CS314 - Network Flow

Note Title 4/14/2010

Announcements

- Turn in HW
- Next HW is posted (written, so due next Wed.)
- Office hours tomorrow changed
  - 10(ish) to noon

(Chapter 7 of book)

Goal: Model transportation networks
(from "Secret" government pub in 1955)

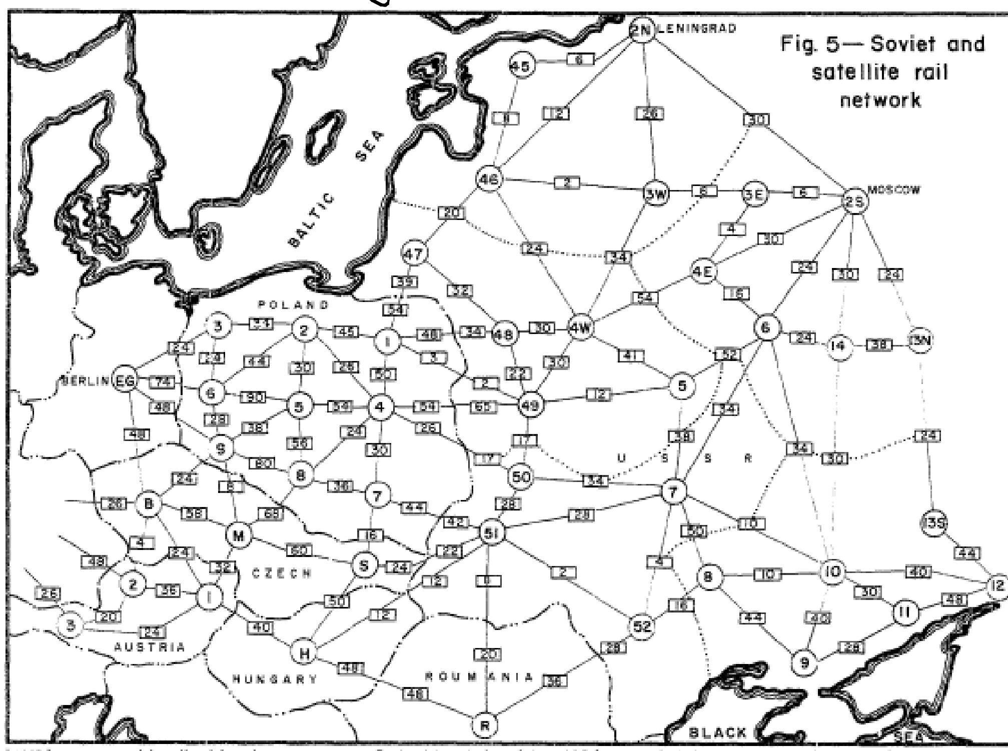


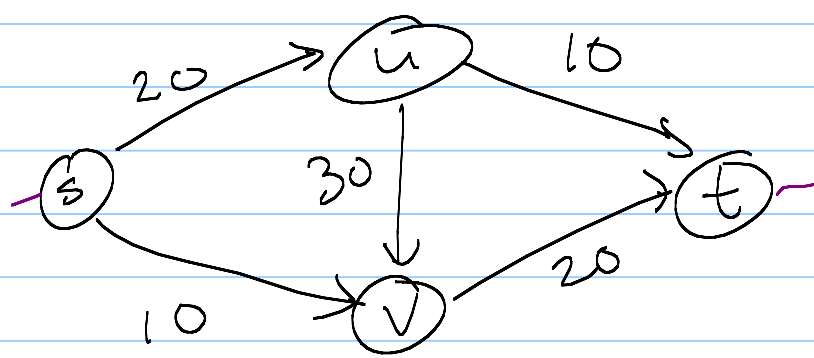
Fig. 5— Soviet and satellite rail network

Legend: International boundary ········· Regional boundaries of the USSR (they are included as a matter of general information)

(7) Operating divisions. Those located in Russia are believed to be accurately located. Some Russian divisions (2, 3, 4 and 13) are located in two regions and are so indicated. Divisions shown in the satellites are indicated according to the authors' best judgment, since intelligence reports are unavailable. Train capacities in Russia are for 1000-net-ton trains or their equivalent. Train capacities in Poland are for 666 net tons (or the equivalent). Train capacities in all other satellites are for 400 net tons (or the equivalent) except in East Germany. In East Germany, train capacities are those of entering lines. The numbers shown in boxes are total interdivisional capacities.

