Algorithm Theory

Chapter 6
Graph Algorithms

Maximum Flow Applications

Fabian Kuhn

Maximum Flow Applications

- Maximum flow has many applications
- Reducing a problem to a max flow problem can even be seen as an important algorithmic technique
- Examples:
  - related network flow problems
  - computation of small cuts
  - computation of matchings
  - computing disjoint paths
  - scheduling problems
  - assignment problems with some side constraints
  - …

Undirected Edges and Vertex Capacities

Undirected Edges:

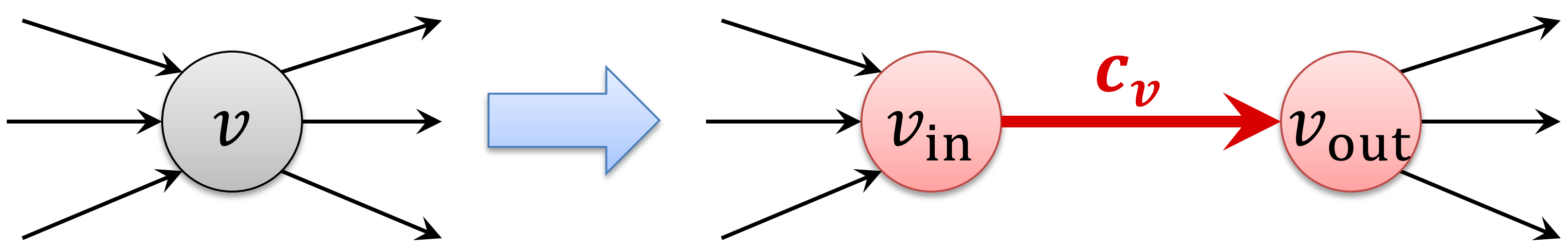
- Undirected edge $\{u, v\}$: add edges (u, v) and (v, u) to network

Vertex Capacities:

- Not only edges, but also (or only) nodes have capacities
- Capacity c_v of node $v \notin \{s, t\}$:

$$f^{\text{in}}(v) = f^{\text{out}}(v) \leq c_v$$

- Replace node v by edge $e_v = \{v_{\text{in}}, v_{\text{out}}\}$:

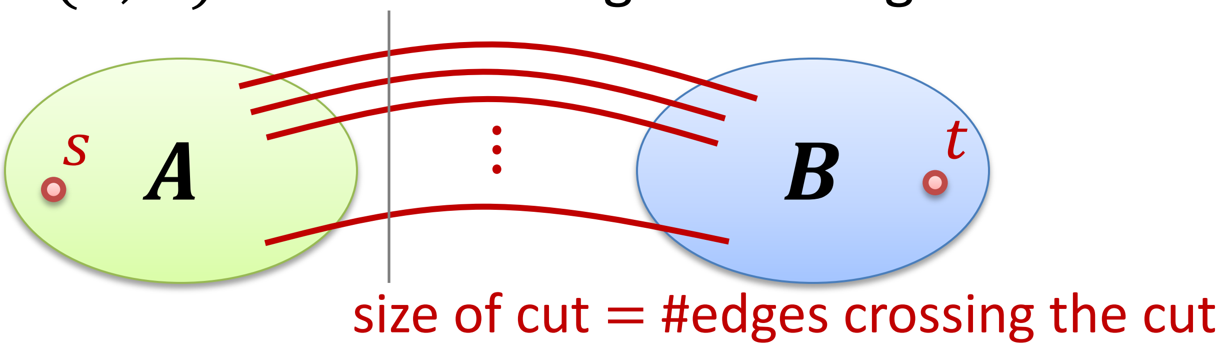


Minimum s-t Cut

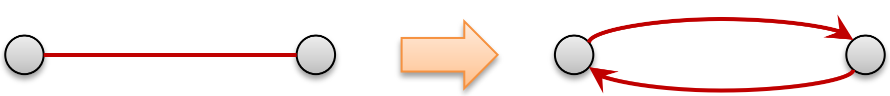
Given: undirected graph $G = (V, E)$, nodes $s, t \in V$

***s*-*t* cut:** Partition (A, B) of V such that $s \in A$, $t \in B$

Size of cut (A, B): number of edges crossing the cut



Objective: find s-t cut of minimum size

- Create flow network:
  - make edges directed: 
  - edge capacities $= 1$

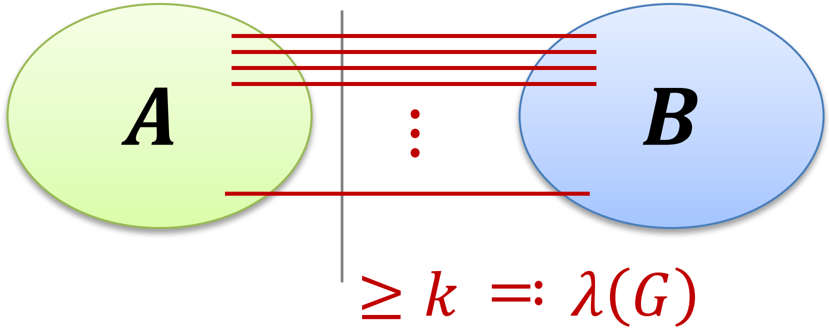
- Size of cut in G = capacity of cut in flow network

Edge Connectivity

Definition: A graph $G = (V, E)$ is k-edge connected for an integer $k \geq 1$ if the graph $G_X = (V, E \setminus X)$ is connected for every edge set

$$X \subseteq E, |X| \leq k - 1.$$

Need to remove $\geq k$ edges to disconnect G



Edge Connectivity $\lambda(G)$

max k such that G is k-edge connected.

