# Network Flows

## Lecture 18

Akshar Varma

7–8 August, 2023

CS3000 Algorithms and Data

Flow Networks

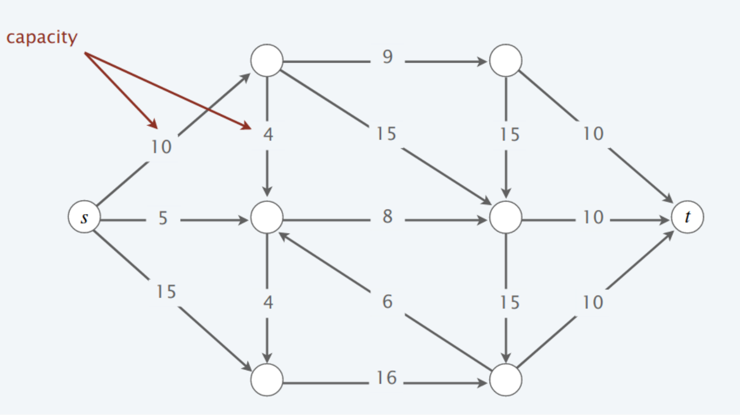
Max Flow-Min Cut Theorem

Reductions and Applications

# 1. Flow Networks

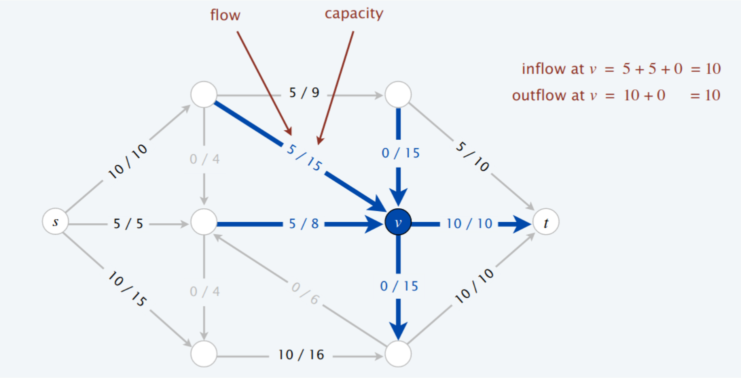
# Flow Networks: $N = (G, c, s, t)$

- Directed graph $G = (V, E)$; edge capacities $c : E \to \mathbb{R}^{\geq 0}$; source $s$, terminal $t$.
- Models transportation networks: water pipelines, electric grids, road traffic, etc.
- Flow through an edge $f : E \to \mathbb{R}^{\geq 0}$. How much can "flow" through?
- What can stop flow from a source vertex $s$ to a terminal vertex $t$?



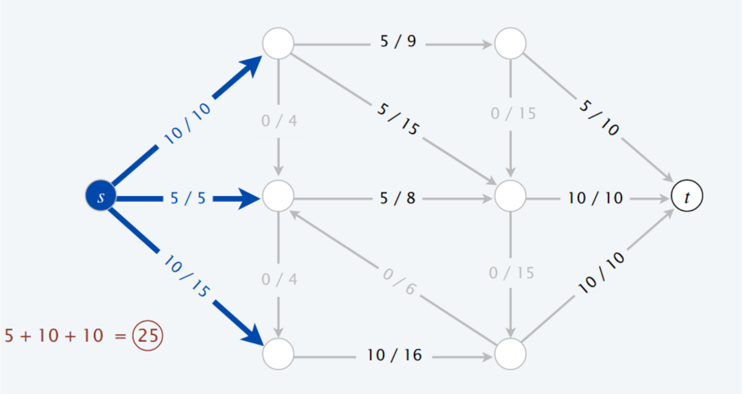
# Flow Conservation and Capacity Limits

- The capacities are constraints; can't sent more than the capacity. $f(e) \leq c(e), \forall e \in E$
- Cannot except 8 lanes of vehicles on a 4 lane road or more water than size of pipe.
- Also, if something enters a node except source and sink, it must leave.
  For all vertices $v \in V - \{s, t\}$ flow must satisfy: $\sum_{e=(u,v)} f(e) = \sum_{e=(v,w)} f(e)$.
- If you enter an *intersection*, you must leave; water coming in, must go out.



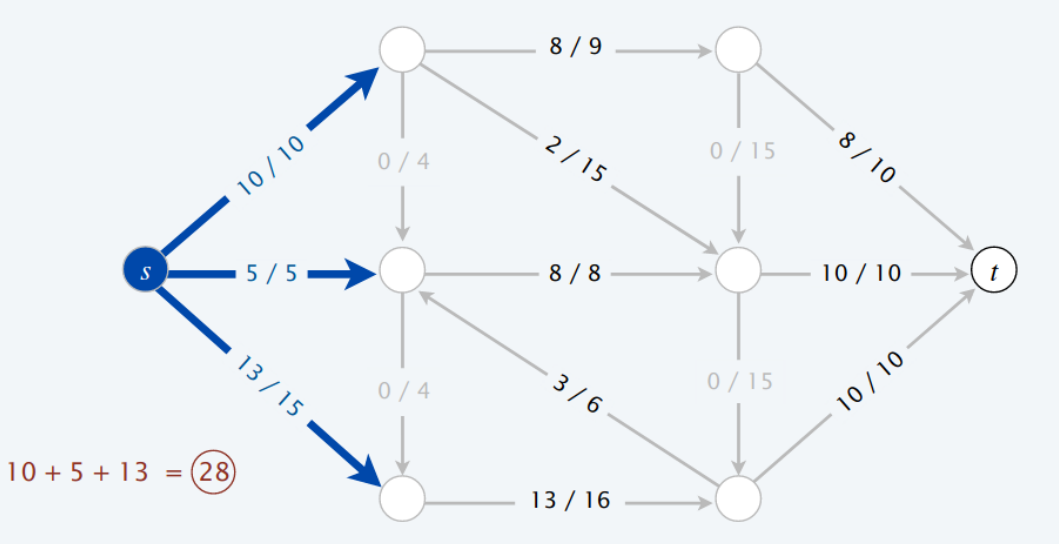
# An example flow satisfying all conditions

Value of flow $|f| = \sum_{e=(s,u)} f(e) - \sum_{e=(w,s)} f(e)$. Total flow *out of* $s$.