More formally:

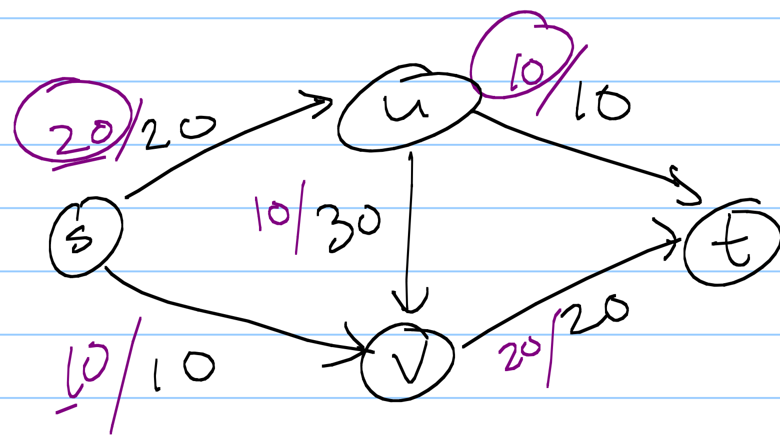
- A directed graph $G=(V,E)$
- Each edge has a maximum capacity c_e
- Two special vertices $s,t \in V$
  - s is the source
  - t is the sink



Note: s has no incoming edges, + t has no outgoing

Think of edges as pipes, roads, network connections, etc ...

Goal is to "push" as much flow from s to t.



$v(f) = 20 + 10$

$= 30$

not a flow

Formally:

A flow is a function $f: E \to \mathbb{R}^+$
(some amount sent along each edge)
such that:

① • capacity constraint: $\forall e \in E$, $\underline{0 \le f(e) \le c_e}$

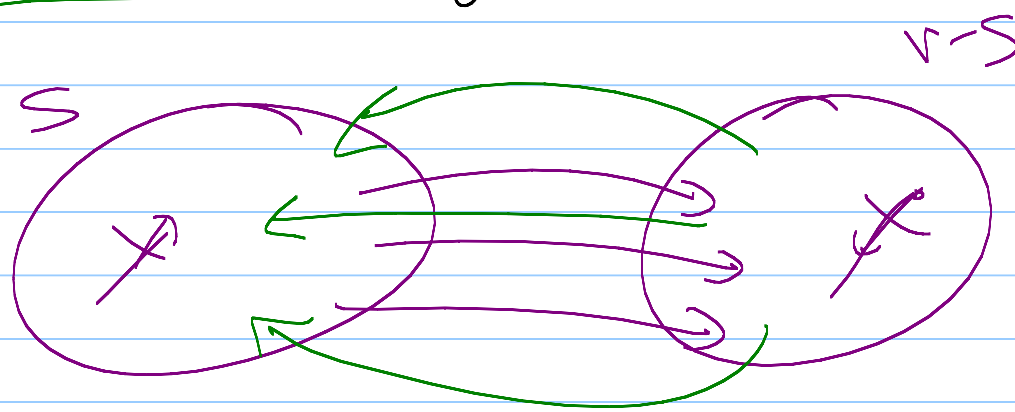
② • conservation constraint: $\forall v \in V$, if $\underline{v \ne s \text{ or } t}$,

$$\underbrace{\sum_{e \text{ into } v} f(e)}_{= f^{in}(v)} = \underbrace{\sum_{e \text{ out of } v} f(e)}_{= f^{out}(v)}$$