Goal: Compute *edge connectivity $\lambda(G)$* of G
(and edge set X of size $\lambda(G)$ that divides G into ≥ 2 parts)

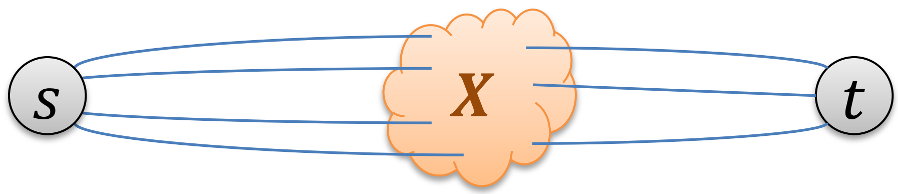
- minimum set X is a minimum s-t cut for some $s, t \in V$
  - Actually for all s, t in different components of $G_X = (V, E \setminus X)$
- Fix s, find min s-t cut for all $t \neq s \Longrightarrow$ running time $O(mn^2)$

Minimum s-t Vertex-Cut

Given: undirected graph $G = (V, E)$, nodes $s, t \in V$

***s*-*t* vertex cut:** Set $X \subset V$ such that $s, t \notin X$ and s and t are in different components of the sub-graph $G[V \setminus X]$ induced by $V \setminus X$

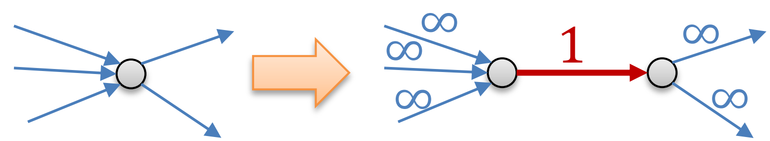
Size of vertex cut: $|X|$



Objective: find s-t vertex-cut of minimum size

- Replace undirected edges $\{u, v\}$ by (u, v) and (v, u)
- Compute max s-t flow for edge capacities ∞ and node capacities

$$c_v = 1 \text{ for } v \neq s, t$$



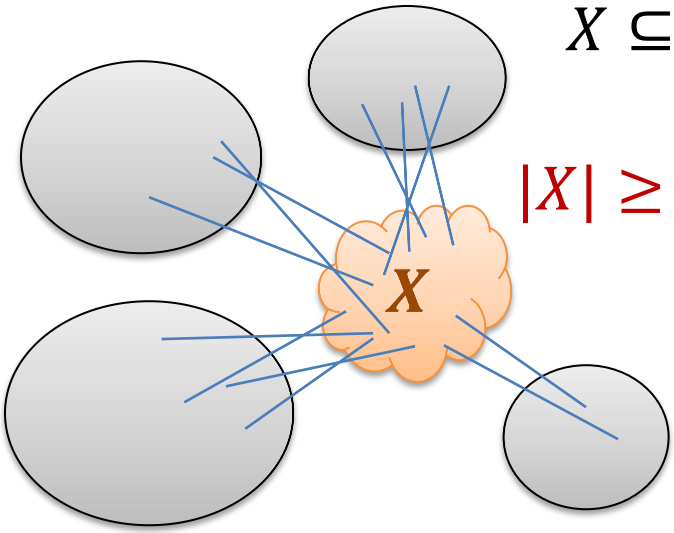
- Replace each node v by v_{in} and v_{out}
- Min edge cut corresponds to min vertex cut in G

Vertex Connectivity

Definition: A graph $G = (V, E)$ is k-vertex connected for an integer $k \geq 1$ if the sub-graph $G[V \setminus X]$ induced by $V \setminus X$ is connected for every vertex set

$X \subseteq V, |X| \leq k - 1.$

Need to remove $\geq k$ nodes to disconnect G



$|X| \geq k =: \kappa(G)$

Vertex Connectivity $\kappa(G)$

max k such that G is k-vertex connected.

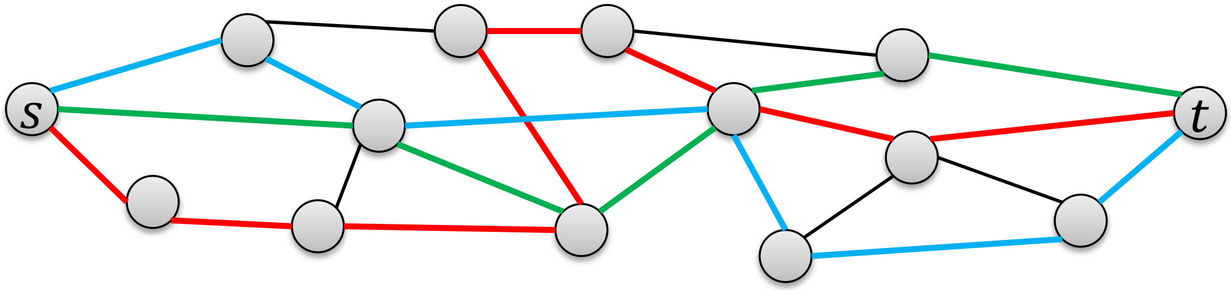
Goal: Compute vertex connectivity $\kappa(G)$ of G
(and node set X of size $\kappa(G)$ that divides G into ≥ 2 parts)

- Compute minimum s-t vertex cut for all s and all $t \neq s$ such that t is not a neighbor of s $\implies$ running time $O(m \cdot n^3)$