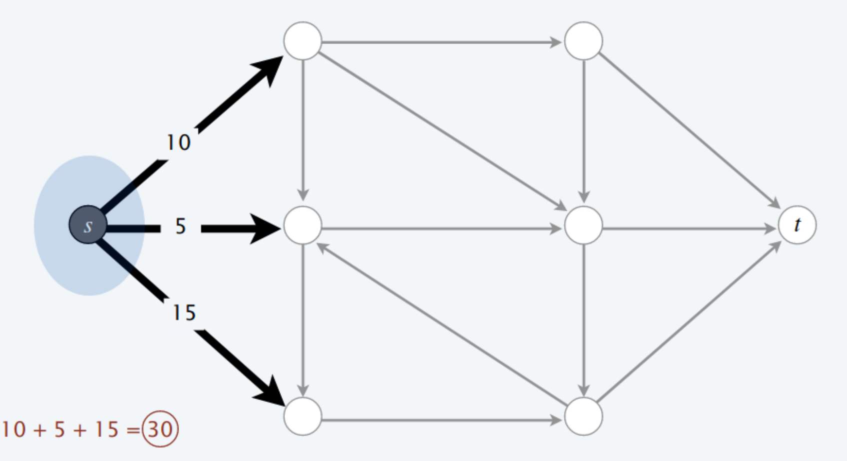

Always equal to total flow *into* $t$ by conservation of flow.

# Maximum Flow possible in this network

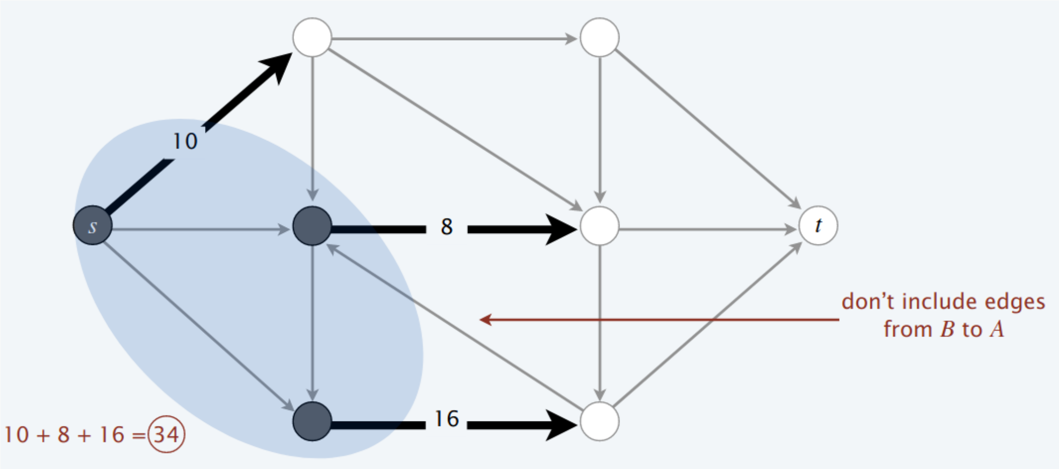


# Capacity of Cut

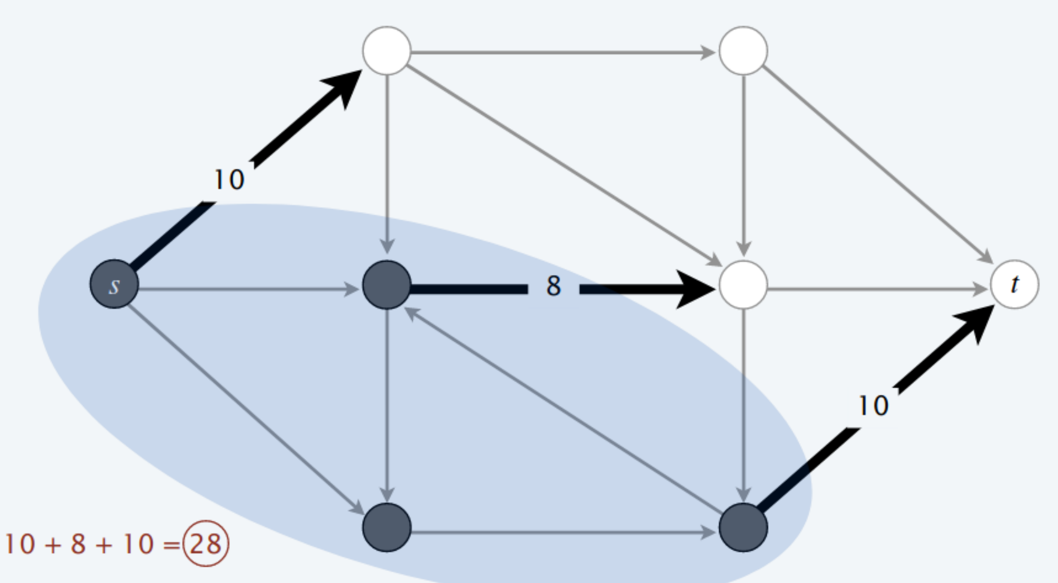
- A cut $(A, B)$ separates source $s \in A$ and terminal $t \in B$.
- Capacity of cut $\|A, B\|$ is sum of capacities of edges from $A$ **to** $B$.



# Capacity of (another) Cut



# Minimum Cut Capacity



# 2. Max Flow-Min Cut Theorem

# Max Flow-Min Cut Theorem and other useful facts

- Lemma: For any flow $f$ and any cut $(A, B)$, $|f|$ equals net flow across $(A, B)$.
- **Weak duality:** *Any* flow value is smaller than *any* cut capacity.
- **Strong duality:** Maximum flow is equal to the minimum cut capacity.
- *Alternatively*: If the $|f| = \|A, B\|$, then it is the max flow and the min cut.
- If all capacities are integral $c : E \to \mathbb{N}^{\geq 0}$, then there is a max-flow with $f : E \to \mathbb{N}^{\geq 0}$.
- Maximum flows (and minimum cuts) can be computed in $O(VE)$ time.
- *Terminology*: Edges "saturated" if $f(e) = c(e)$, "avoided" if $f(e) = 0$.
- Flow can always be decomposed into cycles and paths.
- There is always a flow in which only one of $f(u, v)$ and $f(v, u)$ are non-zero.

