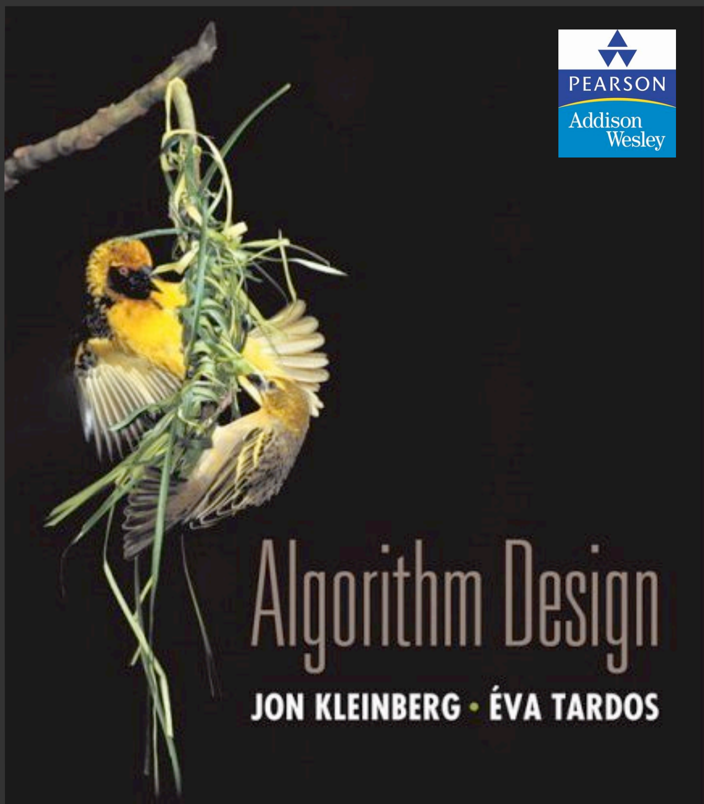


Lecture slides by Kevin Wayne

http://www.cs.princeton.edu/~wayne/kleinberg-tardos

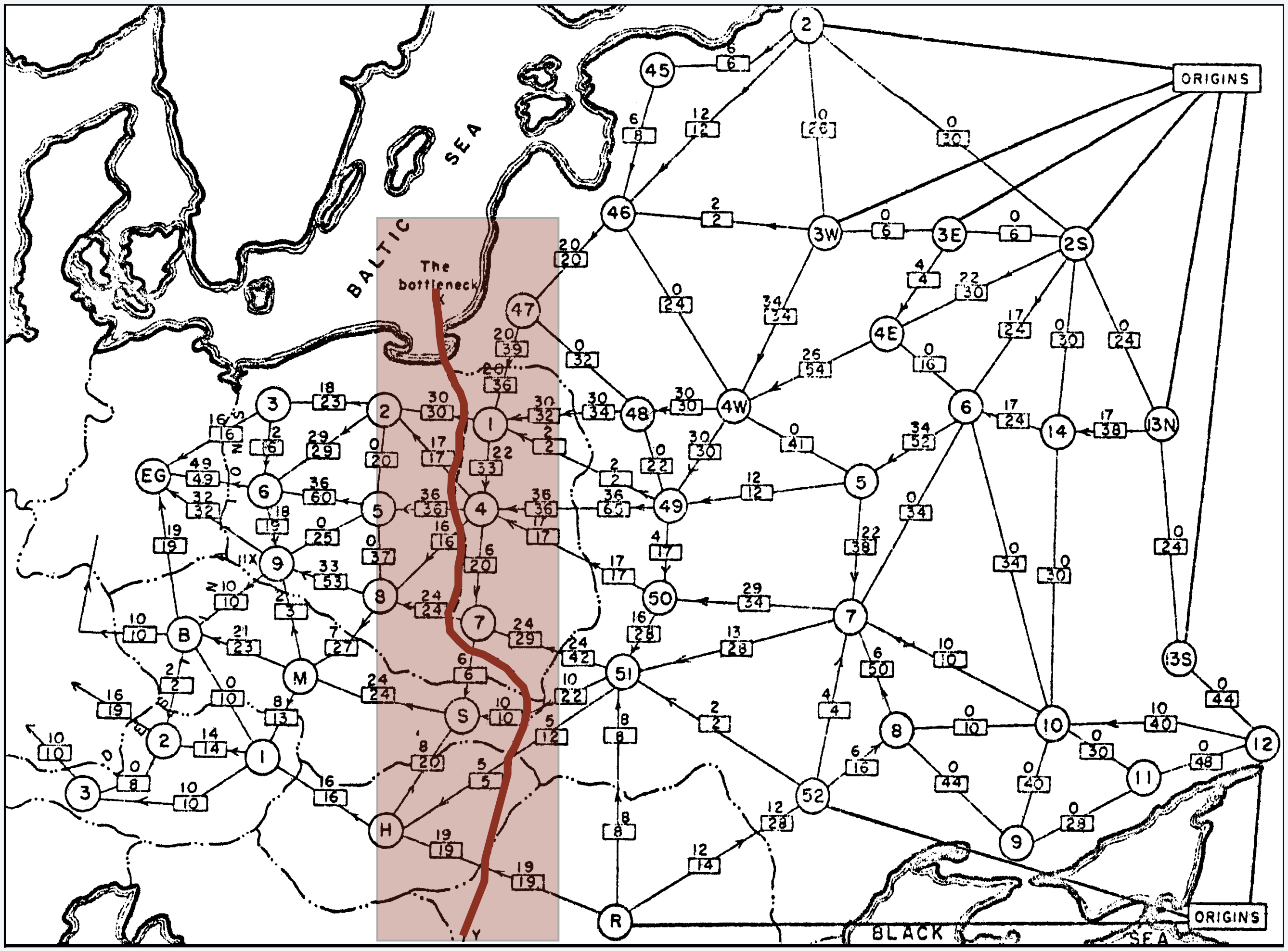
7. Network Flow II

- *bipartite matching*
- *disjoint paths*
- *extensions to max flow*
- *survey design*
- *airline scheduling*
- *image segmentation*
- *project selection*
- *baseball elimination*

Last updated on Mar 31, 2013 3:25 PM

Soviet rail network (1950s)

"Free world" goal. Cut supplies (if cold war turns into real war).

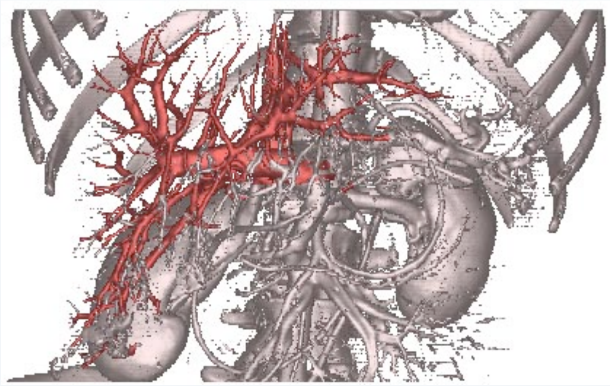


Reference: *On the history of the transportation and maximum flow problems.*
Alexander Schrijver in Math Programming, 91: 3, 2002.

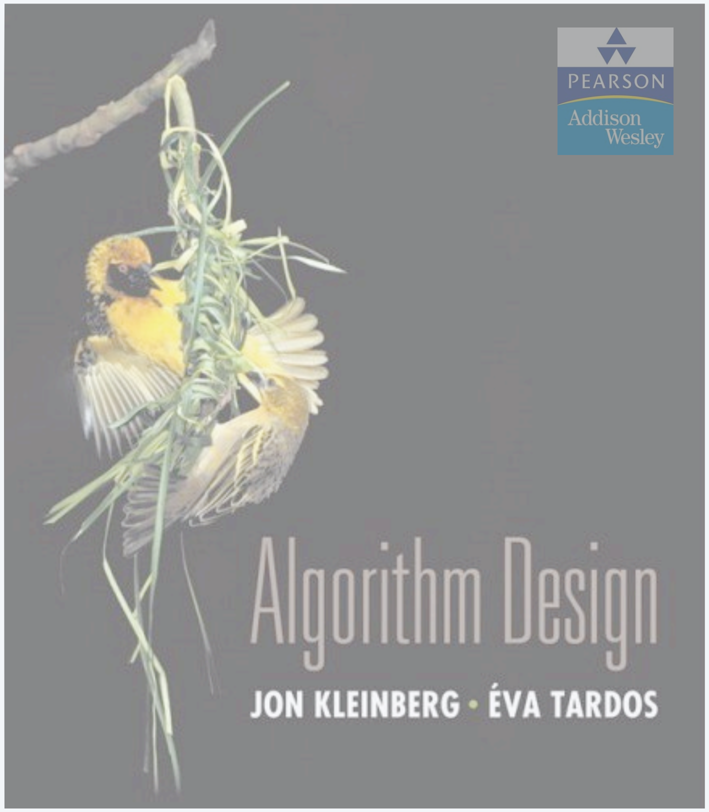
Max-flow and min-cut applications

Max-flow and min-cut are widely applicable problem-solving model.

- Data mining.
- Open-pit mining.
- Bipartite matching.
- Network reliability.
- Baseball elimination.
- Image segmentation.
- Network connectivity.
- Distributed computing.
- Security of statistical data.
- Egalitarian stable matching.
- Network intrusion detection.
- Multi-camera scene reconstruction.
- Sensor placement for homeland security.
- Many, many, more.



liver and hepatic vascularization segmentation



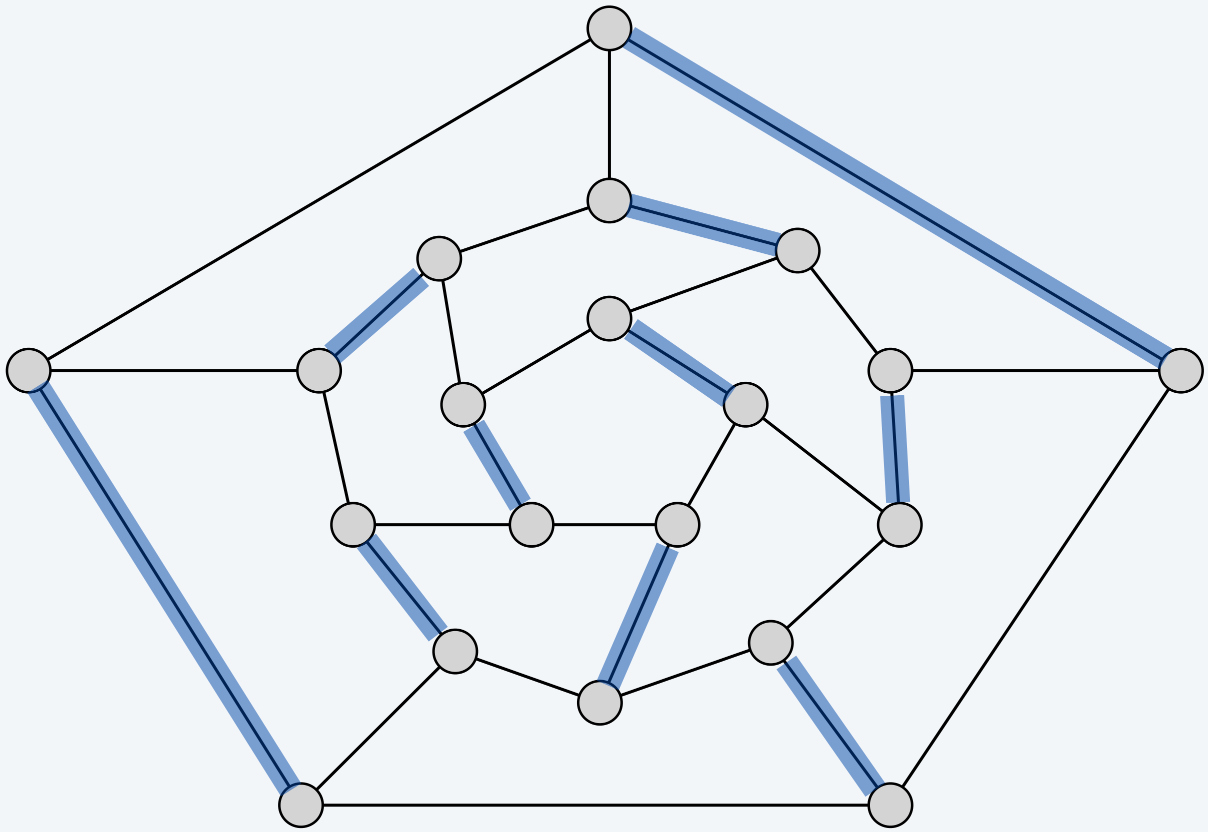
7. NETWORK FLOW II

- *bipartite matching*
- *disjoint paths*
- *extensions to max flow*
- *survey design*
- *airline scheduling*
- *image segmentation*
- *project selection*
- *baseball elimination*

Matching

Def. Given an undirected graph $G = (V, E)$ a subset of edges $M \subseteq E$ is a matching if each node appears in at most one edge in M.

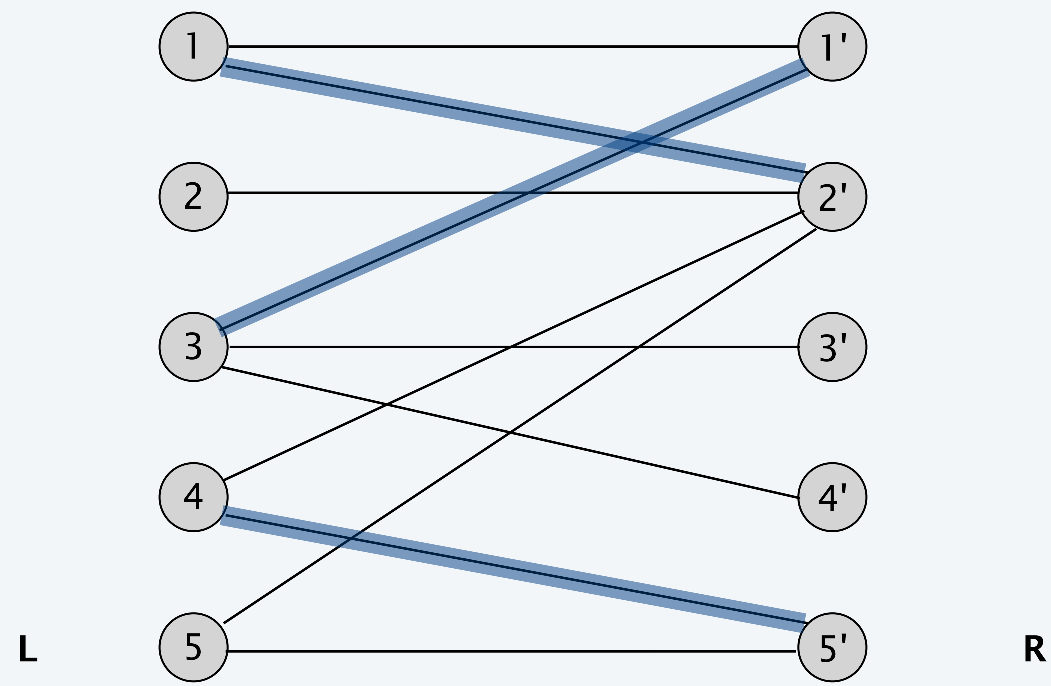
Max matching. Given a graph, find a max cardinality matching.



Bipartite matching

Def. A graph G is bipartite if the nodes can be partitioned into two subsets L and R such that every edge connects a node in L to one in R.

Bipartite matching. Given a bipartite graph $G = (L \cup R, E)$, find a max cardinality matching.

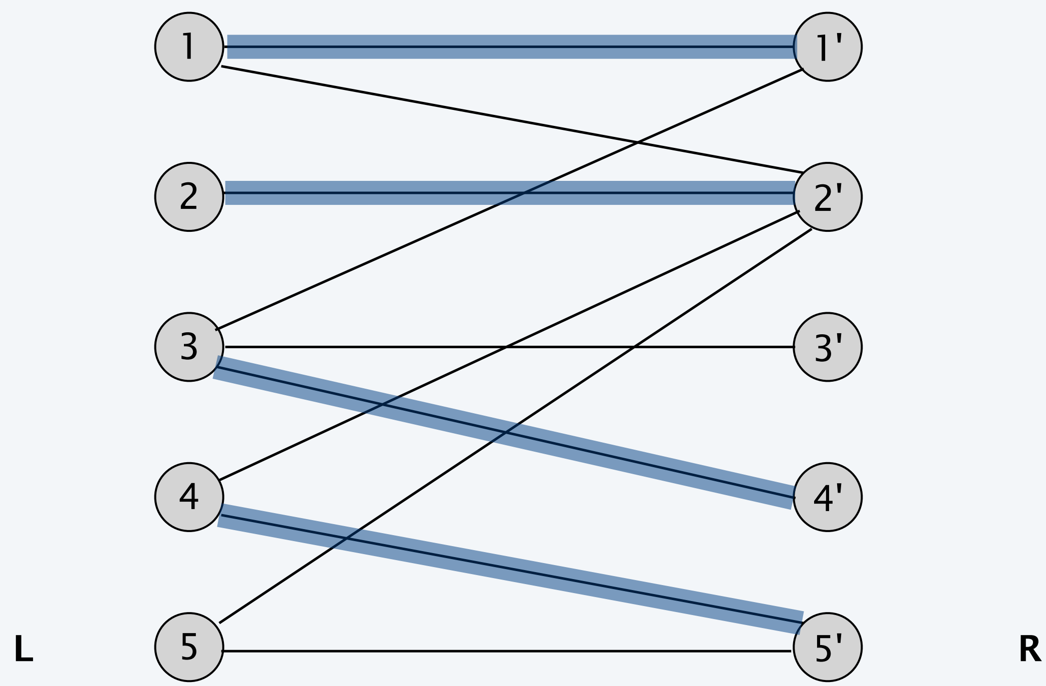


matching: 1–2', 3–1', 4–5'

Bipartite matching

Def. A graph G is bipartite if the nodes can be partitioned into two subsets L and R such that every edge connects a node in L to one in R.

