Lecture 19a: Applications of Max Flow

Version of March 22, 2018

The Max Flow setup can model (surprisingly) many (seemingly) unrelated problems.

The idea is to express the problem as a max flow and then feed individual instances into our max flow solver.

The examples seen below all share the property that they are integer flow problems, e.g., all capacities are integral, so running-time analyses can use FF bound for integral flows.

1. Edge-Disjoint Paths
2. Circulations with Demands
3. Maximum Bipartite Matching
4. Baseball Elimination

1. Edge-Disjoint Paths

2. Circulations with Demands

3. Maximum Bipartite Matching

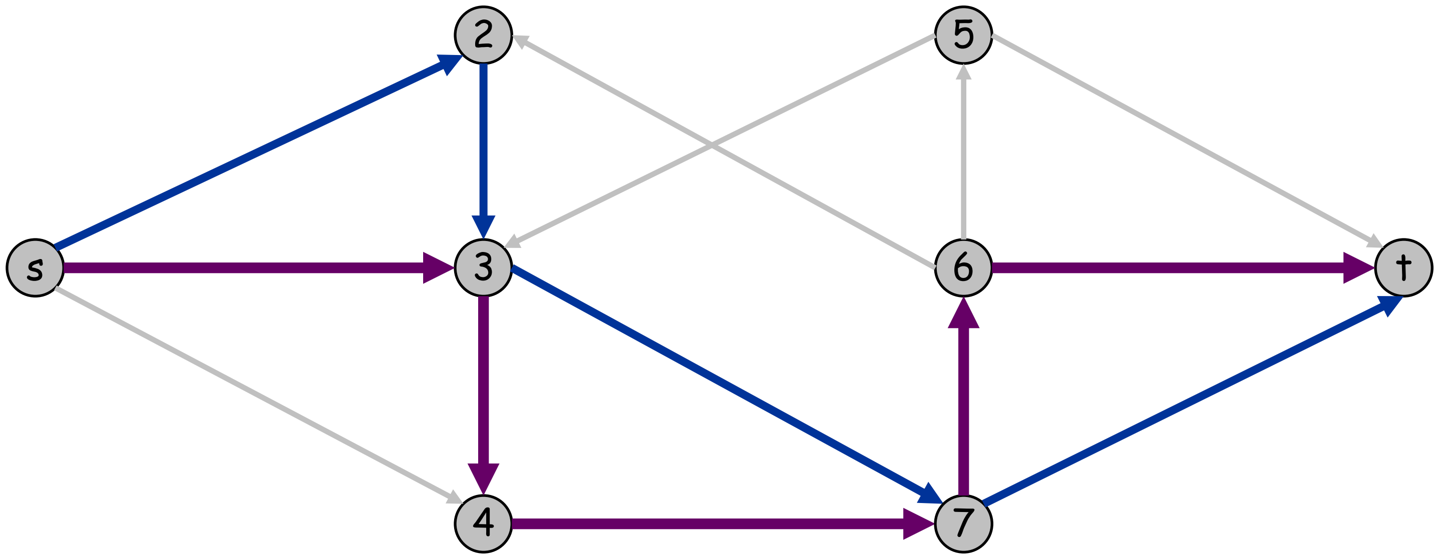
4. Baseball Elimination

Edge Disjoint Paths

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

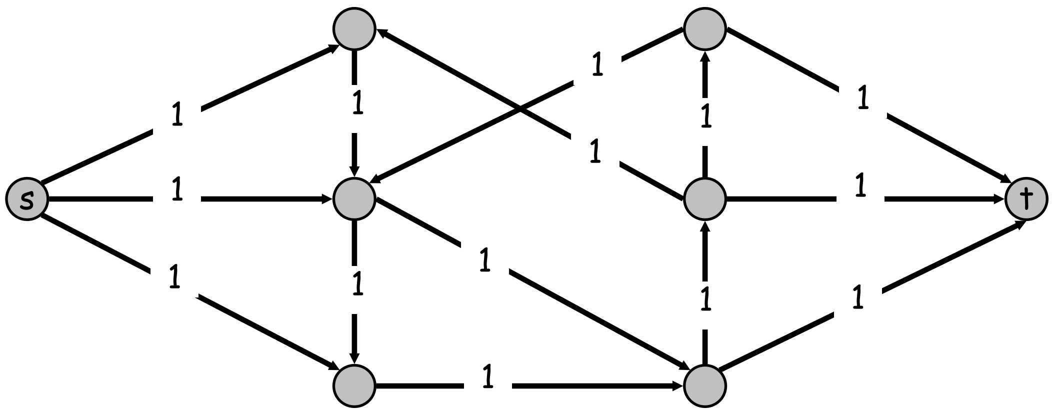
Def. Two paths are edge-disjoint if they have no edge in common.

Application: Communication networks.



Edge Disjoint Paths

Max flow formulation: assign unit capacity to every edge.



