Network Flow

Examples of network flow problems:

- How much data can we send from one point in the network to another point?

- How much traffic does the freeway system sustain for travel between two cities?

- How profitable could a manufacturing supply chain be?

Def. A flow network is a directed graph $G=(V,E)$ with following features:

- Each edge e has a non-negative capacity c_e
- Has a single source node $s \in V$
- Has a single sink node $t \in V$

Assumptions:

- no edges enter s or leave t
- at least one edge is connected to each node
- all capacities are integers

Notation

We will call $f(e)$ flow through edge e. $f(e)$ has the following properties:

1- Capacity constraint:
for each edge $e \in E$, $0 \leq f(e) \leq c_e$

2- Conservation of flow

$$\sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e) \text{, except for } s \& t$$

We are looking for <u>steady state</u> flow.

<u>Def.</u> For a steady state flow, the value of flow $v(f)$ is defined as follows:

$$v(f) = \sum_{e \text{ out of } s} f(e)$$

Max Flow problem

Given a flow network G, find an s-t flow with max. value.

Def. G_f is the residual graph of G with the following definition:

- G_f has the same set of nodes as G
- for each edge $\underline{e}$ w/ $f(e) < C_e$, we include $\underline{e}$ in G_f with capacity $C_e - f(e)$
- for each edge $\underline{e}$ w/ $f(e) > 0$, we include edge e' (opposite direction to $\underline{e}$) in G_f with $f(e)$ units of capacity

To create G_f:

- if $f(e) = 0$

- if $f(e) = C_e$

- if $f(e) < C_e$

Def. If P is a simple path from $\underline{s}$ to $\underline{t}$ in G_f, then bottleneck(P) is the minimum residual capacity of any edge on P.

Overall strategy to find Max Flow

- Find a path from $\underline{s}$ to $\underline{t}$
- Find the bottleneck value for this path
- Push flow through this path with value equal to bottleneck value
- Repeat!

Augment (f, c, P)
let $b =$ bottleneck (P)
for each edge $(v, u) \in P$
if $e = (v, u)$ is a forward edge,
then
else (v, u) is a backward edge
and let $e = (u, v)$
then
endif
endfor

Return (f')

If f is flow before augmentation, and f' is flow after augmentation, we need to show that if f is a valid flow, then f' will also be a valid flow.

Proof: 1- Check capacity condition

Need to show that for each edge $e \in E$, we have $0 \leq f'(e) \leq c_e$

A- If $\underline{e}$ is a forward edge,

bottleneck$(P) \leq$

B- If $\underline{e}$ is a backward edge,

bottleneck$(P) \leq$

2- Check conservation of flow

Since f is a valid flow, for each node v other than s & t we have:

$$\sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e)$$

Ford-Fulkerson algorithm for Max Flow.

```
MaxFlow (G, s, t, c)
    Initially f(e)=0 for all e in G
    While there is an s-t path in G_f
        let P be a simple s-t path in G_f
        f' = augment (f, c, P)
        f = f'
        update G_f
    endwhile
    Return f.
```

Proof of correctness should include:
- Proof of termination
- Proof that f is a Max Flow.

① While loop terminates

② f is a Max Flow

We first need some definitions

Define a <u>cut</u> : A cut divides nodes in the graph into 2 sets A&B such that $s \in A$ and $t \in B$

FACT: Let f be any s-t flow and (A,B) any s-t cut, then

$$v(f) = f^{out}(A) - f^{in}(A)$$

Ford-Fulkerson terminates when the flow f has no s-t paths in G_f.

Claim: If there is no s-t path in G_f, then there is an s-t cut (A^*, B^*) where

$$v(f) = c(A^*, B^*)$$

Proof: Create sets A^* & B^* such that A^* includes all nodes $\underline{v}$ where there is an s-v path in G_f.

$$B^* = V - A^*$$

Ford-Fulkerson algorithm for Max Flow.

```
MaxFlow (G, s, t, C)
    Initially f(e)=0 for all e in G
    While there is an s-t path in G_f
        let P be a simple s-t path in G_f
        f' = augment (f, C, P)
        f = f'
        update G_f
    endwhile
    Return f.
```

Scaled version of Ford-Fulkerson

Initially $f(e)=0$ for all e in G

Set Δ to be the largest power of 2 that is no larger than the Max. cap. out of s.

While $\Delta \geq 1$

While there is an s-t path in $G_f(\Delta)$

let P be a simple s-t path in $G_f(\Delta)$

$f' =$ augment(f, P)

$f = f'$

update $G_f(\Delta)$

endwhile

$\Delta = \Delta/2$

endwhile

Return f

Background:

- During the Δ-scaling phase, each augmentation increases the flow value by at least Δ.