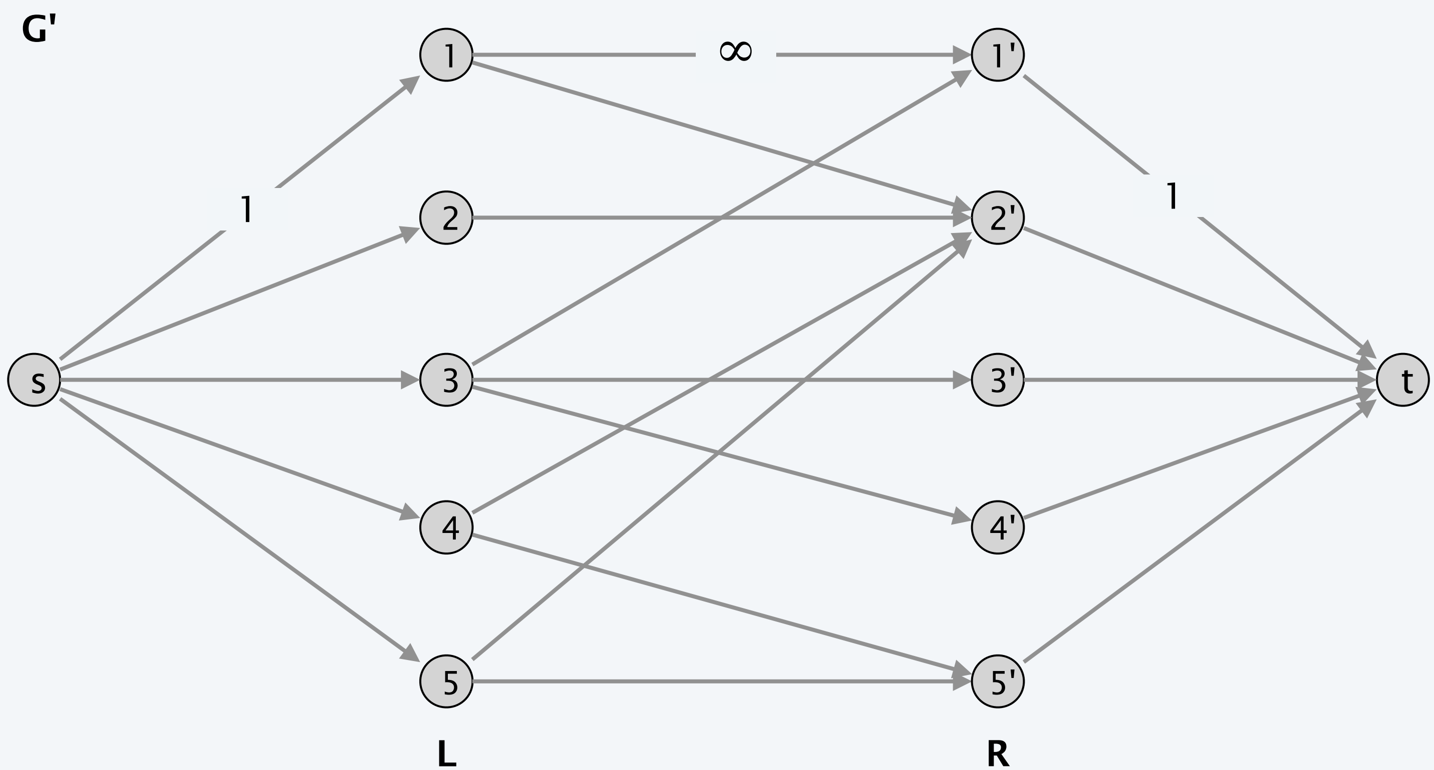
Bipartite matching. Given a bipartite graph $G = (L \cup R, E)$, find a max cardinality matching.



matching: 1–1', 2–2', 3–4', 4–5'

Bipartite matching: max flow formulation

- Create digraph $G' = (L \cup R \cup \{s, t\},\ E'\ )$.
- Direct all edges from L to R, and assign infinite (or unit) capacity.
- Add source s, and unit capacity edges from s to each node in L.
- Add sink t, and unit capacity edges from each node in R to t.

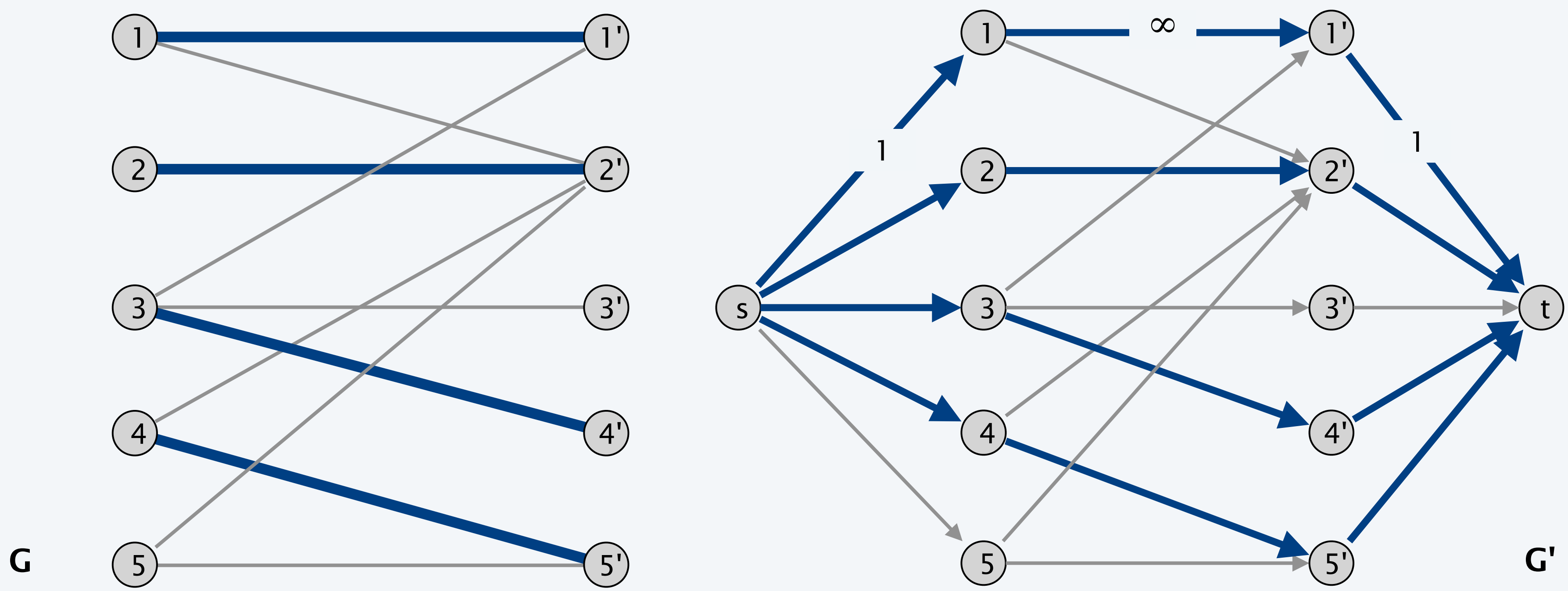


Max flow formulation: proof of correctness

Theorem. Max cardinality of a matching in G = value of max flow in G'.

Pf. $\leq$

- Given a max matching M of cardinality k.
- Consider flow f that sends 1 unit along each of k paths.
- f is a flow, and has value k. ▪

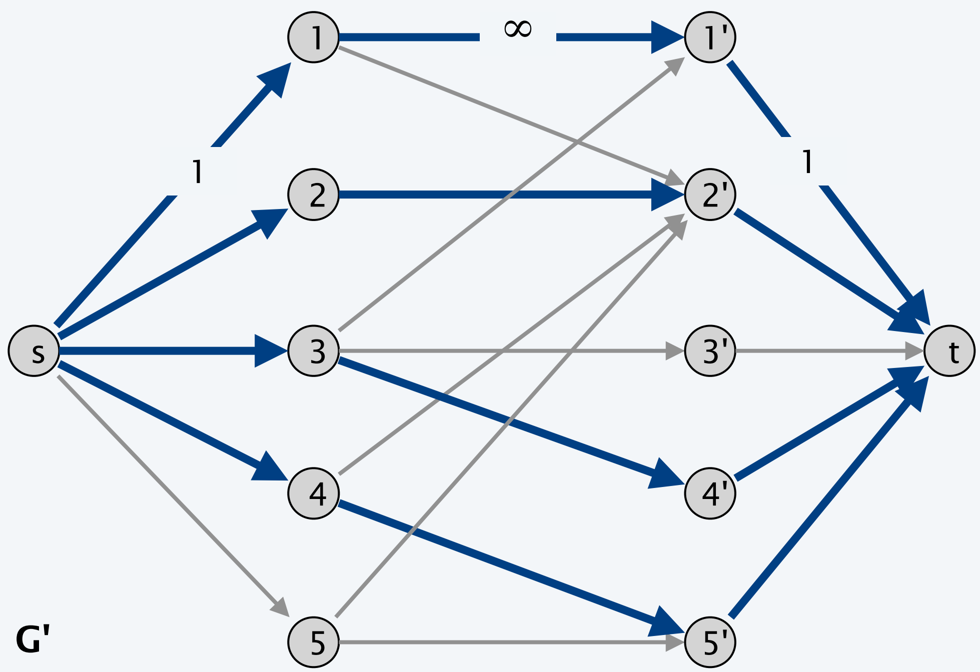


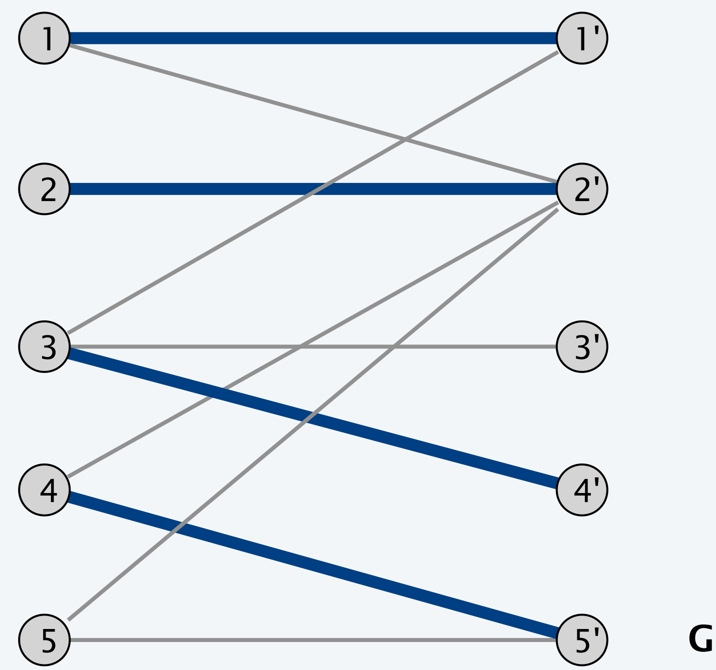
Max flow formulation: proof of correctness

Theorem. Max cardinality of a matching in G = value of max flow in G'.

Pf. $\geq$

- Let f be a max flow in G' of value k.
- Integrality theorem $\Rightarrow$ k is integral and can assume f is 0-1.
- Consider M = set of edges from L to R with $f(e) = 1$.
  - each node in L and R participates in at most one edge in M
  - $|M| = k$: consider cut $(L \cup s, R \cup t)$ ▪





Perfect matching in a bipartite graph

Def. Given a graph $G = (V, E)$ a subset of edges $M \subseteq E$ is a perfect matching if each node appears in exactly one edge in M.

Q. When does a bipartite graph have a perfect matching?

Structure of bipartite graphs with perfect matchings.

- Clearly we must have $|L| = |R|$.
- What other conditions are necessary?
- What conditions are sufficient?

Perfect matching in a bipartite graph

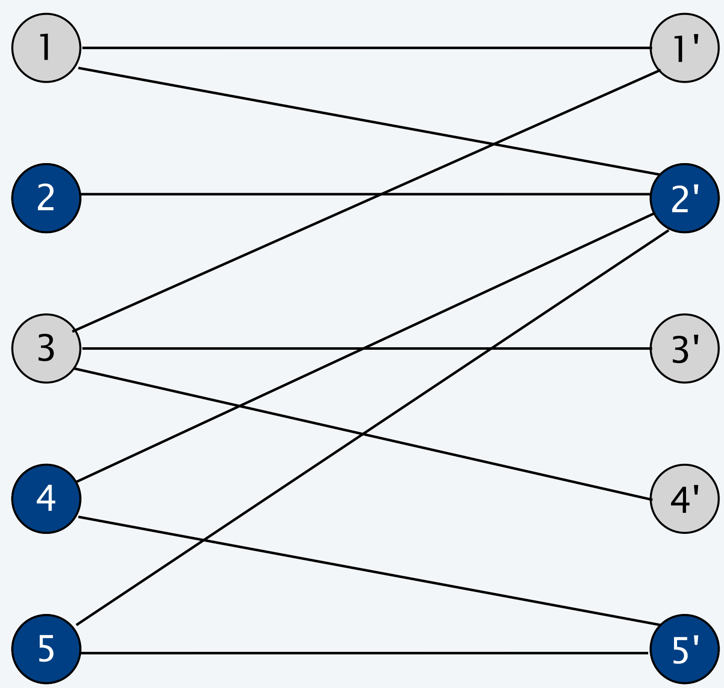
Notation. Let S be a subset of nodes, and let $N(S)$ be the set of nodes adjacent to nodes in S.

Observation. If a bipartite graph $G = (L \cup R, E)$ has a perfect matching, then $|N(S)| \geq |S|$ for all subsets $S \subseteq L$.

Pf. Each node in S has to be matched to a different node in $N(S)$. ▪

S = { 2, 4, 5 }

N(S) = { 2', 5' }



no perfect matching

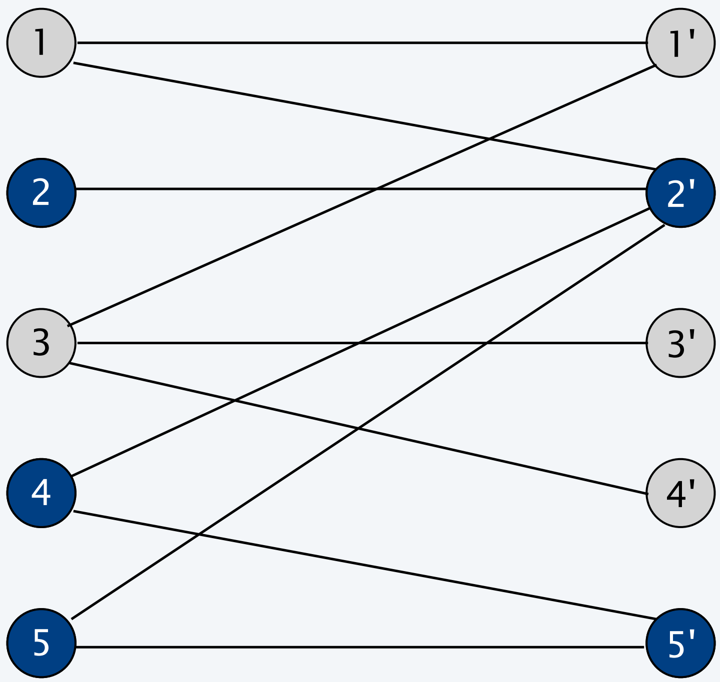
Hall's theorem

Theorem. Let $G = (L \cup R, E)$ be a bipartite graph with $|L| = |R|$.
G has a perfect matching iff $|N(S)| \geq |S|$ for all subsets $S \subseteq L$.

Pf. $\Rightarrow$ This was the previous observation.

S = { 2, 4, 5 }

N(S) = { 2', 5' }

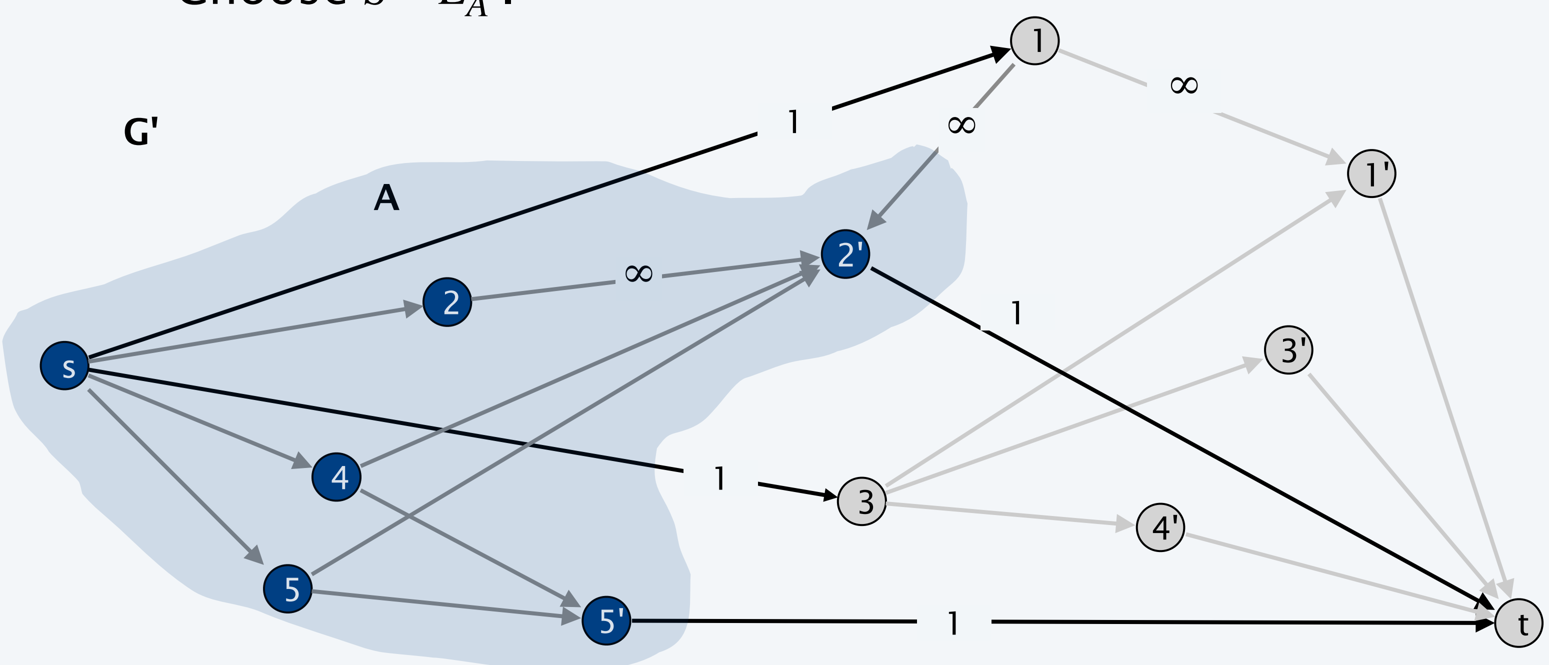


no perfect matching

Proof of Hall's theorem

Pf. $\Leftarrow$ Suppose G does not have a perfect matching.

- Formulate as a max flow problem and let (A, B) be min cut in G'.
- By max-flow min-cut theorem, $cap(A, B) < |L|$.
- Define $L_A = L \cap A$, $L_B = L \cap B$, $R_A = R \cap A$.
- $cap(A, B) = |L_B| + |R_A|$.
- Since min cut can't use ∞ edges: $N(L_A) \subseteq R_A$.
- $|N(L_A)| \leq |R_A| = cap(A, B) - |L_B| < |L| - |L_B| = |L_A|$.
- Choose $S = L_A$. ▪



$L_A = \{2, 4, 5\}$

$L_B = \{1, 3\}$

$R_A = \{2', 5'\}$

$N(L_A) = \{2', 5'\}$

Bipartite matching running time

Theorem. The Ford-Fulkerson algorithm solves the bipartite matching problem in $O(m\, n)$ time.

Theorem. [Hopcroft-Karp 1973] The bipartite matching problem can be solved in $O(m\, n^{1/2})$ time.

SIAM J. COMPUT.
Vol. 2, No. 4, December 1973

AN $n^{5/2}$ ALGORITHM FOR MAXIMUM MATCHINGS IN BIPARTITE GRAPHS*

JOHN E. HOPCROFT† AND RICHARD M. KARP‡

Abstract. The present paper shows how to construct a maximum matching in a bipartite graph with n vertices and m edges in a number of computation steps proportional to $(m + n)\sqrt{n}$.

Key words. algorithm, algorithmic analysis, bipartite graphs, computational complexity, graphs, matching

Nonbipartite matching

Nonbipartite matching. Given an undirected graph (not necessarily bipartite), find a matching of maximum cardinality.

- Structure of nonbipartite graphs is more complicated.
- But well-understood. [Tutte-Berge, Edmonds-Galai]
- Blossom algorithm: $O(n^4)$. [Edmonds 1965]
- Best known: $O(m\, n^{1/2})$. [Micali-Vazirani 1980, Vazirani 1994]

PATHS, TREES, AND FLOWERS

JACK EDMONDS

1. Introduction. A *graph G* for purposes here is a finite set of elements called *vertices* and a finite set of elements called *edges* such that each edge *meets* exactly two vertices, called the *end-points* of the edge. An edge is said to *join* its end-points.