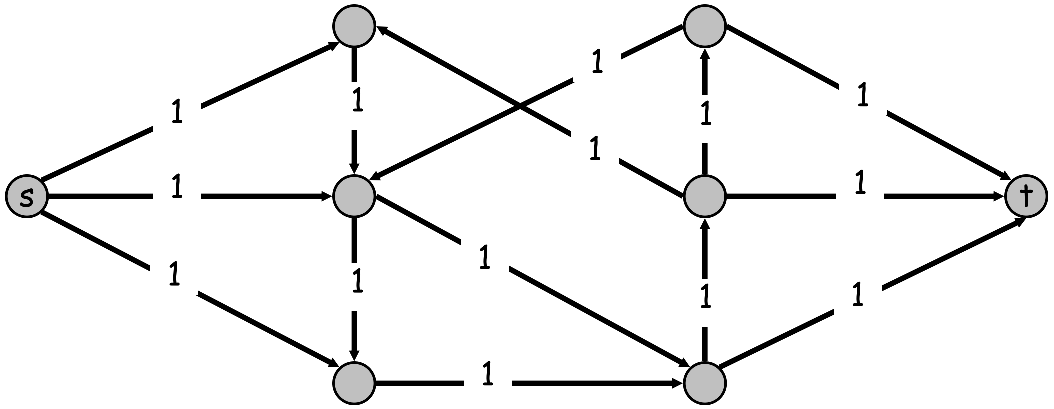
Theorem. Max number edge-disjoint s-t paths equals max flow value.

Proof. $\leq$

- Suppose there exists 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.



Proof. $\geq$

- Let f be a max flow in G' of value k computed by Ford-Fulkerson
- $f(e) = 1$ or 0 for every edge e (integrality property).

- Consider any edge (s, u) with $f(s, u) = 1$.
  - By conservation, there exists edge (u, v) with $f(u, v) = 1$
  - Continue to find the next unused edge out of v until reaching t.

- After finding one path, flow value decreases by 1.
- Repeat the process k times to find k edge-disjoint paths.
- The proof above also provides an **algorithm.**

1. Edge-Disjoint Paths
2. Circulations with Demands
3. Maximum Bipartite Matching
4. Baseball Elimination

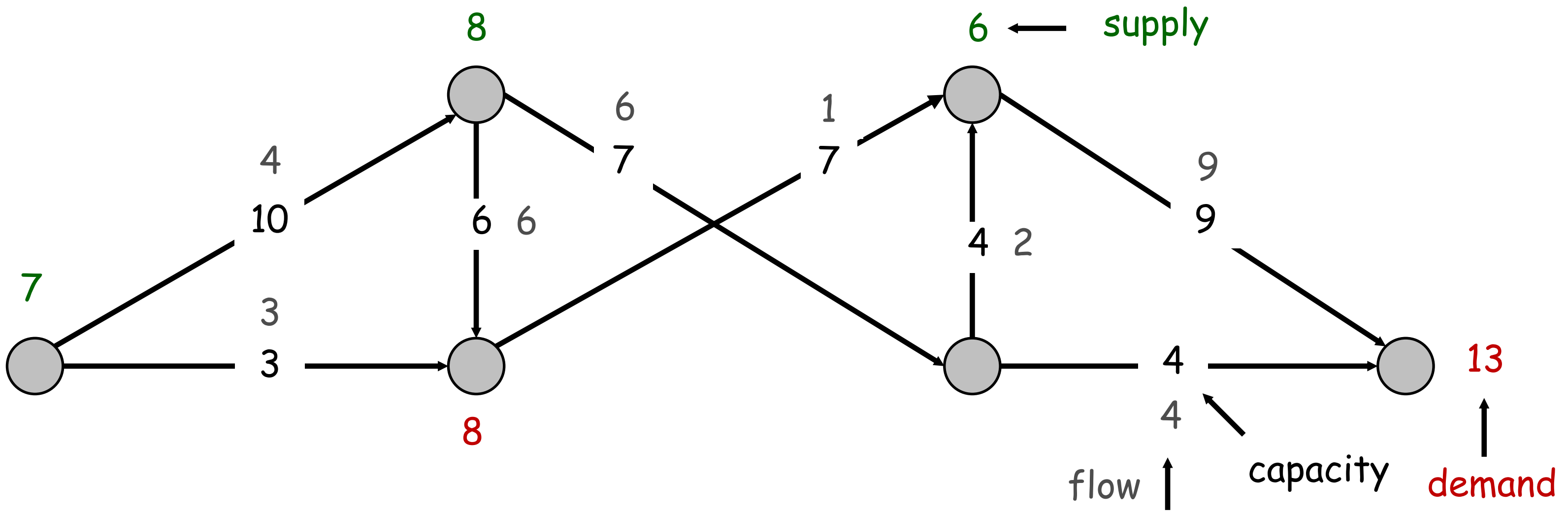
Circulation with Demands

Input: A directed connected graph $G = (V, E)$, in which

- every edge $e \in E$ has a capacity $c(e)$;
- a number of source vertices $s_1, s_2, \ldots$, each with a supply of $sup(s_i)$ and a number of target vertices $t_1, t_2, \ldots$, each with a demand of $dem(t_i)$;
- $\sum_i sup(s_i) \geq \sum_i dem(t_i)$