Notation: for any $S \subseteq V$,

$$f^{out}(S) = \underbrace{\sum_{e \text{ out of } S} f(e)}$$

$$\left(\& \ \underbrace{f^{in}(S)} \text{ similarly} \right)$$



Maximum Flow Problem

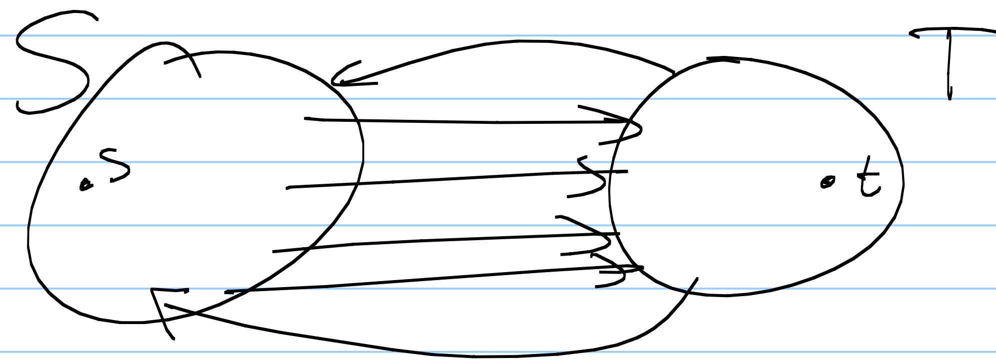
- The value of a flow is $\sum_{e \text{ out of } s} f(e) = \sum_{e \text{ into } t} f(e)$ $v(f)$

Goal: Find flow with maximum value.
(Arrange the traffic as efficiently as possible.)

Basic obstacle

For any $S \subseteq V$ with $s \in S$, $t \in V - S = T$,

all flow must leave S & enter T.



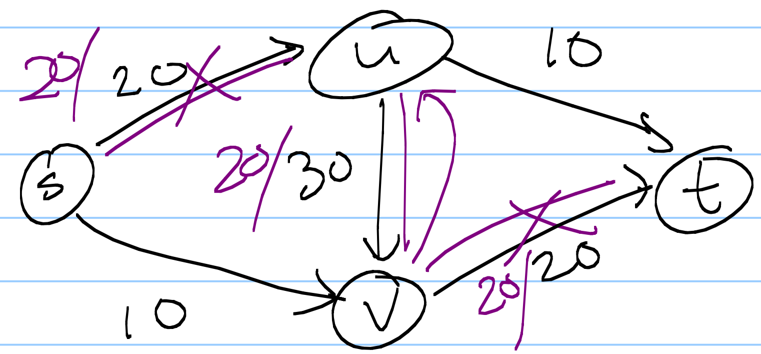
so flow $\leq$ sum of edge capacities from S to T

(this is called (S,T)-cut)

Computing Flow

Ideas?

find an s to t path + push as much flow as we can



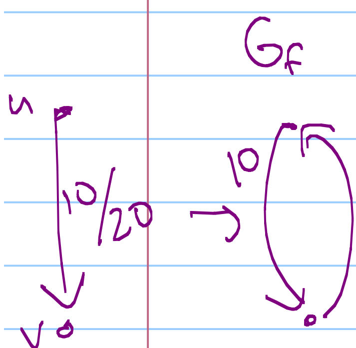
Now - no s to t paths

Problem: We can get stuck!

So we may need to "unpush" flow.

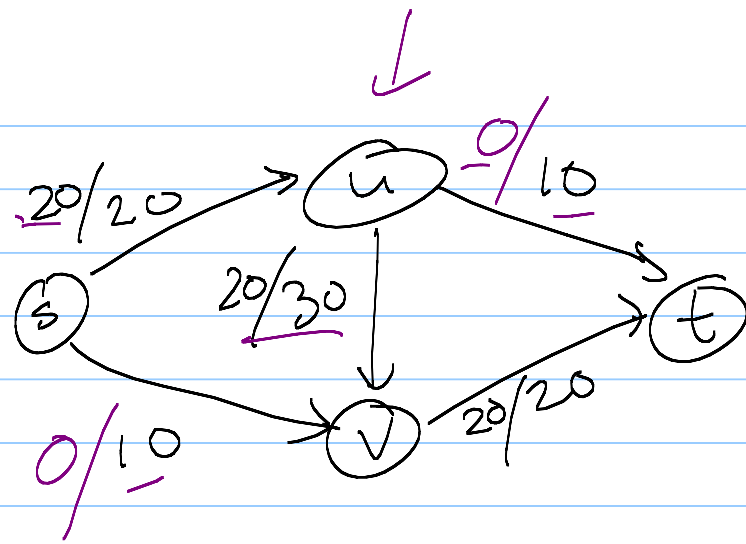
Dfn: The residual graph G_f of G with respect to a flow f is a graph with:

- G_f has same vertex set as G



- For each edge (u,v) in G with $f(uv) < c_{uv}$, add an edge to G_f from u to v with weight $c_{uv} - f(uv)$
- If $f(e) > 0$ where $e = (u,v)$, add edge vu in G_f of value $f(e)$