A *matching* in *G* is a subset of its edges such that no two meet the same vertex. We describe an efficient algorithm for finding in a given graph a matching of maximum cardinality. This problem was posed and partly solved by C. Berge; see Sections 3.7 and 3.8.

COMBINATORICA
Akadémiai Kiadó – Springer-Verlag

COMBINATORICA 14 (1) (1994) 71–109

A THEORY OF ALTERNATING PATHS AND BLOSSOMS FOR PROVING CORRECTNESS OF THE $O(\sqrt{V}E)$ GENERAL GRAPH MAXIMUM MATCHING ALGORITHM

VIJAY V. VAZIRANI[1]

Received December 30, 1989
Revised June 15, 1993

Historical significance (Jack Edmonds 1965)

2. Digression. An explanation is due on the use of the words "efficient algorithm." First, what I present is a conceptual description of an algorithm and not a particular formalized algorithm or "code."

For practical purposes computational details are vital. However, my purpose is only to show as attractively as I can that there is an efficient algorithm. According to the dictionary, "efficient" means "adequate in operation or performance." This is roughly the meaning I want—in the sense that it is conceivable for maximum matching to have no efficient algorithm. Perhaps a better word is "good."

I am claiming, as a mathematical result, the existence of a *good* algorithm for finding a maximum cardinality matching in a graph.

There is an obvious finite algorithm, but that algorithm increases in difficulty exponentially with the size of the graph. It is by no means obvious whether *or not* there exists an algorithm whose difficulty increases only algebraically with the size of the graph.

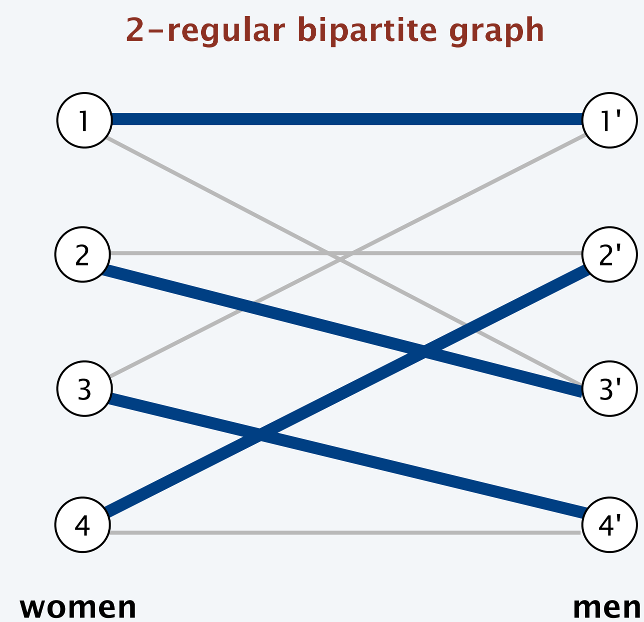
k-regular bipartite graphs

Dancing problem.

- Exclusive Ivy league party attended by n men and n women.
- Each man knows exactly k women; each woman knows exactly k men.
- Acquaintances are mutual.
- Is it possible to arrange a dance so that each woman dances with a different man that she knows?

Mathematical reformulation. Does every k-regular bipartite graph have a perfect matching?

Ex. Boolean hypercube.



k-regular bipartite graphs have perfect matchings

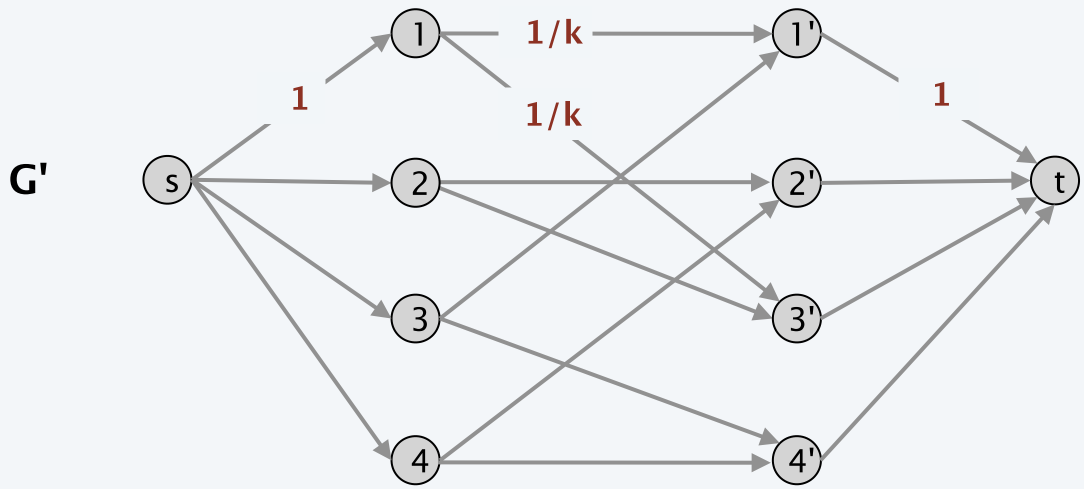
Theorem. Every k-regular bipartite graph G has a perfect matching.

Pf.

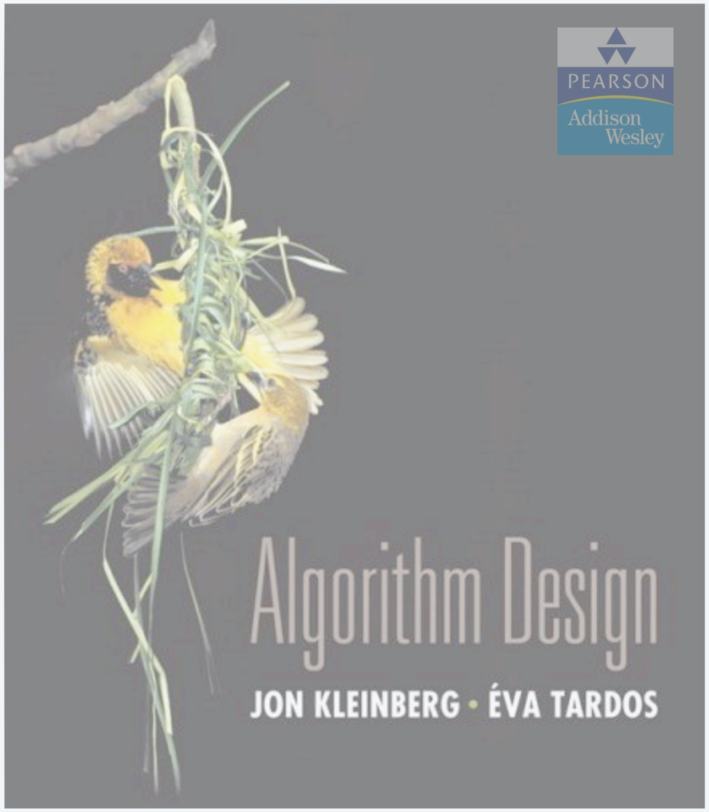
- Size of max matching = value of max flow in G'.
- Consider flow

$$f(u,v) = \begin{cases} 1/k & \text{if } (u,v) \in E \\ 1 & \text{if } u = s \text{ or } v = t \\ 0 & \text{otherwise} \end{cases}$$

- f is a flow in G' and its value = $n \Rightarrow$ perfect matching. ▪



a feasible flow f of value n



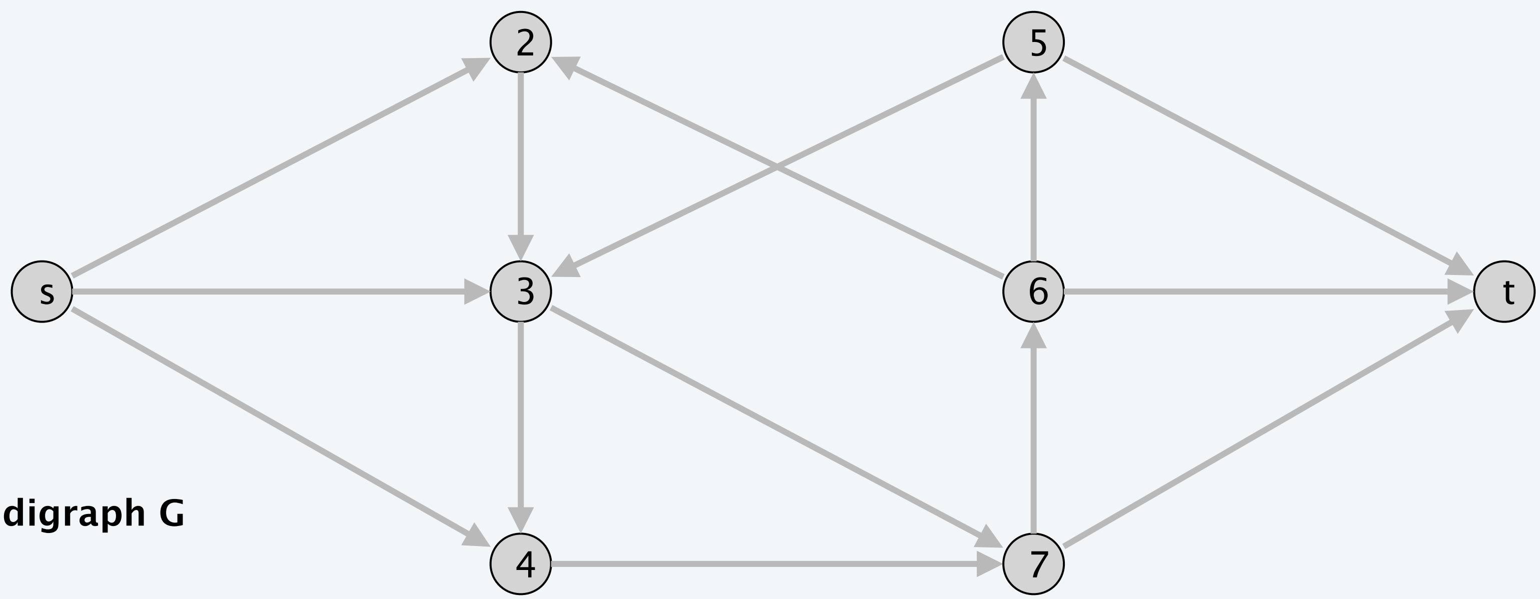
7. Network Flow II

- *bipartite matching*
- *disjoint paths*
- *extensions to max flow*
- *survey design*
- *airline scheduling*
- *image segmentation*
- *project selection*
- *baseball elimination*

Edge-disjoint paths

Def. Two paths are edge-disjoint if they have no edge in common.

Disjoint path problem. Given a digraph $G = (V, E)$ and two nodes s and t, find the max number of edge-disjoint $s \rightsquigarrow t$ paths.



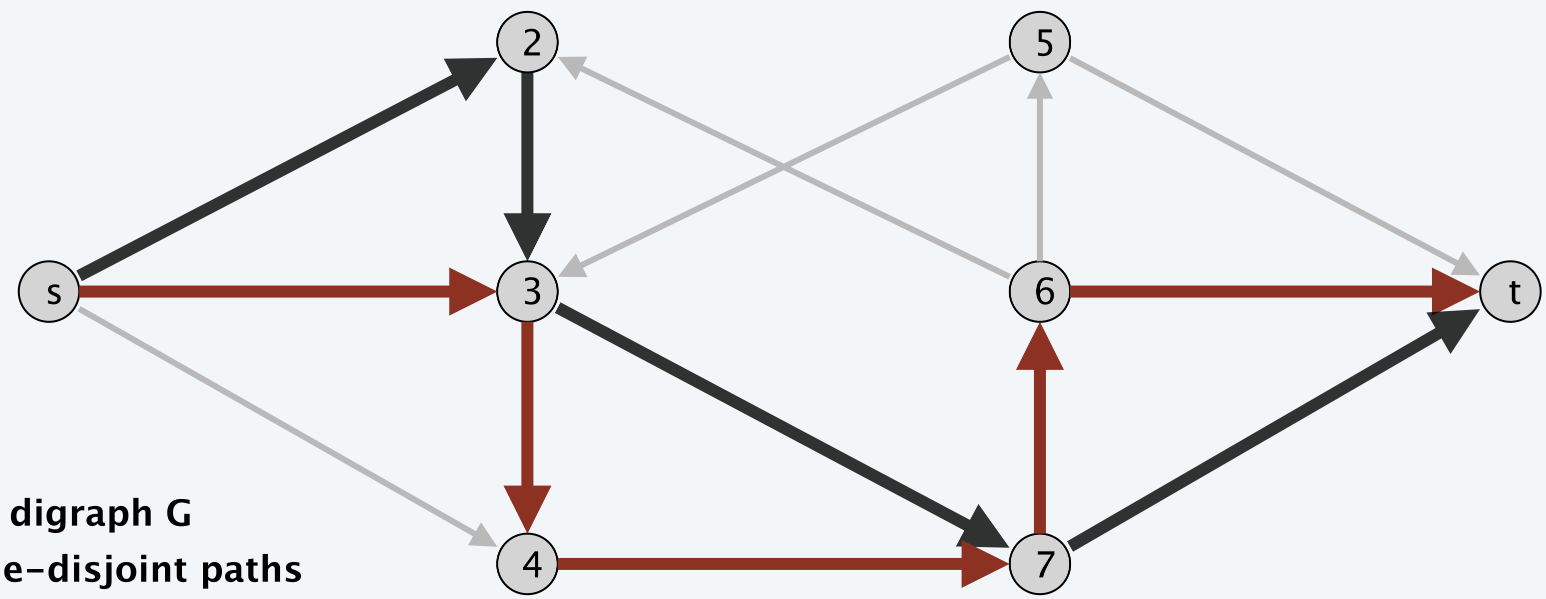
digraph G

Edge-disjoint paths

Def. Two paths are edge-disjoint if they have no edge in common.

Disjoint path problem. Given a digraph $G = (V, E)$ and two nodes s and t, find the max number of edge-disjoint $s \rightsquigarrow t$ paths.

Ex. Communication networks.



digraph G
2 edge-disjoint paths

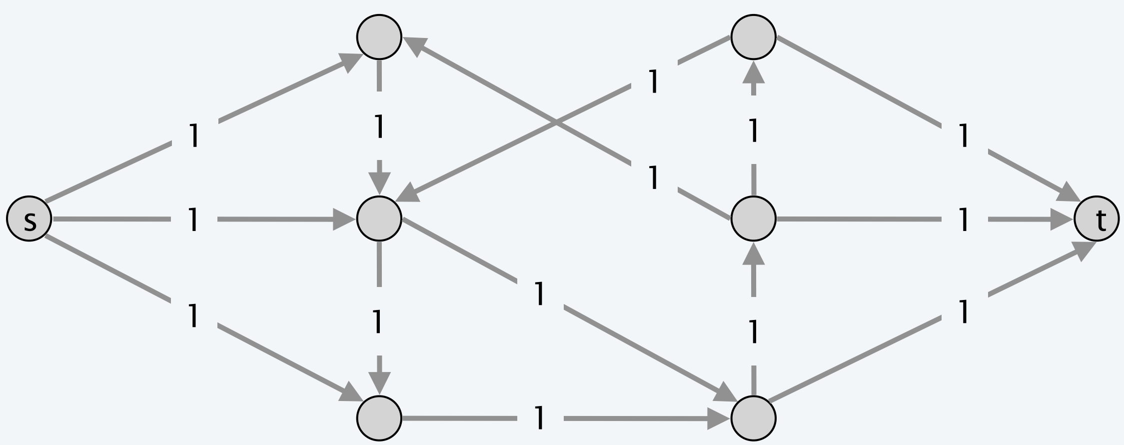
Edge-disjoint paths

Max flow formulation. Assign unit capacity to every edge.

Theorem. Max number edge-disjoint $s \rightsquigarrow t$ paths equals value of max flow.

Pf. $\leq$

- Suppose there are k edge-disjoint $s \rightsquigarrow t$ paths $P_1, \ldots, P_k$.
- Set $f(e) = 1$ if e participates in some path P_j ; else set $f(e) = 0$.
- Since paths are edge-disjoint, f is a flow of value k. ▪



Edge-disjoint paths

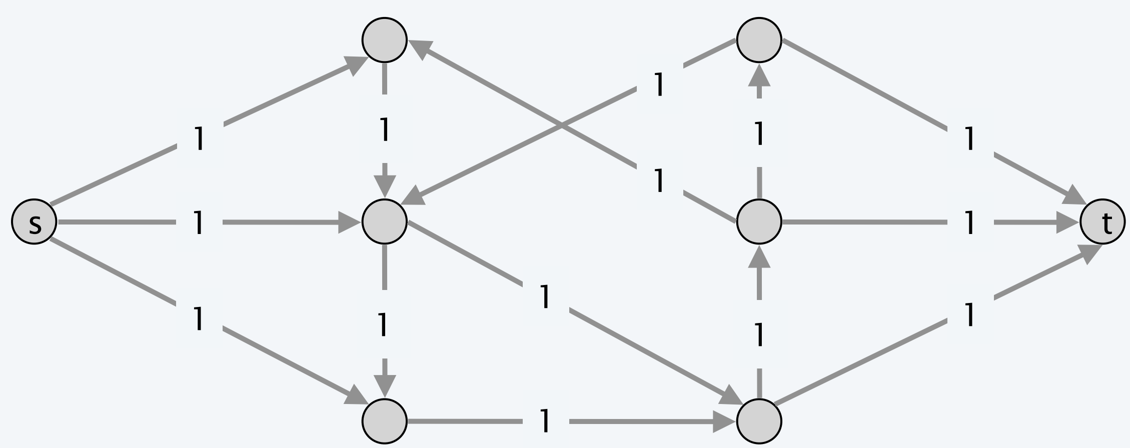
Max flow formulation. Assign unit capacity to every edge.

Theorem. Max number edge-disjoint $s \rightsquigarrow t$ paths equals value of max flow.