Output: A flow f that meets capacity and conservation conditions, and

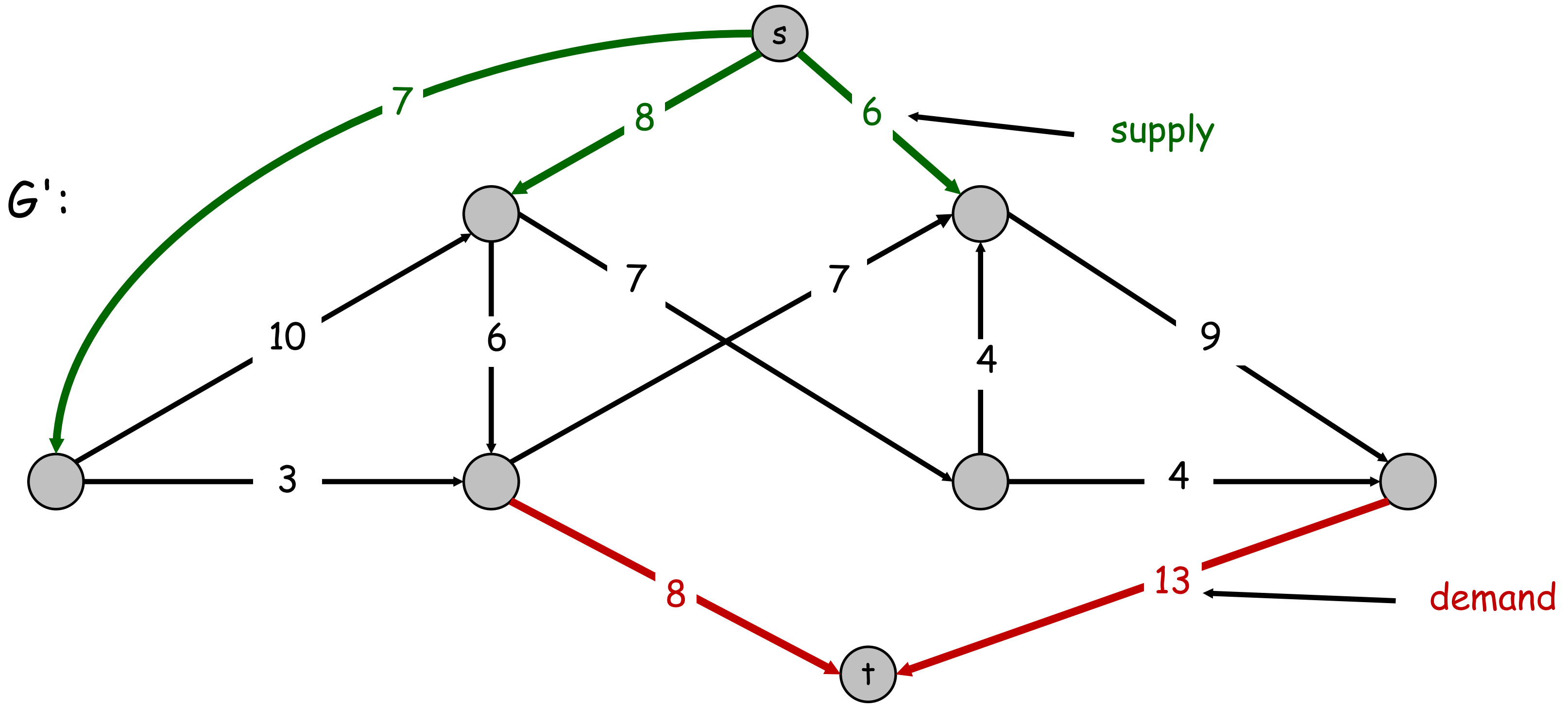
- At each source vertex s_i, $\sum_{e \text{ out of } s_i} f(e) - \sum_{e \text{ into } s_i} f(e) \leq sup(s_i)$;
- At each target vertex t_i, $\sum_{e \text{ into } t_i} f(e) - \sum_{e \text{ out of } t_i} f(e) = dem(t_i)$.



Solving Circulation with Demands using Max Flow

Algorithm:

- Add a “super source” s and a “super target” t.
- Add an edge from s to each s_i with capacity $sup(s_i)$.
- Add an edge from each t_i to t with capacity $dem(t_i)$.
- Compute the max flow f.
- If $|f| = \sum_i dem(t_i)$, then return f; else return “no solution”.



1. Edge-Disjoint Paths

2. Circulations with Demands

3. Maximum Bipartite Matching

4. Baseball Elimination

Maximum Bipartite Matching

A graph $G = (V, E)$ is Bipartite if there exists partition
$V = X \cup Y$ with $X \cap Y = \emptyset$ and $E \subseteq X \times Y$.

A Matching is a subset $M \subseteq E$
such that $\forall v \in V$ at most one edge in M is incident upon v.

