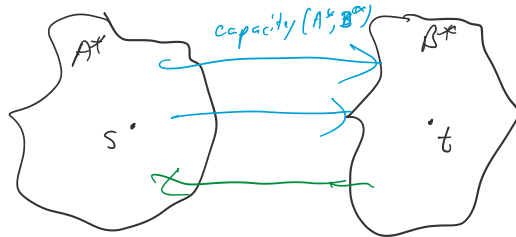


Recall: Last time we showed that min-cut & max-flow problems are dual   
 different ways of seeing the same problem.

Equivalence: Ford-Fulkerson keeps augmenting paths until no  $s-t$  path exists in  $G_{f^*}$ , so there is a cut  $(A^*, B^*)$ .



$v(f^*) = \text{capacity}(A^*, B^*)$

Also, can't have higher  $v(f^*)$

(need saturation or else still  $s-t$  path in  $G_{f^*}$ )

$\Rightarrow f^*$  is a max-flow &  $(A^*, B^*)$  is a min-cut.

Note: Ford-Fulkerson only works with integer weights for  $O(mC)$  time.   
 For irrational weights, need to deal with cycles explicitly, but a variation   
 Edmonds-Karp works in  $O(\sqrt{VE}^2)$  time.

Reductions

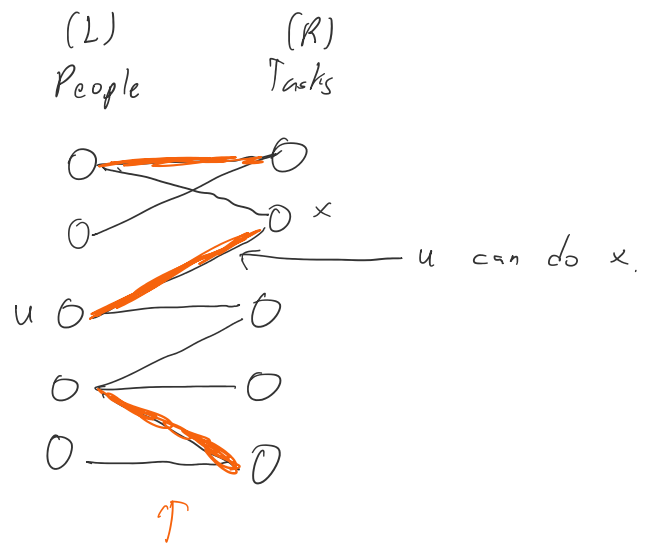
A reduction rewrites one problem  $A$  as a different problem  $A'$ , so that if we can solve  $A'$ , then we can get a solution of  $A$ .

Ex. We rewrite problems as shortest path to use things like Dijkstra on  $A^*$ .

Maximum Bipartite Matching

- Set of people  $L$  & tasks  $R$
- Each person can only do certain jobs
- Bipartite graph

A matching assigns people to tasks   
 A maximum matching contains as many edges as possible



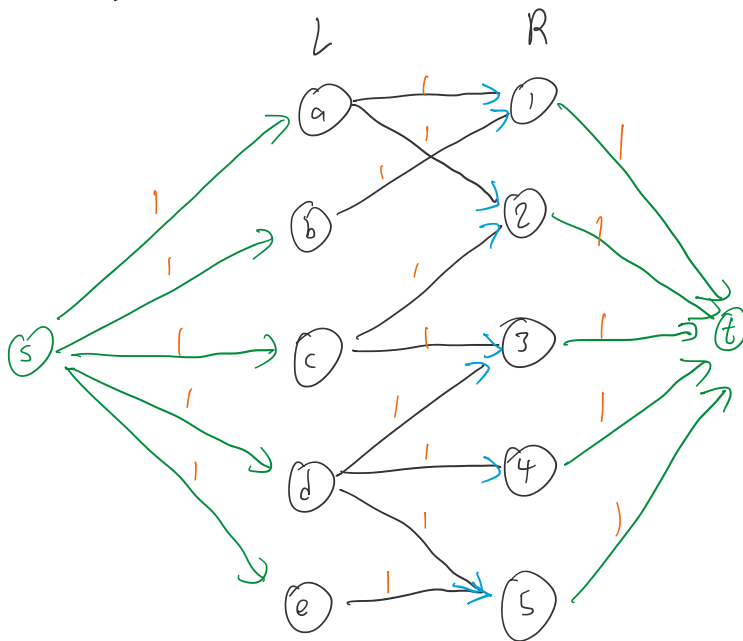
not maximum matching

Formally: Given bipartite graph  $G = (L \cup R, E)$ , find  $S \subseteq E \subseteq L \times R$  that