Edge-Disjoint Paths

Given: Graph $G = (V, E)$ with nodes $s, t \in V$

Goal: Find as many edge-disjoint s-t paths as possible



Solution:

- Find max s-t flow in G with edge capacities $c_e = 1$ for all $e \in E$

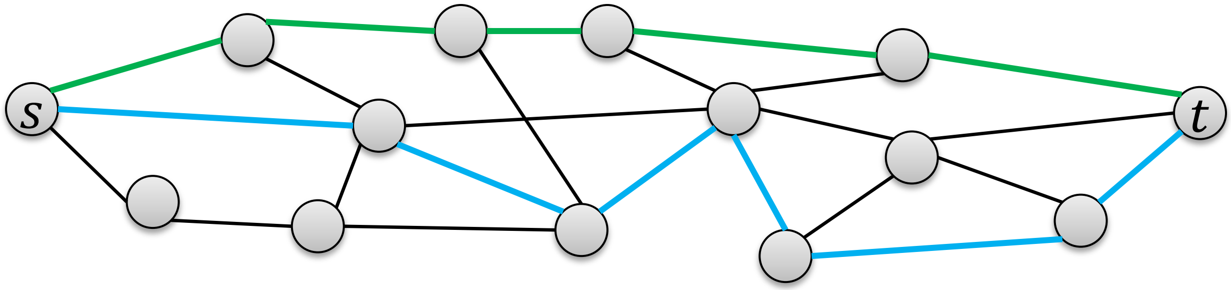
Flow f induces $|f|$ edge-disjoint paths:

- Integral capacities → can compute integral max flow f
- Get $|f|$ edge-disjoint paths by greedily picking them
- Correctness follows from flow conservation $f^{\text{in}}(v) = f^{\text{out}}(v)$

Vertex-Disjoint Paths

Given: Graph $G = (V, E)$ with nodes $s, t \in V$

Goal: Find as many internally vertex-disjoint s-t paths as possible



Solution:

- Find max s-t flow in G with node capacities $c_v = 1$ for all $v \in V$

Flow f induces $|f|$ vertex-disjoint paths:

- Integral capacities → can compute integral max flow f
- Get $|f|$ vertex-disjoint paths by greedily picking them
- Correctness follows from flow conservation $f^{\text{in}}(v) = f^{\text{out}}(v)$

Menger's Theorem

Theorem: (edge version)
For every graph $G = (V, E)$ with nodes $s, t \in V$, the size of the minimum s-t (edge) cut equals the maximum number of pairwise edge-disjoint paths from s to t.

Theorem: (node version)
For every graph $G = (V, E)$ with non-adjacent nodes $s, t \in V$, the size of the minimum s-t vertex cut equals the maximum number of pairwise internally vertex-disjoint paths from s to t.

- Both versions can be seen as a special case of the max flow min cut theorem

Baseball Elimination

Team i	Wins w_i	Losses ℓ_i	To Play r_i	Against = r_{ij}				
				NY	Balt.	T. Bay	Tor.	Bost.
New York	81	69	12	-	2	5	2	3
Baltimore	79	77	6	2	-	2	1	1
Tampa Bay	79	74	9	5	2	-	1	1
Toronto	76	80	6	2	1	1	-	2
Boston	71	84	7	3	1	1	2	-

- Only wins/losses possible (no ties), winner: team with most wins
- Which teams can still win (as least as many wins as top team)?
- Boston is eliminated (cannot win):
  - Boston can get at most 78 wins, New York already has 81 wins
- If for some i, j: $w_i + r_i < w_j$ → team i is eliminated
- Sufficient condition, but not a necessary one!