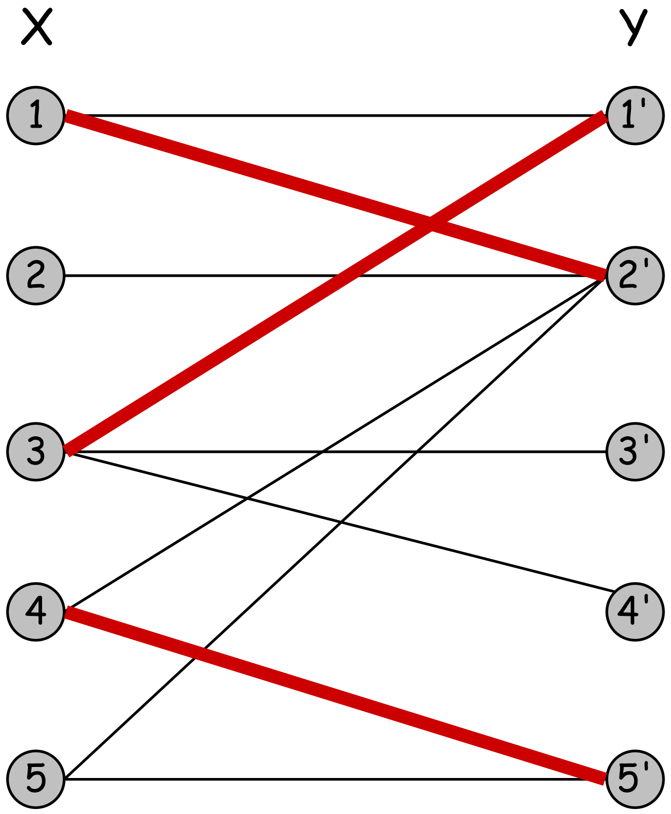
The Size $|M|$ is the number of edges in M.

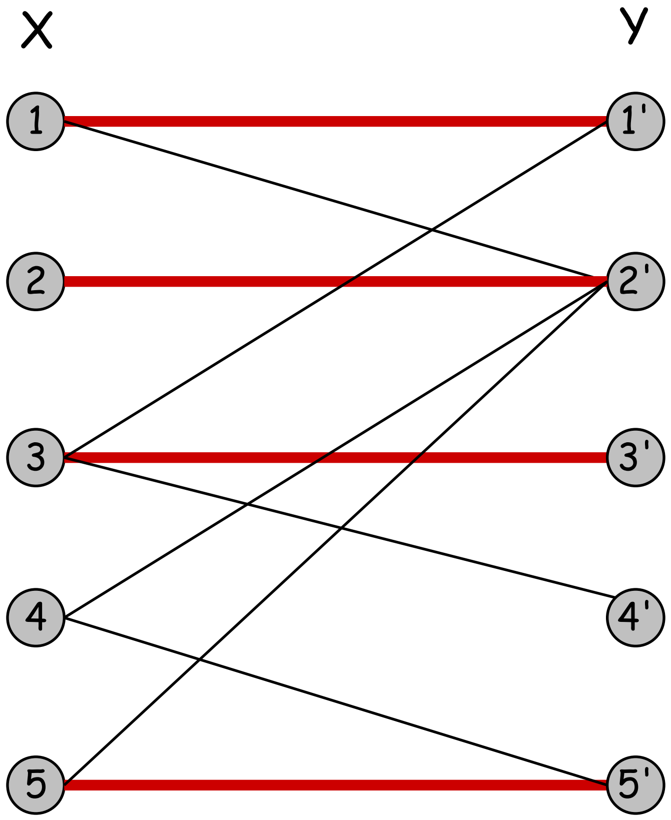
A Maximum Matching is matching M such that
every other matching M' satisfies $|M'| \leq M$.

Problem: Given bipartite graph G, find a Maximum Matching.

Applications: Assign jobs to people, tasks to machines, etc.

