Flows and cuts:

we have a graph $V = \{\text{vertices}\}$
$E = \{\text{edges}\}$

make this graph <u>directed</u>

there is a unique vertex s (the source): this has only outgoing edges

and d (the drain): this has only incoming edges

- each edge has an associated <u>capacity</u> (which is integer, ≥ 0)
- flow through the edge may not exceed its capacity
- count incoming flows +ve, outgoing -ve
- Kirchoff's law applies for all but s and d

$$\sum_{e \in e(v)} f_e = 0$$

<u>Problem</u>: what is the maximum flow from s to d. ?

- many practical applications
- many algorithms follow.

<u>Augmenting path</u>:

- a path from s to d
    - undirected
    - all forward ($s \to d$) arrows below capacity
    - all backward ($d \to s$) arrows greater than zero

1) if an augmenting path exists, flow is not maximal
<u>Because</u> we can increase flow along this path.

2) if flow is maximal, there is no A.P.
<u>Because</u> if there were, we could augment

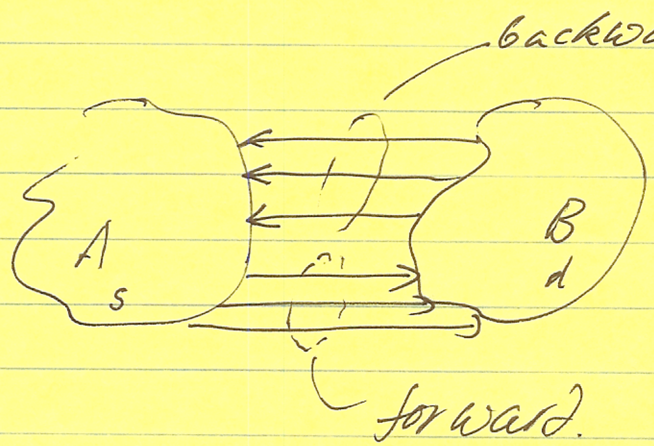
Algorithm outline

- find augmenting path
- max flow on this path

- Efficiency depends on details of how path is found

Now, consider a disconnecting cut

- set of edges that disconnects s, d
- divide verts into A, B st $A \cup B = V$ and $A \cap B = \emptyset$ and $s \in A$, $d \in B$



- total flow $A \to B = \Sigma \text{forward} - \Sigma \text{backward}$

Value of the cut

$= \sum$ capacities forward

~~$- \sum$ capacities backward~~

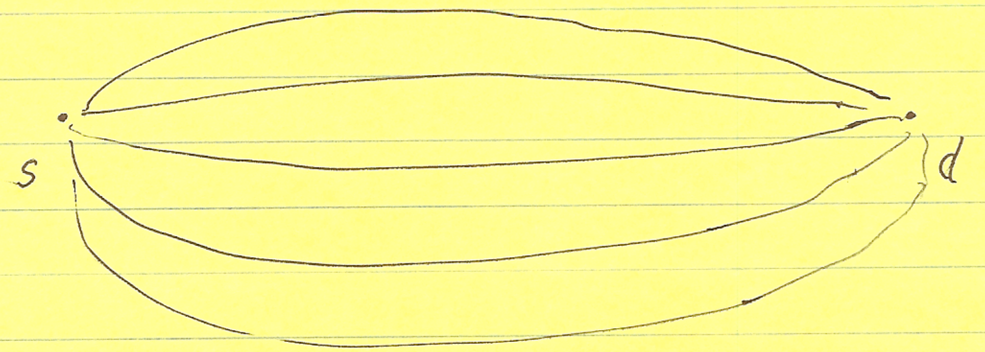
$\Rightarrow$ If flow is maximal, there is at least 1 cut so that

- all forward are at capacity
- all backward are zero

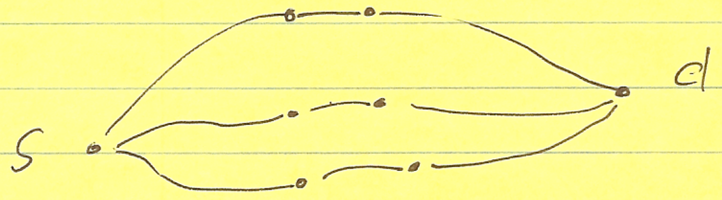
$\Leftarrow$ if such a cut exists, flow is maximal.