Pf. $\geq$

- Suppose max flow value is k.
- Integrality theorem $\Rightarrow$ there exists 0-1 flow f of value k.
- Consider edge (s, u) with $f(s, u) = 1$.
  - by conservation, there exists an edge (u, v) with $f(u, v) = 1$
  - continue until reach t, always choosing a new edge
- Produces k (not necessarily simple) edge-disjoint paths. ▪

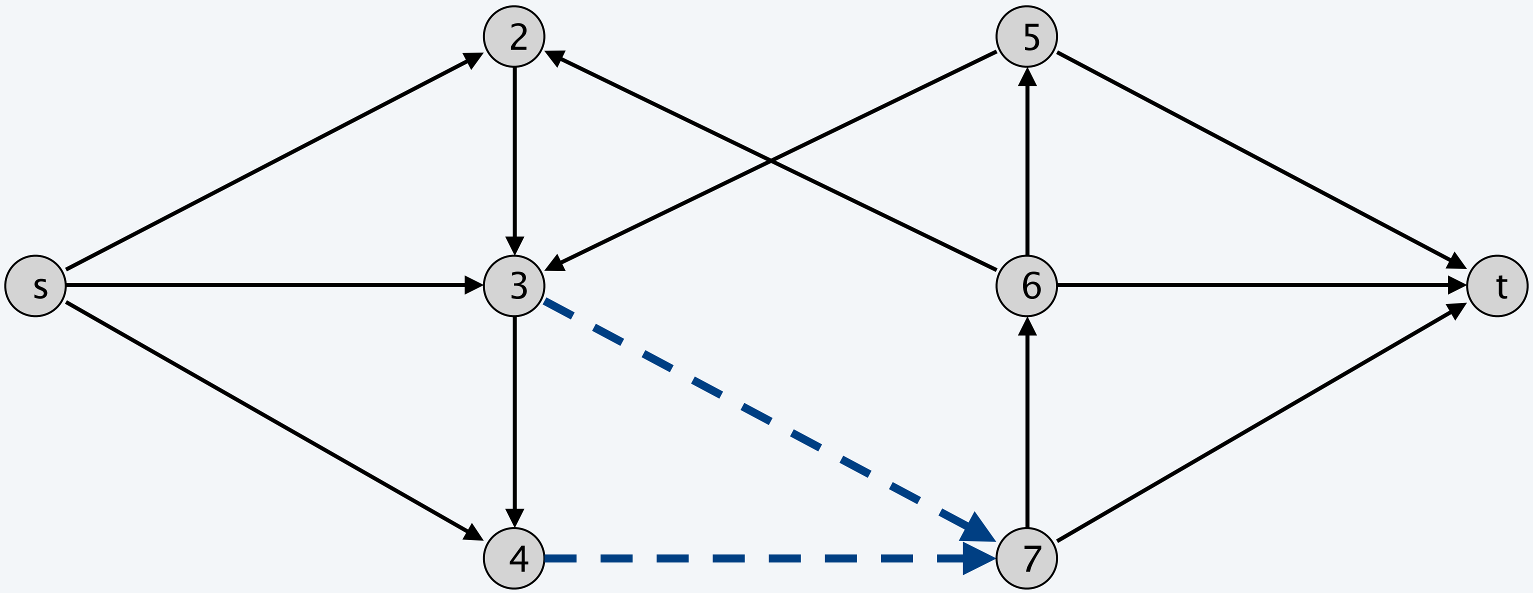
can eliminate cycles to get simple paths in O(mn) time if desired (flow decomposition)



Network connectivity

Def. A set of edges $F \subseteq E$ disconnects t from s if every $s \rightsquigarrow t$ path uses at least one edge in F.

Network connectivity. Given a digraph $G = (V, E)$ and two nodes s and t, find min number of edges whose removal disconnects t from s.

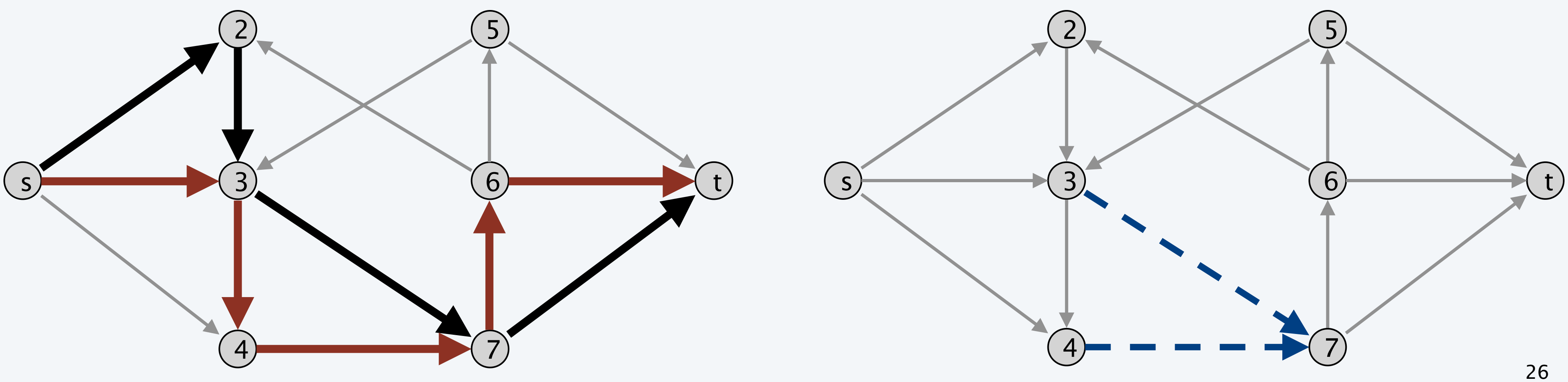


Menger's theorem

Theorem. [Menger 1927] The max number of edge-disjoint $s \rightsquigarrow t$ paths is equal to the min number of edges whose removal disconnects t from s.

Pf. $\leq$

- Suppose the removal of $F \subseteq E$ disconnects t from s, and $|F| = k$.
- Every $s \rightsquigarrow t$ path uses at least one edge in F.
- Hence, the number of edge-disjoint paths is $\leq k$. ▪

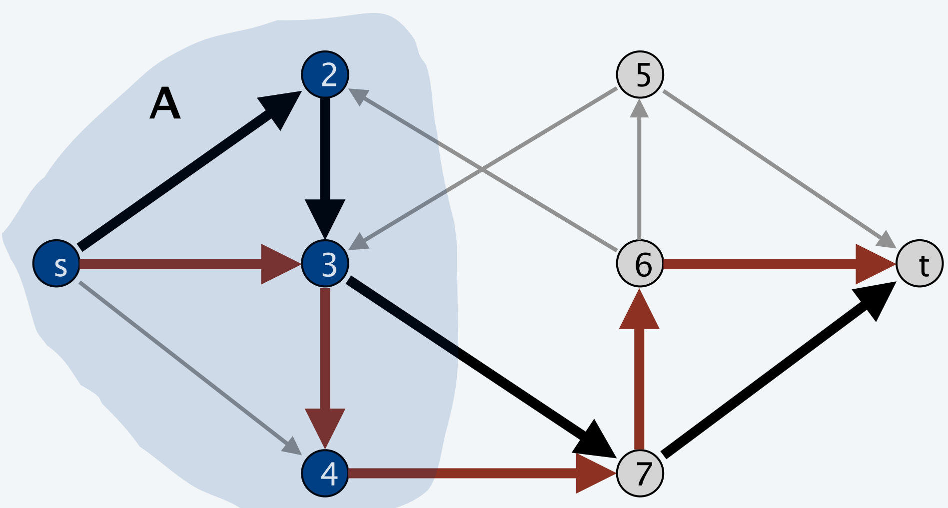


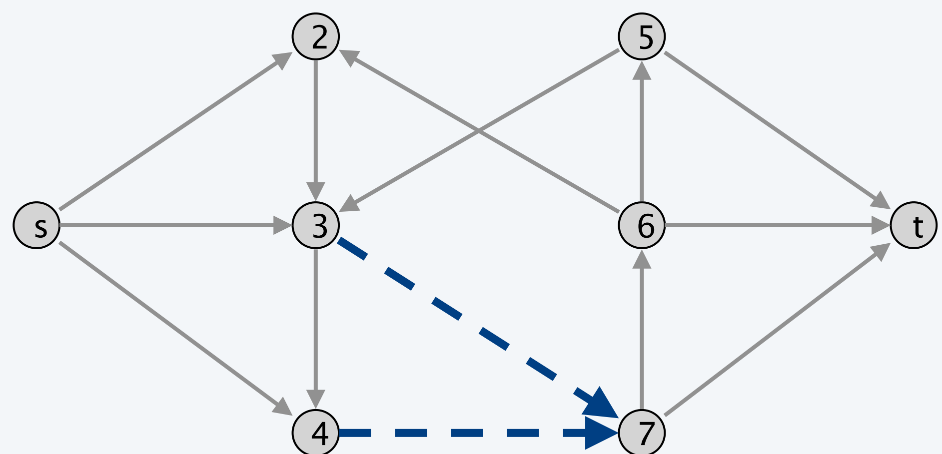
Menger's theorem

Theorem. [Menger 1927] The max number of edge-disjoint $s \rightsquigarrow t$ paths equals the min number of edges whose removal disconnects t from s.

Pf. $\geq$

- Suppose max number of edge-disjoint paths is k.
- Then value of max flow = k.
- Max-flow min-cut theorem $\Rightarrow$ there exists a cut (A, B) of capacity k.
- Let F be set of edges going from A to B.
- $|F| = k$ and disconnects t from s. ▪

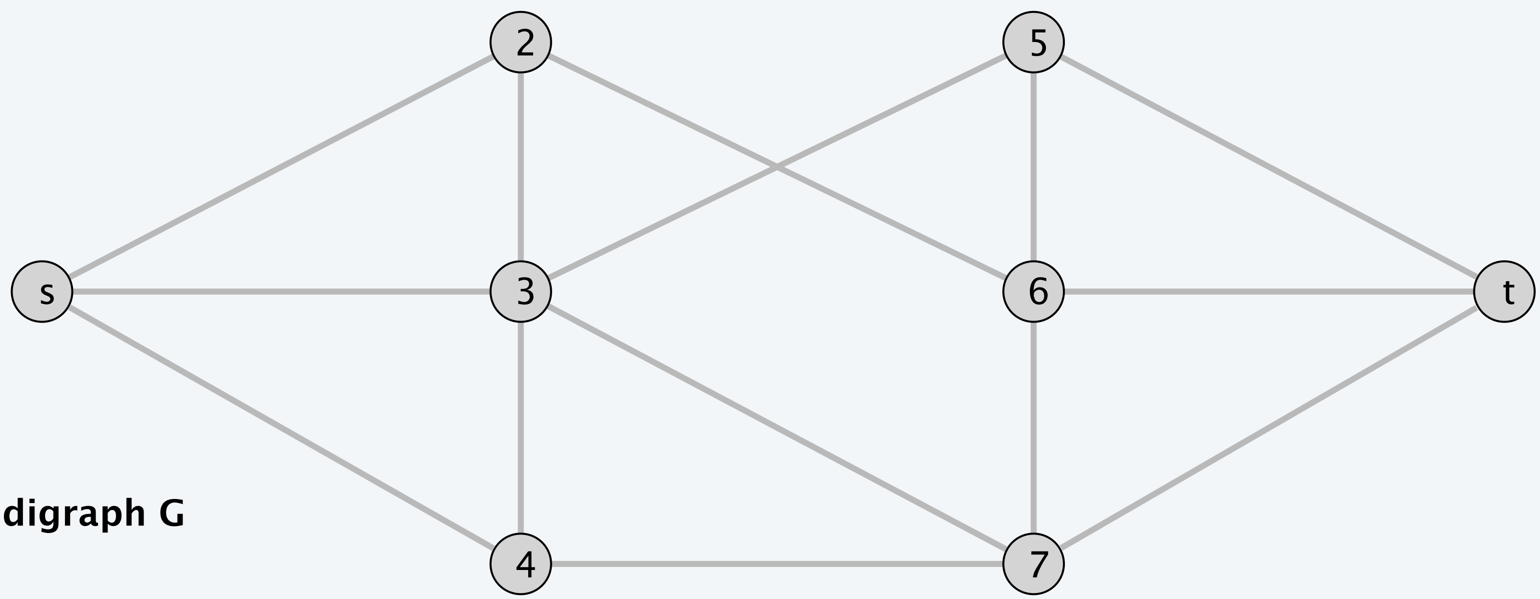




Edge-disjoint paths in undirected graphs

Def. Two paths are edge-disjoint if they have no edge in common.

Disjoint path problem in undirected graphs. Given a graph $G = (V, E)$ and two nodes s and t, find the max number of edge-disjoint s-t paths.

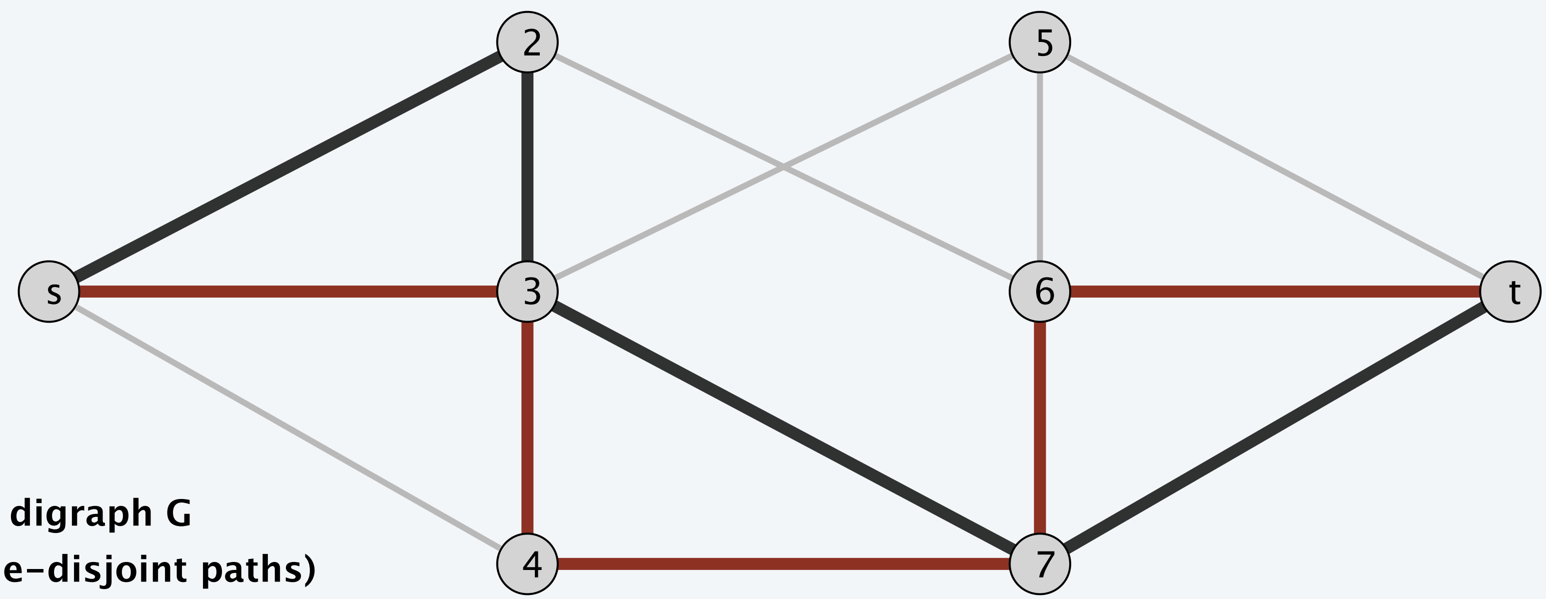


digraph G

Edge-disjoint paths in undirected graphs

Def. Two paths are edge-disjoint if they have no edge in common.

Disjoint path problem in undirected graphs. Given a graph $G = (V, E)$ and two nodes s and t, find the max number of edge-disjoint s-t paths.

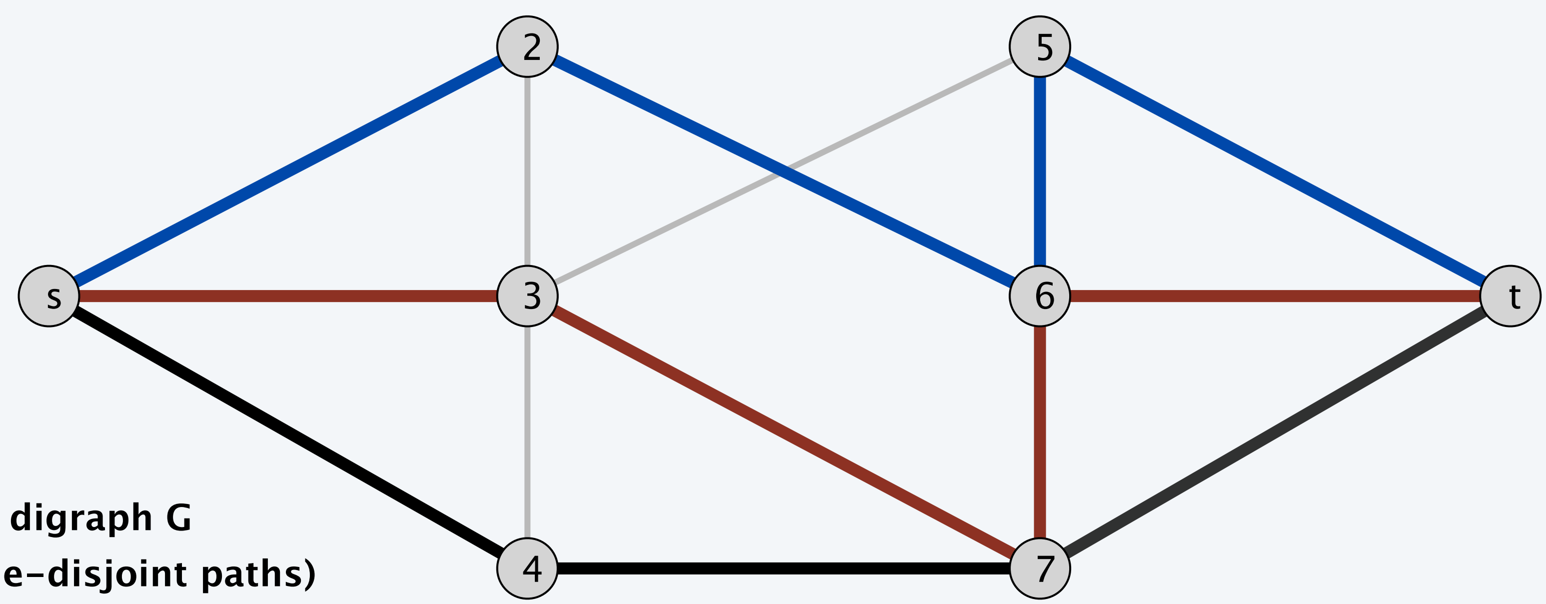


digraph G
(2 edge-disjoint paths)

Edge-disjoint paths in undirected graphs

Def. Two paths are edge-disjoint if they have no edge in common.

Disjoint path problem in undirected graphs. Given a graph $G = (V, E)$ and two nodes s and t, find the max number of edge-disjoint s-t paths.



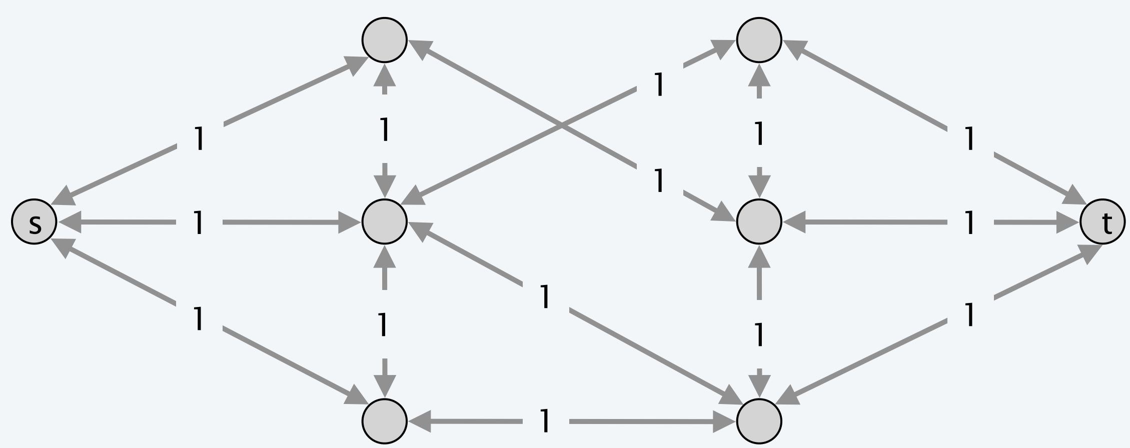
digraph G
(3 edge-disjoint paths)

Edge-disjoint paths in undirected graphs

Max flow formulation. Replace edge edge with two antiparallel edges and assign unit capacity to every edge.