# 3. Reductions and Applications

# Reductions to Max Flow-Min Cut

- Max flow and min cut can solve a large variety of find the “best” problems.
- $N = (G, c, s, t)$ can represent many types of problems.
- In these cases finding max flow (value)/min cut (capacity) gives the solution.
- We use the terminology “reduction” when we convert a problem to another.
- Intuitively: the problem difficulty reduces to that of something we know how to do.
- Flow network must be created so that its solution easily solves original problem.
- Requires converting flow (value)/cut (capacity) into original problem solution.
- Just a matter of interpreting appropriately; sometimes requires minimal conversion.
- Given a problem solution pair $P, S$, map it to a flow network: $R(P) = (G, c, s, t)$.
- Our $R(P)$ must be such that we can easily compute $R'(f, |f|, (A, B), \|A, B\|) = S$.

# Applications of Max Flow-Min Cut

We'll solve the following problems by reducing to a flow problem:

- Number of edge disjoint paths from $s$ to $t$.
- Vertex capacities and number of vertex disjoint paths from $s$ to $t$.
- Bipartite Matching.
- Tuple Selection (generalizes bipartite matching).
- Extending flow networks to cases where there are:
    - Multiple sources/sinks
    - Circulations with supplies, demands
    - Capacity lower bounds
- Minimum Cost Circulations
- Survey Design: for customers (constraints on products/customers/questions etc.)
- Airline Scheduling: schedule equipment and crew for most customer satisfaction.
- Image Segmentation: divide images into coherent/meaningful regions.
- Project Selection: choose projects to maximize revenue with prerequisite constraints.

# Disjoint Paths given $G = (V, E), s, t$

- We want the maximum number of paths from $s$ to $t$ that are disjoint from each other.
- One example application of this is in communication networks.
- *Edge disjoint paths:* must have no edges in common between two paths.
- Cannot have same channel being used for the same conversation.
- How many conversations can keep happening simultaneously?
- On the flip side, how many links broken completely prevents $s$ communicating to $t$?
- *Vertex disjoint paths:* must have no common vertices among any two paths.
- There may be limits on how much each cell tower can handle/transmit.
- How many cell towers needed for expected call volume?

# Edge Disjoint Paths in Graphs

- Assign capacity 1 to every edge in the graph. $G'$
- Flow $|f|$ will equal the number of edge disjoint paths $k$. Why?
- Each edge can contribute to at most one path since capacity 1. (Integrality!)
- Find the paths by traversing from $s$ to $t$ using $f(e) = 1$ edges.
- Remove paths found, and repeat until all paths found.

- What if graph was undirected?
- Make every edge $\{u, v\}$ into two antiparallel edges $(u, v)$ and $(v, u)$.
- Reduction! Undirected graph edge disjoint paths $\longrightarrow$ Digraph edge disjoint paths.

- A subset of edges $F \subseteq E$ disconnects $t$ from $s$ if each $s - t$ path has some $e \in F$.
- If we remove edges from $F$, then no path from $s$ to $t$ will remain.
- **Network Connectivity:** Find minimum sized $F$ which disconnects $t$ from $s$.
- **Menger's Theorem:**
  Max number of edge disjoint $s - t$ paths $=$ min size for $F \subseteq E$ to disconnect $t$ from $s$.
- Our earlier reduction will also allow us to find out about network connectivity.
- In fact, the min cut capacity in that reduction *is* the size of the best $F \subseteq E$.
- Menger's theorem is a special case of max flow-min cut theorem; for capacity 1 edges.

# Vertex Capacities and Vertex Disjoint Paths

- We've seen a lot about edge capacities, what if we want capacities on vertices $c(v)$?
- Do we need to come up with new algorithms, theorems, and so on?
- No! Come up with a reduction!
- Replace every vertex $v$ with $v_{in}, v_{out}$, add $(v_{in}, v_{out})$, s. t $c(v_{in}, v_{out}) = c(v)$.
- Every edge into $v$ now goes into $v_{in}$ and every edge out of $v$ comes out of $v_{out}$.

- This reduction of making a vertex into an edge gives us more power (conceptually).
- We can think in terms of vertex capacities in our reduction from this point.
- Vertex disjoint paths $\longrightarrow$ reduce using $c(v) = 1 \longrightarrow v_{in}, v_{out}$ gives edge disjoint paths.

# Vertex Capacities Reduction

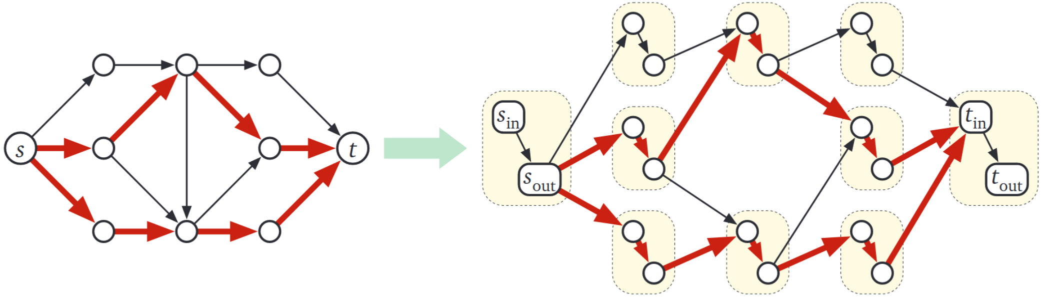


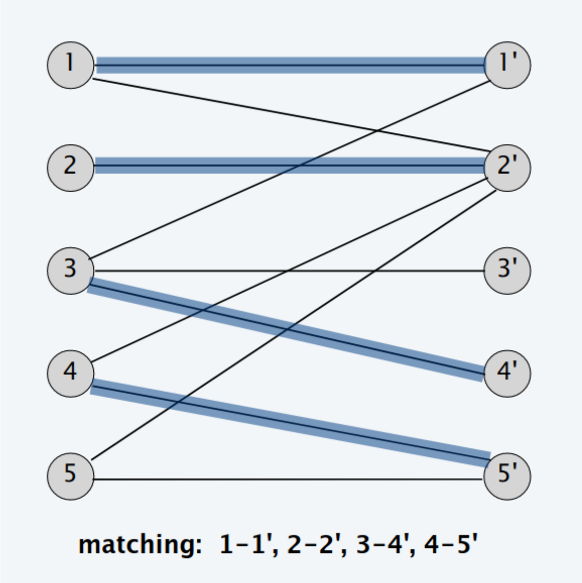

**Figure 11.1.** Reducing vertex-disjoint paths in $G$ to edge-disjoint paths in $\bar{G}$.

Figure from Jeff Erickson's book

# Bipartite Matching

- “Match” up vertices on one side of a bipartite graph with vertices on the other side.
- Formally: A subset of edges, such that no vertex in two edges.
- Maximum Matching: The largest matching that exists, as many pairs matched up.
- Original application: Matching doctors and hospitals based on their preferences.
- Doctors list hospitals they are willing to work at.
- Hospitals list doctors they’re willing to hire.
- Bipartite graph: Doctors and hospitals are vertices. Edge iff vertices okay to match.
- Maximum bipartite matching: find largest matching in this graph.
- Match as many doctor-hospital pairs up.