Bipartite Matching Example





<u>Matching</u>

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

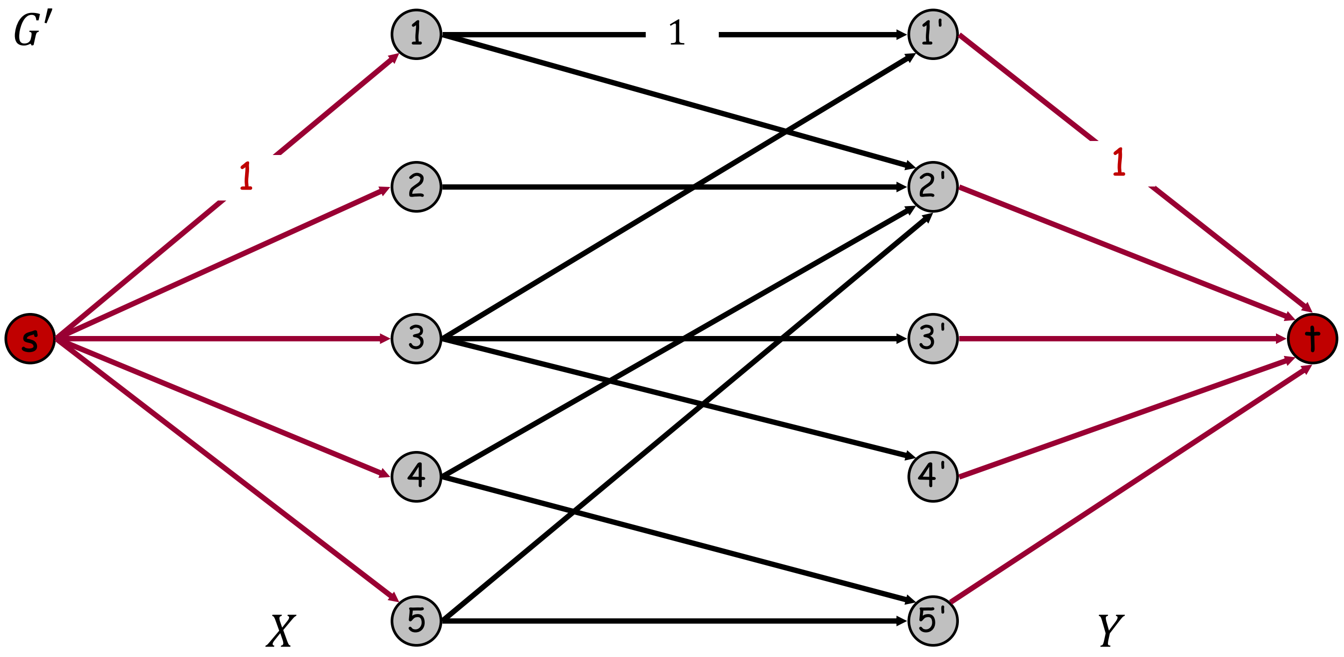
<u>Max Matching</u>

1-1', 2-2', 3-3' 4-4'

From bipartite matching to flow

Max flow formulation.

- Create directed graph $G' = (X \cup Y \cup \{s, t\}, E')$.
- Direct all edges from X to Y, and assign them capacity 1.
- Add source s, and unit capacity edges from s to each node in X.
- Add target t, and unit capacity edges from each node in Y to t.

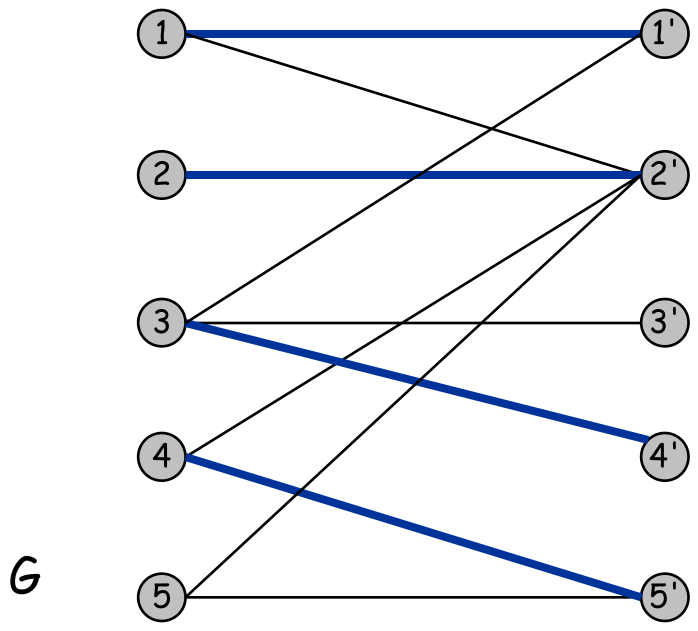


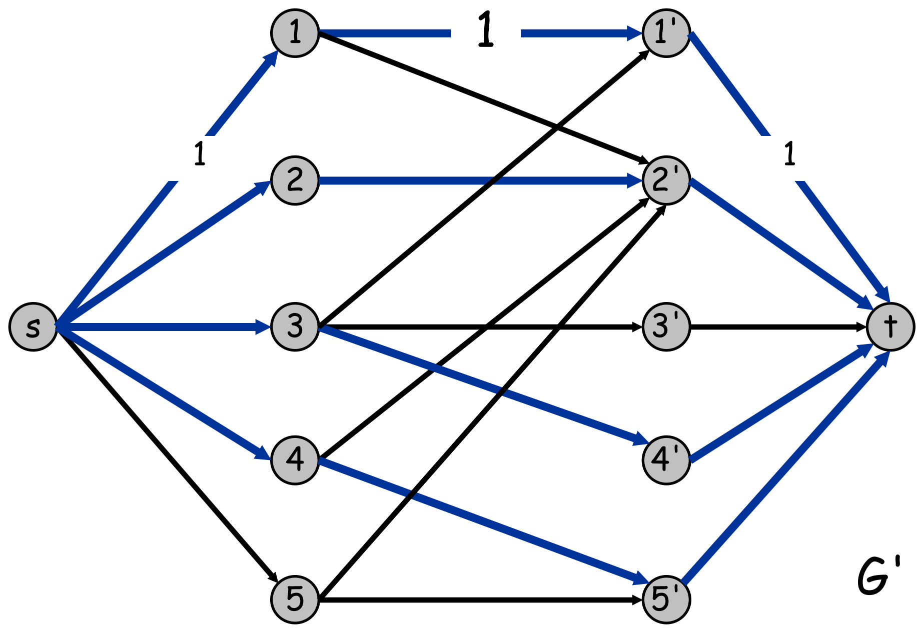
Maximum Bipartite Matching: Proof of Correctness

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 the $3k$ edges.
  (s,x), (x,y), (y,t) where (x,y) is an edge in the matching
- f is a flow, and has cardinality k.