Observation. Two paths P_1 and P_2 may be edge-disjoint in the digraph but not edge-disjoint in the undirected graph.

if P_1 uses edge (u, v)
and P_2 uses its antiparallel edge (v, u)



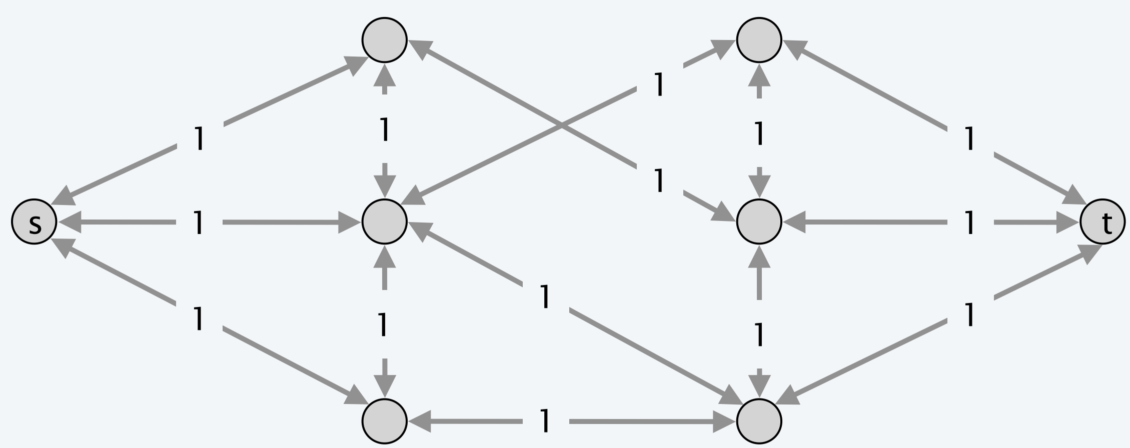
Edge-disjoint paths in undirected graphs

Max flow formulation. Replace edge edge with two antiparallel edges and assign unit capacity to every edge.

Lemma. In any flow network, there exists a maximum flow f in which for each pair of antiparallel edges e and e', either $f(e) = 0$ or $f(e') = 0$ or both. Moreover, integrality theorem still holds.

Pf. [by induction on number of such pairs of antiparallel edges]

- Suppose $f(e) > 0$ and $f(e') > 0$ for a pair of antiparallel edges e and e'.
- Set $f(e) = f(e) - \delta$ and $f(e') = f(e') - \delta$, where $\delta = \min \{ f(e), f(e') \}$.
- f is still a flow of the same value but has one fewer such pair. ▪

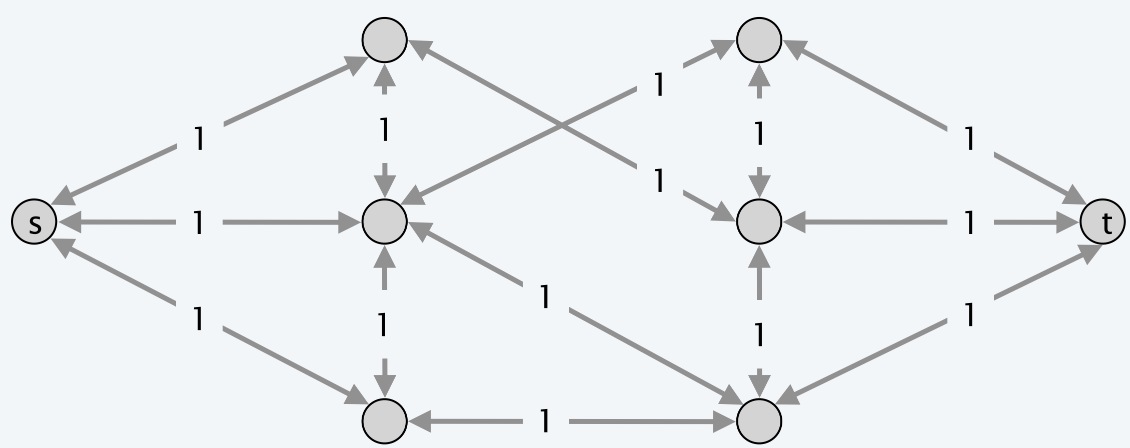


Edge-disjoint paths in undirected graphs

Max flow formulation. Replace edge edge with two antiparallel edges and assign unit capacity to every edge.

Lemma. In any flow network, there exists a maximum flow f in which for each pair of antiparallel edges e and e', either $f(e) = 0$ or $f(e') = 0$ or both. Moreover, integrality theorem still holds.

Theorem. Max number edge-disjoint $s \rightsquigarrow t$ paths equals value of max flow.
Pf. Similar to proof in digraphs; use lemma.

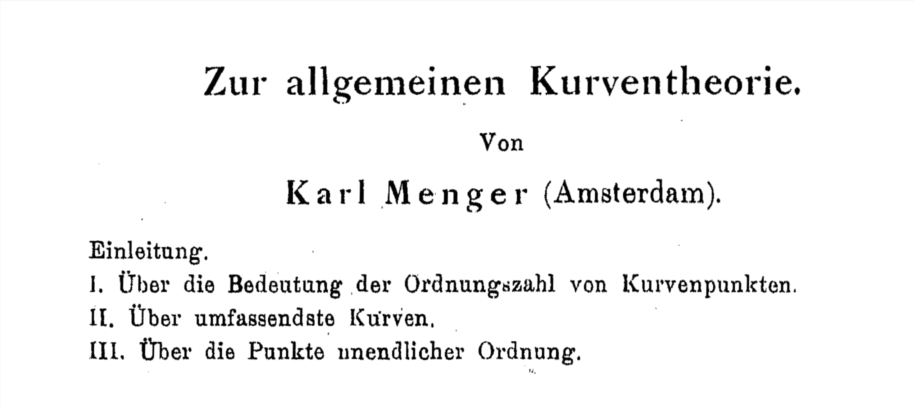


Menger's theorems

Theorem. Given an undirected graph with two nodes s and t, the max number of edge-disjoint s-t paths equals the min number of edges whose removal disconnects s and t.

Theorem. Given a undirected graph with two nonadjacent nodes s and t, the max number of internally node-disjoint s-t paths equals the min number of internal nodes whose removal disconnects s and t.

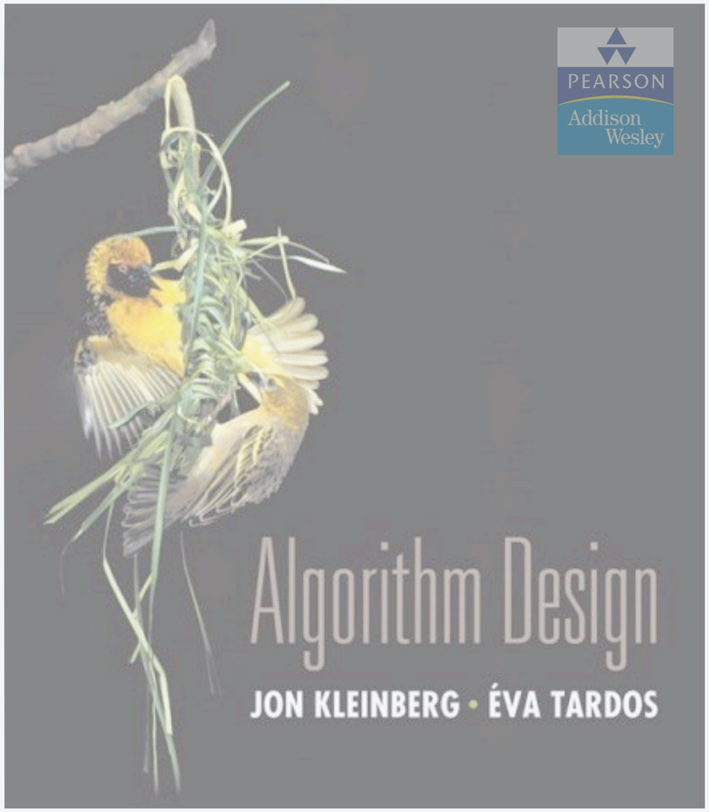
Theorem. Given an directed graph with two nonadjacent nodes s and t, the max number of internally node-disjoint $s \rightsquigarrow t$ paths equals the min number of internal nodes whose removal disconnects t from s.


Zur allgemeinen Kurventheorie.

Von

Karl Menger (Amsterdam).

Einleitung.
I. Über die Bedeutung der Ordnungszahl von Kurvenpunkten.
II. Über umfassendste Kurven.
III. Über die Punkte unendlicher Ordnung.



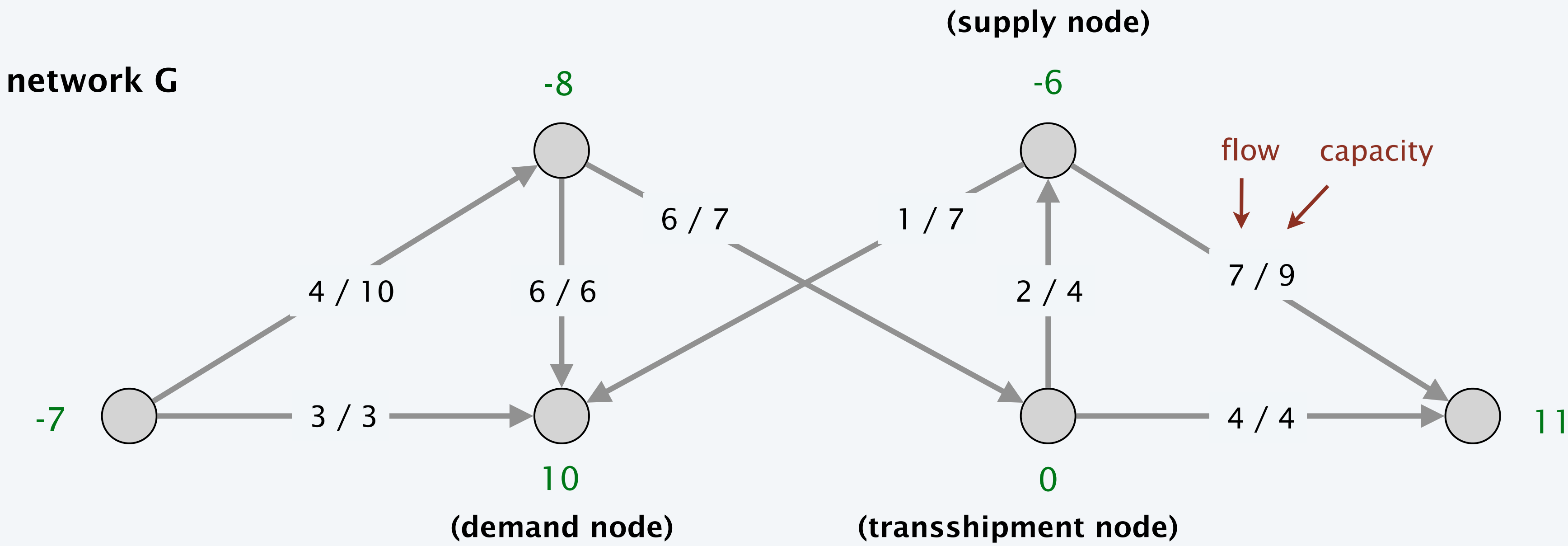
7. Network Flow II

- *bipartite matching*
- *disjoint paths*
- *extensions to max flow*
- *survey design*
- *airline scheduling*
- *image segmentation*
- *project selection*
- *baseball elimination*

Circulation with demands

Def. Given a digraph $G = (V, E)$ with nonnegative edge capacities $c(e)$ and node supply and demands $d(v)$, a circulation is a function that satisfies:

- For each $e \in E$: $\quad 0 \;\le\; f(e) \;\le\; c(e)$ (capacity)
- For each $v \in V$: $\quad \sum_{e \text{ in to } v} f(e) \;-\; \sum_{e \text{ out of } v} f(e) \;=\; d(v)$ (conservation)

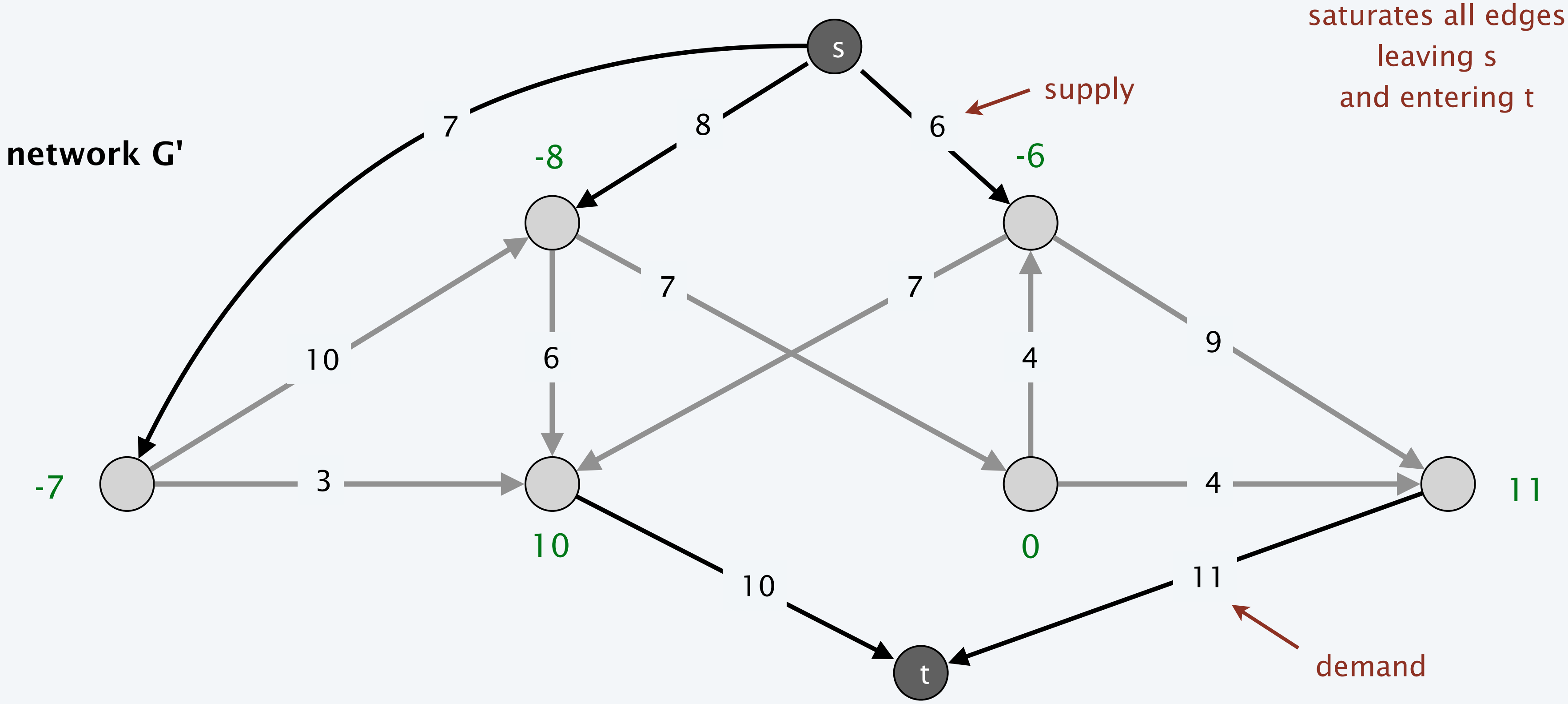


Circulation with demands: max-flow formulation

- Add new source s and sink t.
- For each v with $d(v) < 0$, add edge (s, v) with capacity $-d(v)$.
- For each v with $d(v) > 0$, add edge (v, t) with capacity $d(v)$.

Claim. G has circulation iff G' has max flow of value $D = \sum_{v\,:\,d(v)>0} d(v) = \sum_{v\,:\,d(v)<0} -d(v)$

saturates all edges leaving s and entering t



Circulation with demands

Integrality theorem. If all capacities and demands are integers, and there exists a circulation, then there exists one that is integer-valued.

Pf. Follows from max-flow formulation + integrality theorem for max flow.

Theorem. Given (V, E, c, d), there does not exists a circulation iff there exists a node partition (A, B) such that $\Sigma_{v \in B} d(v) > cap(A, B)$.

Pf sketch. Look at min cut in G'.

demand by nodes in B exceeds
supply of nodes in B plus
max capacity of edges going from A to B

Circulation with demands and lower bounds

Feasible circulation.

- Directed graph $G = (V, E)$.
- Edge capacities $c(e)$ and lower bounds $\ell(e)$ for each edge $e \in E$.
- Node supply and demands $d(v)$ for each node $v \in V$.

Def. A circulation is a function that satisfies:

- For each $e \in E$: $\ell(e) \le f(e) \le c(e)$ (capacity)
- For each $v \in V$: $\sum_{e \text{ in to } v} f(e) - \sum_{e \text{ out of } v} f(e) = d(v)$ (conservation)

Circulation problem with lower bounds. Given (V, E, ℓ, c, d), does there exists a feasible circulation?

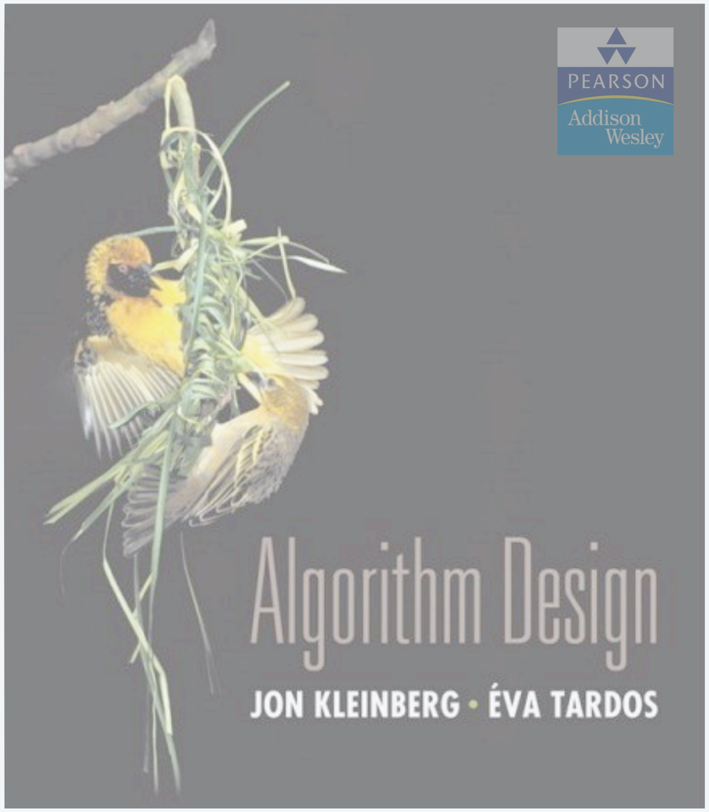
Circulation with demands and lower bounds

Max flow formulation. Model lower bounds as circulation with demands.

- Send $\ell(e)$ units of flow along edge e.
- Update demands of both endpoints.

Theorem. There exists a circulation in G iff there exists a circulation in G'. Moreover, if all demands, capacities, and lower bounds in G are integers, then there is a circulation in G that is integer-valued.

Pf sketch. $f(e)$ is a circulation in G iff $f'(e) = f(e) - \ell(e)$ is a circulation in G'.



7. Network Flow II

- *bipartite matching*
- *disjoint paths*
- *extensions to max flow*
- *survey design*
- *airline scheduling*
- *image segmentation*
- *project selection*
- *baseball elimination*

Survey design

- Design survey asking n_1 consumers about n_2 products. ← one survey question per product
- Can only survey consumer i about product j if they own it.
- Ask consumer i between c_i and c_i' questions.
- Ask between p_j and p_j' consumers about product j.