Formally: Given bipartite graph  $G = (L \cup R, E)$ , find  $S \subseteq E \subseteq L \times R$  that is a matching and as large as possible

- Note:
- Given  $L$  &  $R$ , so don't need to find them, though could use BFS
  - $S$  is a perfect matching if every vertex is matched.
  - Maximum vs. maximal
    - ↑ as big as possible
    - ↑ can't make any bigger w/o undoing a choice

Reduction: Convert problem into network flow problem to solve, where the solution to the flow problem is easily converted back into original matching problem.



turn into directed graph by pointing all arrows from  $L$  to  $R$ .

Add vertex  $s$  pointing to all  $v \in L$   
Add vertex  $t$  pointing from all  $v \in R$

Assign all edges capacity 1

Solve max-flow on new graph  $G'$  using Ford-Fulkerson.

Analysis: Ford-Fulkerson augments with  $s-t$  paths, each one having value 1.

Flows  $\Rightarrow$  matchings

Let  $M$  be the set of  $L \rightarrow R$  edges we use in flow  $f$ .

$M$  must be a matching because if two edges are used touching a node, the flow  $f$  will be unbalanced, since flow through a node is  $\leq 1$  by construction.

Also,  $|M| = v(f)$  because each edge implies a different unique node in  $L$  used, which must have 1 flow going through it from  $s$ .

$\Rightarrow$  max-flows found by Ford-Fulkerson correspond to matchings, of same value.

All matchings correspond to an integral flow

Given a matching simply push flow into all nodes of  $L$  with a selected

All matchings correspond to an integral flow

Given a matching simply push flow into all nodes of L with a selected out-edge, and out of all nodes of R with a selected in-edge.

⇒ A max-flow is also a maximum matching

Runtime:

Ford-Fulkerson takes  $O(m' C)$  time, where  $m'$  is number of edges in flow network

edges in G =  $m + 2n$  ← new edges

$$C = \sum_{e \in \text{Out}(s)} c_e = |\text{Out}(s)| = |L| = n$$

$$= O((m + 2n)n) = O(mn + 2n^2) = O(mn) \text{ time.}$$

for maximum bipartite matching.

Even though maximum bipartite matching doesn't look like a flow problem, we can reduce it to a flow problem.

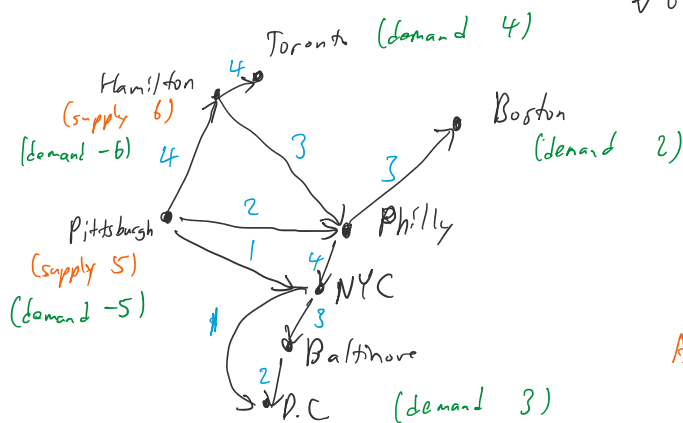
Max-flow extensions

The max-flow problem had just a single source  $s$  and target  $t$ . (sink)

What if multiple sources & targets?