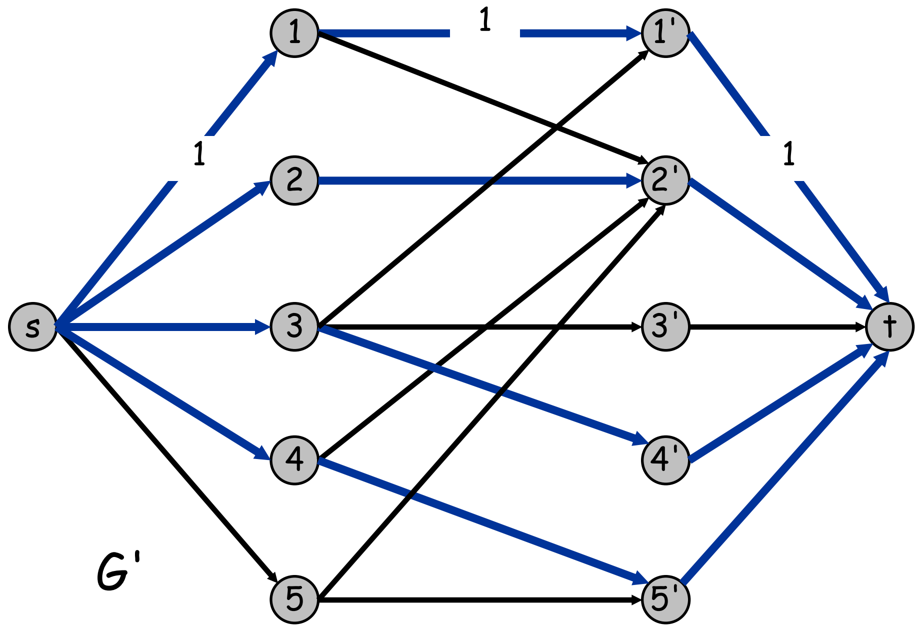


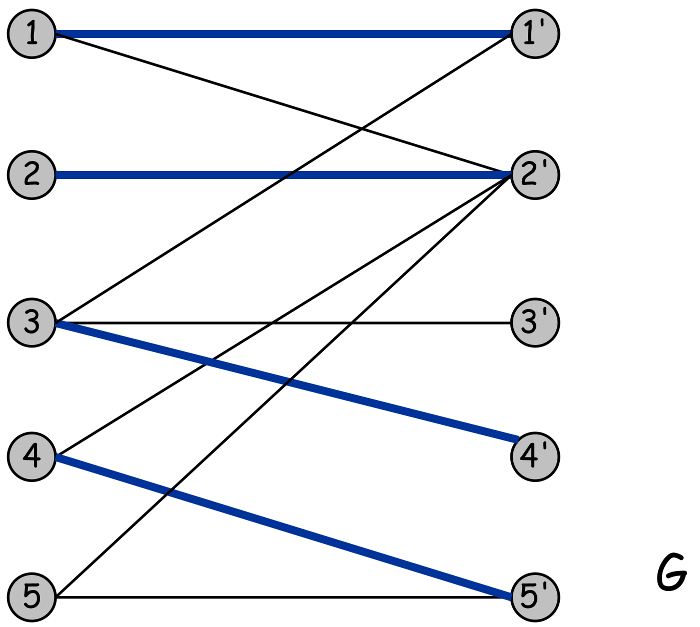
Maximum Bipartite Matching: Proof of Correctness

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 computed by Ford-Fulkerson
- $f(e) = 1$ or 0 for every edge e (because of integrality of F-F solution)
- Consider M = set of edges from X to Y with $f(e) = 1$.
  - each node in X and Y participates in at most one edge in M
  - $|M| = k$





Maximum Bipartite matching: Running time

Algorithm:

- Run F-F on the constructed graph G'
- Report all original edges from G that have flow 1
- Correct by analysis on previous page
- Correct no matter how augmenting paths are chosen

Running time: $O(VE)$

- Each iteration increases $|f|$ by at least 1.
- $|f^*| \le V/2$
- Each iteration takes $O(E)$ time.

Specialized algorithms

- $O(\sqrt{V}E)$ [Hopcroft–Karp, 1973]
- $O(V^{2.376})$ using matrix multiplication [Mucha-Sankowski, 2004]
- $O(E^{10/7})$ [Madry, 2013]
- But in practice, they are not as good as using max flow algorithms.

1. Edge-Disjoint Paths
2. Circulations with Demands
3. Maximum Bipartite Matching
4. Baseball Elimination

Baseball Elimination

Team i	Wins w_i	To play r_i	Remaining Against = r_{ij}			
			1	2	3	4
1	3	2	-	1	1	0
2	2	3	1	-	1	1
3	2	3	1	1	-	1
4	0	2	0	1	1	-

Rule: Order teams by the number of wins.