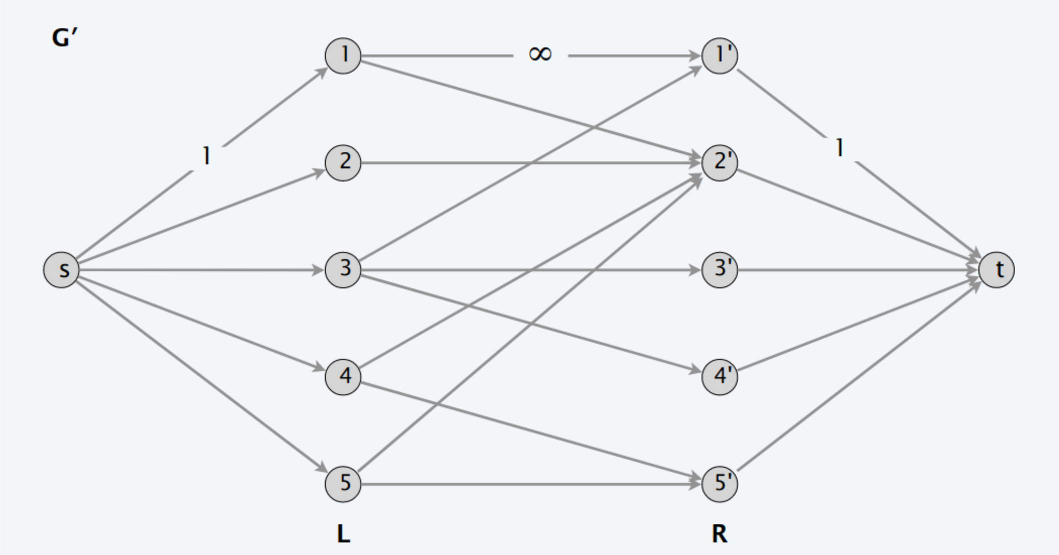
# Bipartite Matching Example



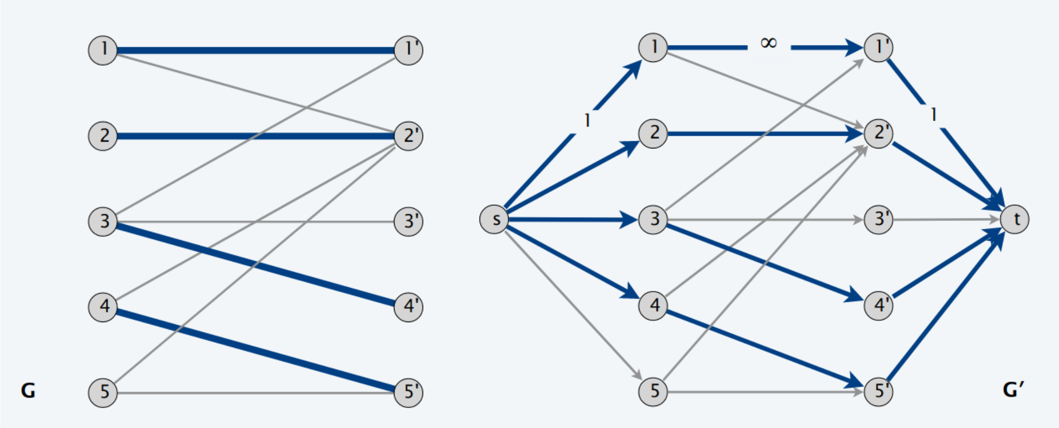
# Reducing Bipartite Matching to Network Flow

- $G = (V, E)$ where $V = L \cup R$ is union of two sets of vertices (left and right).
- Edges describe all pairings that are acceptable to both sides.
- We want to create $N = (G', c, s, t)$ given $G$.
- $N$ must have property that some of $f, |f|, (A, B), \|A, B\|$ gives maximum matching.
- Ideas?
- Vertices would be new $s, t$ with all old vertices.
- Every edge $\{l, r\}$ which existed becomes a directed edge $(l, r)$ with $\infty$ capacity.
- Add edge $(s, \ell)$ for all $\ell \in L$ with capacity 1.
- Add edge $(r, t)$ for all $r \in R$ with capacity 1.
- $N = \left(G' = (V \cup \{s, t\}, \{(\ell, r), (s, \ell), (r, t)\}), c, s, t\right)$
- There is matching of size $|f|$ where $|f|$ is the maximum flow value.
- Edges with $f(\ell, r) = 1$ give the actual matching edges.

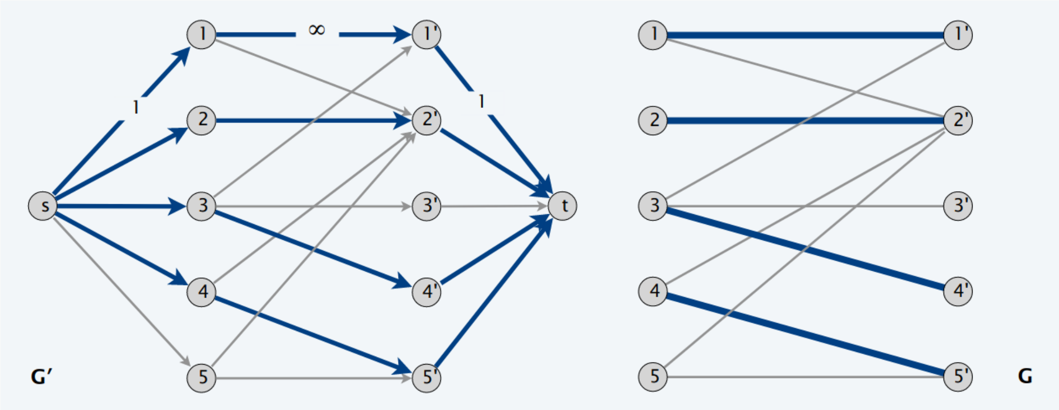
# Bipartite Matching Reduction Visualization