(Easy; there can be no augmenting path) $\square$

$\Rightarrow$: consider all paths, $s \to d$



- Each has, at least, either an edge which is forward and at capacity or backward and at zero



(there might be more than one)

Now cut each path at some such edge.

→ is this a cut?

<u>No</u>: Because some verts could appear on both s-side and d side

<u>But</u>: we can get a cut out of it

assume $v_1 \in V(s)$, and $v_1 \in V(d)$

can't move from $V(d)$ to $V(s)$

<u>because</u> otherwise there would be an augmenting path □

Notice

Value of this cut

= Max flow.

But there cannot be a cut with <u>lower</u> value, because then flow would be smaller.

(Getting a min-cut from a max-flow:

⇒ wasn't constructive (cycles)

cut all capacity, 0 edges.

→ if this is a cut, stop

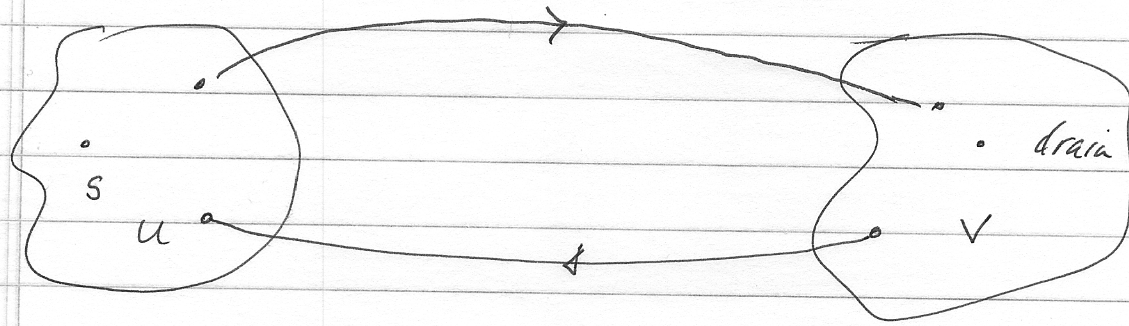
→ otherwise, put other cc's into V(s), V(d) at will

→ restore edges in V(s), V(d)

→ this is a cut)

Min-cut = easy 0-1 QP

- Consequence - if we have an easy 0-1 QP, we can do it with any fast min-cut alg we have.



have x, 1 per vertex:

U: $x_i = 0$
V: $x_i = 1$

$$\text{value of cut} = \sum_{\substack{a \in \text{edges} \\ \text{from} \\ U \text{ to } V}} c_a$$

$$= \sum_{\text{all edges}} (1 - x_i)(x_j) \cdot c_{i \to j}$$

So we have

$$\min \sum_{i,j} (1 - x_i)(x_j)\, C_{i \to j}$$

$$\text{s.t.} \quad x \in \{0, 1\}$$

<u>But</u> this is

$$\max \sum_{i,j} (x_i - 1)\, x_j\, C_{i \to j}$$

$$\text{st} \quad x \in \{0, 1\}$$

<u>and we have</u>

$$\max \quad x^T \underline{A}\, x \quad + \quad b^T x$$

↑
non neg

$$\text{st} \quad x \in \{0, 1\}.$$

Goldberg - Tarjan Push - Relabel:

- One of many alg's, v. good complexity properties

- Idea: - use infeasible flows ("preflows")
  - keep approximate track of path length to t
  - make flows "more feasible"

- Defns:
  - work on $(V, V \times V)$

    ↳ i.e. all directed edges

  - any edge not in G gets $c = 0$

  - flow is

$$f : V \times V \to \mathbb{R} \quad \text{st}$$

$$f(v \to w) \leq c(v \to w)$$

$$f(v \to w) = -f(w \to v)$$

$$\sum_u f(u \to v) = 0 \quad \text{for all } v \in V - \{s, t\}$$

<u>The value of a flow is</u>

$$\sum_v f(v \to t)$$

- a <u>preflow</u> is $f : V \times V \to R$

  such that
  $$f(v \to w) \leq c(v \to w)$$
  $$f(v \to w) = -f(w \to v)$$
  $$\sum_u f(u \to v) \geq 0$$

  $\sum_u f(u \to v) = e(v)$ is the <u>excess</u>

- we must push excess to the target
- residual capacity for some preflow f is

  $$r_f(v \to w) = c(v \to w) - f(v \to w)$$

  if $r_f(v \to w) > 0$, can move flow through this edge.