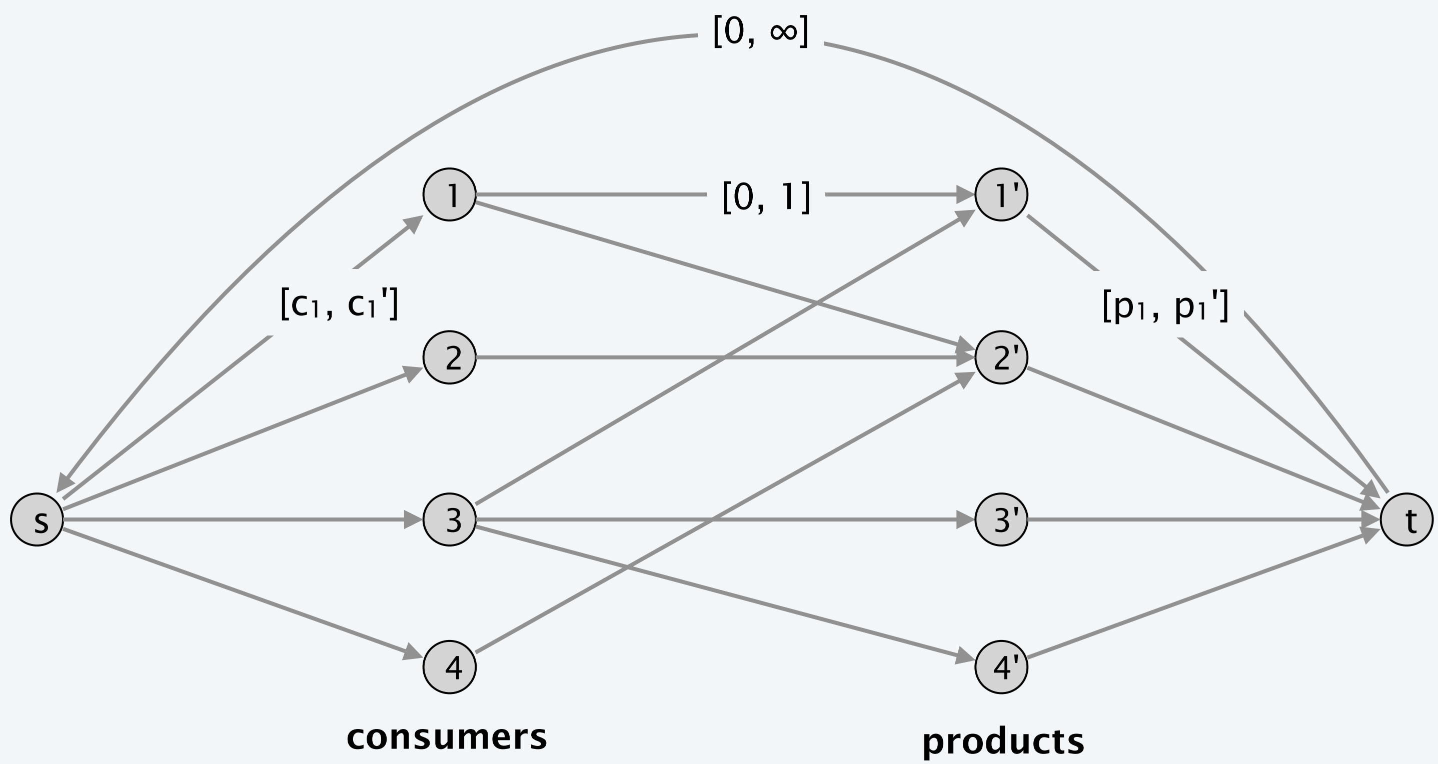
Goal. Design a survey that meets these specs, if possible.

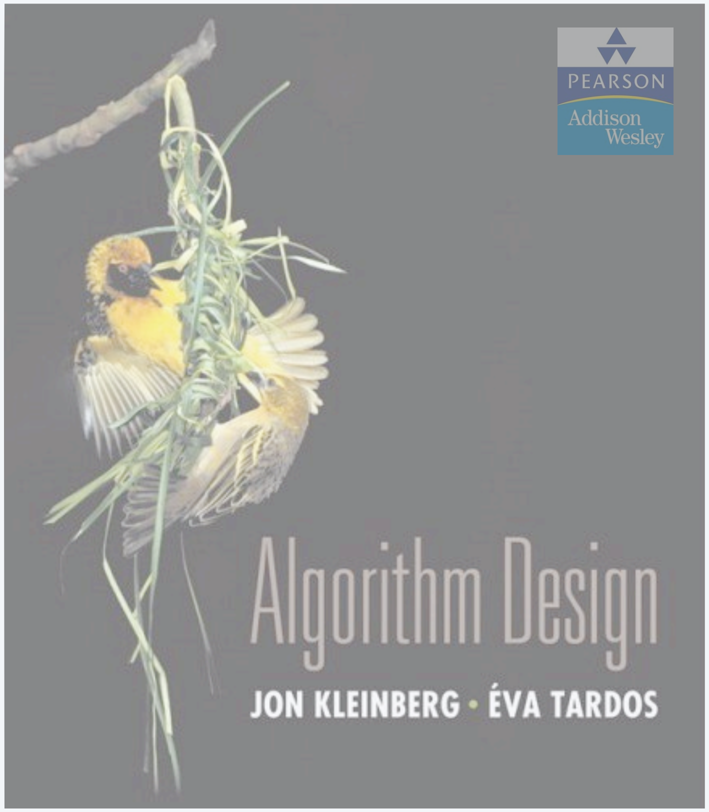
Bipartite perfect matching. Special case when $c_i = c_i' = p_j = p_j' = 1$.

Survey design

Max-flow formulation. Model as circulation problem with lower bounds.

- Add edge (i, j) if consumer j owns product i.
- Add edge from s to consumer j .
- Add edge from product i to t.
- Add edge from t to s.
- Integer circulation $\Leftrightarrow$ feasible survey design.





7. Network Flow II

- *bipartite matching*
- *disjoint paths*
- *extensions to max flow*
- *survey design*
- ***airline scheduling***
- *image segmentation*
- *project selection*
- *baseball elimination*

Airline scheduling

Airline scheduling.

- Complex computational problem faced by nation's airline carriers.
- Produces schedules that are efficient in terms of:
  - equipment usage, crew allocation, customer satisfaction
  - in presence of unpredictable issues like weather, breakdowns
- One of largest consumers of high-powered algorithmic techniques.

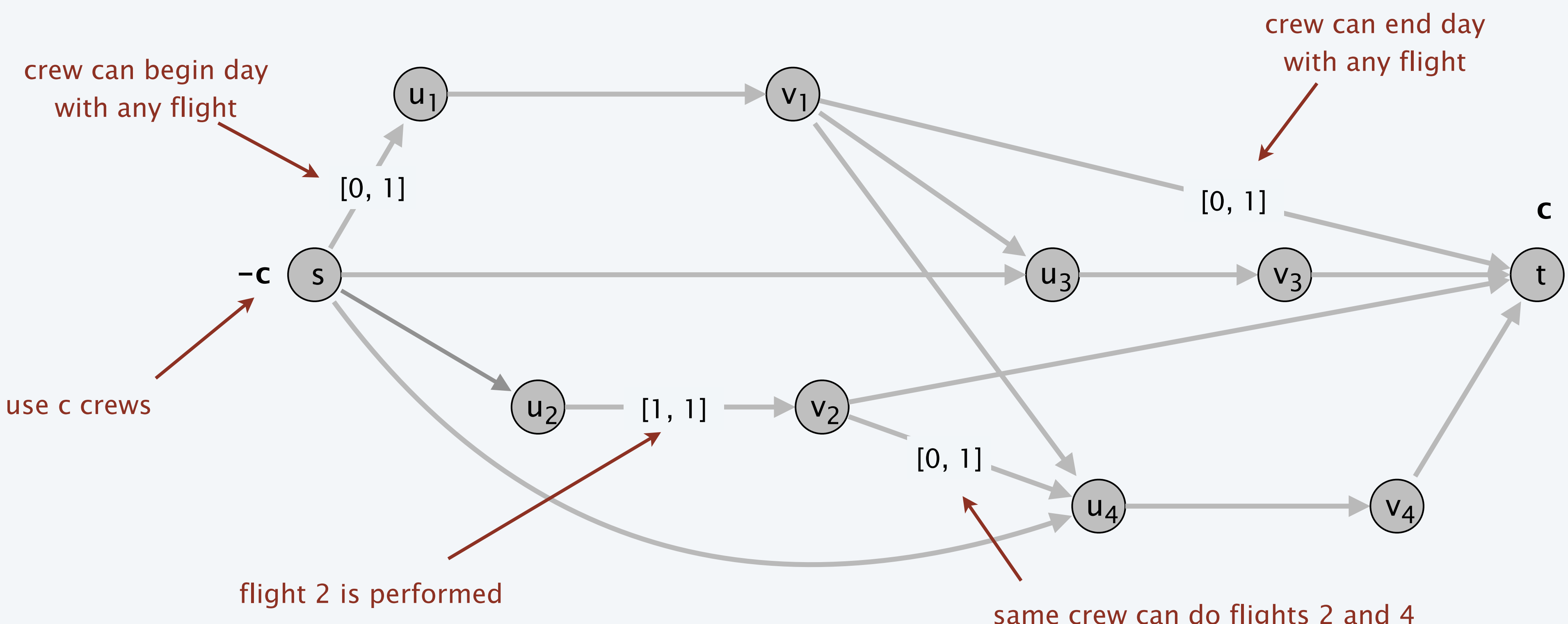
"Toy problem."

- Manage flight crews by reusing them over multiple flights.
- Input: set of k flights for a given day.
- Flight i leaves origin o_i at time s_i and arrives at destination d_i destination at time f_i.
- Minimize number of flight crews.

Airline scheduling

Circulation formulation. [to see if c crews suffice]

- For each flight i, include two nodes u_i and v_i.
- Add source s with demand $-c$, and edges (s, u_i) with capacity 1.
- Add sink t with demand c, and edges (v_i, t) with capacity 1.
- For each i, add edge (u_i, v_i) with lower bound and capacity 1.
- if flight j reachable from i, add edge (v_i, u_j) with capacity 1.



Airline scheduling: running time

Theorem. The airline scheduling problem can be solved in $O(k^3 \log k)$ time.

Pf.

- k = number of flights.
- c = number of crews (unknown).
- $O(k)$ nodes, $O(k^2)$ edges.
- At most k crews needed.

  $\Rightarrow$ solve $\lg k$ circulation problems. ← binary search for optimal value c*
- Value of the flow is between 0 and k.

  $\Rightarrow$ at most k augmentations per circulation problem.
- Overall time = $O(k^3 \log k)$.

Remark. Can solve in $O(k^3)$ time by formulating as minimum flow problem.

Airline scheduling: postmortem

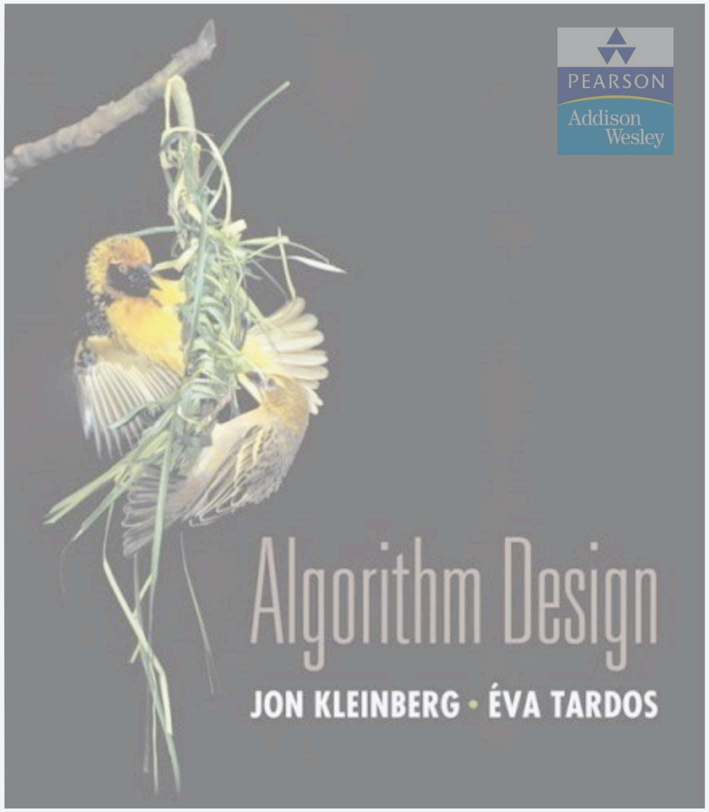
Remark. We solved a toy problem.

Real-world problem models countless other factors:

- Union regulations: e.g., flight crews can only fly certain number of hours in given interval.
- Need optimal schedule over planning horizon, not just one day.
- Deadheading has a cost.
- Flights don't always leave or arrive on schedule.
- Simultaneously optimize both flight schedule and fare structure.

Message.

- Our solution is a generally useful technique for efficient reuse of limited resources but trivializes real airline scheduling problem.
- Flow techniques useful for solving airline scheduling problems (and are widely used in practice).
- Running an airline efficiently is a very difficult problem.



7. Network Flow II

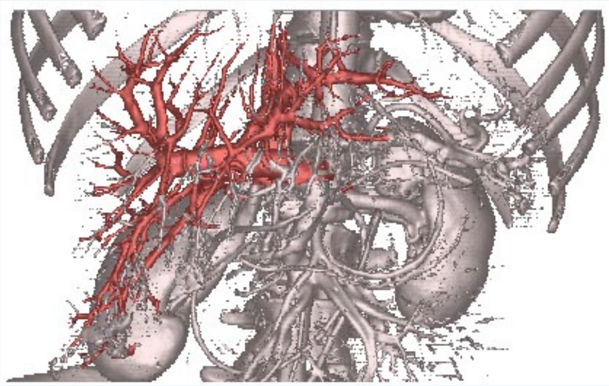
- bipartite matching
- disjoint paths
- extensions to max flow
- survey design
- airline scheduling
- ***image segmentation***
- project selection
- baseball elimination

Image segmentation

Image segmentation.

- Central problem in image processing.
- Divide image into coherent regions.

Ex. Three people standing in front of complex background scene.
Identify each person as a coherent object.

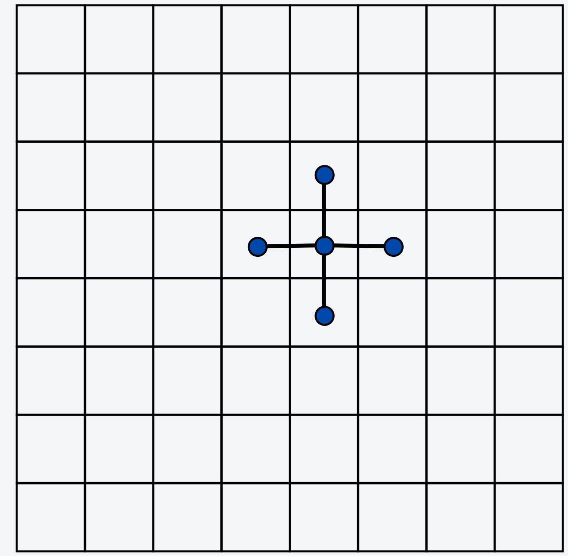


liver and hepatic vascularization segmentation

Image segmentation

Foreground / background segmentation.

- Label each pixel in picture as belonging to foreground or background.
- V = set of pixels, E = pairs of neighboring pixels.
- $a_i \geq 0$ is likelihood pixel i in foreground.
- $b_i \geq 0$ is likelihood pixel i in background.
- $p_{ij} \geq 0$ is separation penalty for labeling one of i and j as foreground, and the other as background.



Goals.

- Accuracy: if $a_i > b_i$ in isolation, prefer to label i in foreground.
- Smoothness: if many neighbors of i are labeled foreground,we should be inclined to label i as foreground.
- Find partition (A, B) that maximizes: $\displaystyle\sum_{i \in A} a_i + \sum_{j \in B} b_j - \sum_{\substack{(i,j) \in E \\ |A \cap \{i,j\}| = 1}} p_{ij}$

foreground background

Image segmentation

Formulate as min cut problem.

- Maximization.
- No source or sink.
- Undirected graph.

Turn into minimization problem.

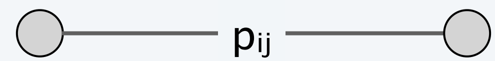
- Maximizing $$\sum_{i\in A} a_i + \sum_{j\in B} b_j - \sum_{\substack{(i,j)\in E \\ |A\cap\{i,j\}|=1}} p_{ij}$$
- is equivalent to minimizing $$\underbrace{\left(\sum_{i\in V} a_i + \sum_{j\in V} b_j\right)}_{\text{a constant}} - \sum_{i\in A} a_i - \sum_{j\in B} b_j + \sum_{\substack{(i,j)\in E \\ |A\cap\{i,j\}|=1}} p_{ij}$$
- or alternatively $$\sum_{j\in B} a_j + \sum_{i\in A} b_i + \sum_{\substack{(i,j)\in E \\ |A\cap\{i,j\}|=1}} p_{ij}$$

Image segmentation

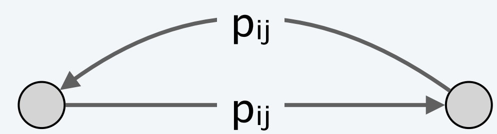
Formulate as min cut problem $G' = (V', E')$.

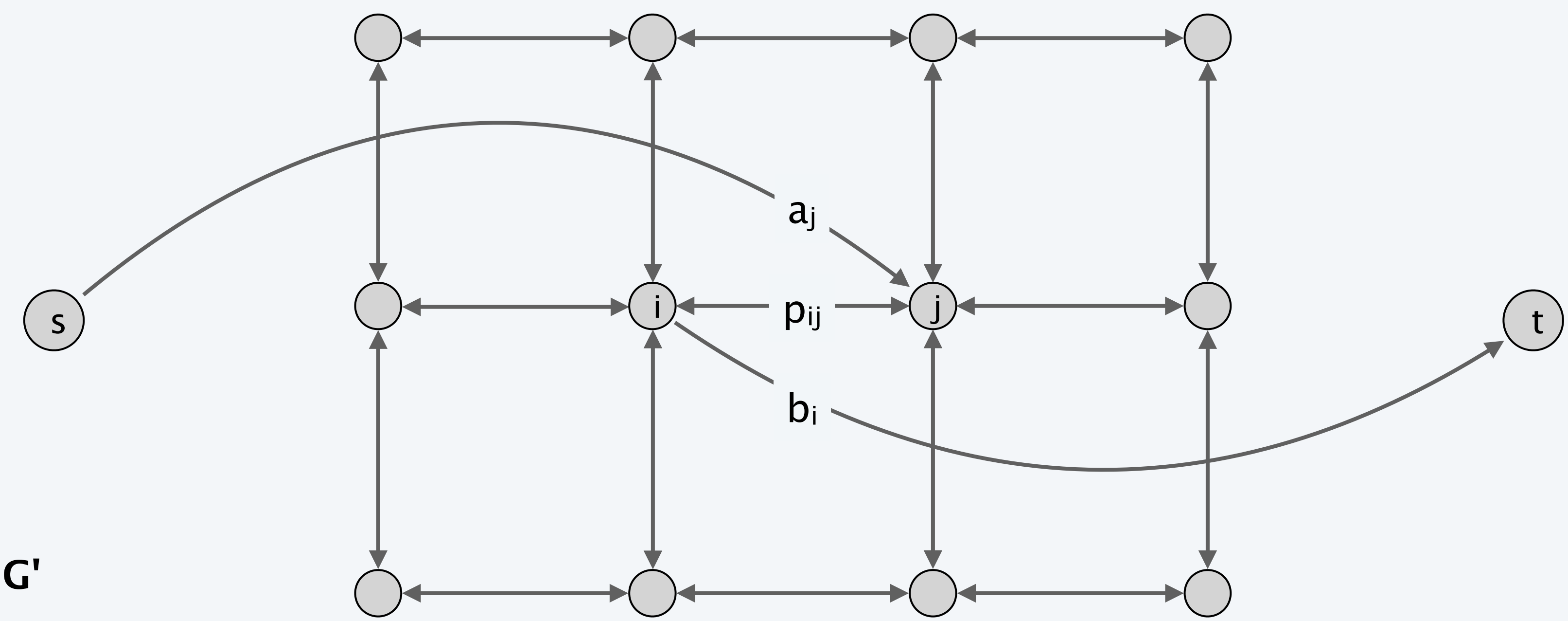
- Include node for each pixel.
- Use two antiparallel edges instead of undirected edge.
- Add source s to correspond to foreground.
- Add sink t to correspond to background.

edge in G



two antiparallel edges in G'





G'

Image segmentation

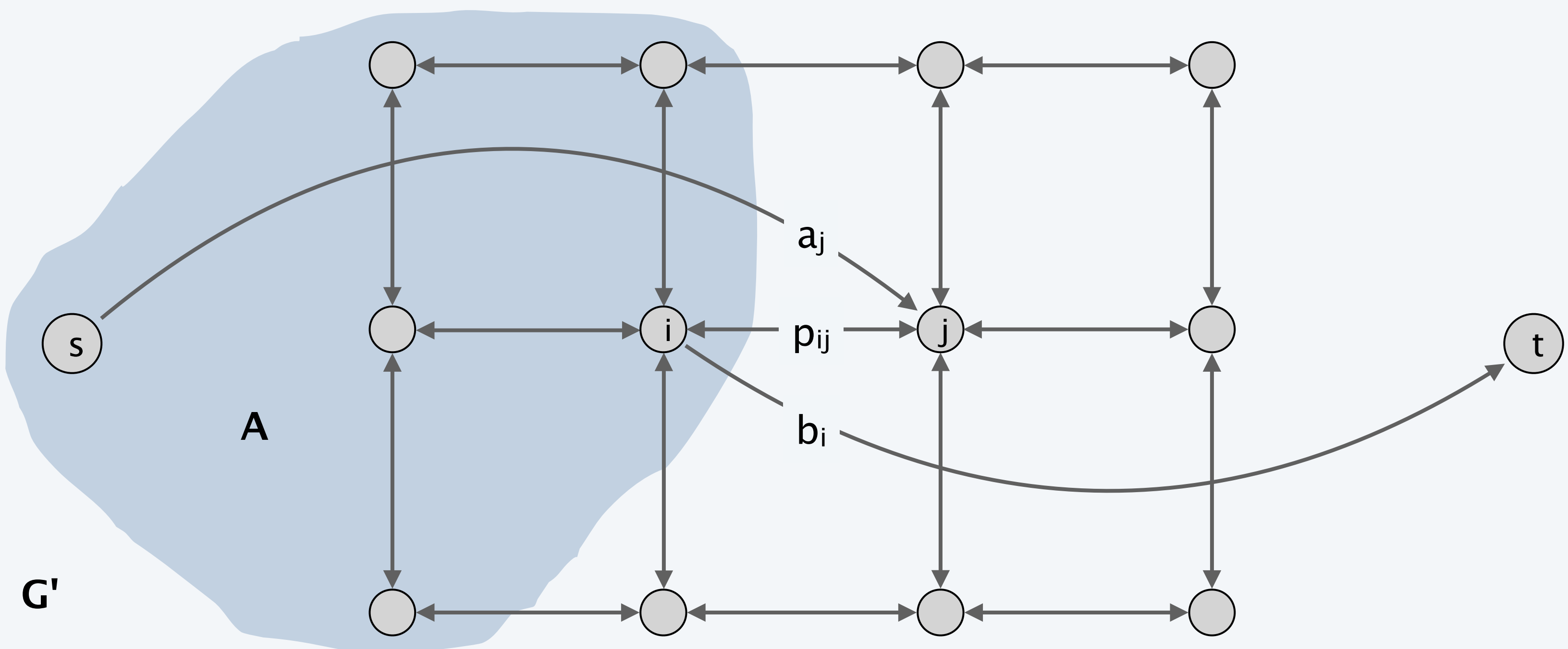
Consider min cut (A, B) in G'.

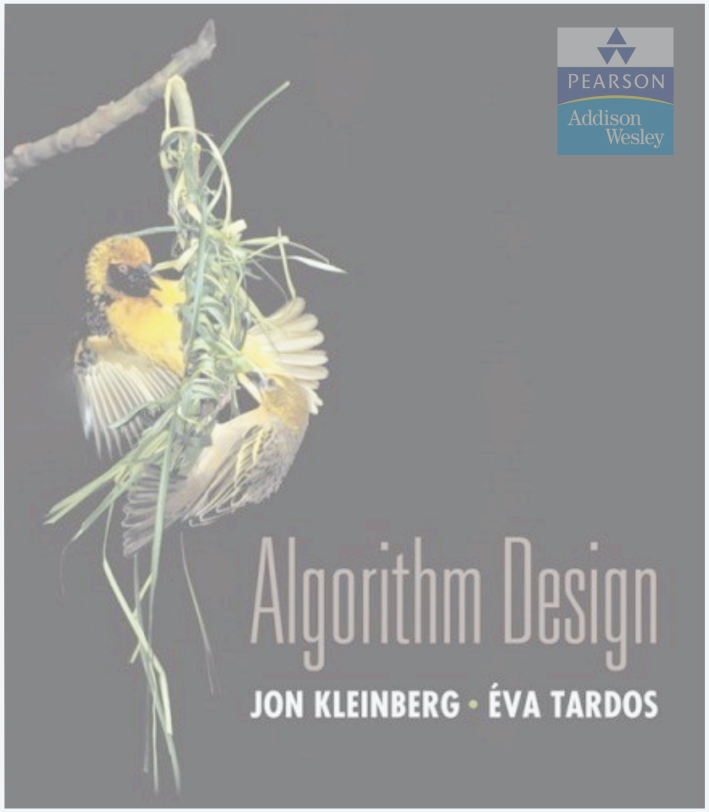
- A = foreground.

$$cap(A,B) \;=\; \sum_{j \in B} a_j + \sum_{i \in A} b_i \;+\; \sum_{\substack{(i,j) \in E \\ i \in A,\ j \in B}} p_{ij}$$

if i and j on different sides, p_{ij} counted exactly once

- Precisely the quantity we want to minimize.





7. Network Flow II

- *bipartite matching*
- *disjoint paths*
- *extensions to max flow*
- *survey design*
- *airline scheduling*
- *image segmentation*
- *project selection*
- *baseball elimination*

Project selection

Projects with prerequisites.

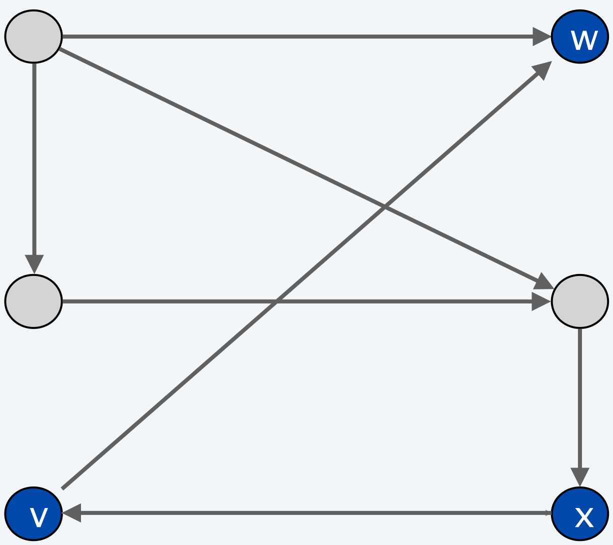
can be positive
or negative

- Set of possible projects P: project v has associated revenue p_v.
- Set of prerequisites E: if $(v, w) \in E$, can't do project v unless also do project w.
- A subset of projects $A \subseteq P$ is feasible if the prerequisite of every project in A also belongs to A.

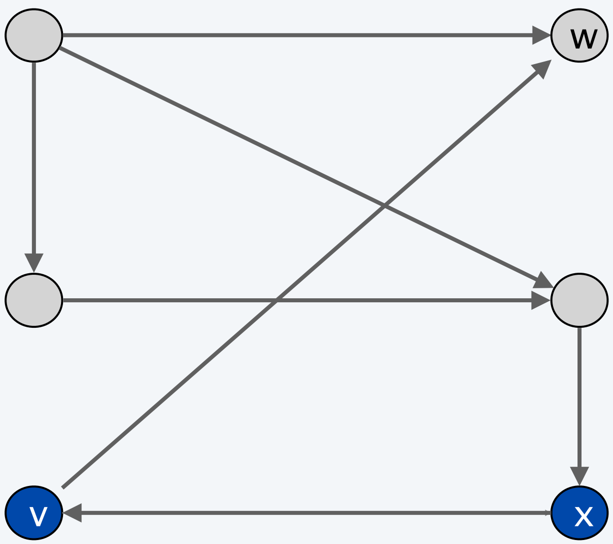
Project selection problem. Given a set of projects P and prerequisites E, choose a feasible subset of projects to maximize revenue.

Project selection: prerequisite graph

Prerequisite graph. Add edge (v, w) if can't do v without also doing w.



{ v, w, x } is feasible

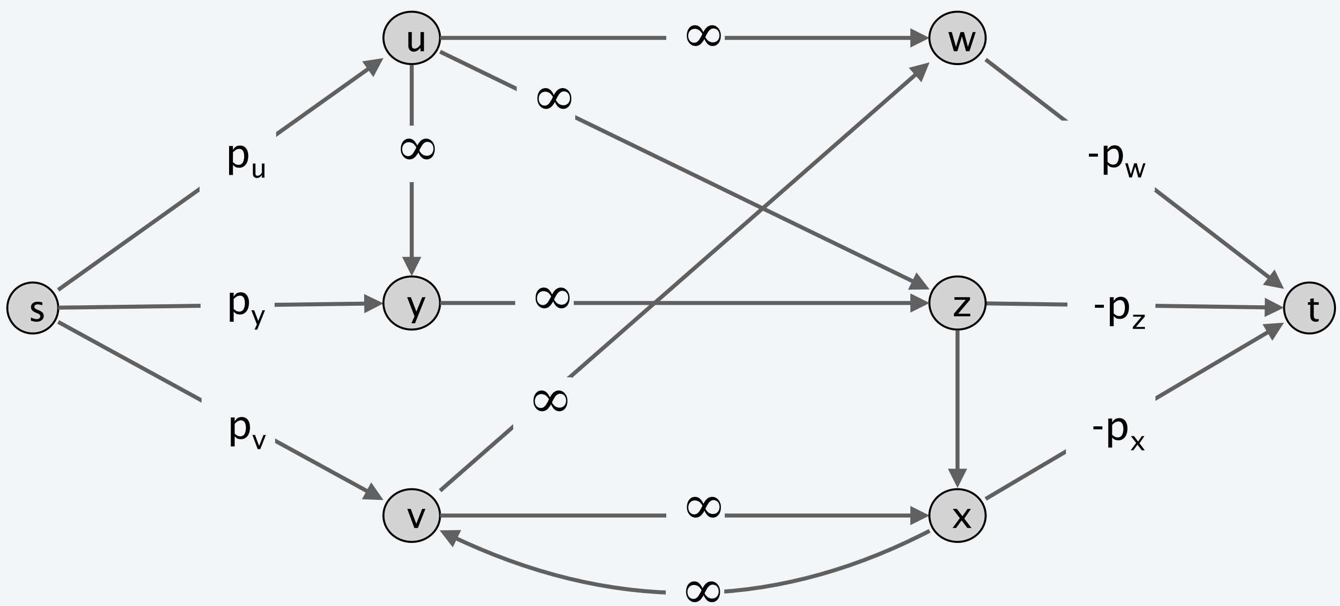


{ v, x } is infeasible

Project selection: min-cut formulation

Min-cut formulation.

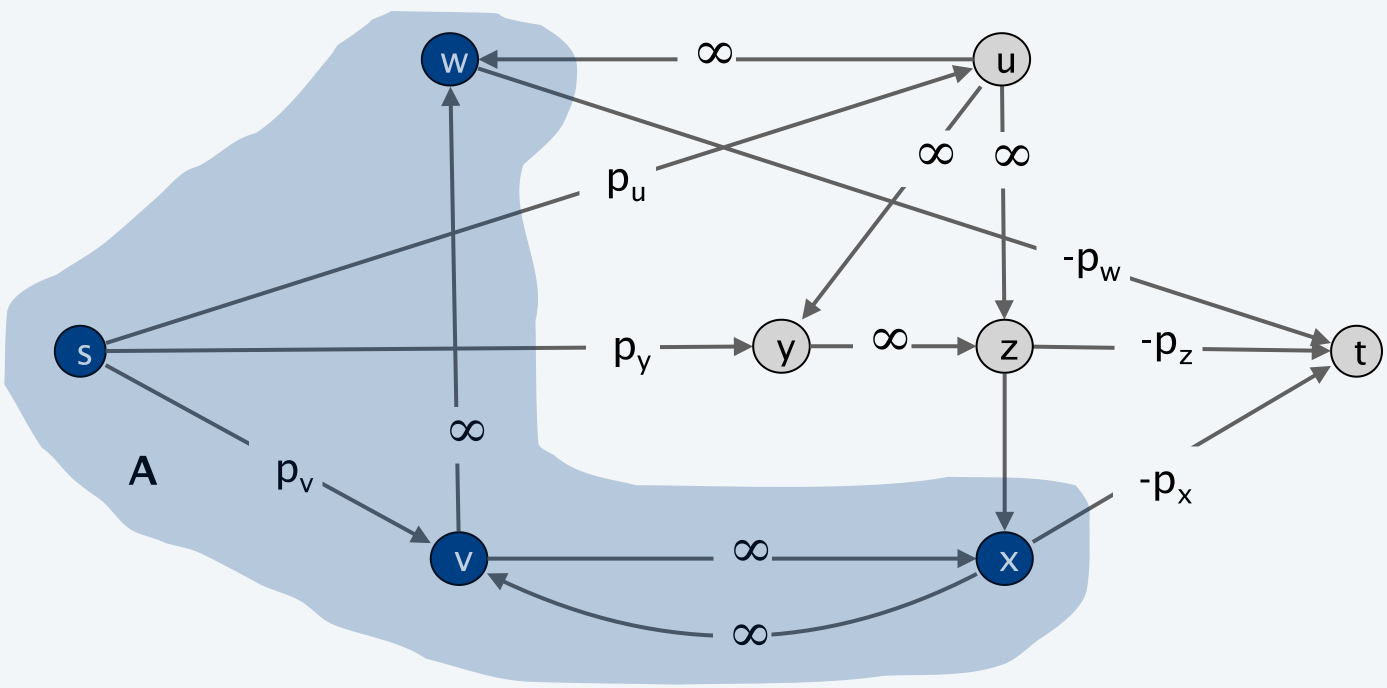
- Assign capacity ∞ to all prerequisite edge.
- Add edge (s, v) with capacity p_v if $p_v > 0$.
- Add edge (v, t) with capacity $-p_v$ if $p_v < 0$.
- For notational convenience, define $p_s = p_t = 0$.



Project selection: min-cut formulation

Claim. (A, B) is min cut iff $A - \{s\}$ is optimal set of projects.

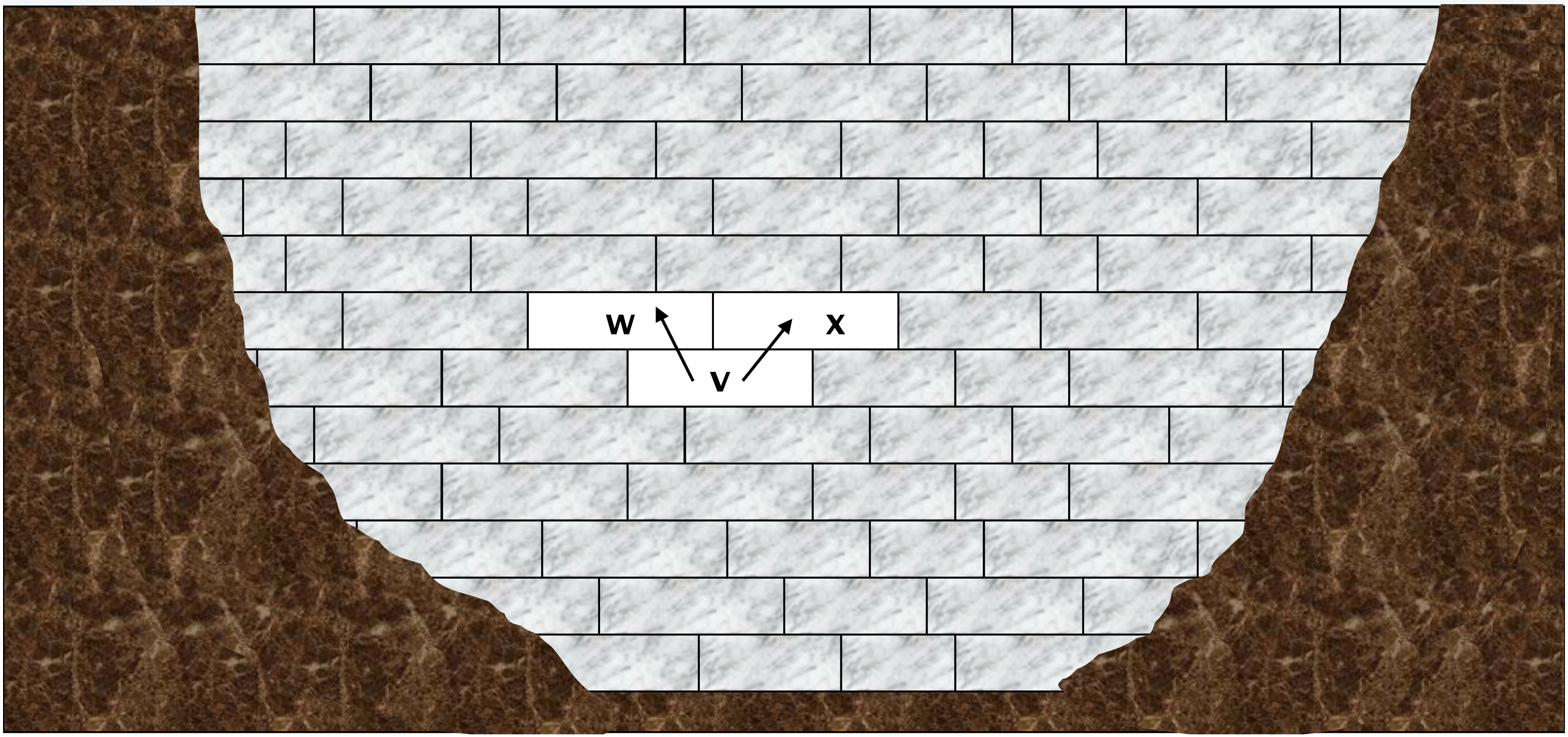
- Infinite capacity edges ensure $A - \{s\}$ is feasible.
- Max revenue because: $$cap(A, B) = \sum_{v \in B:\, p_v > 0} p_v + \sum_{v \in A:\, p_v < 0} (-p_v)$$ $$= \underbrace{\sum_{v:\, p_v > 0} p_v}_{\text{constant}} - \sum_{v \in A} p_v$$

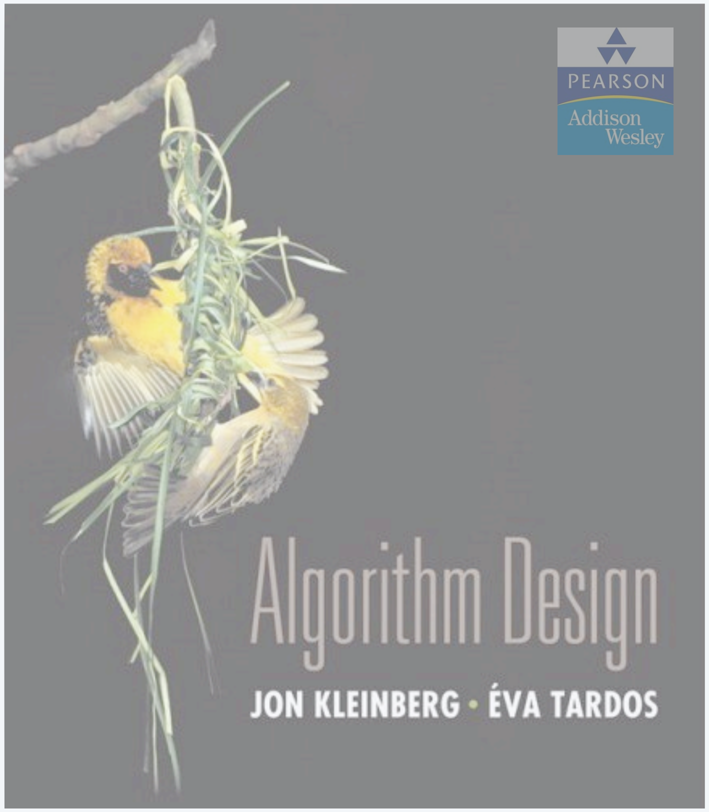


Open-pit mining

Open-pit mining. (studied since early 1960s)

- Blocks of earth are extracted from surface to retrieve ore.
- Each block v has net value p_v = value of ore – processing cost.
- Can't remove block v before w or x.