# Matching to Flow



# Flow to Matching



# Tuple Selection

- Bipartite matching is special case of a more general “assignment” type problem.
- You now have many sets $X_1, X_2, \ldots, X_d$.
- Want to select as many $d-$tuples as possible subject to various capacity constraints:
    1. $\forall i$, we have that $x \in X_i$ can appear in at most $c(x)$ tuples.
    2. $\forall i$, we have that $x \in X_i, y \in X_{i+1}$ can appear in at most $c(x, y)$ tuples.
- The $c(x), c(x, y)$ values are usually some small non-negative number or $\infty$.
- Maximum matching: $d = 2$, each $x$ has $c(x) = 1$, each pair $x, y$ has $c(x, y) \in \{0, 1\}$.
- Reduction:
    - Each $x \in X_i, \forall i$ has a vertex with capacity $c(x)$. Special vertices $s, t$.
    - Edges $(s, w)$ for all $w \in X_1$ and $(z, t)$ for all $z \in X_d$ with capacity 1.
    - Edges $(x, y)$ for $x \in X_i, y \in X_{i+1}$ for all $i$ with capacity $c(x, y)$.

# Tuple Selection Flow Network

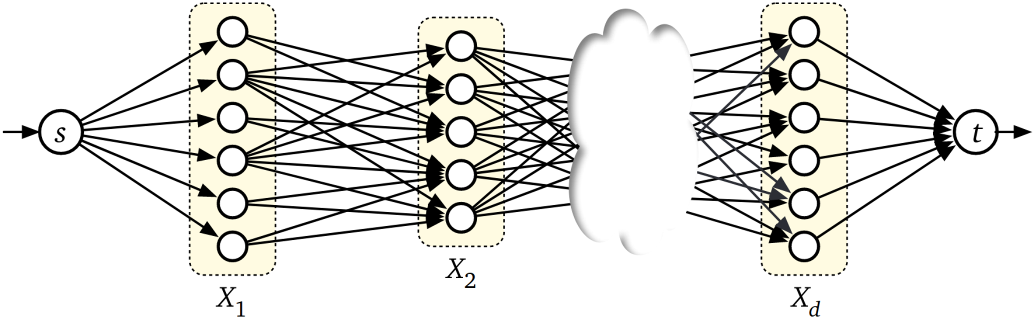


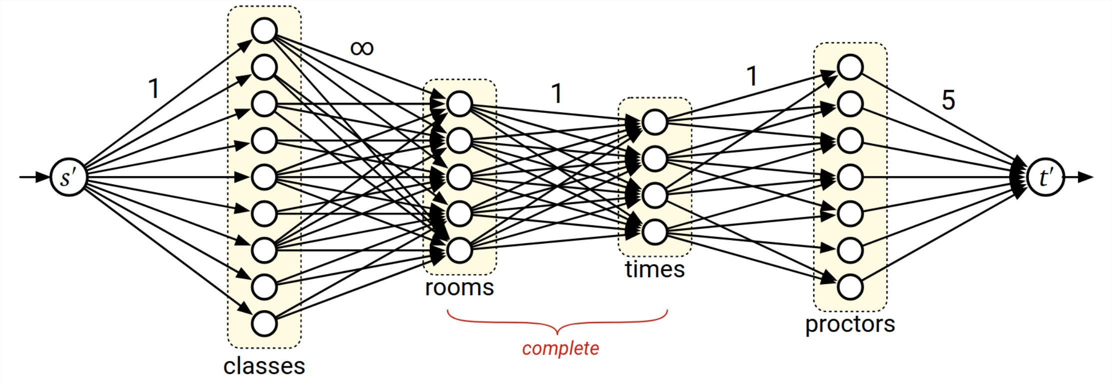

**Figure 11.4.** The flow network for a tuple selection problem.

Figure from Jeff Erickson's book

# Exam Scheduling at Uskees University

- $n$ different classes, to be scheduled into one of $r$ rooms, with $t$ available time slots.
- At most one class in one room in one time slot.
- Classes cannot be split into different rooms or different time slots.
- There are $p$ proctors to oversee the exam. One proctor oversees one exam at a time.
- Proctors available at different time slots; each can proctor at most 5 exams.
- You know enrollment $E[i]$ in class $i$ via $E[1 \ldots n]$; size $S[j]$ of room $j$ via $S[1 \ldots r]$.
- Scheduling class $i$ in room $j$ requires that $E[i] \leq S[j]$.
- Availability of proctors for each time slot $A[k, \ell] \in \{0, 1\}$ known via $A[1 \ldots t, 1 \ldots p]$.
- Fits into tuple selection framework; 4 resources: classes, rooms, times, proctors.
- Any ideas on how this becomes a network?

# How Would the Exam Scheduling Network Look Like?




Exam scheduling network flow formulation

# Exam Scheduling Network Definition

- Create $s, t$ and vertices for each class $c_i$, room $r_j$, time slot $t_k$ and proctor $p_\ell$.
- Add edges $s$ to $c_i$ with capacity 1. (each class holds one final)
- Add edges $c_i$ to $r_j$ of $\infty$ capacity iff $E[i] \leq S[j]$. (class vs. room size limits)
- Add edges $r_j$ to $t_k$ with capacity 1 for all $j, k$. (1 exam in room $j$ in slot $k$).
- Add edges $t_k$ to $p_\ell$ with capacity 1 iff $A[k, \ell] = 1$. (proctor's availability)
- Add edges $p_\ell$ to $t$ with capacity 5. (can proctor at most 5 exams)

