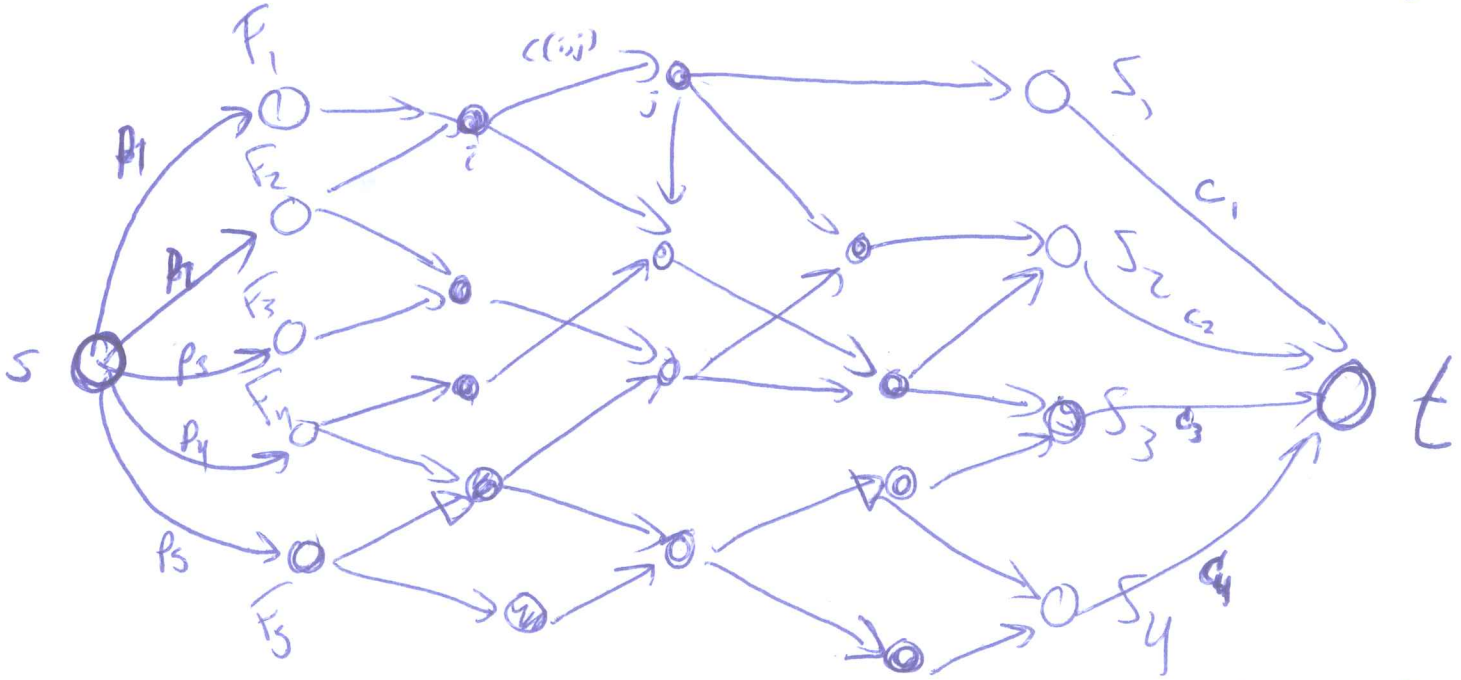
Chapter 3. Maximum Flow Problems

A company owns factories $F_1, \dots, F_n$ that each make the same item which is sold at stores $S_1, \dots, S_m$.

The holiday's newest toy!

Problem: Factory F_i can only make p_i toys per day,
Store S_j can only sell c_j toys per day,
AND we need to get the toys to the stores!
Every route has a finite capacity:
City i to city j has capacity $c(i,j)$ toys per day.



Want to maximize # of toys sold without saturating the market, overtaxing a factory, or overloading a shipping route.

Other Applications:

Internet Streaming
Water Flow
Traffic?
Trains, Planes, ...