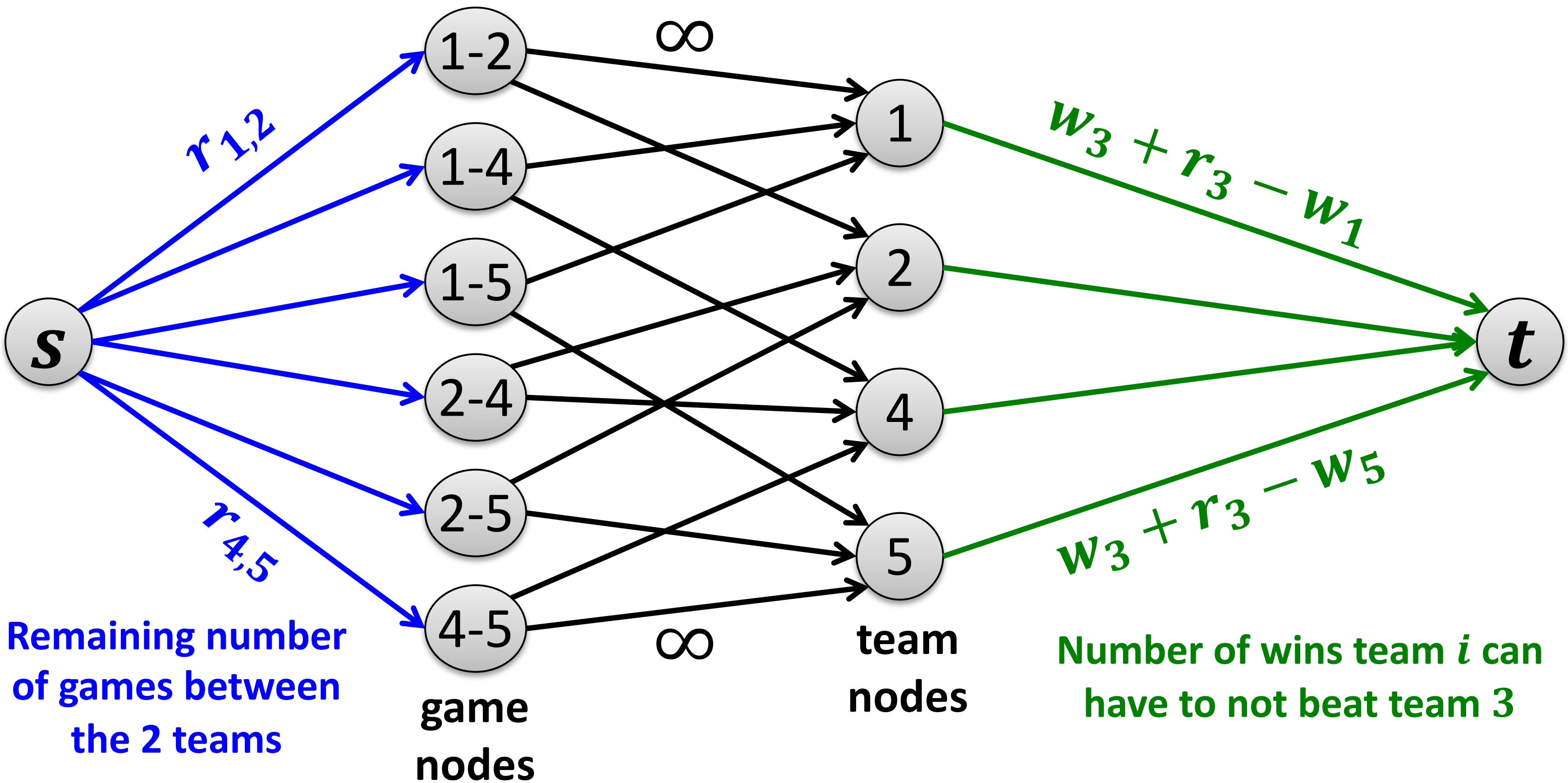
Baseball Elimination

Team i	Wins w_i	Losses ℓ_i	To Play r_i	Against = r_{ij}				
				NY	Balt.	T. Bay	Tor.	Bost.
New York	81	69	12	-	2	5	2	3
Baltimore	79	77	6	2	-	2	1	1
Tampa Bay	79	74	9	5	2	-	1	1
Toronto	76	80	6	2	1	1	-	2
Boston	71	84	7	3	1	1	2	-

- Can Toronto still finish first?
- Toronto can get $82 > 81$ wins, but:
  NY and Tampa have to play 5 more times against each other
  → if NY wins two, it gets 83 wins, otherwise, Tampa has 83 wins
- Hence: Toronto cannot finish first
- How about the others? How can we solve this in general?

Max Flow Formulation

- Can team 3 finish with most wins?



- Team 3 can finish first iff all source-game edges are saturated

Reason for Elimination

AL East: Aug 30, 1996

Team i	Wins w_i	Losses ℓ_i	To Play r_i	Against = r_{ij}				
				NY	Balt.	Bost.	Tor.	Detr.
New York	75	59	28	-	3	8	7	3
Baltimore	71	63	28	3	-	2	7	4
Boston	69	66	27	8	2	-	0	0
Toronto	63	72	27	7	7	0	-	0
Detroit	49	86	27	3	4	0	0	-

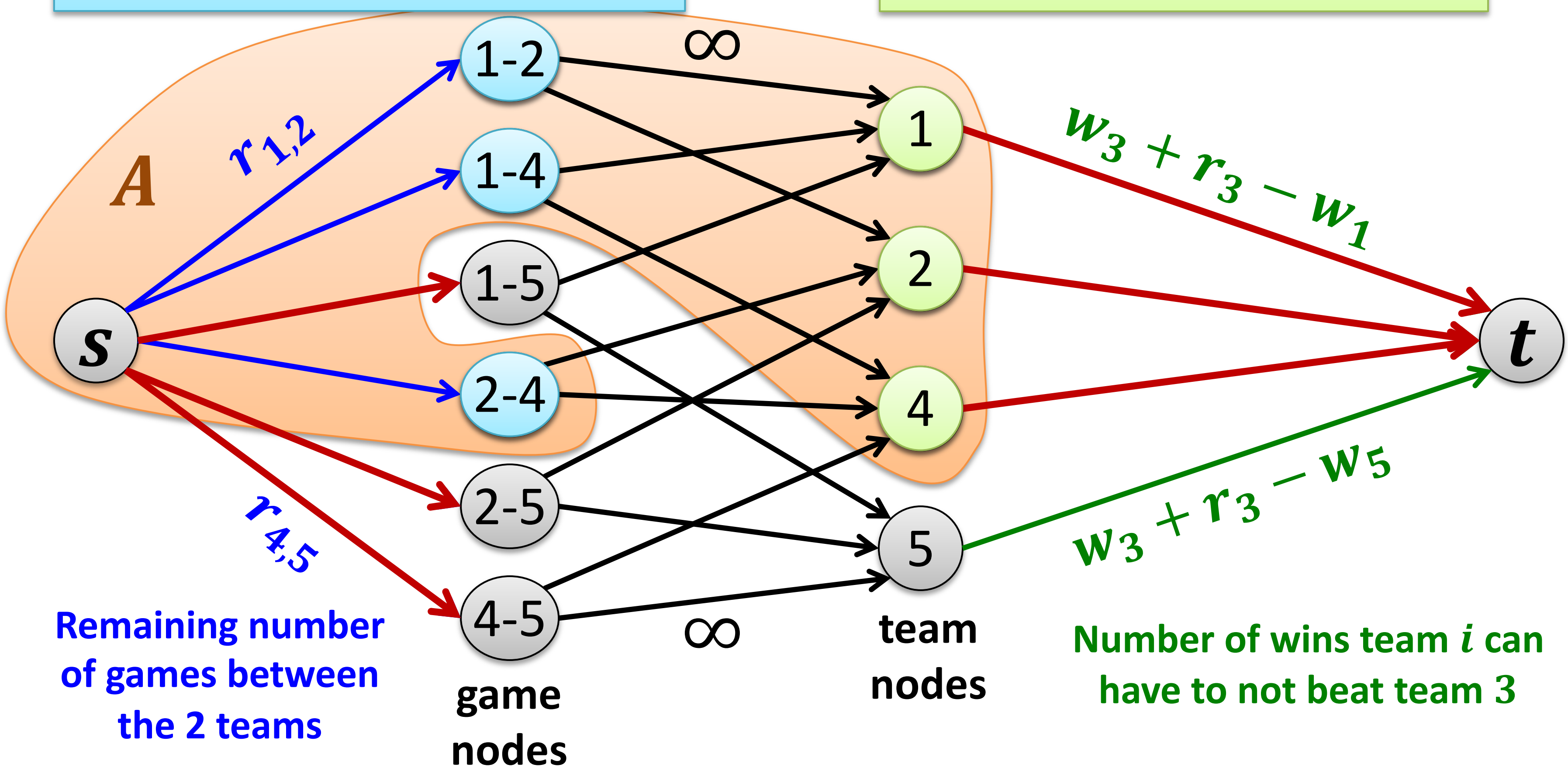
- Detroit could finish with $49 + 27 = 76$ wins
- Consider $R = \{\text{NY}, \text{Bal}, \text{Bos}, \text{Tor}\}$
  - Have together already won $w(R) = 278$ games
  - Must together win at least $r(R) = 27$ more games
- On average, teams in R win $\frac{278+27}{4} = 76.25$ games