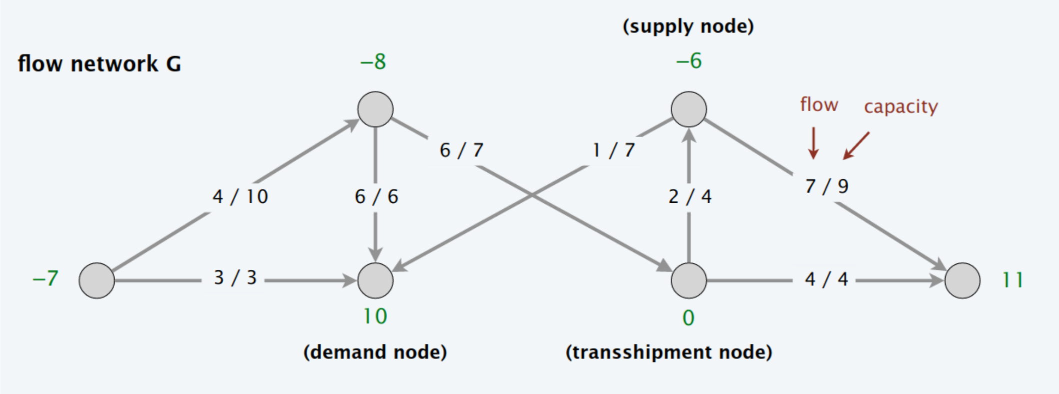
# Extending Max Flow

- We saw how to extend max flow to deal with vertex capacities. We can do more.

- What if there are multiple sources and multiple sinks?
- Create a super source and connect to each source; similarly use a super sink.

- Circulations with supplies and demands: each vertex has a demand $d(v) \in \mathbb{R}$.
- No special source or sink, products need to *circulate* in network.
- No concept of transport from source to a terminal vertex.
- Question: Is there a flow that satisfies circulation constraints?

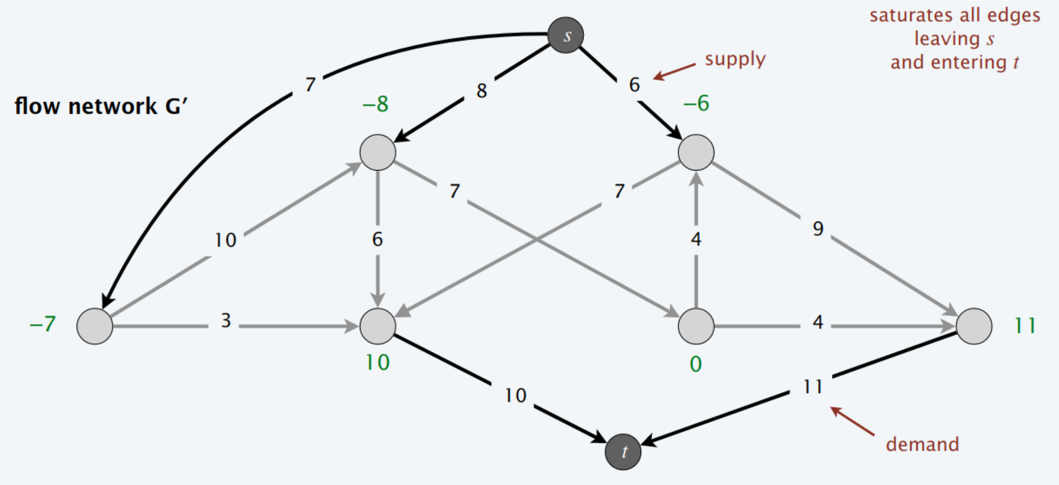
# Circulation Network with Supplies and Demands



# Reducing Circulations with Supplies and Demands

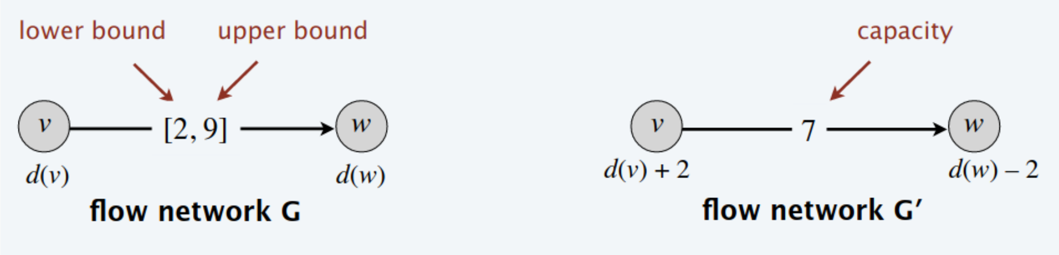
- Now we have a $d(v)$ for every vertex, not a capacity but a "demand".
- $d(v) > 0$ is a supply node and $d(v) < 0$ is a demand node.
- Reduction: Create super sink, super source.
- For every $d(v) < 0$, connect sink to $v$ with capacity $-d(v)$ (positive).
- For $d(v) > 0$, connect $v$ to sink with capacity $d(v)$.
- Circulation $\iff$ max flow is $\sum_{v:d(v)>0} d(v) = \sum_{v:d(v)<0} -d(v)$
- Same as checking if edges leaving $s$ and entering $t$ are *saturated*.

# Reduction for Circulation Network with Supplies and Demands



# Flow Lower Bounds

- What if flow must satisfy a lower bound at each edge: $\ell(e) \leq f(e) \leq c(e)$?
- We reduce this to circulations with demands.
- "Send" $\ell(e)$ units of flow through the edge and adjust the demands on vertices.
- Start vertex creates and end vertex consumes.



# Minimum Cost Circulations

- Everything until now, we've reduced to max flow.
- However, that is not the most broad framework of this kind.
- Circulations can have costs in addition to upper+lower bounds and demands.
- Let $p(e)$ denote a price associated with sending flow through an edge.
- The total cost, which we want to minimize is $\sum_{e \in E} p(e) \cdot f(e)$.
- Of course, this still has to be subject to flow conservation and capacity bounds.

$$\min \sum_{e \in E} p(e) \cdot f(e)$$

s.t.

$$\ell(e) \leq f(e) \leq u(e) \qquad \text{Capacity Bounds}$$

$$\sum_{e=(u,v)\in E} f(e) - \sum_{e=(v,w)\in E} f(e') = d(v) \qquad \text{Flow Conservation}$$