Label function

$$d : V \to N_0 \cup \{\infty\}$$

is a <u>valid labelling</u> ~~f~~for f if

$$d(s) = |V| \quad , \quad d(t) = 0$$

$$d(v) \leq d(w) + 1 \qquad \text{for } v \to w \text{ st}$$

$$r_f(v \to w) > 0$$

(Notice gives an approx of "dist" to t along paths where we can push

Algorithm:

- initialize at preflow and valid labelling

eg $f(s \to v) = c(s \to v)$ for all v

$f(u \to v) = 0$ for all $u \neq s, v$

$$d(s) = |V| \;,\; d(t) = 0 \;,\; d(v) = 0 \;,\; v \in V - \{s, t\}$$

Algorithm:

- initialize w/ preflow and valid labelling

  eg. $f(s \to v) = c(s \to v)$ for all v

  $f(u \to v) = 0$ for all $u \neq s$, v

  $d(s) = |V|$, $d(t) = 0$, $d(v) = 0$, $v \in V - \{s, t\}$

- while there is an active vertex

[active is ($v \neq s, t$, $e(v) > 0$, $d(v) < \infty$)]

  - choose one and execute an admissible operation for that vertex.

Operations are:

Push: (on an edge $v \to w$)

- admissible if v is active
  and $r_f(v \to w) > 0$
  and: $d(v) = d(w) + 1$

- $\delta = \min\{ e(v),\ r_f(v \to w) \}$

- $f(v \to w) = f(\cancel{w} \to w) + \delta$
  $f(w \to \cancel{v}) = f(w \to v) - \delta$

$r_f(v \to w) = r_f(v \to w) - \delta$
$r_f(w \to v) = r_f(w \to \cancel{v}) + \delta$

$e(v) \to e(v) - \delta$

$e(w) \to e(w) + \delta$

Relabel:

- admissible if v is active,
  and if $r_f(v \to w) > 0$
  always implies $d(v) \leq d(w)$

$$d(v) = \min\{d(w) + 1\} \text{ over edges } r_f(v \to w) > 0$$

Facts

- ~~If~~ f is a preflow, d a valid labelling, v an active vertex
  then either a push or a relabel is admissible for v
- ~~f is always a pre~~
  If f starts a preflow, d a valid labelling, always remain so during alg
  (INDUCTION)

• Let f be a preflow on Graph, and d an arbitrary valid labelling on V.

Write G_f for (V, E_f)

$$E_f = \{v \to w \in E : r_f(v \to w) > 0\}$$

If the algorithm terminates and

Then ~~label~~ t is ~~a~~ not accessible from s in G_f f is a flow, and is maximal

Proof:

~~Proo~~ assume there is ~~a~~ path.

no admissible vert ⇒ flow

$s = v_0 - v_1 - v_2 \cdots v_m = t$

maximal because no

now $d(v_i) \leq d(v_{i+1}) + 1$ augmenting path

so $d(v_0) = d(s) \leq d(t) + m < |V|$

~~Th~~ because: $m \leq |V| - 1$ terminates $d(t) = 0$ after

but $d(s) = |V|$ (for valid) labelling.

admissible operations

- If the algorithm terminates, and all labels are finite, then at term f is a flow, and is maximal.

<u>Proof</u>: • terminates because no admissible vert ⇒ flow

• maximal, because no augmenting path.

• <u>Theorem</u>: Alg terminates after at most $O(|V|^2 |E|)$ admissible operations

Flows, cuts and linear programs.

