IE170: Algorithms in Systems Engineering: Lecture 23

Jeff Linderoth

Department of Industrial and Systems Engineering
Lehigh University

March 26, 2007

Taking Stock

Last Time

- All-Pairs Shortest Paths

This Time

- Transitive Closure (Fast)
- Flows in Networks

Transitive Closure

Transitive Closure

- Given directed graph $G = (V, E)$.
- Compute graph $\mathcal{TC}(G) = (V, E^*)$ such that $e = (i, j) \in E^* \Leftrightarrow \exists$ path from i to j in G

- Transitive closure can be thought of as establishing a data structure that makes it possible to solve reachability questions (can I get to x from y?) efficiently. After the preprocessing of constructing the transitive closure, all reachability queries can be answered in constant time by simply reporting a matrix entry.
- Transitive closure is fundamental in propagating the consequences of modified attributes of a graph G.

Applications of Transitive Closure

- Consider the graph underlying any spreadsheet model, where the vertices are cells and there is an edge from cell i to cell j if the result of cell j depends on cell i. When the value of a given cell is modified, the values of all reachable cells must also be updated. The identity of these cells is revealed by the transitive closure of G.
- Many database problems reduce to computing transitive closures, for analogous reasons.
- Doing it fast is important

Transitive Closure Algorithms

1. Perform BFS or DFS from each vertex and keep track of the vertices encountered: $O(V(V+E))$. (Good for sparse graphs)
2. Find Strongly Connected Components. (All vertices in each component are mutually reachable). Do BFS or DFS on component graph. (In which component A is connected to component B if there exists an edge from a vertex in A to a vertex in B)
3. You can use Warshall's Algorithm with weights 1. (In fact you can use "bits" and make things very efficient as well)

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 $s \rightsquigarrow v \rightsquigarrow t \quad \forall v \in V$

Flows

- A positive flow is a function $p : V \times V \rightarrow \mathbb{R}^{|V| \times |V|}$ that satisfies two conditions:

1. Capacity Constraints:

$$0 \leq p(u, v) \leq c(u, v) \ \forall u \in V, v \in V$$

2. Flow Conservation:

$$\sum_{v \in V} p(v, u) = \sum_{v \in V} p(u, v) \ \forall u \in V \setminus \{s, t\}$$

- We will assume that a positive flow either goes from u to v or from v to u but not both.
- If not, we can "cancel" the flow, and preserve the conditions

Net Flows

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

1. Capacity Constraints:

$$0 \leq f(u, v) \leq c(u, v)$$

2. Skew Symmetry:

$$f(u, v) = -f(v, u) \ \forall u \in V, v \in V$$

3. Flow Conservation:

$$\sum_{v \in V} f(u, v) = 0 \ \forall u \in V \setminus \{s, t\}$$

Another way to think of flow conservation:

$$\sum_{v \in V \ | \ f(v,u) > 0} f(v, u) = \sum_{v \in V \ | \ f(u,v) > 0} f(u, v)$$

Different Yet Same

- There are two difference between positive flow p and net flow f
  1. $p(u, v) \geq 0$ (while not true for f)
  2. f satisfies the skew symmetric condition
- However the functions are really equivalent. Given p, define f as

$$f(u, v) = p(u, v) - p(v, u)$$

  This satisfies flow conservation and capacity constraints
- Given f define p as

$$p(u, v) = \begin{cases} f(u, v) & \text{if } f(u, v) > 0 \\ 0 & \text{if } f(u, v) \leq 0 \end{cases}$$

More Flow

- So from here on out, we will use net flow instead of positive 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.

Σ's Scare Me!

- We'll introduce a shorthand notation for summing between sets of vertices.
- Given $X \subseteq V$, $Y \subseteq V$

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

- Therefore flow conservation is

$$f(\{u\}, V) = 0 \quad \forall u \in V \setminus \{s, t\}.$$

Lemma, Lemma, Lemma

- With this shorthand notatation, writing down useful flow properties is easy. Can you prove the following?

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

Example...

A Simple Upper Bound

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

Proof

$$\begin{aligned} f(S,T) &= f(S,V) - f(S,S) & \text{Since } S \cup T = V, S \cap T = \emptyset \\ &= f(S,V) \\ &= f(\{s\},V) + f(S \setminus \{s\}, V) & \text{flow conservation} \\ &= f(\{s\},V) \\ &= |f| \end{aligned}$$

Coronary :-)

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

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

Residual Network

- 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):

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

- 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.

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:

Augmenting Flow Lemma

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 in edge in P_{st}
- (Think of it as a sequence of pipes along which we can quirt more flow from s to t
- How much more?

Augmenting Paths

- How much more?

$$c_f(P_{st}) = \min\{c_f(u,v) \mid (u,v) \text{ is on path } P_{st}\}.$$

- 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 $|f| \le c(S,T)$ (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 $i \leftarrow 1$ **to** n
do $f[u, v] \leftarrow f[v, u] \leftarrow 0$
while $\exists$ 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.

Can We Do Better!?

- Two Smart Guys had the following idea.
- Instead of augmenting on an *arbitrary* augmenting path, why don't we sugment 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.

This/Next Time

- Continuation of TSP lab.
  - Will give you some "test graphs".
  - Will also give you a bit more homework (on max flows)
  - No Late Homework Accepted
- Quiz: April 4
- Programming Quiz: April 23