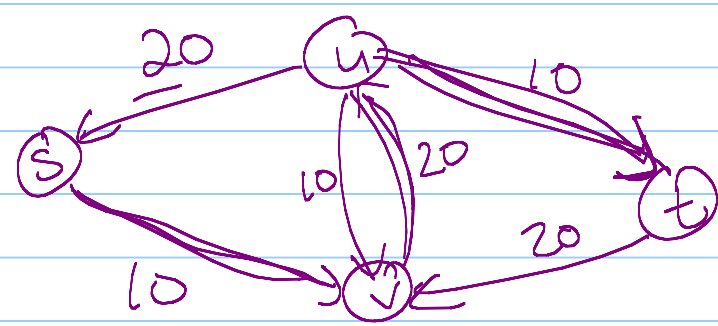
Ex:



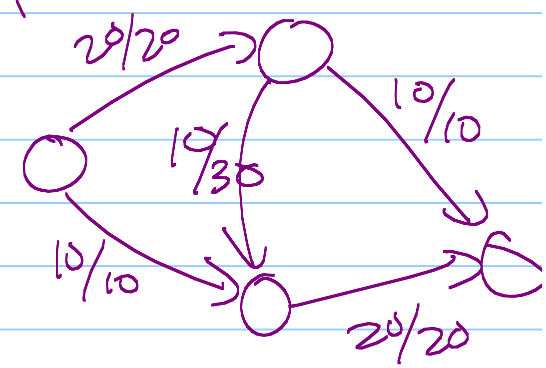
G & f

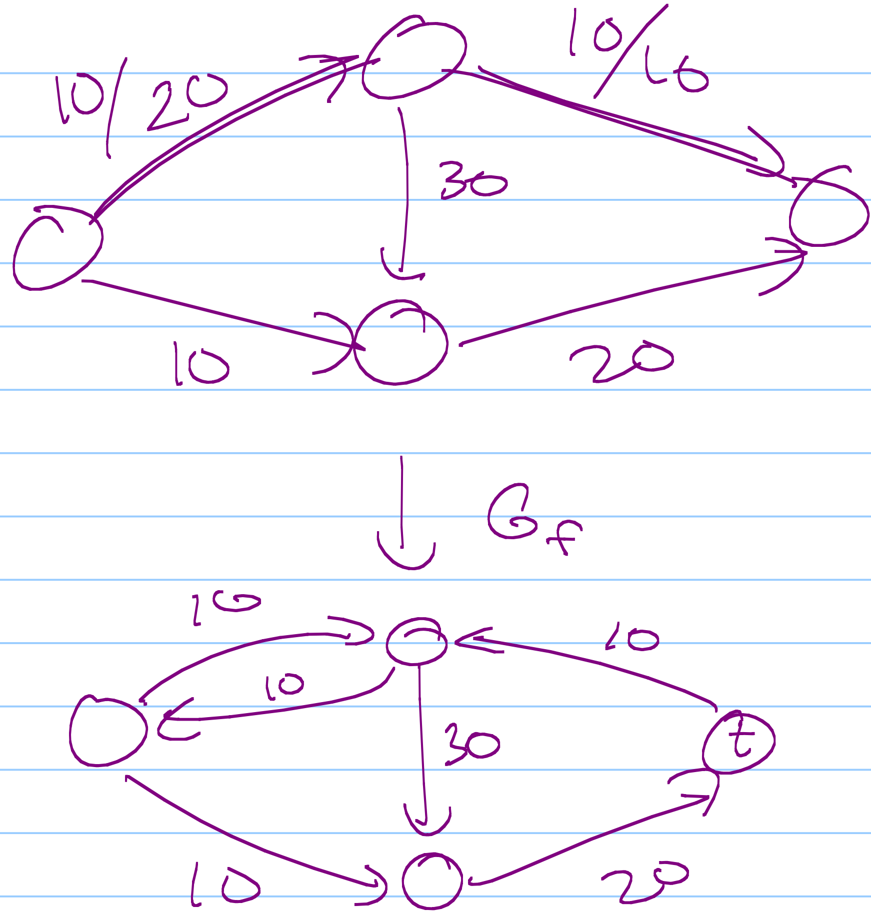
$\downarrow$

G_f



push along s-to-t path



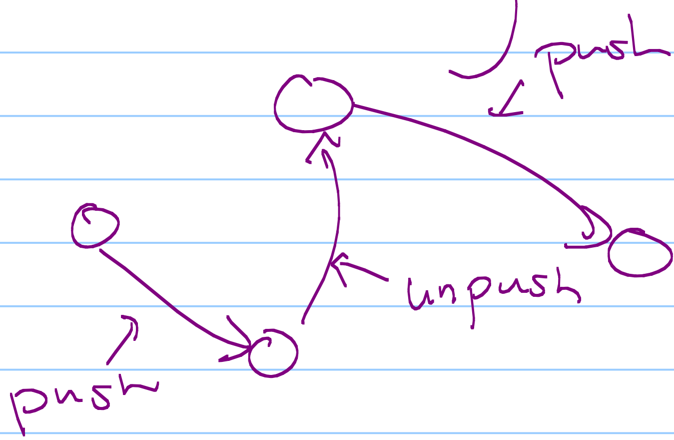


u

v

So G_f does have an s-to-t path!
(Notice that path "unpushes" some flow.)

We find s-to-t path in G_f,
+ either increase or decrease
flow along each edge in that path

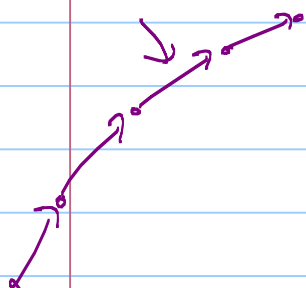


Claim: New flow f' is a valid flow.

pf: Need to verify 2 things:

① capacity constraint: only changed flow for edges on path P.

let $e = (u,v) \in P$.



w(bottleneck edge on P) $\leq (c_e - f(e))$

Let bottleneck edge in P = min weight edge of P in G_f.

So adding w(bottleneck) to every edge in P cannot exceed capacity of each edge.

(2) conservation

before, in = out for every vertex.

The only flow change is along P.

$$\xrightarrow[+10]{P} \; v \; \xrightarrow[+10]{P}$$

Change each vertex in P along 2 of its edges, by the same amount,

So flow in is still = flow out.

Our Algorithm: [Ford-Fulkerson 1956]

- Find a path from s to t in G_f
- Push flow along s-to-t path