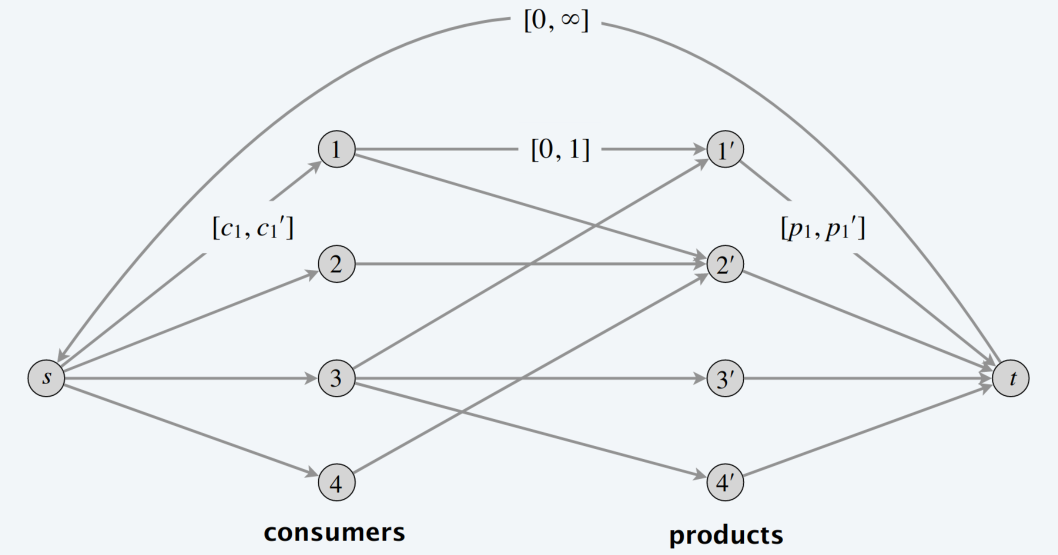
# Survey Design

- Design a survey to ask $n$ consumers about $m$ products.
- There must be one survey question per product.
- You can only survey consumer $i$ about product $j$ is they own it.
- Consumer $i$ must be asked between $c_i$ and $c_i'$ questions.
- At least $p_j$ and at most $p_j'$ consumers need to be surveyed for product $j$.
- Is there a survey design that meets these requirements?
- If there is, design it, or correctly show why it isn't possible.

# Survey Design Network Formulation

- We'll need lower bounds, so let us use circulations with lower bounds.
- We don't need any demands on vertices so all $d(v) = 0$.
- Create a $s, t$ and vertices for each consumer and product.
- Add edge $(i, j)$ if consumer $i$ owns product $j$; set capacity bounds $[0, 1]$.
- Add edges from $s$ to consumer $i$; set capacity bounds $[c_i, c'_i]$.
- Add edges from product $j$ to $t$; set capacity bounds $[p_j, p'_j]$.
- Add an edge from $t$ to $s$; set capacity bounds $[0, \infty]$.
- If there is an integral circulation, then survey possible.

# Survey Design Network Visualization



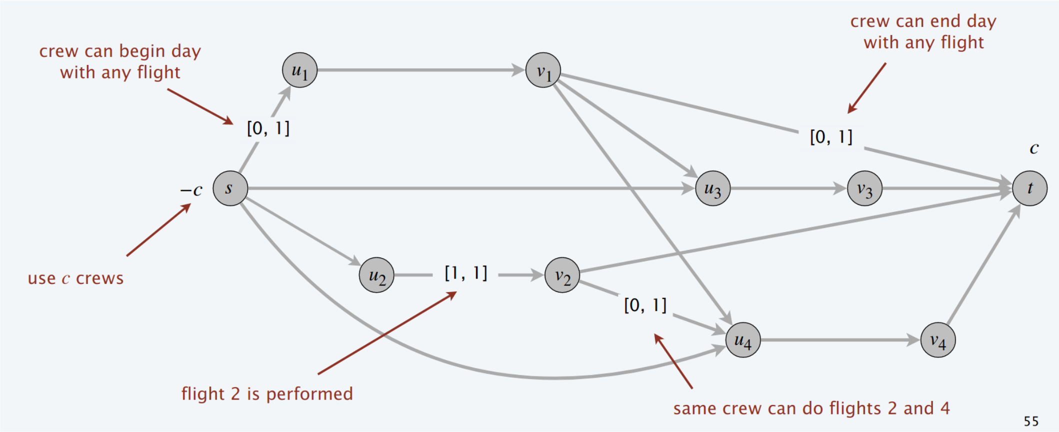
# Airline Scheduling

- Need to manage allocation of equipment, crew, customer satisfaction and so on.
- These require scheduling where equipment and crew should be at all times.
- Toy setup: minimize the number of flight crews needed given these constraints:
- Set of $k$ flights each day.
- Flight $i$ leaves origin $o_i$ at time $s_i$ and reaches destination $d_i$ at time $f_i$.

# Airline Scheduling via Circulations

- For each flight $i$ create two vertices $u_i$ (start of flight) and $v_i$ (end of flight).
- Add source $s$ with demand $-c$, connect to each $u_i$ with capacity bounds $[0, 1]$.
- Add sink $t$ with demand $c$, connected from each $v_i$ with capacity bounds $[0, 1]$.
- For each flight $i$, add edge $(u_i, v_i)$ with capacity bounds $[1, 1]$.
- If same crew can service flights $i$ & $j$ add $(v_i, u_j)$ with bounds $[0, 1]$.

# Airline Scheduling Network Visualization



# Image Segmentation

- Separate image into foreground and background.
- The pixels becomes vertices and neighboring vertices are neighbors in graph.
- We want to have as few foreground pixels end up in the background and vice versa.
- For pixel $i$, let $a_i \geq 0, b_i \geq 0$ be foreground/background likelihood respectively.
- We also want some smoothness in the foreground/background separation.
- So we'll penalize our separation whenever $i$ is a different side than most neighbors.
- We'll use $p_{ij} \geq 0$ as penalty for labeling pixel $i$ and $j$ differently.
- So we want:

$$\min \sum_{i \in B} a_i + \sum_{j \in A} b_j + \sum_{\substack{(i,j) \in E \\ |A \cap \{i,j\}| = 1}} p_{ij}$$

# Minimum Cut Modeling of Image Segmentation

- Create a node for each pixel; add antiparallel edges between neighbors.
- Source $s$ acts as "foreground" side, sink $t$ acts as "background".
- Capacities $a_j$ from $s$ to pixels, $b_i$ from pixels to sink; $p_{ij}$ on antiparallels.
- Find minimum cut; it will give separation into foreground and background segments.