Q: Does Team 4 still have a chance to finish in first place (tie is OK)?

A: No, obviously.

Baseball Elimination

Team i	Wins w_i	To play r_i	Remaining Against = r_{ij}			
			1	2	3	4
1	3	2	-	1	1	0
2	2	3	1	-	1	1
3	2	3	1	1	-	1
4	1	2	0	1	1	-

Q: Does Team 4 still have a chance to finish in first place (tie is OK)?

A: No, because
- Team 4 has to win both remaining games against team 2 and 3.
- Team 1 has to lose both remaining games against team 2 and 3.
- Then 2 and 3 will both have 3 wins.
- The game between team 2 and 3 will give one of them one more win.

Suppose you need to do this for MLB / Premier League…

Baseball Elimination: Formal Definition

Input:

- n teams: $1, 2, \dots, n$
- One particular team, say n (without loss of generality)
- Team i has won w_i games already
- Team i and j still need to play r_{ij} games, $r_{ij} = 0$ or 1.
- Team i has a total of $r_i = \sum_j r_{ij}$ games to play

Output:

- "Yes", if there is an outcome for each remaining game such that team n finishes with the most wins (tie is OK).
- "No", if no such possibilities.

Brute-force algorithm:

- For each remaining game, consider two possible outcomes.
- Try all 2^r possible combinations, where $r = \sum_{i,j} r_{ij}$

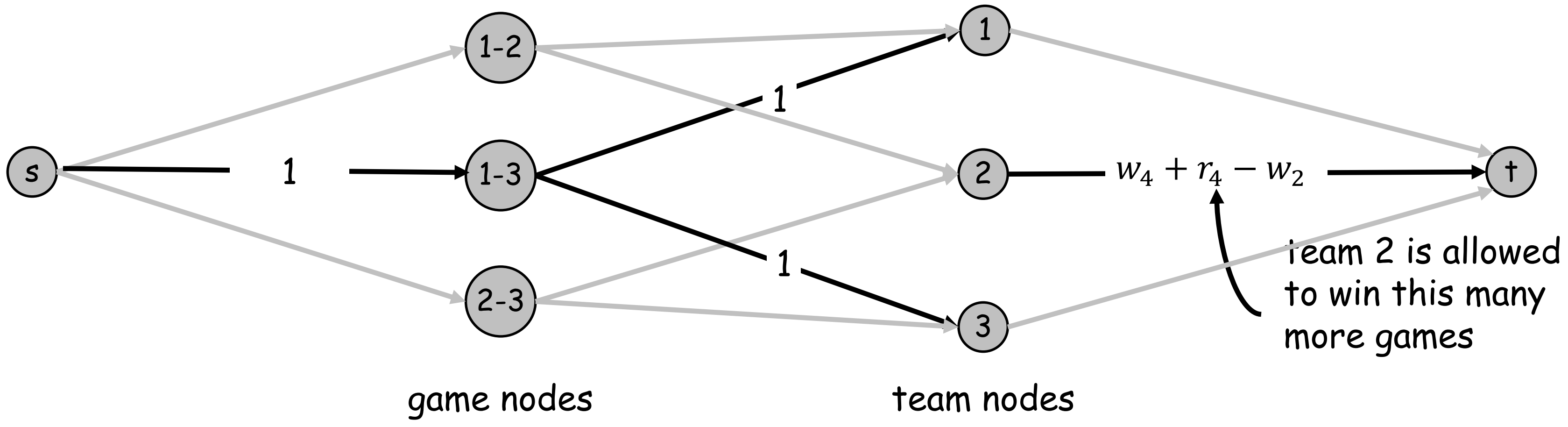
Baseball Elimination: Max Flow Formulation

Can team n finish with most wins?

- Assume team n wins all remaining games $\Rightarrow w_n + r_n$ wins.
- All other teams must have $\leq w_n + r_n$ wins.

Flow network construction:

- A source s and a target t
- A node for each remaining game (i, j); and an edge from s to it with capacity 1
- A node for each team $i = 1, 2, \dots, n-1$; and an edge from it to t with capacity $w_n + r_n - w_i$
- Game node (i, j) has edges to team node i and j, with capacity 1



