c(u, v) = 1 \ \forall (u, v) \in E'$

Observations

(You can see the book for more formal proofs)

- There is a matching M in G of size $|M|$ if and only if there is an (integer-valued) flow f in G' of value $|f| = |M|$.
- Thus a maximum-matching in a bipartite graph G is the value of the maximum flow in the flow network G'
- What is a cut in G'?

Next Time

- More applications of Max Flow
- FINALLY Start going over homework and problem sets
- Quiz: April 4
- Programming Quiz: April 23