Reason for Elimination

Team 3 eliminated $\Leftrightarrow$ min cut $(A, V \setminus A)$ of cap. < "all blue edges"

A contains all game nodes for teams in R

A contains team nodes R with $R \neq \emptyset$

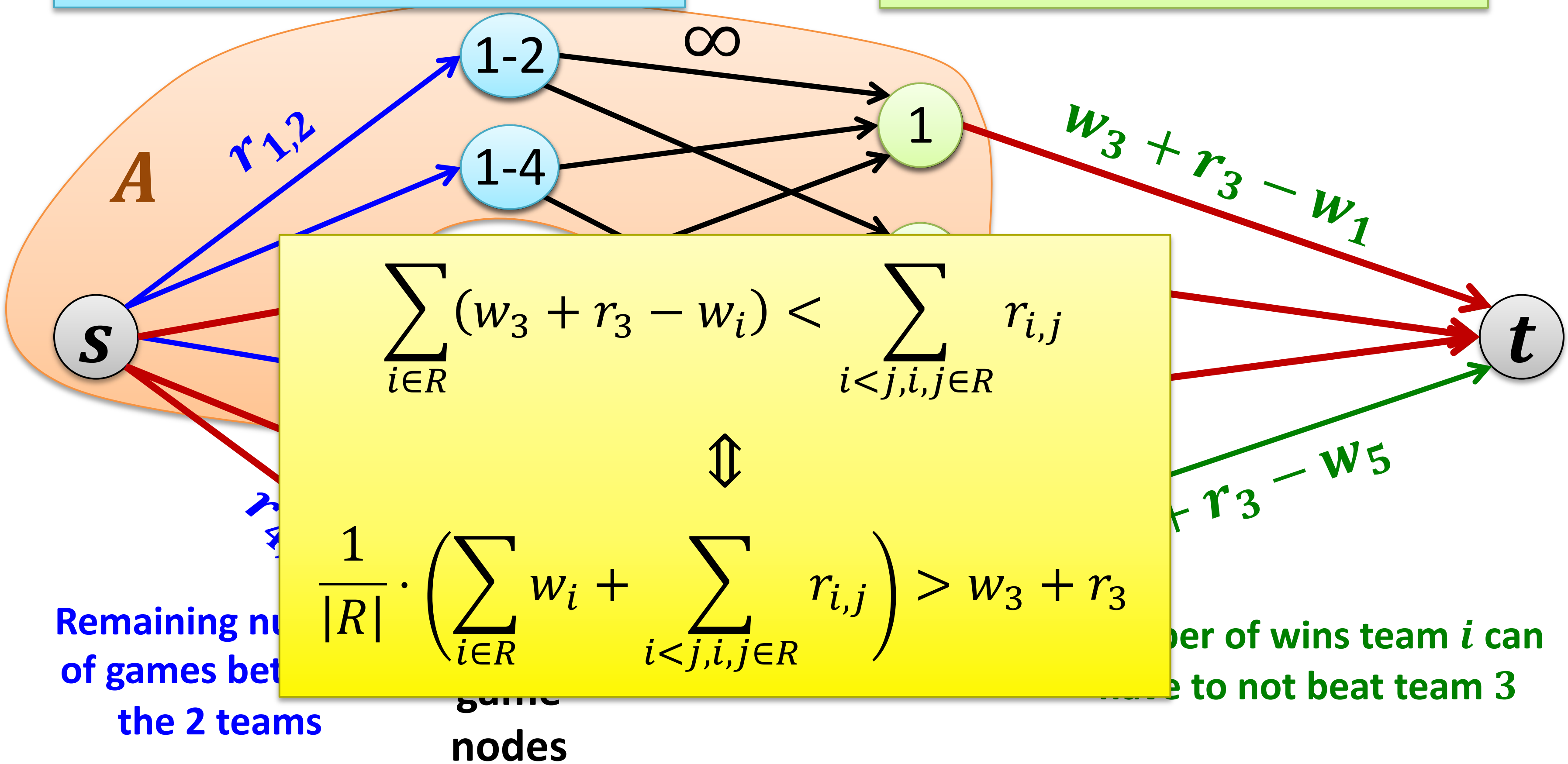


Reason for Elimination

Team 3 eliminated $\Leftrightarrow$ min cut $(A, V \setminus A)$ of cap. < "all blue edges"