Baseball Elimination: Max Flow Formulation

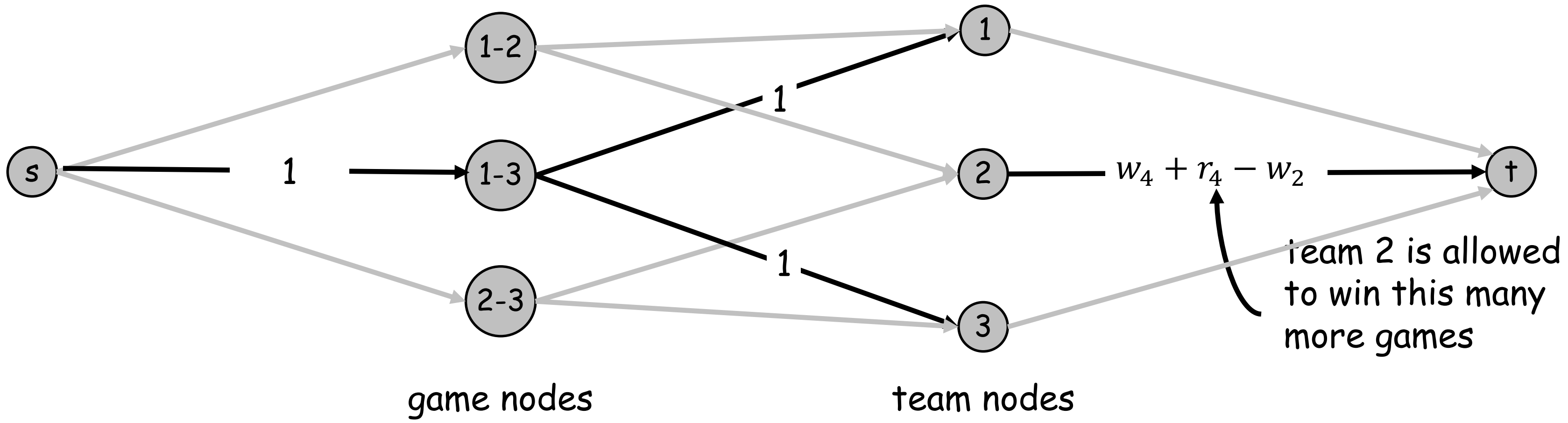
Claim: There is a way for team n to finish in the first place iff the max flow has value $r = \sum_{i,j} r_{ij}$.

Proof: "⇒": Suppose there is an outcome for each remaining game such that team n finishes the first. First set $f\big(s,(i,j)\big) = 1$ for all (i,j).

For each remaining game (i,j):

- if i wins, set $f\big((i,j),i\big) = 1$ and $f\big((i,j),j\big) = 0$;
- if j wins, set $f\big((i,j),j\big) = 1$ and $f\big((i,j),i\big) = 0$.

Team i wins $\leq w_n + r_n - w_i$ games, so it can send all incoming flow to t.



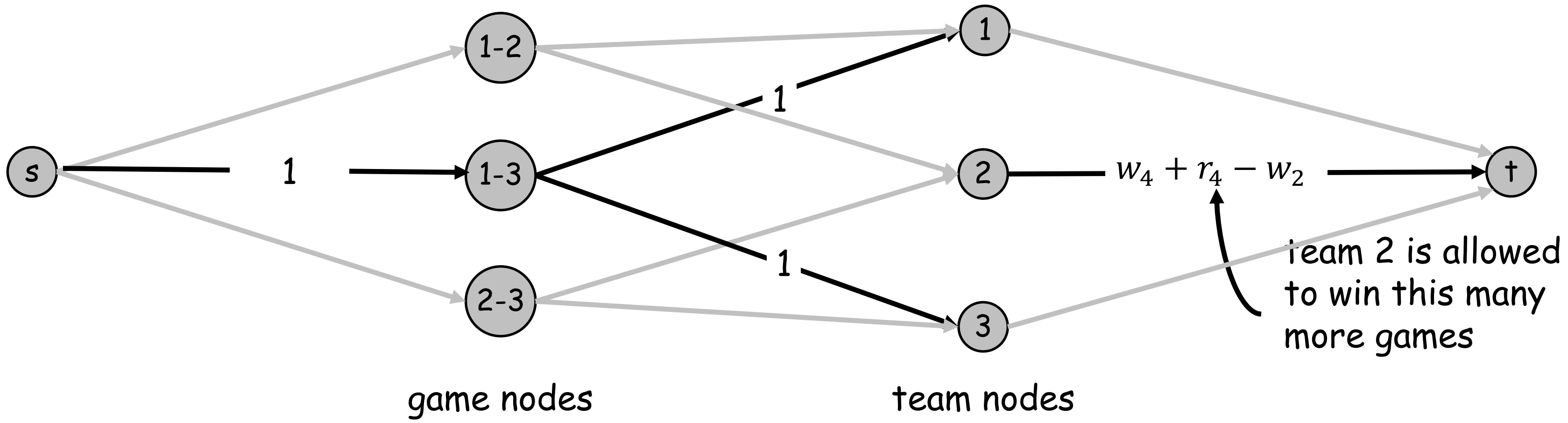
Baseball Elimination: Max Flow Formulation

Proof: "$\Leftarrow$": Suppose the max flow f has $|f| = r$. It must saturate all edges out of s.

Look at each game node (i,j). Exactly one of its outgoing edges must have 1 unit of flow (integrality property):

- If $f\big((i,j),i\big) = 1$, let i win the game;
- If $f\big((i,j),j\big) = 1$, let j win the game.

Team node i receives $\leq w_n + r_n - w_i$ units of flow, each corresponding to one win, so it cannot beat team n.



Baseball Elimination: Extensions

Q: What if r_{ij} can be more than 1?

Q: Can this be used for football (soccer) leagues?

- Using the old rule: Winner takes 2 points, loser 0 point; each team gets 1 point in case of a tie.