~~For~~ a <u>directed</u> graph, write an incidence matrix A

$$\begin{array}{c|c} & \text{edges} \longrightarrow \\ \hline \text{verts} \downarrow & a_{ij} = \begin{cases} 0 & - \ e_j \text{ not incident on } v_i \\ 1 & e_j \longrightarrow v_i \ (\text{i.e } e_j \text{ arrives at } v_i) \\ -1 & e_j \longleftarrow v_i \ (\text{i.e } e_j \text{ leaves } v_i) \end{cases} \end{array}$$

Make row 1 correspond to s (source)
2 csp to d. (drain)

<u>versions</u>:

$$\max \quad v$$

$$\text{st} \quad A f + v \begin{bmatrix} -1 \\ 1 \\ 0 \\ \vdots \end{bmatrix} = 0$$

$$f \geq 0 \qquad f \leq c$$

flow

Another version:

$$A_R = \begin{bmatrix} \text{all rows of } A \\ \text{but the first} \\ \text{two} \end{bmatrix}$$

$w^T =$ ~~first~~ second row of A (which csp target)

$$\max \quad w^T f$$

$$\text{s.t.} \quad A_R f = 0 \quad \text{(Kirchoffs law.}$$

$$f \geq 0 \quad f \leq c$$

We will do max-flow, min cut by LP-duality

Step 1:

A, A_R are TUM

Proof: (induction)

- consider a Minor B, dim k
- 3 cases

  - some col is all zeros, $\det(B) = 0$
  - B has 1 col that has exactly 1 non-zero

$$\det(B) = \det\begin{bmatrix} 1 & b^T \\ 0 & \tilde{B} \end{bmatrix} \times (\pm 1)$$

by induction

$$\det B = \pm \det \tilde{B}$$

  - B has all cols with 2 non-zeros

    then $1^T B = 0$, $\det B = 0$

Duals of linear programs

1) Assume we have a linear program

$$\min \quad -w^T f$$

$$\text{s.t.} \quad A_R f = 0$$

$$f \geq 0$$

$$c - f \geq 0$$

(This isn't the standard form — just convenient for flow)

its Lagrangian is

$$\mathcal{L}(f, \lambda_e, \lambda_i^1, \lambda_i^2)$$

$$= -w^T f + \lambda_e^T (A_R f) - \lambda_i^{1T} f - \lambda_i^{2T} (c-f)$$

we want the Lagrange dual.

$$\inf_{f} \mathcal{L}(f, \lambda_e, \lambda_i^1, \lambda_i^2)$$

$$= \begin{cases} -\lambda_i^{2T} c & \text{if } [-\omega^T + \lambda_e^T A_R - \lambda_i^{1T} + \lambda_i^{2T}] = 0 \\ -\infty & \text{otherwise.} \end{cases}$$

Now the dual problem is to <u>max</u> this

i.e.

$$\max \quad -\lambda_i^{2T} c$$

$$\text{s.t.} \quad -\omega^T + \lambda_e^T A_R - \lambda_i^{1T} + \lambda_i^{2T} = 0$$

$$\lambda_i^1 \geq 0$$

$$\lambda_i^2 \geq 0$$

We can clean this up a bit.

③-③

$$\max \quad -\lambda_i^{2^T} c$$

$$\text{s.t.} \quad A_e^T \lambda_e + \lambda_i^2 \geq w^T$$

$$\lambda_i^2 \geq 0$$

Step 2:

- Set up Max Flow as L.P.

$$\max \quad w^T f$$

$$\text{st} \quad A_R f = 0 \qquad f \geq 0 \qquad f \leq c$$

Step 3:

- Take the dual; because feasible set has interior point, max flow = min dual
- Dual is:

$$\min \quad y^T c$$

$$\text{st} \quad y \geq 0 \qquad y^T + z^T A_R \geq w^T$$

Step 4:

- at solution, y and z are INTEGER.

to see this, write constraints as

$$\begin{bmatrix} Id & A_R \end{bmatrix} \begin{bmatrix} y \\ z \end{bmatrix} \geq w$$

$[Id \;\; A_R]$ is TUM

(Because minors are minors of A_R)

<u>Step 5</u>:

- write $\hat{z} = [\,\underset{\substack{\uparrow \\ \text{source}}}{0}\,;\, \underset{\substack{\uparrow \\ \text{drain}}}{-1}\,;\, z\,]$

- then constraint is

$$y^T + \hat{z}^T \underset{\sim}{A} \geq 0$$

Step 6:

- A solution to the dual yields a cut

$$U = \{ v \in \text{Verts} \mid \hat{z}_v \geq 0 \}$$

so U contains s, <u>not</u> drain.