A contains all game nodes for teams in R

A contains team nodes R with $R \neq \emptyset$



$$\sum_{i \in R} (w_3 + r_3 - w_i) < \sum_{i<j,\, i,j \in R} r_{i,j}$$

$$\Updownarrow$$

$$\frac{1}{|R|} \cdot \left(\sum_{i \in R} w_i + \sum_{i<j,\, i,j \in R} r_{i,j} \right) > w_3 + r_3$$

Reason for Elimination

Certificate of elimination:

$$R \subseteq X, \qquad w(R) := \underbrace{\sum_{i \in R} w_i}_{\text{\#wins of nodes in } R}, \qquad r(R) := \underbrace{\sum_{i,j \in R} r_{i,j}}_{\text{\#remaining games among nodes in } R}$$

- Team $x \in X$ is eliminated by $R \subseteq X \setminus \{x\}$ if

$$\frac{w(R) + r(R)}{|R|} > w_x + r_x.$$

- If team $x \in X$ is eliminated, there exists $R \subseteq X \setminus \{x\}$ such that team x is eliminated by R.
  - R can be constructed by looking at a minimum cut

Circulations with Demands

Given: Directed network with positive edge capacities

Sources & Sinks: Instead of one source and one destination, several sources that generate flow and several sinks that absorb flow.

Supply & Demand: sources have supply values, sinks demand values

Goal: Compute a flow such that source supplies and sink demands are exactly satisfied

- The circulation problem is a feasibility rather than a maximization problem

Circulations with Demands: Formally

Given: Directed network $G = (V, E)$ with

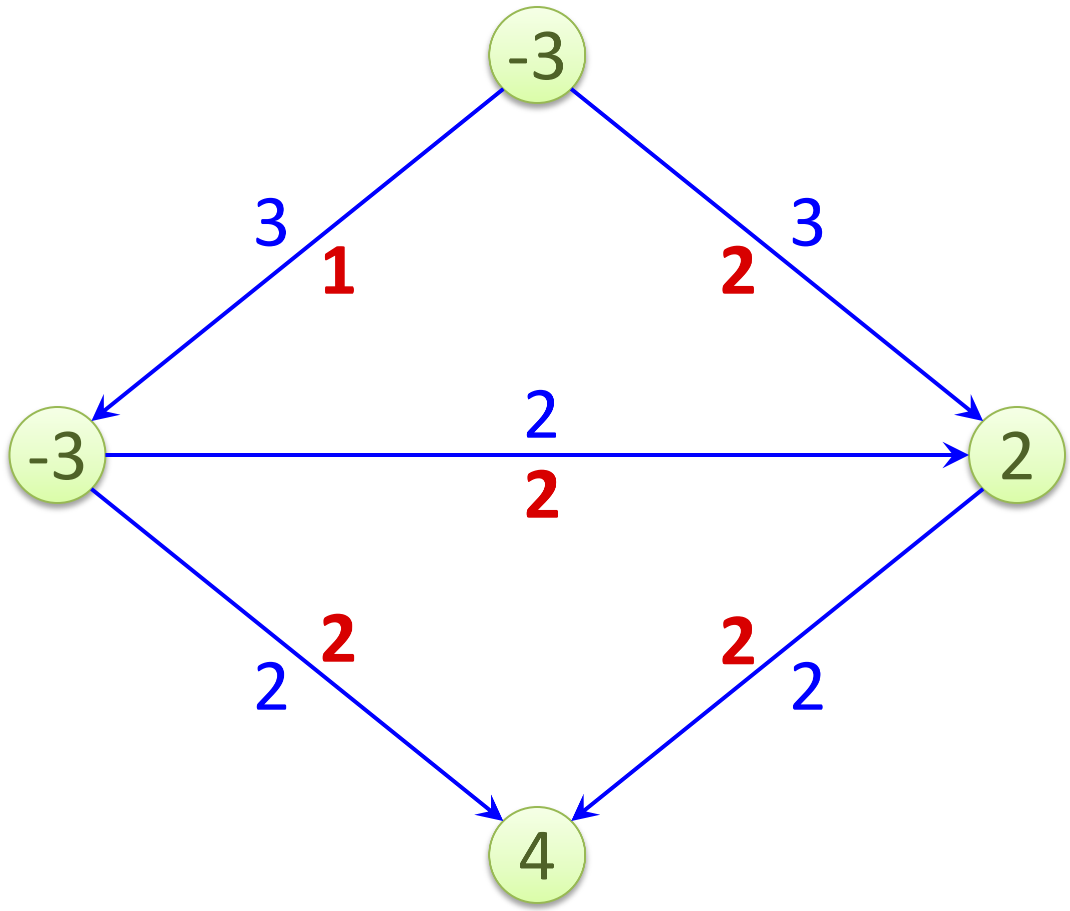
- Edge capacities $c_e \geq 0$ for all $e \in E$
- Node demands $d_v \in \mathbb{R}$ for all $v \in V$
  - $d_v > 0$: node needs flow and therefore is a sink
  - $d_v < 0$: node has a supply of $-d_v$ and is therefore a source
  - $d_v = 0$: node is neither a source nor a sink

Flow: Function $f: E \to \mathbb{R}_{\geq 0}$ satisfying

- *Capacity Conditions*: $\forall e \in E$: $\quad 0 \leq f(e) \leq c_e$
- *Demand Conditions*: $\forall v \in V$: $\quad f^{\text{in}}(v) - f^{\text{out}}(v) = d_v$

Objective: Does a flow f satisfying all conditions exist?
If yes, find such a flow f.