- Let f be the flow at the end of the Δ-scaling phase.

  Claim: There is an s-t cut (A,B) in G for which $c(A,B) \leq v(f) + m\Delta$

Claim: The number of augmentations in a scaling phase is at most $2m$.

<u>First scaling phase</u>

How many times can we use each edge going out of S?

<u>Other scaling phases</u>

At the end of the Δ-scaling phase, we have already shown that $C(A,B) \leq v(f) + m\Delta$

If f^* is Max flow, Then $v(f^*) \leq C(A,B)$

$\Rightarrow v(f^*) \leq v(f) + m\Delta$

So, in the next scaling phase, where $\Delta' = \Delta/2$ how many iterations can we have?

Scaled version of Ford-Fulkerson

Initially $f(e)=0$ for all e in G

Set Δ to be the largest power of 2

that is no larger than the Max. cap. out of s.

While $\Delta \geq 1$

While there is an s-t path in $G_f(\Delta)$

let P be a simple s-t path in $G_f(\Delta)$

$f' = \text{augment}(f, P)$

$f = f'$

update $G_f(\Delta)$

endwhile

$\Delta = \Delta/2$

endwhile

Return f

strongly versus weakly polynomial
(relevant if input consists of integers)

An algorithm runs in strongly polynomial time if the no. of operations is bounded by a polynomial in the number of integers in the input.

An algorithm runs in weakly polynomial time if the no. of operations is bounded by a polynomial in the number of bits in the input, but not in the number of integers in the input.

Edmonds-Karp

Same as Ford-Fulkerson, except that each augmenting path must be a shortest path with available capacity.

Can be shown to have running time $O(nm^2)$

Ford-Fulkerson	$O(Cm)$	pseudo-polynomial
Scaled version of FF	$O(m^2 \lg C)$	weakly polynomial
Edmonds-Karp	$O(nm^2)$	strongly polynomial
Orlin + KTR	$O(nm)$	〃 〃

Recently developed methods solve max flow in close to linear time WRT m.

approximation

Discussion 8

1. You have successfully computed a maximum s-t flow f for a network G = (V; E) with integer edge capacities. Your boss now gives you another network G' that is identical to G except that the capacity of exactly one edge is decreased by one. You are also explicitly given the edge whose capacity was changed. Describe how you can compute a maximum flow for G' in O(|V| + |E|) time.

2. You need to transport iron-ore from the mine to the factory. We would like to determine how long it takes to transport. For this problem, you are given a graph representing the road network of cities, with a list of *k* of its vertices ($t_1, t_2, \ldots, t_k$) which are designated as factories, and one vertex *S* (the iron-ore mine) where all the ore is present.
We are also given the following:

- Road Capacities (amount of iron that can be transported per minute) for each road (edges) between the cities (vertices).
- Factory Capacities (amount of iron that can be received per minute) for each factory (at $t_1, t_2, \ldots, t_k$)
- The amount of ore to be transported from the mine, *C*

Give a polynomial-time algorithm to determine the minimum amount of time necessary to transport and receive all the iron-ore at factories.

3. In a daring burglary, someone attempted to steal all the candy bars from the CS department. Luckily, he was quickly detected, and now, the course staff and students will have to keep him from escaping from campus. In order to do so, they can be deployed to monitor strategic routes.

More formally, we can think of the USC campus as a graph, in which the nodes are locations, and edges are pathways or corridors. One of the nodes (the instructor's office) is the burglar's starting point, and several nodes (the USC gates) are the escape points — if the burglar reaches any one of those, the candy bars will be gone forever. Students and staff can be placed to monitor the edges. As it is hard to hide that many candy bars, the burglar cannot pass by a monitored edge undetected.

Give an algorithm to compute the minimum number of students/staff needed to ensure that the burglar cannot reach any escape points undetected (you don't need to output the corresponding assignment for students — the number is enough). As input, the algorithm takes the graph G = (V,E) representing the USC campus, the starting point s, and a set of escape points $P \subseteq V$. Prove that your algorithm is correct and runs in polynomial time.

4. We define a most vital edge of a network as an edge whose deletion causes the largest decrease in the maximum s-t-flow value. Let f be an arbitrary maximum s-t-flow. Either prove the following claims or show through counterexamples that they are false:

(a) A most vital edge is an edge e with the maximum value of c(e).
(b) A most vital edge is an edge e with the maximum value of f (e).
(c) A most vital edge is an edge e with the maximum value of f (e) among edges belonging to some minimum cut.
(d) An edge that does not belong to any minimum cut cannot be a most vital edge.
(e) A network can contain only one most vital edge.