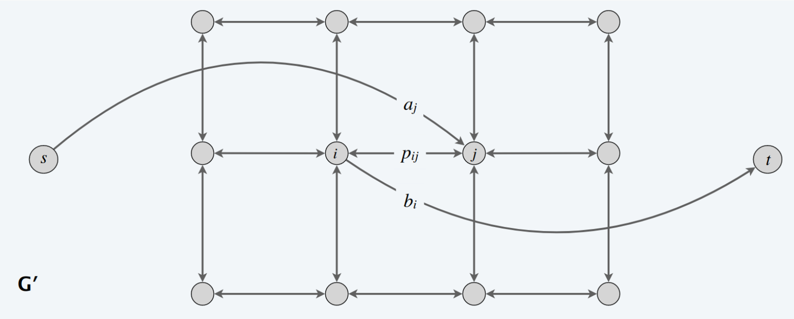
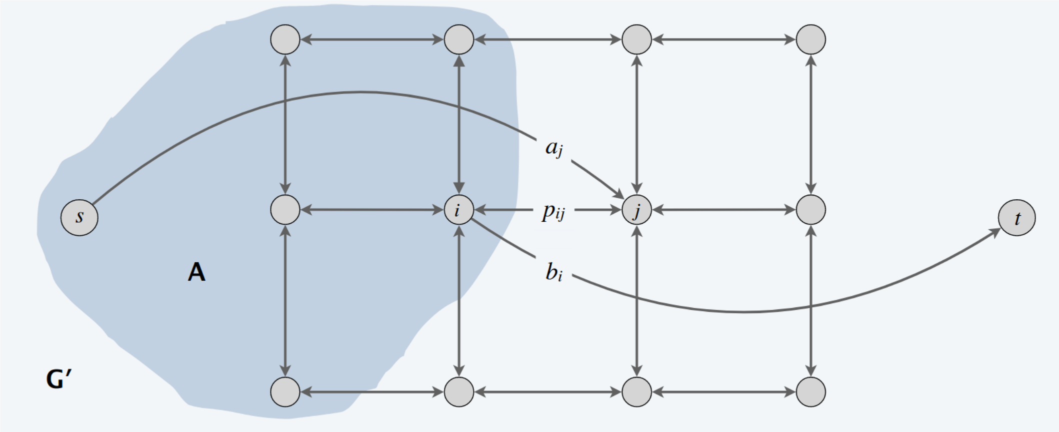



Image Segmentation Minimum Cut Formulation

# Minimum Cut Visualization for Image Segmentation

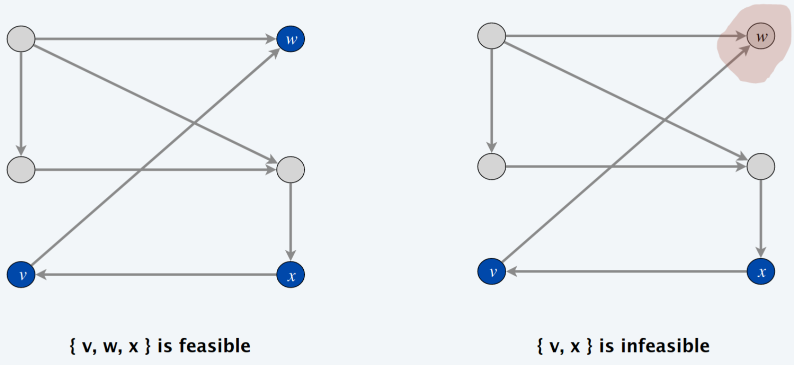
$$\|A, B\| = \sum_{i \in B} a_i + \sum_{j \in A} b_j + \sum_{\substack{(i,j) \in E \\ |A \cap \{i,j\}| = 1}} p_{ij}$$

A is foreground



# Project Selection

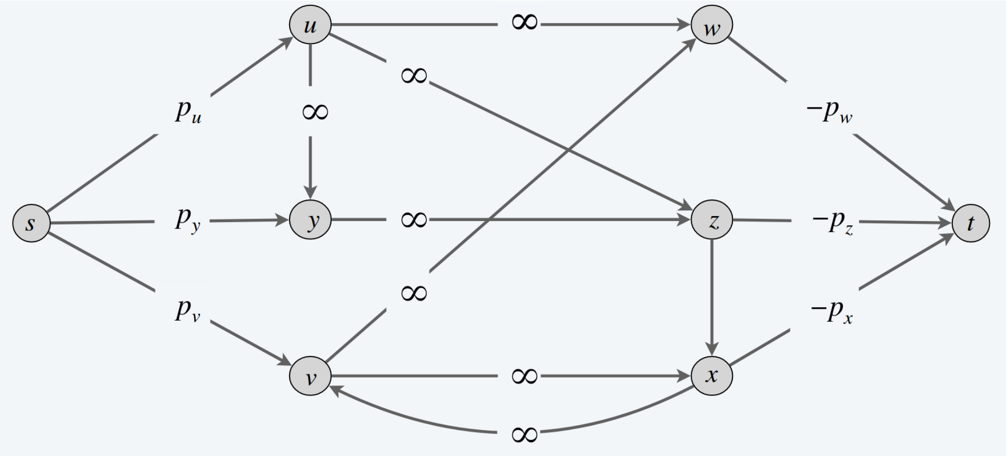
- Set of projects $P$ with revenue $p_v$ for project $v \in P$.
- Prerequisites E: $(v, w) \in E \implies w$ is a prerequisite for $v$.
- A subset of projects $A \subseteq P$ feasible if all prerequisites of $p \in A$ present in $A$.




- Given a sets $P, E$, choose a feasible subset of projects to maximize revenue.

# Minimum Cut Formulation of Project Selection

- We'll model using minimum cut.
- Assign capacity $\infty$ to each prerequisite edge since they must *not* be cut.
- Add $(s, v)$ with capacity $p_v$ if $p_v > 0$ and $(v, t)$ with capacity $p_v$ if $p_v < 0$.



# Minimum Cut Visualization for Project Selection

- Output: $A - \{s\}$ from min-cut $(A, B)$. Due to $\infty$ capacities, $A$ must be feasible.

$$\|A, B\| = \sum_{v \in B: p_v > 0} p_v + \sum_{v \in A: p_v < 0} (-p_v) = \sum_{v: p_v > 0} p_v - \sum_{v \in A} p_v$$

- Min-Cut Capacity is a constant minus total revenue of chosen projects.
- Minimizing this is the same as maximizing revenue.