Example



Condition on Demands

Claim: If there exists a feasible circulation with demands d_v for $v \in V$, then

$$\sum_{v \in V} d_v = 0.$$

Proof:

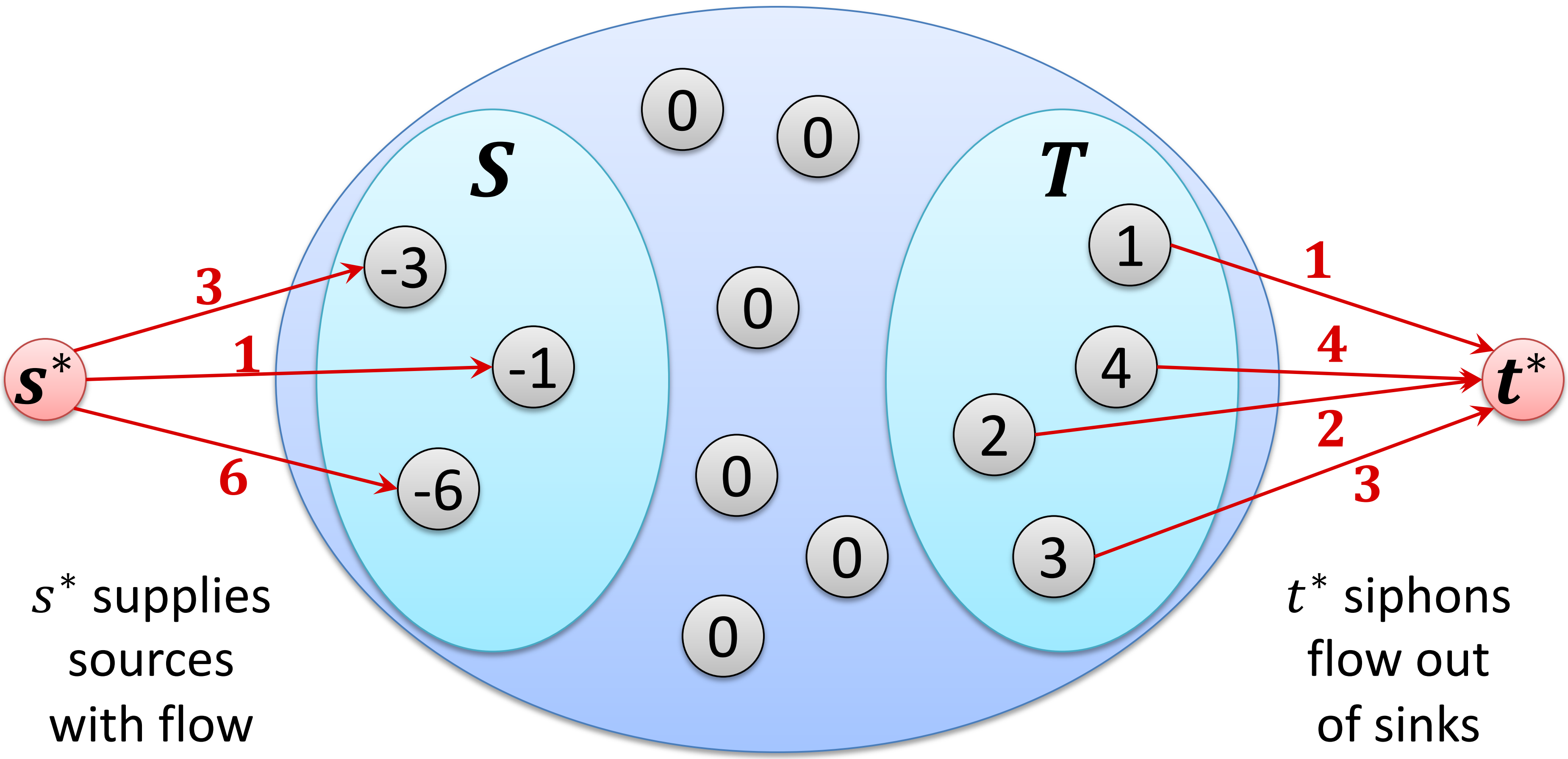
- $\sum_v d_v = \sum_v \left(f^{\text{in}}(v) - f^{\text{out}}(v)\right)$
- $f(e)$ of each edge e appears twice in the above sum with different signs → overall sum is 0

Total supply = total demand:

$$\text{Define } \boldsymbol{D} \coloneqq \sum_{v:d_v<0} -\boldsymbol{d_v} = \sum_{v:d_v>0} \boldsymbol{d_v}$$