Step 7:

- value of cut

$$= \sum_{\substack{i \in \text{edges} \\ \text{leaving } U}} c_i = c(\delta^{out}(U))$$

- it is enough to show

$$c(\delta^{out}(U)) \leq y^T c = \text{max flow value, from 3}$$

because we must have $c(\delta^{out}(U)) \geq$ flow $\cancel{y^T c}$ for any ~~other~~ flow $\hat{y}$

Step 8:

- to show this, it is enough to show that for each edge a in the cut, $y_a \geq 1$

  (because then $y^T c \geq \sum_{i \in \text{edges in cut}} c_i \; = c(\delta^{out}(u))$)

- $a = u \to v$,

  so $\hat{z}_u \geq 0$ (by defn of cut)

  $\hat{z}_v \leq -1$ (integer, less than zero)

  $y^T + \hat{z}^T A \geq 0$ implies

  $y_a + z_v - z_u \geq 0$

  so

  $y_a \geq z_u - z_v \geq 1$

Done

Proof. Consider the proof of Theorem 10.6. We do at most ϕ iterations, while each iteration takes $O(n+t)$ time, where t is the number of arcs deleted. ∎

Hence, similarly to Corollary 10.6a one has:

Corollary 10.11a. *For integer capacities, a maximum flow can be found in time $O(n(\phi+m))$, where ϕ is the maximum flow value.*

Proof. Similar to the proof of Corollary 10.6a. ∎

Therefore,

Corollary 10.11b. *For integer capacities, a maximum flow can be found in time $O(nm\log C)$.*

Proof. In the proof of Theorem 10.10, a maximum flow with respect to c' can be obtained from $2f''$ in time $O(nm)$ (by Corollary 10.11a), since the maximum flow value in the residual graph $D_{f''}$ is at most m. ∎

10.8b. Complexity survey for the maximum flow problem

Complexity survey ($*$ indicates an asymptotically best bound in the table):

	$O(n^2mC)$	Dantzig [1951a] simplex method
	$O(nmC)$	Ford and Fulkerson [1955,1957b] augmenting path
	$O(nm^2)$	Dinits [1970], Edmonds and Karp [1972] shortest augmenting path
	$O(n^2m\log nC)$	Edmonds and Karp [1972] fattest augmenting path
	$O(n^2m)$	Dinits [1970] shortest augmenting path, layered network
	$O(m^2\log C)$	Edmonds and Karp [1970,1972] capacity-scaling
	$O(nm\log C)$	Dinits [1973a], Gabow [1983b,1985b] capacity-scaling
	$O(n^3)$	Karzanov [1974] (preflow push); cf. Malhotra, Kumar, and Maheshwari [1978], Tarjan [1984]
	$O(n^2\sqrt{m})$	Cherkasskiĭ [1977a] blocking preflow with long pushes
	$O(nm\log^2 n)$	Shiloach [1978], Galil and Naamad [1979,1980]
	$O(n^{5/3}m^{2/3})$	Galil [1978,1980a]

≫

continued

	$O(nm\log n)$	Sleator [1980], Sleator and Tarjan [1981,1983a] dynamic trees
$*$	$O(nm\log(n^2/m))$	Goldberg and Tarjan [1986,1988a] push-relabel+dynamic trees
	$O(nm+n^2\log C)$	Ahuja and Orlin [1989] push-relabel + excess scaling
	$O(nm+n^2\sqrt{\log C})$	Ahuja, Orlin, and Tarjan [1989] Ahuja-Orlin improved
$*$	$O(nm\log((n/m)\sqrt{\log C}+2))$	Ahuja, Orlin, and Tarjan [1989] Ahuja-Orlin improved + dynamic trees
$*$	$O(n^3/\log n)$	Cheriyan, Hagerup, and Mehlhorn [1990,1996]
	$O(n(m+n^{5/3}\log n))$	Alon [1990] (derandomization of Cheriyan and Hagerup [1989,1995])
	$O(nm+n^{2+\varepsilon})$	(for each $\varepsilon>0$) King, Rao, and Tarjan [1992]
$*$	$O(nm\log_{m/n}n+n^2\log^{2+\varepsilon}n)$	(for each $\varepsilon>0$) Phillips and Westbrook [1993,1998]
$*$	$O(nm\log_{\frac{m}{n\log n}}n)$	King, Rao, and Tarjan [1994]
$*$	$O(m^{3/2}\log(n^2/m)\log C)$	Goldberg and Rao [1997a,1998]
$*$	$O(n^{2/3}m\log(n^2/m)\log C)$	Goldberg and Rao [1997a,1998]

Here $C:=\|c\|_\infty$ for integer capacity function c. For a complexity survey for unit capacities, see Section 9.6a.

Research problem: Is there an $O(nm)$-time maximum flow algorithm?

For the special case of *planar* undirected graphs:

	$O(n^2\log n)$	Itai and Shiloach [1979]
	$O(n\log^2 n)$	Reif [1983] (minimum cut), Hassin and Johnson [1985] (maximum flow)
	$O(n\log n\log^* n)$	Frederickson [1983b]
$*$	$O(n\log n)$	Frederickson [1987b]

For *directed* planar graphs:

	$O(n^{3/2}\log n)$	Johnson and Venkatesan [1982]
	$O(n^{4/3}\log^2 n\log C)$	Klein, Rao, Rauch, and Subramanian [1994], Henzinger, Klein, Rao, and Subramanian [1997]
$*$	$O(n\log n)$	Weihe [1994b,1997b]

Now, notice:

- in augmenting path, we always had a flow
  - is this a cut? not necessarily
  - e.g.

$$s \xrightarrow{1\ (3)} \bullet \xrightarrow{1\ (3)} \bullet \xrightarrow{1\ (3)} d$$

  - flow that isn't a cut, BECAUSE there is an augmenting path.
  - c.f. discussion of aug paths.
    - a flow is ONLY a cut if there is no augmenting path.
      i.e. if its maximal.

In push-relabel, we have manipulating <u>CUTS</u> (by thm, above), and ONLY had a flow at optimality.

~~These two are dual~~

Max-flow, min. cut are <u>DUAL</u>

MAX FLOW	MIN CUT
min $-w^T f$	max $-\lambda_i^{2\,T} c$
s.t. $A_R f = 0$	s.t. $-w + A_R^T \lambda_e - \lambda_i^1 + \lambda_i^2 = 0$
$f \geq 0$	$\lambda_i^1 \geq 0$
$c - f \geq 0$	$\lambda_i^2 \geq 0$

These problems are coupled by complementarity conditions

$$(\lambda_i^1)_K \, f_K = 0 \qquad K = 1 \cdots \#E$$

$$(\lambda_i^2)_K \, \cancel{f_4} \, (c_K - f_K) = 0 \qquad K = 1 \cdots \#E$$

(i.e. either the inequality is <u>active</u> or its Lagrange multiplier is 0)

Flow's Now $\quad (\lambda_i^1)_K = 0 \quad$ implies that flow is > 0

$\qquad\qquad (\lambda_i^2)_K = 0 \quad$ " " $< c$

- The non-zero λ_i's identify edges that could disconnect (i.e. any path ~~not~~ containing one is not augmenting)
- If the non-zero λ_is meet the <u>equality</u>, we have a <u>cut</u>

There is a <u>general point</u> here

- consider $(f, \lambda_i^1, \lambda_i^2, \lambda_e) = (p, d)$

$$\underbrace{f}_{\text{primal vars}}, \quad \underbrace{\lambda_i^1, \lambda_i^2, \lambda_e}_{\text{dual vars.}}$$

1) ~~at a solution~~ if p is soln to Primal, d is soln to dual

p, d are <u>complementary</u>

(follows from KKT).

2) if (p, d) are <u>complementary</u> <u>and</u>

p is primal feasible and

d is dual feasible ~~then~~

$\Updownarrow$

p solves primal $\quad$ d solves dual.

(This follows from KKT conditions).

<u>Hence</u> two kinds of <u>strategy</u>

primal - feasible, dual infeasible (Aug path)

primal - infeasible, dual-feasible (Push relabel