Circulation with demands has multiple sources  $S$  and multiple sinks  $T$ . (supply) (demand)

Supply = negative demand.  $\forall s \in S$ , let  $d_s = -\text{supply}$   
 $\forall t \in T$ , let  $d_t = \text{demand}$



Feasible flow must have

$$\sum_{s \in S} d_s + \sum_{t \in T} d_t = 0$$

Let  $D = \sum_{t \in T} d_t$

Also usual constraints:

$$\forall e \in E, 0 \leq f(e) \leq c_e$$

$$\forall v \in V, f^{in}(v) - f^{out}(v) = d_v$$

( $d_v = 0$  for nodes not in  $S$  or  $T$ )

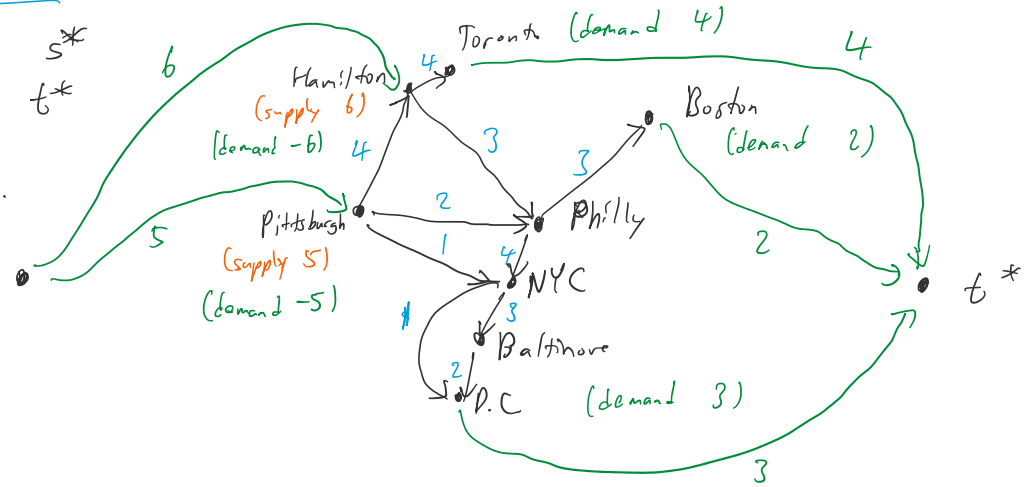
Reduction to max-flow

Add new super-source  $s^*$



Reduction to max flow

Add new super-source  $s^*$   
 ↓  
 super-sink  $t^*$   
 $(s^*, s)$  has capacity  $-d_s$ .  
 $(t, t^*)$  has capacity  $d_t - s^*$



Feasible circulation if and only if there is a flow of value  $D$ , from  $s^*$  to  $t^*$ .

Circulation with lower bounds

Suppose Pittsburgh made a commitment to always ship at least 1 ton of steel on the Pittsburgh-Philly line. Can we modify our algorithm?

New constraints:  $\forall e \in E, l_e \leq f(e) \leq c_e$   
 $\forall v \in V, f^{in}(v) - f^{out}(v) = d_v$

We want to convert into ordinary circulation w/ demands problem.

Idea: Start by preallocating the flow demanded by  $l_e$  as part of the demands by the nodes.

Define flow  $f_0$  by setting  $f_0(e) = l_e \forall e \in E$ .

- meets capacity constraint but not demand constraint.

Define  $L_v = f_0^{in}(v) - f_0^{out}(v)$ .

(normally, we need  $f^{in}(v) - f^{out}(v) = d_v$  for demand constraint.)

New demand constraint:  $f^{in}(v) - f^{out}(v) = d_v - L_v$

and new capacity constraint:  $0 \leq f(e) \leq c_e - l_e$ .

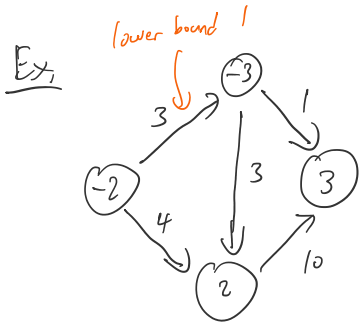
(since we used some of the flow  $f_0$  already)

⇒ new standard circulation problem.

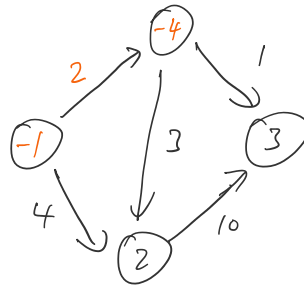
lower bound!

(-16)

→ new standard circulation



lower bound messes up obvious solution



(Reduction) Transformation to equivalent problem w/o lower bounds

Serial reductions:

circulation with lower bounds



circulation w/o lower bounds



max flow

Summary Can now