Focus on Trucks hauling toys. From r (source) to s (sink). Each truck T_ℓ is given a route, i.e. an r,s-path P_ℓ, to follow. Let $P_1, \dots, P_t$ be a list of routes for t trucks.

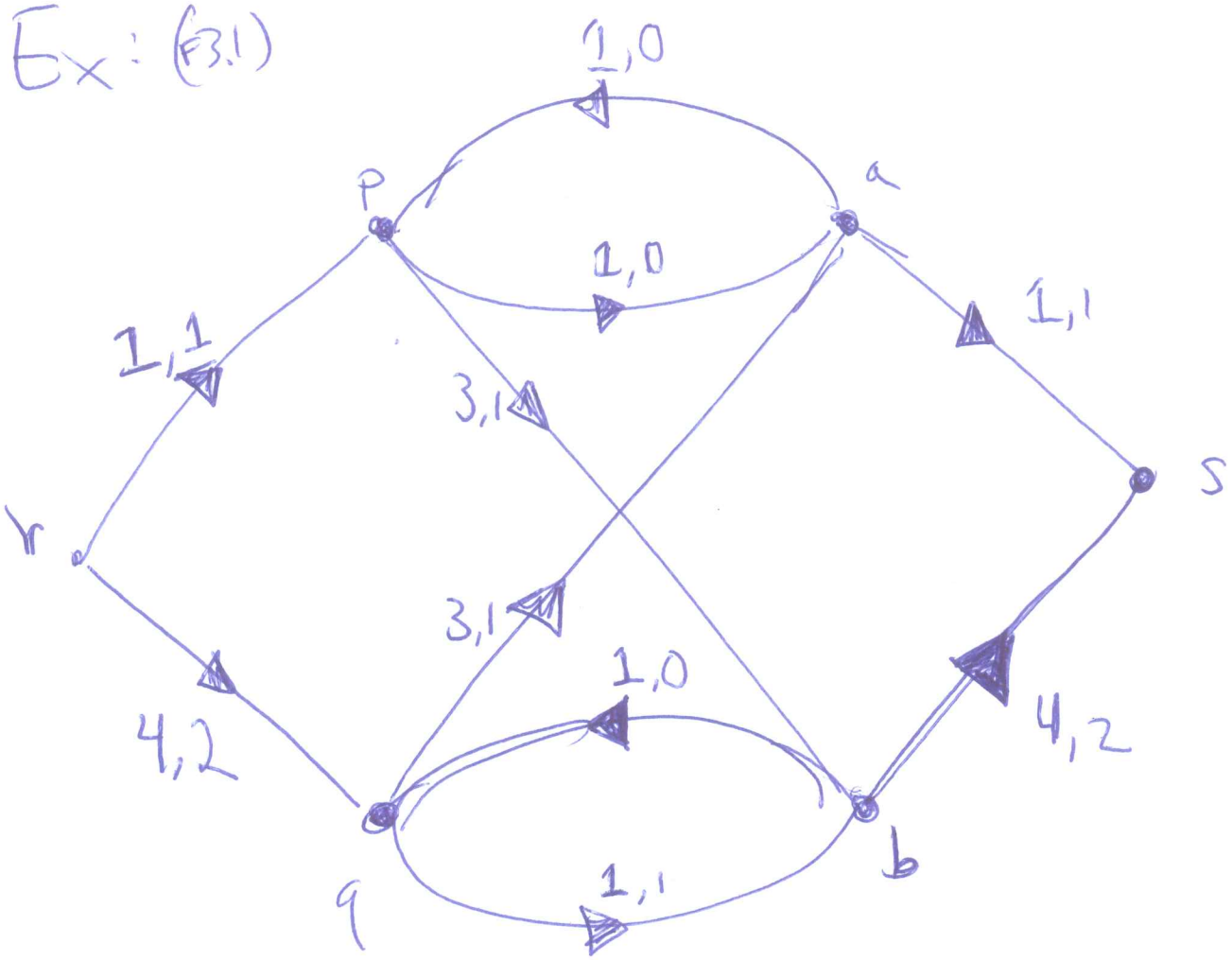
Every vertex $v \notin \{r, s\}$ has the same # of incoming & outgoing edges. So, if we let x_{uv} = # of paths using edge $uv \in E$, then

(Conservation) $\underbrace{\sum_{u : uv \in E} x_{u,v} - \sum_{w : vw \in E} x_{v,w}}_{f_x(v)} = 0 \quad (v \notin \{r, s\})$

Also every edge uv has a capacity $c(u,v) \geq 0$

If we are sending discrete trucks, then $x_{u,v} \geq 0$ is integer.

Ex: (F3.1)



$P_1 =$ rpbs

$P_2 =$ rqbs

$P_3 =$ rqasb

Path Packing Problem:

Maximize t such that t r-s-paths $P_1, \dots, P_t$ exist such that every edge $x_{u,v}$ is used in at most $c(u,v)$ of them.

?

gives a solution.

Integral Max Flows Problem:

$$\max \quad f_{\underline{x}}(s)$$

$$\text{s.t.} \quad f_{\underline{x}}(v) = 0 \qquad v \notin \{r,s\}$$

$$0 \le x_{u,v} \le c(u,v) \qquad uv \in E(G)$$

$$x_{u,v} \quad \text{integer.}$$

Feasible solution: an (integral) feasible (r,s)-flow of value $f_{\underline{x}}(s)$

Prop. 31: There exists a family $(P_1, \dots, P_t)$ of r-s-paths such that $|\{i : P_i \text{ uses } e\}| \le c(e)$ for all $e \in E$ if and only if there exists an integral feasible (r,s)-flow of value t.

$$\max \sum_{\substack{P_j \\ \text{is a path}}} x_{P_j}$$

$$\text{s.t.} \sum_{P_j \ni e} x_{P_j} \le c(e) \quad \forall e$$

$$x_{P_j} \ge 0, \quad \text{integer.}$$

Pf of 3.1: ($\Rightarrow$) Assign $x_e = \sum_{P_j \ni e} x_{P_j}$. ✓

($\Leftarrow$) ~~If there exists s-path~~ Strengthened Induction: $\exists$ a path packing w/ $\sum_{P_j \ni e} x_{P_j} \le x_e$

Induct on $t \ge 0$, secondarily on $\sum x_e$

If $t = 0$, then assigning $x_{P_j} = 0$ also has value 0.

Now, let $t > 0$. We must show that some rs-path P_j has

$$\min \{x_e : e \in P_j\} \ge 1.$$

Since $\text{val}(f_{\underline{x}}) = t$, we have

$$t = f_{\underline{x}}(s) = \sum_{v : sv \in E} x_{v,s} - \sum_{w : sw \in E} x_{s,w} = t$$

We will search backward along edges with positive value, until reaching either r (and we have an rs-path) or a vertex already in the walk.

We will terminate this way, due to conservation constraints and G is finite.

1. We reach r. Then our walk (in reverse) is an rs-path P_j.

   Assign $x_{P_j} = \min \{x_e : e \in P_j\}$.

   Since all e's have real positive, integer value, x_{P_j} is positive, integer value.

   Define $x'_e = \begin{cases} x_e - x_{P_j} & e \in P_j \\ x_e & \text{otherwise.} \end{cases}$

   $\underline{x}'$ is an integer feasible flow w/ $f_{\underline{x}'}(r) = t - x_{P_j} < t$.

   Induct to find a path packing of value $t - x_{P_j}$, and x_{P_j} to get a path packing of value t.

   ↑ True by strengthened induction! (Capacities are not violated)

2. If we reach a vertex v already in our walk, then we have a cycle!

   Remove value from the cycle. □

Skolem: Every s-t flow of nonnegative value is the sum of at most $m = |E|$ flows, each of which is a path flow, or a circuit flow (cycle flow).

Duality (Combinatorial Perspective)

"Duality is how competing entities find an equilibrium solution."

Max Flow: A terrorist communication network is modeled as a graph, where the leader sends orders down through a network. He wants to send as much information, but certain links are less secure than others. More secure ≡ Higher Capacity.

Dual: The NSA has wire tapped to discover the entire communication network, and wants to shut it down. It wants to remove communication links subject to

More Secure ≡ Higher Cost to Shut Down.

Min Cut:

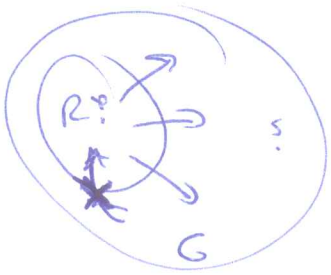
$$\min \sum_{e \in E} c(e)\, y_e$$

$$\text{s.t.} \sum_{e \in P_j} y_e \geq 1$$

$$y_e \geq 0 \quad \text{(integer)}$$

Recall Edge-Centric flows, Path-Centric Flows, and Edge-based Cuts.

For $R \subseteq V$, $\delta(R) = \{uv : u \in R,\ v \notin R\}$



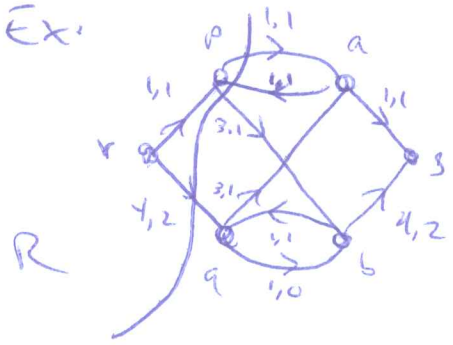
Edges cut of R, not into R.

$\delta(R)$ is an rs-cut if $r \in R$, $s \notin R$.

Prop 3.3. For any rs-cut $\delta(R)$ and any rs-flow $\underline{x}$, we have

$$x(\delta(R)) - x(\delta(\bar{R})) = f_{\underline{x}}(s).$$

Ex:



$R = \{r, p\}$,
$\delta(R) = \{pa, pb, rq\}$
$\delta(\bar{R}) = \{ap\}$

Pf: Add the equations $f_{\underline{x}}(v) = 0$ for $v \in \bar{R} \setminus \{s\}$ and the identity $f_{\underline{x}}(s) = f_{\underline{x}}(s)$.

RHS: $f_{\underline{x}}(s)$.

LHS: All edges with both ends in $\bar{R}$ cancel out.
What is left: $+1 \cdot x_{u,v}$ if $uv \in \delta(R)$
$-1 \cdot x_{u,v}$ if $uv \in \delta(\bar{R})$. □

Given an rs-cut $\delta(R)$, the capacity of the cut is

$$c(\delta(R)) = \sum_{uv \in \delta(R)} c(u,v).$$

Cor 3.4: For any feasible s-t flow x, and any s-t cut $\delta(R)$, we have

$$f_x(s) \le c(\delta(R)).$$

(Weak Duality!)

Pf: $f_x(s) = x(\delta(R)) - x(\delta(\bar{R})) \le x(\delta(R)) \le c(\delta(R))$. □

Thm 3.5: (Max-Flow/Min-Cut Theorem):

If there is a maximum s-t flow, then

$$\max\{f_x(s) : x \text{ a feasible } s\text{-}t \text{ flow}\} = \min\{c(\delta(R)) : \delta(R) \text{ is an } s\text{-}t \text{ cut}\}.$$

Our proof is to give an algorithm that solves both problems simultaneously.

Idea: Start w/ a feasible s-t flow x

Improve it by sending flow along an x-augmenting path

If you cannot improve it, you have an s-t cut that matches!

Let $P = v_1 \cdots v_k$ be an undirected path in G. (It can use edges in either direction.)

P is x-incrementing if for every edge $v_i v_{i+1}$ in P, we have:

if $v_i v_{i+1}$ is a forward edge in G, then $x_{v_i, v_{i+1}} < c(v_i, v_{i+1})$.

if v_i, v_{i+1} is a backward edge in G, then $x_{v_{i+1}, v_i} > 0$.

P is x-augmenting if it is an x-incrementing s-t path.

3 things:

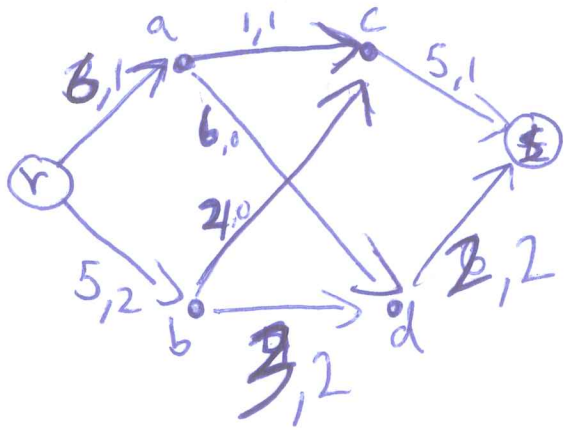
1. What are augmenting/incrementing paths?
2. How do we use one to modify/improve a flow?
3. How do we find one?

1. Recall definitions: χ-incrementing,
   χ-augmenting
   $\varepsilon = \varepsilon(P) \leftarrow \chi$-width of P.

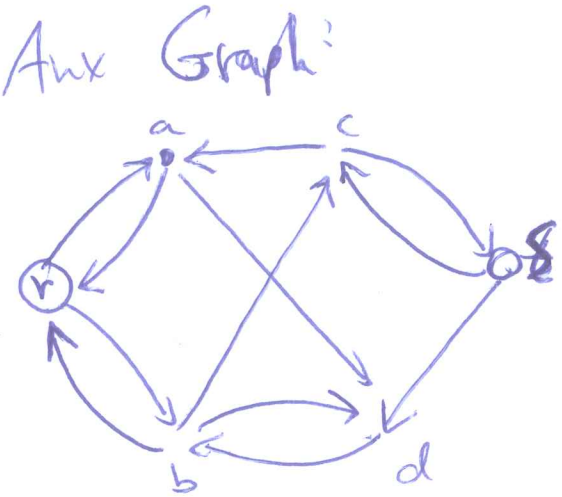
2. Forward Edges: $+\varepsilon$
   Backward Edges: $-\varepsilon$

3. Build Auxiliary Digraph.

Ex:

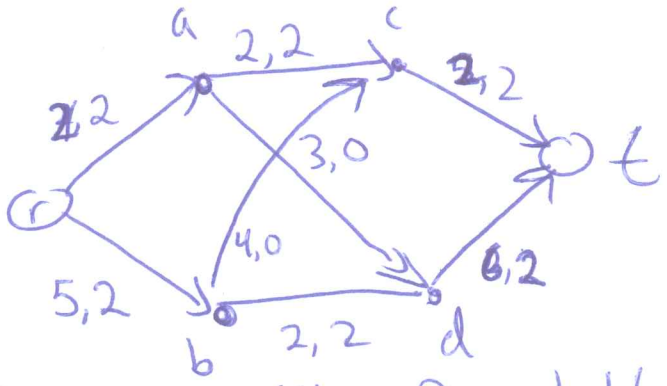


Aux Graph:



rs-Paths: radbcs, rbcs

Ex!



$P_1 = racte$ (2) $P_2 = rbdte$ (2)

$P_3 = rbcadte$ (2)

Let $\varepsilon = \varepsilon(P) = \min\left\{ c(u,v) - x_{u,v} : uv \text{ is } \underline{\text{forward}} \right\} \cup \left\{ x_{u,v} : vu \text{ is backward} \right\}$.

We augment x by some ε "along" P.

Restate MFMC: If max flow, then max flow = min integer cut.

Pf of Max Flow/Min Cut: Let x be max flow.

No x-augmenting path exists!

Let $R = \{v \in V : \exists \text{ an } x\text{-incrementing path from } r \text{ to } v\}$.

Then $c(\delta(R)) = x(\delta(R))$

and $c(\delta(\bar{R})) = 0$.

Thus, $f_x(s) = \underline{c(\delta(R))}$. □

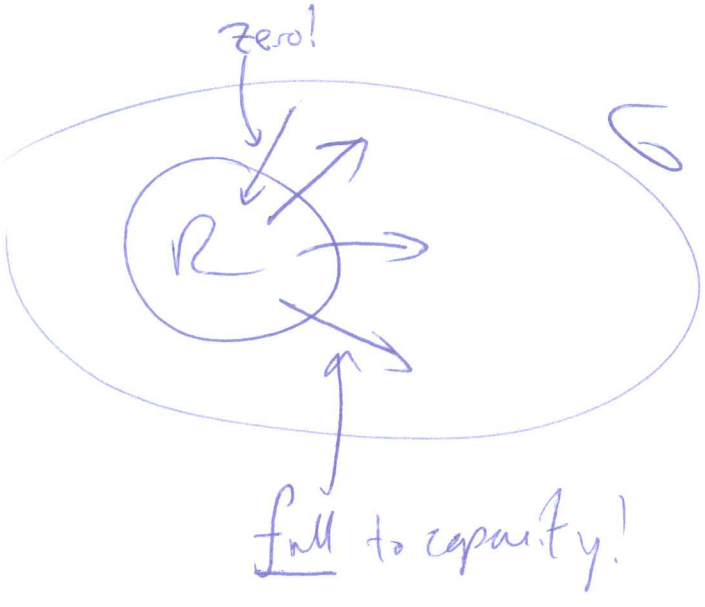
Cor: A flow x is maximum iff no x-augmenting path exists.

Cor: If all edge capacities have integer values, then there exists a maximum flow with integer values.

Pf: All $\varepsilon(P)$ are integers!

Cor: If $\underline{x}$ is a feasible rs-flow, and $\delta(R)$ is an rs-cut, then $\underline{x}$ is <u>maximum</u> if and only if

$$x_e = c(e) \quad \forall e \in \delta(R) \quad \text{and} \quad x_e = 0 \quad \text{for all } e \in \delta(\bar{R}).$$



Pf: Recall: $f_{\underline{x}}(S) = \underline{x}(\delta(R)) - \underline{x}(\delta(\bar{R})) = c(\delta(R))$.

So, $x(\delta(R)) = c(\delta(R))$

and $x(\delta(\bar{R})) = 0$. (by $x_e \geq 0$). □

Augmenting Paths Algorithm:

(BFS on Auxiliary Digraph).

Augmenting Path Algorithm (Ford-Fulkerson)

Init: Let $\underline{x} = \underline{0}$, the zero flow.

Iteration: Build Auxiliary digraph $G(\underline{x})$:

$V(G(\underline{x})) = V(G)$.

$E(G(\underline{x})) = \{uv : uv \in E \text{ and } x_{u,v} < c(uv)\} \cup \{uv : vu \in E \text{ and } x_{u,v} > 0\}$.

* Find an rs-path P in $G(\underline{x})$.
  - Compute $\varepsilon = \varepsilon_{\underline{x}}(P)$ (the $\underline{x}$-width of P)
  - For all $uv \in P$,
    - Set $x_{uv} \leftarrow x_{u,v} + \varepsilon$ if uv is forward.
    - Set $x_{vu} \leftarrow x_{v,u} - \varepsilon$ if uv is backward.
  - If no path, then output $\delta(R)$.

* Breadth-First Search:

Init: $R = \{r\}$, $S = \emptyset$

$d(r,v) = \infty$ for all $v \neq r$

$d(r,r) = 0$

$p(r) = r$, $p(v) = \emptyset$ otherwise

While $S \neq R$:

- Select $v \in R \setminus S$, minimizing $d(r,s)$
- $S \leftarrow S \cup \{v\}$.
- for all u s.t. $vu \in E$:
  - if $u \notin R$, then
    - $R \leftarrow R \cup \{u\}$
    - $d(r,u) = d(r,v) + 1$.
    - $p(u) = v$

If $s \in R$, then output reverse($s, p(s), p(p(s)), \ldots, r$)

Else output $\delta(R)$.