- Repeat until G_f contains no s to t paths

Pseudocode

Maxflow(G):

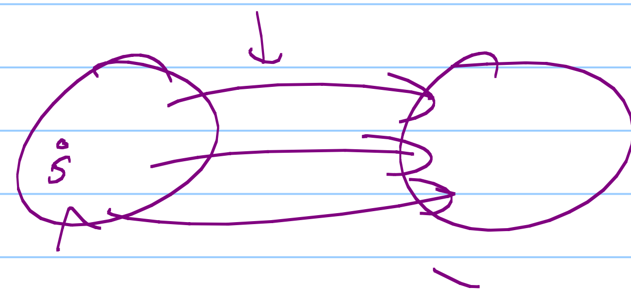
```
f(e) ← 0   ∀e ∈ E
G_f ← G
while there is an s-t path in G_f
    Let P ← s-t path in G_f
    f' = Augment(f, P)
    f ← f'
    Update G_f
return f
```

Augment (f, P):
    $b \leftarrow$ bottleneck edge of P
    for each edge $(u,v) \in P$
        if $e = (u,v)$ is forward in G
            $f(e) \leftarrow f(e) + b$
        if $e = (u,v)$ is backwards in G
            $f(e) \leftarrow f(e) - b$
    return f

We know this returns a valid flow, but haven't shown it returns the *maximum* flow.

Dfn: An *s-t cut* is a partition of V into 2 sets (S, T) with $s \in S$, $t \in T$.

The *capacity* of a cut $c(S,T) = \sum_{e \text{ out of } S} c_e$



Strategy: 2 things

① Thm: Let f be any s-t flow, and (S,T) any s-t cut.

Then $v(f) \le c(S,T)$

② Given a flow f where there is no s-to-t path in G_f, we can find a cut (S^*,T^*) with:

$$v(f) = c(S^*,T^*).$$

max flow   min cut

First, a lemma:

Lemma: Let f be any s-t flow, and (S,T) any S-T cut. Then $v(f) = f^{out}(S) - f^{in}(S)$.

pf: