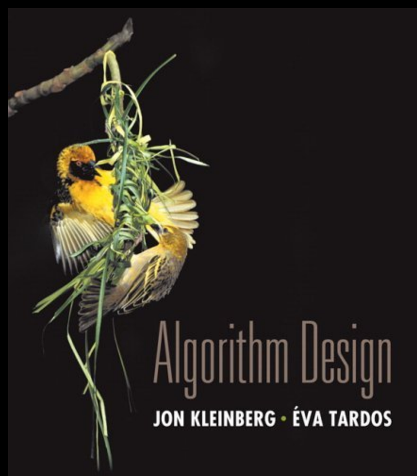


Chapter 7

Network Flow

Slides by Kevin Wayne.
Copyright © 2005 Pearson-Addison Wesley.
All rights reserved.

Network Flow Algorithms: Running Time

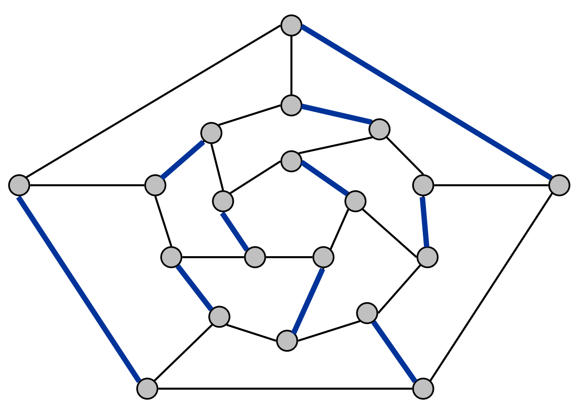
- Generic augmenting path: O(m val(f*))
- Capacity scaling: $O(m^2 \log C)$.
  - C denotes the sum of capacities of all edges out of s.
- Shortest augmenting path: $O(m^2n)$.
  - Proved by Dinitz (also by Edmonds and Karp)
- Preflow-Push algorithm: $O(n^3)$
  - Textbook 7.4
- Conclusion: Network Flow problem can be solved within polynomial time.

7.5 Bipartite Matching

Matching

Matching.

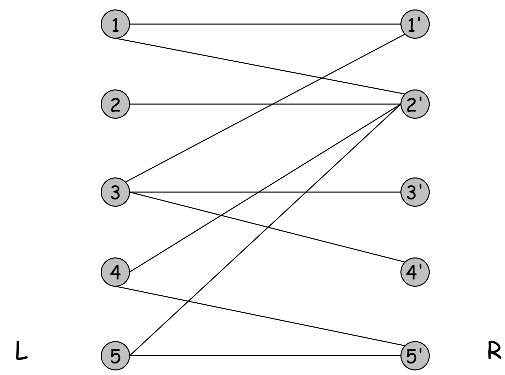
- Input: undirected graph $G = (V, E)$.
- $M \subseteq E$ is a matching if each node appears in at most edge in M.
- Max matching: find a max cardinality matching.



4

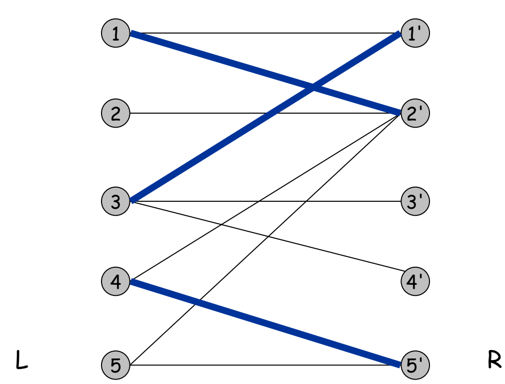
Bipartite Matching

Bipartite matching.
- Input: undirected, bipartite graph $G = (L \cup R, E)$.
- $M \subseteq E$ is a matching if each node appears in at most edge in M.
- Max matching: find a max cardinality matching.



Bipartite Matching

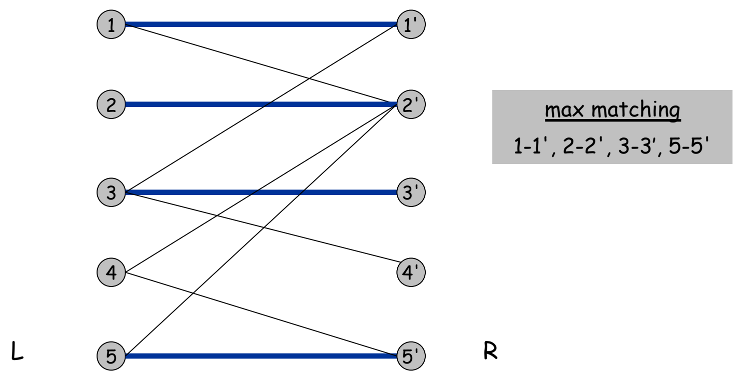
Bipartite matching.
- Input: undirected, bipartite graph $G = (L \cup R, E)$.
- $M \subseteq E$ is a matching if each node appears in at most edge in M.
- Max matching: find a max cardinality matching.



matching
1-2', 3-1', 4-5'

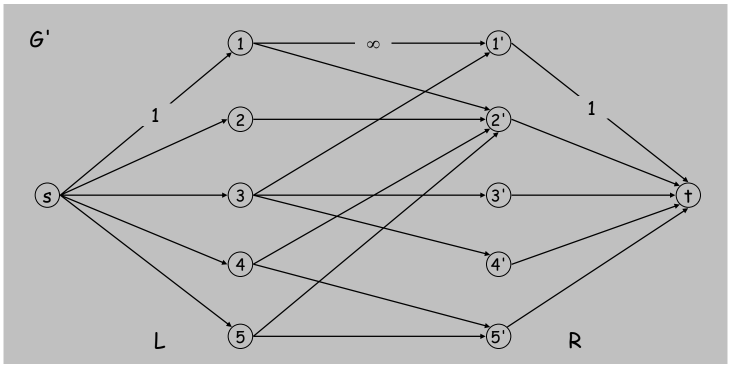
Bipartite Matching

Bipartite matching.
- Input: undirected, bipartite graph $G = (L \cup R, E)$.
- $M \subseteq E$ is a matching if each node appears in at most one edge in M.
- Max matching: find a max cardinality matching.



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.

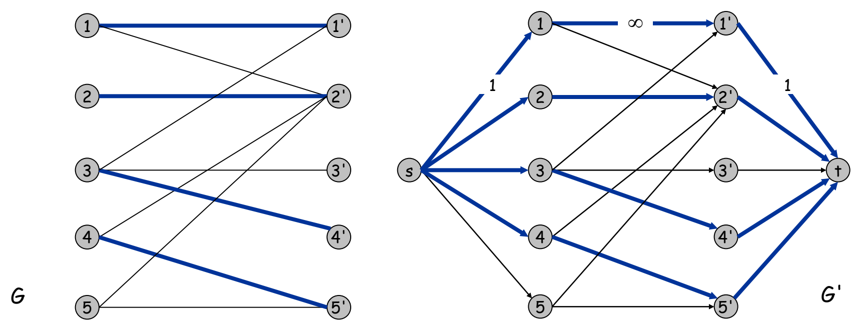


Bipartite Matching: Proof of Correctness

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

Pf. $\leq$

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

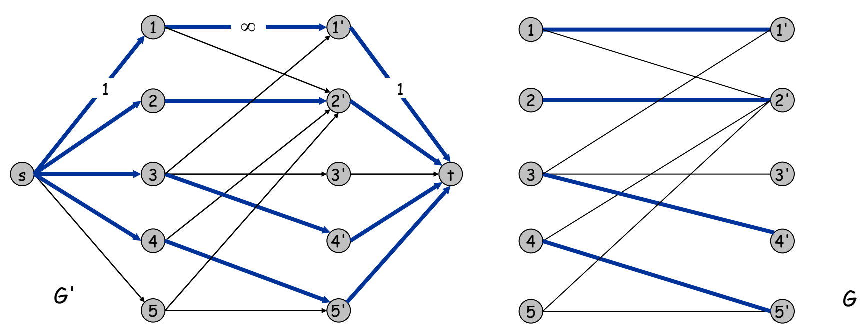


Bipartite Matching: Proof of Correctness

Theorem. Max cardinality 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

Def. A matching $M \subseteq E$ is perfect if each node appears in exactly one edge in M.

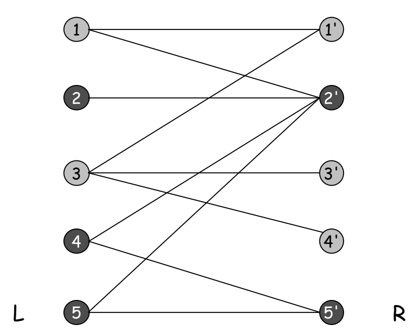
Q. When does a bipartite graph have a perfect matching?

Structure of bipartite graphs with perfect matchings.
- Clearly we must have $|L| = |R|$.
- What other conditions are necessary?
- What conditions are sufficient?

Perfect Matching

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).

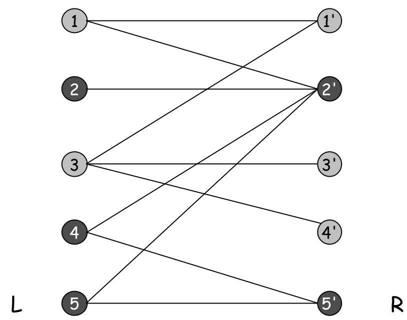


<u>No perfect matching:</u>
S = { 2, 4, 5 }
N(S) = { 2', 5' }.

Marriage Theorem

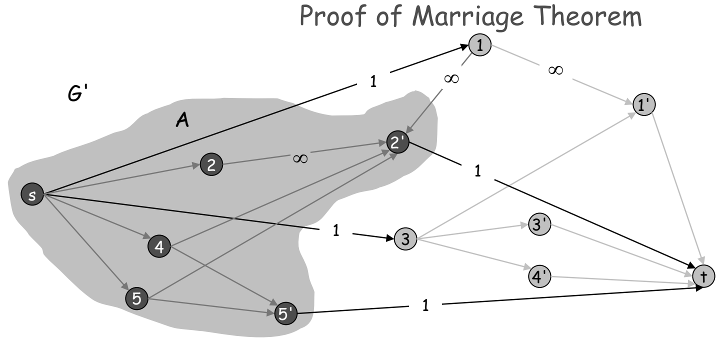
Marriage Theorem. [Frobenius 1917, Hall 1935] Let $G = (L \cup R, E)$ be a bipartite graph with $|L| = |R|$. Then, G has a perfect matching iff $|N(S)| \geq |S|$ for all subsets $S \subseteq L$.

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



No perfect matching:
$S = \{ 2, 4, 5 \}$
$N(S) = \{ 2', 5' \}$.

Proof of Marriage Theorem



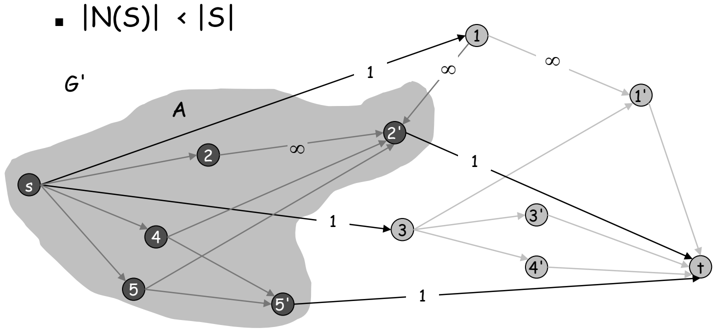
$L_A = \{2, 4, 5\}$
$L_B = \{1, 3\}$
$R_A = \{2', 5'\}$
$N(L_A) = \{2', 5'\}$

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

Proof of Marriage 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, $cap(A, B) < |L|$.
- 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: $N(L_A) \subseteq R_A$.
- $|N(L_A)| \le |R_A| = cap(A, B) - |L_B| < |L| - |L_B| = |L_A|$.
- Choose $S = L_A$, $|N(L_A)| < |L_A|$,
- $|N(S)| < |S|$



$L_A = \{2, 4, 5\}$
$L_B = \{1, 3\}$
$R_A = \{2', 5'\}$
$N(L_A) = \{2', 5'\}$

15

Bipartite Matching: Running Time

Which max flow algorithm to use for bipartite matching?

- Generic augmenting path: $O(m \text{ val}(f^*))$
- Capacity scaling: $O(m^2 \log C)$.
  - C denotes the sum of capacities of all edges out of s.
- Shortest augmenting path: $O(m^2 n)$.
  - Proved by Dinitz (also by Edmonds and Karp)
- Preflow-Push algorithm: $O(n^3)$

16

7.6 Disjoint Paths

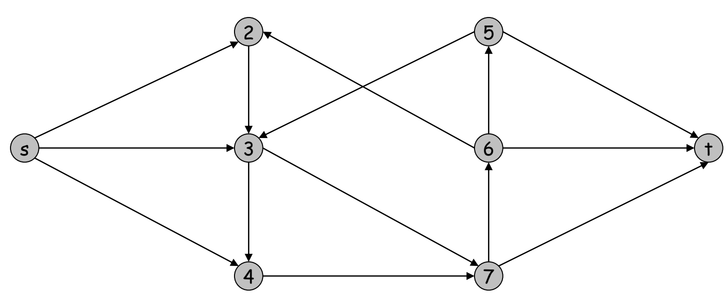
Edge Disjoint Paths

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

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

Ex: communication networks.

Class Exercise: Find the max number of edge-disjoint s-t paths!



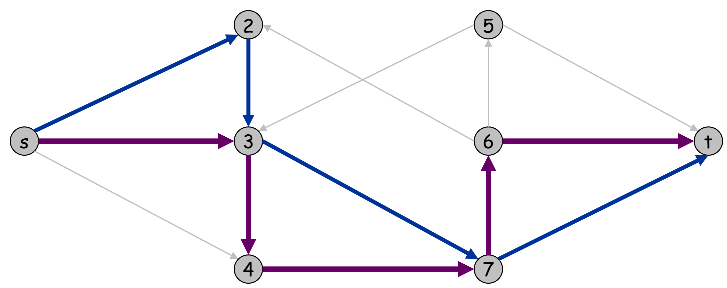
18

Edge Disjoint Paths

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

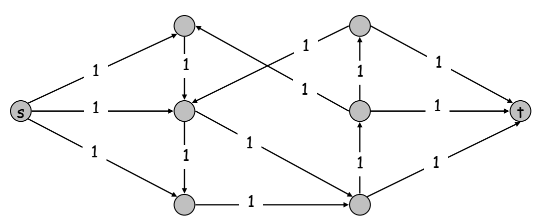
Def. Two paths are edge-disjoint if they have no edge in common.

Ex: communication networks.



Edge Disjoint Paths

Max flow formulation: assign unit capacity to every edge.



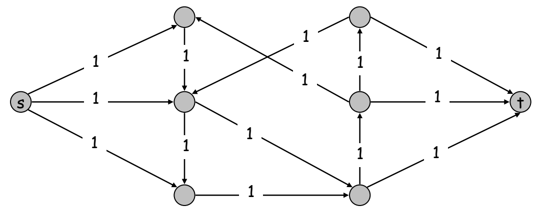
Theorem. Max number edge-disjoint s-t paths equals max flow value.

Pf. $\leq$

- Suppose there are k edge-disjoint paths $P_1, \ldots, P_k$.
- Set $f(e) = 1$ if e participates in some path P_i; 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-t paths equals max flow value.

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. ▪

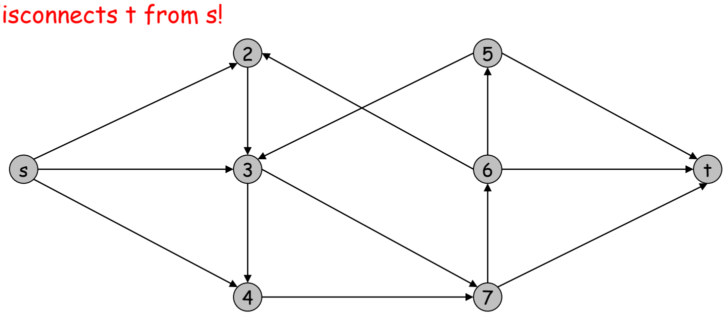
can eliminate cycles to get simple paths if desired

Network Connectivity

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.

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

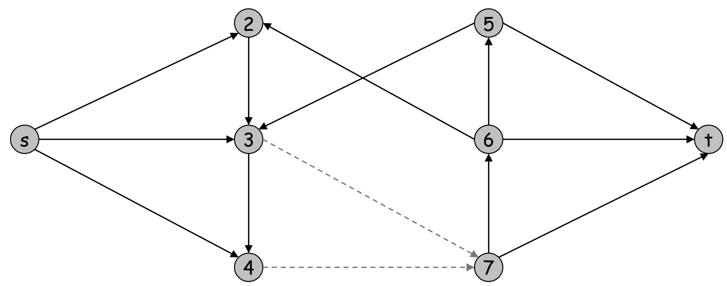
Class Exercise: Find the min number of edges whose removal disconnects t from s!



Network Connectivity

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.

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

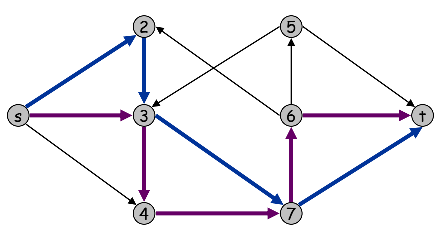


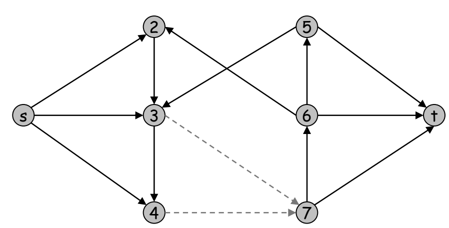
Edge Disjoint Paths and Network Connectivity

Theorem. [Menger 1927] The max number of edge-disjoint s-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-t path uses at least one edge in F.
  Hence, the number of edge-disjoint paths is at most k. ▪



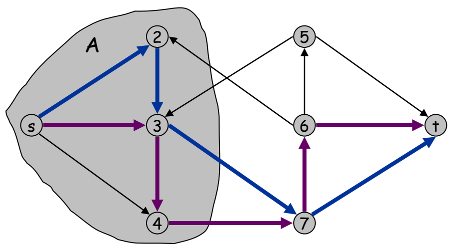


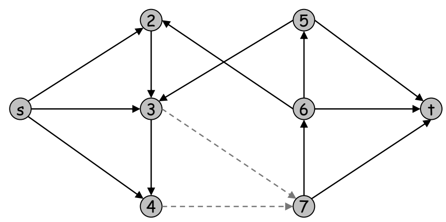
Disjoint Paths and Network Connectivity

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

Pf. $\geq$

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





25

7.7 Extensions to Max Flow

Circulation with Demands

Circulation with demands.
- Directed graph G = (V, E).
- Edge capacities c(e), e ∈ E.
- Node supply and demands d(v), v ∈ V.

demand if d(v) > 0; supply if d(v) < 0; transshipment if d(v) = 0

Def. A circulation is a function that satisfies:
- For each e ∈ E: $0 \leq f(e) \leq c(e)$ (capacity)
- For each v ∈ V: $\sum_{e \text{ in to } v} f(e) - \sum_{e \text{ out of } v} f(e) = d(v)$ (conservation)

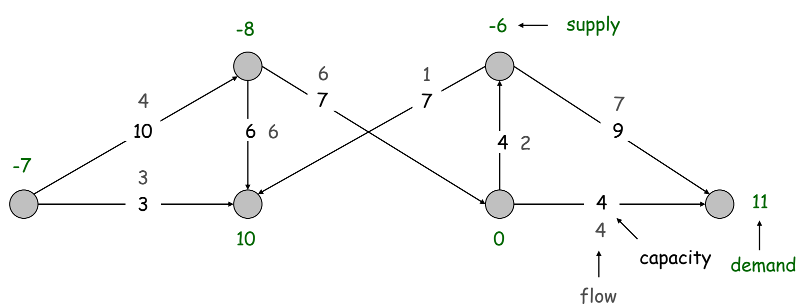
Circulation problem: given (V, E, c, d), does there exist a circulation?

Circulation with Demands

Necessary condition: sum of supplies = sum of demands.

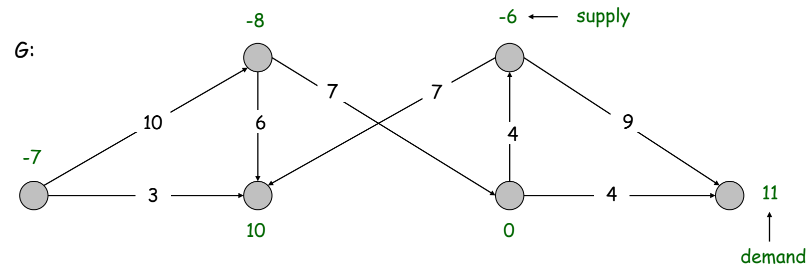
$$\sum_{v\,:\,d(v)>0} d(v) = \sum_{v\,:\,d(v)<0} -d(v) =: D$$

Pf. Sum conservation constraints for every demand node v.



Circulation with Demands

Max flow formulation.



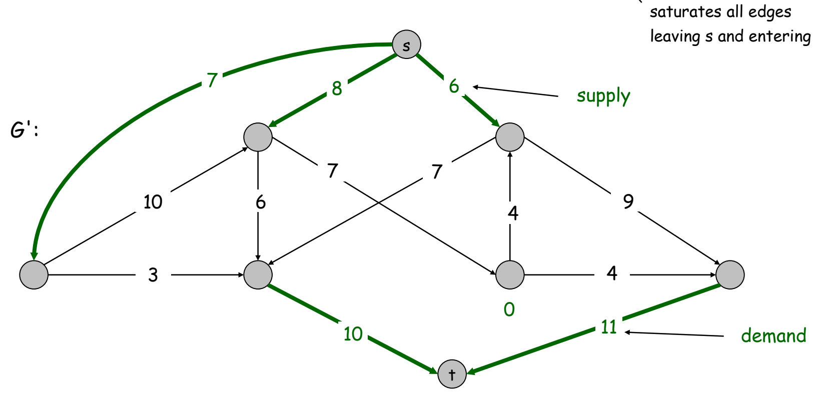
Circulation with Demands

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.

$$\sum_{v : d(v) > 0} d(v) = \sum_{v : d(v) < 0} -d(v) =: D$$

saturates all edges leaving s and entering t



Circulation with Demands

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

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

Pf idea. Look at min cut in G'.

32

Circulation with Demands and Lower Bounds

Feasible circulation.
- Directed graph G = (V, E).
- Edge capacities c(e) and lower bounds ℓ (e), e $\in$ E.
- Node supply and demands d(v), v $\in$ V.

Def. A circulation is a function that satisfies:
- For each e $\in$ E: $\ell(e) \leq f(e) \leq c(e)$ (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 exist an circulation?

33

Circulation with Demands and Lower Bounds

Idea. Model lower bounds 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'. 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) - ℓ(e) is a circulation in G'.

Max-flow and Circulation Comparison

Max-flow
- G = (V, E) = directed graph,
- Two distinguished nodes:
  - s = source, t = sink.
- c(e) = capacity of edge e.
- $0 \le f(e) \le c(e)$
  $\sum_{e \text{ in to } v} f(e) = \sum_{e \text{ out of } v} f(e)$
- max flow = min cut
- Algorithms:
- Generic augmenting path:
  - O(m val(f*)).
- Capacity scaling:
  - $O(m^2 \log C)$
- *Shortest augmenting path:
  - $O(m^2 n)$.
- * Preflow-Push:
  - $O(m\, n^2)$ or $O(n^3)$.

Circulation with demands
- Node supply and demands d(v), v ∈ V.
- demand if d(v) > 0;
- supply if d(v) < 0;
- transshipment if d(v) = 0

▪Conservation $\sum_{e \text{ in to } v} f(e) - \sum_{e \text{ out of } v} f(e) = d(v)$

▪Necessary condition to have a circulation

$$\sum_{v : d(v) > 0} d(v) = \sum_{v : d(v) < 0} -d(v) =: D$$

▪Convert to network flow:
- 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 (saturates all edges leaving s and entering t)
- with Demands and Lower Bound:
  $\ell(e) \le f(e) \le c(e)$

▪Transfer each edge e: (v, w):

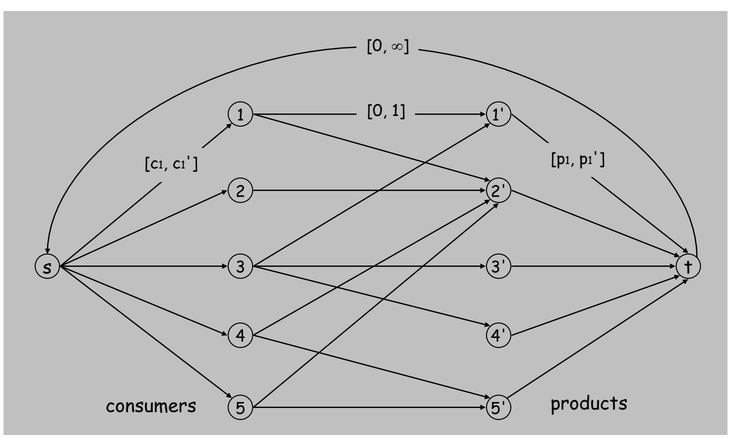
▪d(v)=d(v)+l(e); d(w)=d(w)-l(e); c(e)=c(e)-l(e)

7.8 Survey Design

Survey Design

Algorithm. Formulate as a circulation problem with lower bounds.

- Include an edge (i, j) if consumer j owns product i.
- Integer circulation $\Leftrightarrow$ feasible survey design.



Survey Design Example

Survey design.

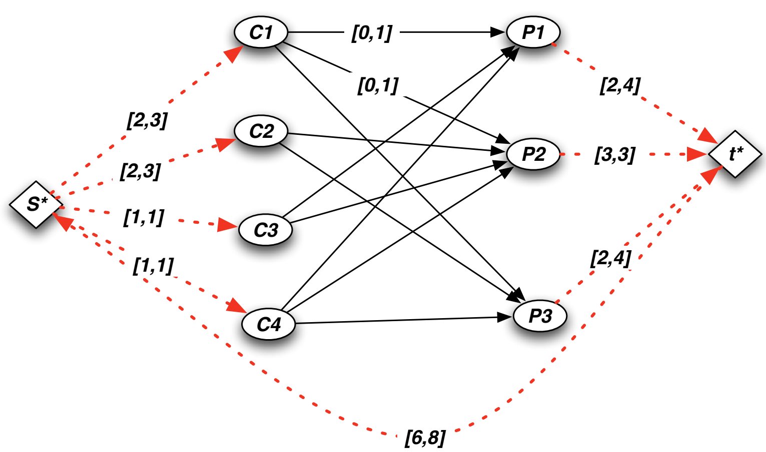
- Design survey asking 4 consumers about 3 products.
- Can only survey consumer i about product j if they own it, see table below.
- Ask consumer 1, consumer 2 each between 2 and 3 questions.
- Ask consumer 3, consumer 4 each 1 question only.
- Ask between 2 and 4 consumers about product 1.
- Ask 3 consumers about product 2.
- Ask between 2 and 4 consumers about product 3.

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

	P1	P2	P3
C1	1	1	1
C2		1	1
C3	1	1	
C4	1	1	1

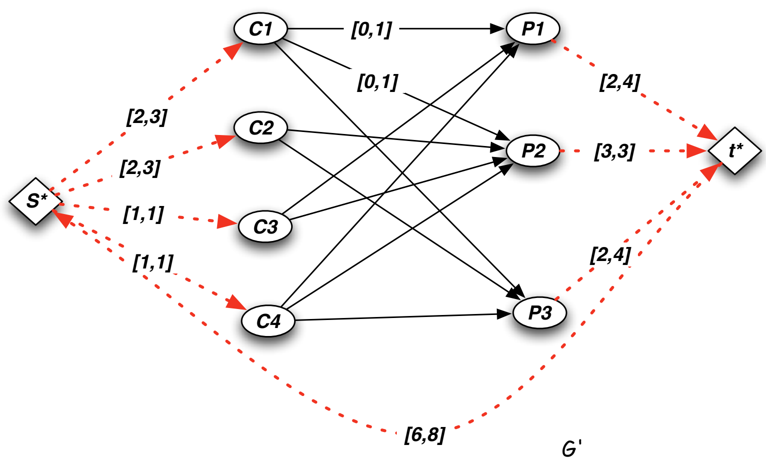
38

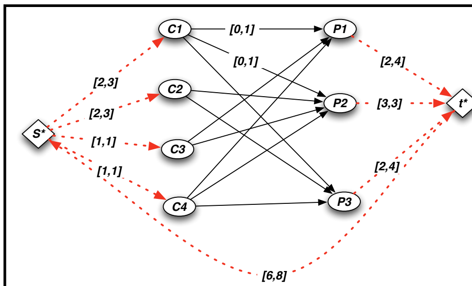
Convert to Network Flow Problem



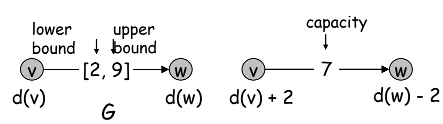
39

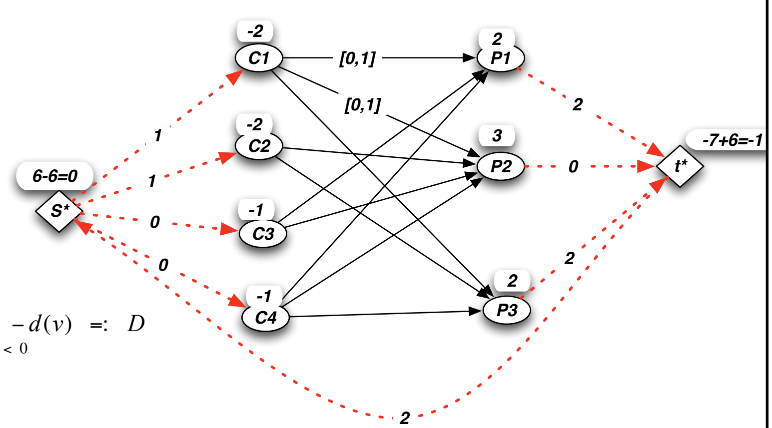
Model lower bounds with demands





Model lower bounds with demands

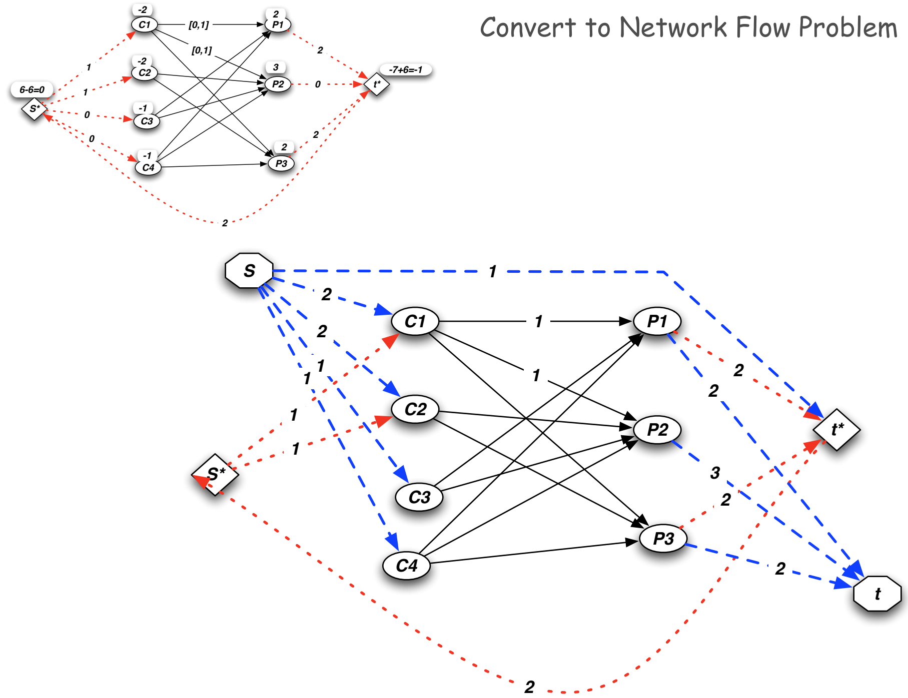




$$\sum_{v\,:\,d(v)>0} d(v) \;=\; \sum_{v\,:\,d(v)<0} -d(v) \;=:\; D$$

D = 7

Convert to Network Flow Problem

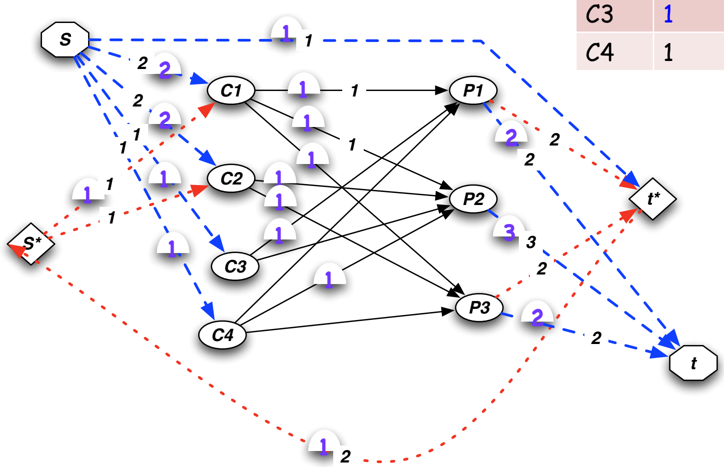


2

Convert to Network Flow Problem

D = 7, F = 7

	P1	P2	P3
C1	1	1	1
C2		1	1
C3	1	1	
C4	1	1	1



43

7.11 Project Selection

Project Selection

can be positive or negative ↓

Projects with prerequisites.

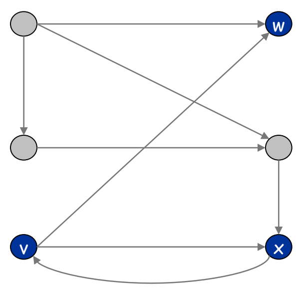
- Set P of possible projects. Project v has associated revenue p_v.
  - some projects generate money: create interactive e-commerce interface, redesign web page
  - others cost money: upgrade computers, get site license
- Set of prerequisites E. If $(v, w) \in E$, can't do project v and unless also do 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. Choose a feasible subset of projects to maximize revenue.

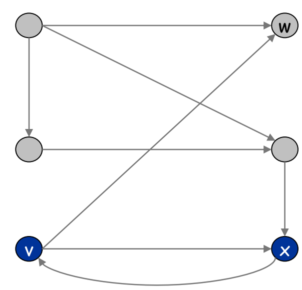
Project Selection: Prerequisite Graph

Prerequisite graph.

- Include an edge from v to w if can't do v without also doing w.
  - w is pre-requisites of v
- {v, w, x} is feasible subset of projects.
- {v, x} is infeasible subset of projects.



feasible

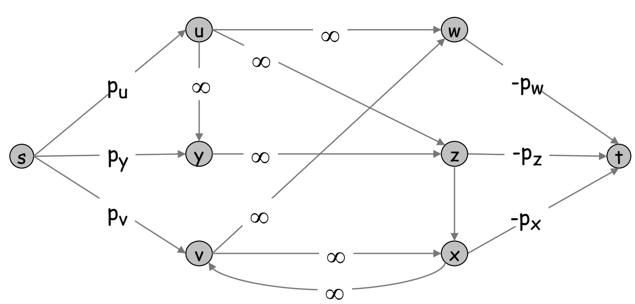


infeasible

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 $p_s = p_t = 0$.

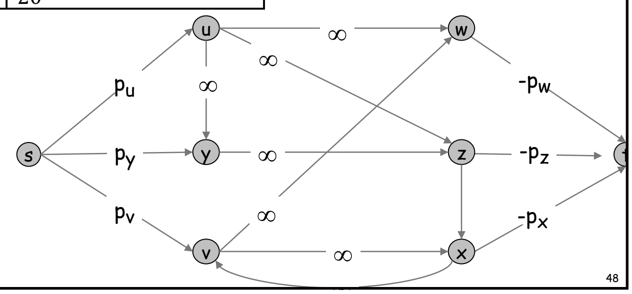


Project Selection: Min Cut Formulation

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	Prerequisites	Profit
P1		-10
P2	P1	15
P3	P1, P2	-5
P4	P2	10
P5	P3	20



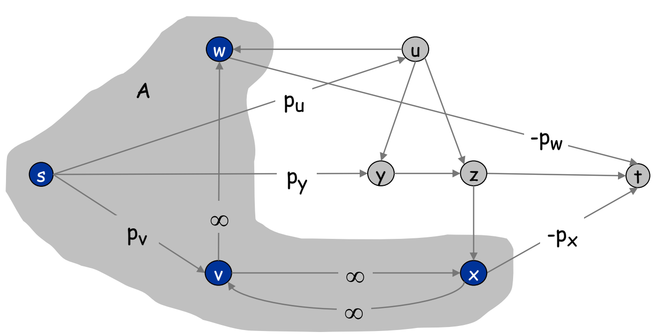
48

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:

$$\begin{aligned} cap(A, B) &= \sum_{v \in B:\, p_v > 0} p_v + \sum_{v \in A:\, p_v < 0} (-p_v) \\ &= \underbrace{\sum_{v:\, p_v > 0} p_v}_{\text{constant}} - \sum_{v \in A} p_v \end{aligned}$$



50

How to Find Min-Cut

From Ford-Fulkerson, we get capacity of minimum cut.

How to find a minimum cut?
Use residual graph.

Following are steps to find a minimum cut:

1) Run Ford-Fulkerson algorithm and consider the final residual graph G_f

2) Perform BFS or DFS from source s to find set A that including all reachable vertices from s in the residual graph G_f.

3) Define set B = V- A, then return (A, B) as a min-cut.

Solved Exer. 2: Doctor Assignment