7. Network Flow II

- bipartite matching
- disjoint paths
- extensions to max flow
- survey design
- airline scheduling
- image segmentation
- project selection
- **baseball elimination**

Baseball elimination

TUESDAY, SEPTEMBER 10, 1996

San Francisco Chronicle

The Gate
Sports Online
▶ http://www.sfgate.com

SPORTING G

49ers, Young Get Big Brea

Quarterback m

By Gary Swan
Chronicle Staff Writer

The bye week has come at a perfect time for the 49ers and quarterback Steve Young. If they had a game next Sunday, there's a good chance Young would not play.

But the pulled groin muscle on his up-

Giants Officially Leave the NL West Race

By Nancy Gay
Chronicle Staff Writer

With the smack of another National League West bat 500 miles away, the Giants' run at the division title ended last night, just as they were handing the visiting St. Louis Cardinals an even bigger lead in the NL Central.

CARDINALS 6
GIANTS 2

In San Diego, Greg Vaughn's three-run homer in the eighth pushed the Padres over the Pirates and officially shoved the rest of the Giants' season into the background. On the heels of their tedious 6-2 loss before an announced crowd of 10,307 at Candlestick Park, the Giants fell 19½ games off the lead.

As it is, the worst the Padres (80-65) can finish is 80-82. The Giants have fallen to 59-83 with 20 games left; they cannot win 80 games. Coming off a miserable 2-8 mark on a three-city road trip that saw their road record drop to 27-47, the Giants were hoping to get off on the right foot in their longest homestand of the year (15 games, 14 days).

Financing in Place For Giants' New Stadium
SEE PAGE B1, MAIN NEWS

"Where we are, you're going to be eliminated sooner or later," Baker said quietly. "But it doesn't alter the fact that we've still got to play ball. You've still got to play hard, the fans come out to watch you play. You've got to play for the fact of loving to play, no matter where you are in the standings.

"You've got to play the role of spoiler, to not make it easier on

GIANTS: Page D5 Col. 3

Baseball elimination problem

Q. Which teams have a chance of finishing the season with the most wins?

i	team		wins	losses	to play	ATL	PHI	NYM	MON
0		Atlanta	83	71	8	–	1	6	1
1		Philly	80	79	3	1	–	0	2
2		New York	78	78	6	6	0	–	0
3		Montreal	77	82	3	1	2	0	–

Montreal is mathematically eliminated.

- Montreal finishes with ≤ 80 wins.
- Atlanta already has 83 wins.

Remark. This is the only reason sports writers appear to be aware of — conditions are sufficient but not necessary!

Baseball elimination problem

Q. Which teams have a chance of finishing the season with the most wins?

i	team		wins	losses	to play	ATL	PHI	NYM	MON
0		Atlanta	83	71	8	–	1	6	1
1		Philly	80	79	3	1	–	0	2
2		New York	78	78	6	6	0	–	0
3		Montreal	77	82	3	1	2	0	–

Philadelphia is mathematically eliminated.

- Philadelphia finishes with ≤ 83 wins.
- Either New York or Atlanta will finish with ≥ 84 wins.

Observation. Answer depends not only on how many games already won and left to play, but on whom they're against.

Baseball elimination problem

Current standings.

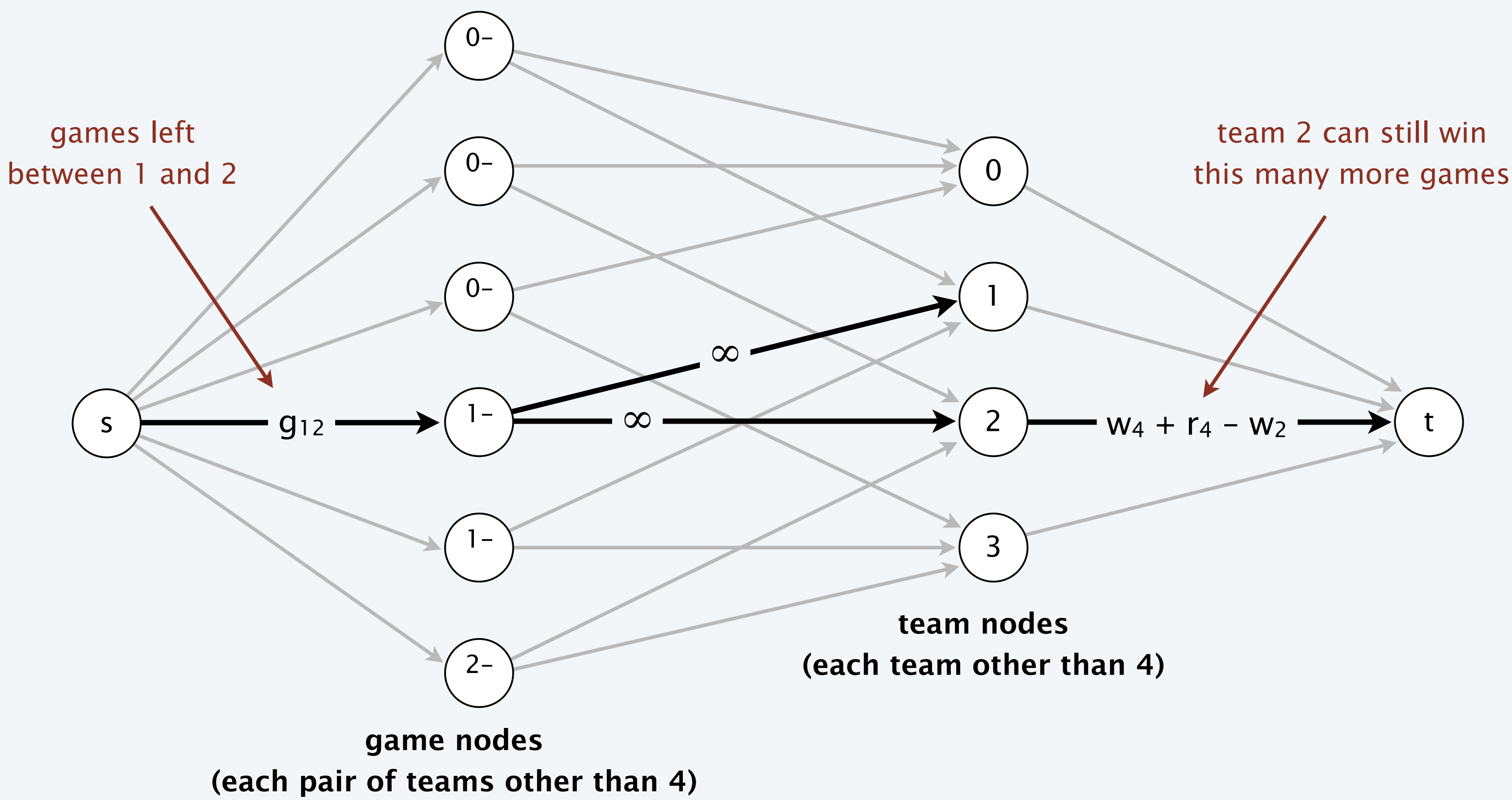
- Set of teams S.
- Distinguished team $z \in S$.
- Team x has won w_x games already.
- Teams x and y play each other r_{xy} additional times.

Baseball elimination problem. Given the current standings, is there any outcome of the remaining games in which team z finishes with the most (or tied for the most) wins?

Baseball elimination problem: max-flow formulation

Can team 4 finish with most wins?

- Assume team 4 wins all remaining games $\Rightarrow w_4 + r_4$ wins.
- Divvy remaining games so that all teams have $\leq w_4 + r_4$ wins.

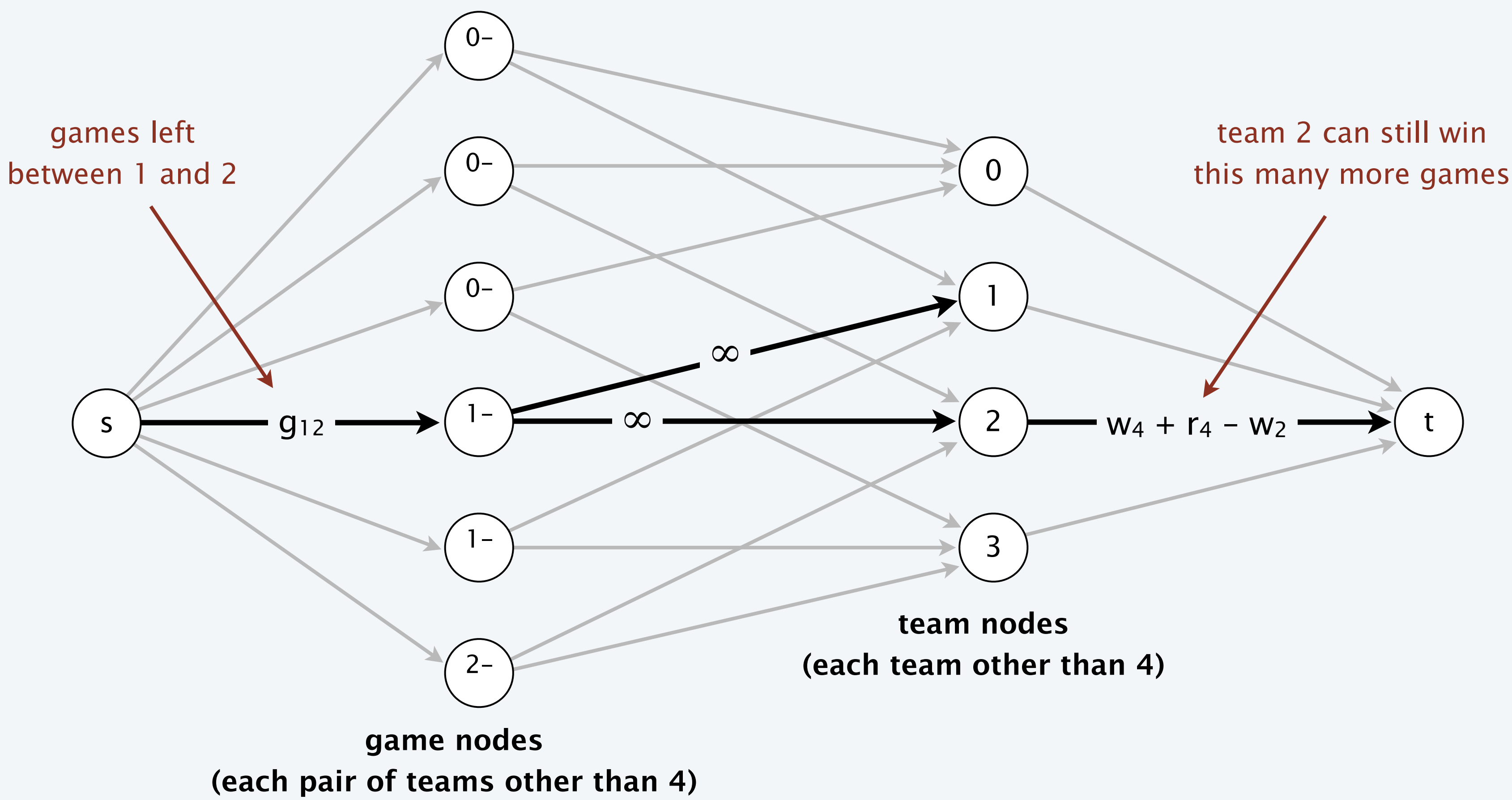


Baseball elimination problem: max-flow formulation

Theorem. Team 4 not eliminated iff max flow saturates all edges leaving s.

Pf.

- Integrality theorem $\Rightarrow$ each remaining game between x and y added to number of wins for team x or team y.
- Capacity on (x, t) edges ensure no team wins too many games. ▪



Baseball elimination: explanation for sports writers

Q. Which teams have a chance of finishing the season with the most wins?

i	team	wins	losses	to play	NYY	BAL	BOS	TOR	DET
0	New York	75	59	28	–	3	8	7	3
1	Baltimore	71	63	28	3	–	2	7	4
2	Boston	69	66	27	8	2	–	0	0
3	Toronto	63	72	27	7	7	0	–	0
4	Detroit	49	86	27	3	4	0	0	–

AL East (August 30, 1996)

Detroit is mathematically eliminated.

- Detroit finishes with ≤ 76 wins.
- Wins for $R = \{ \text{NYY, BAL, BOS, TOR} \} = 278$.
- Remaining games among $\{ \text{NYY, BAL, BOS, TOR} \} = 3 + 8 + 7 + 2 + 7 = 27$.
- Average team in R wins $305/4 = 76.25$ games.

Baseball elimination: explanation for sports writers

Certificate of elimination.

$$T \subseteq S, \quad w(T) := \overbrace{\sum_{i \in T} w_i}^{\#\text{ wins}}, \quad g(T) := \overbrace{\sum_{\{x,y\} \subseteq T} g_{xy}}^{\#\text{ remaining games}},$$

Theorem. [Hoffman-Rivlin 1967] Team z is eliminated iff there exists a subset T^* such that

$$w_z + g_z < \frac{w(T^*) + g(T^*)}{|T^*|}$$

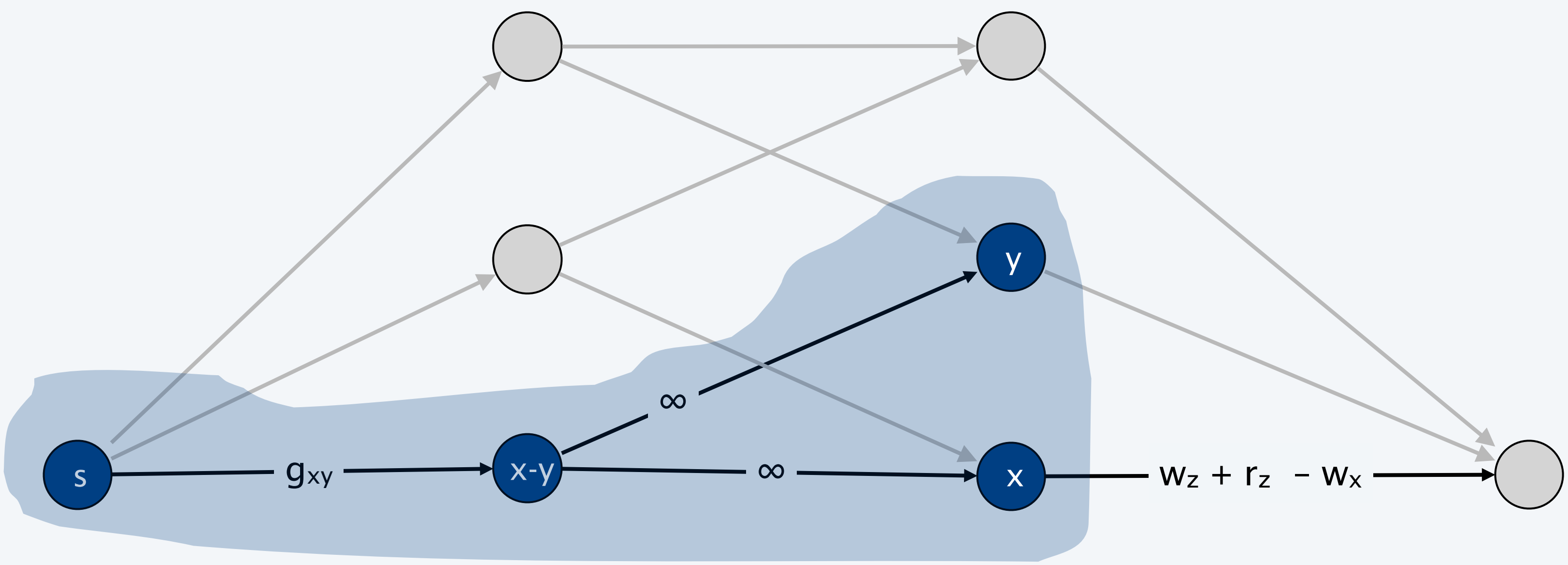
Pf. $\Leftarrow$

- Suppose there exists $T^* \subseteq S$ such that $w_z + g_z < \dfrac{w(T^*) + g(T^*)}{|T^*|}$.
- Then, the teams in T^* win at least $(w(T^*) + g(T^*)) / |T^*|$ games on average.
- This exceeds the maximum number that team z can win. ▪

Baseball elimination: explanation for sports writers

Pf. $\Rightarrow$

- Use max-flow formulation, and consider min cut (A, B).
- Let T^* = team nodes on source side A of min cut.
- Observe that game node $x\text{-}y \in A$ iff both $x \in T^*$ and $y \in T^*$.
  - infinite capacity edges ensure if $x\text{-}y \in A$, then both $x \in A$ and $y \in A$
  - if $x \in A$ and $y \in A$ but $x\text{-}y \notin A$, then adding $x\text{-}y$ to A decreases the capacity of the cut by g_{xy}



Baseball elimination: explanation for sports writers

Pf. $\Rightarrow$

- Use max-flow formulation, and consider min cut (A, B).
- Let T^* = team nodes on source side A of min cut.
- Observe that game node x-$y \in A$ iff both $x \in T^*$ and $y \in T^*$.
- Since team z is eliminated, by max-flow min-cut theorem,

$$\begin{aligned} g(S - \{z\}) &> cap(A, B) \\ &= \overbrace{g(S - \{z\}) - g(T^*)}^{\text{capacity of game edges leaving s}} + \overbrace{\sum_{x \in T^*} (w_z + g_z - w_x)}^{\text{capacity of team edges entering t}} \\ &= g(S - \{z\}) - g(T^*) - w(T^*) + |T^*|(w_z + g_z) \end{aligned}$$

- Rearranging terms: $w_z + g_z < \dfrac{w(T^*) + g(T^*)}{|T^*|}$ ▪