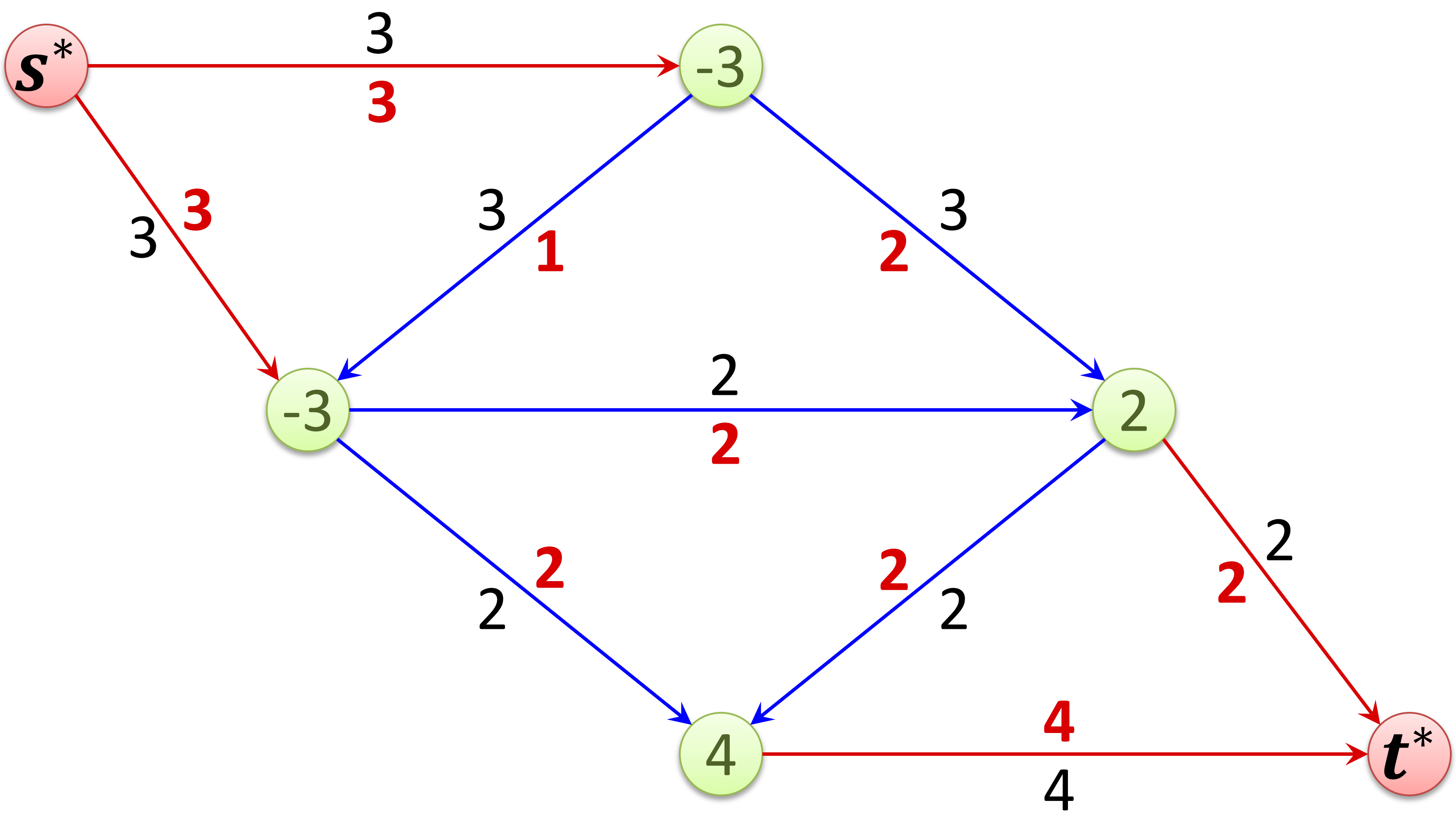
Reduction to Maximum Flow

- Add "super-source" s^* and "super-sink" t^* to network



- valid circulations ⇔ valid s^*-t^* flow that saturates all red edges.

Example



Formally…

Reduction: Get graph G' from graph as follows

- Node set of G' is $V \cup \{s^*, t^*\}$
- Edge set is E and edges
  - (s^*, v) for all v with $d_v < 0$, capacity of edge is $-d_v$
  - (v, t^*) for all v with $d_v > 0$, capacity of edge is d_v

Observations:

- Capacity of min s^*-t^* cut is at most D (e.g., the cut $(s^*, V \cup \{t^*\})$)
- A feasible circulation on G can be turned into a feasible flow of value D of G' by saturating all (s^*, v) and (v, t^*) edges.
- Any flow of G' of value D induces a feasible circulation on G
  - (s^*, v) and (v, t^*) edges are saturated
  - By removing these edges, we get exactly the demand constraints

Circulation with Demands

Theorem: There is a feasible circulation with demands d_v, $v \in V$ on graph G if and only if there is a flow of value D on G'.

- If all capacities and demands are integers, there is a valid integer circulation (if there is a valid circulation)

The max flow min cut theorem also implies the following:

Theorem: The graph G has a feasible circulation with demands d_v, $v \in V$ if and only if the sum of all demands is zero and for all cuts (A, B),

$$\sum_{v \in B} d_v \leq c(A, B).$$

Circulation: Demands and Lower Bounds

Given: Directed network $G = (V, E)$ with

- Edge capacities $c_e > 0$ and **lower bounds** $\mathbf{0 \leq \ell_e \leq c_e}$ **for** $\boldsymbol{e \in E}$
- Node demands $d_v \in \mathbb{R}$ for all $v \in V$
  - $d_v > 0$: node needs flow and therefore is a sink
  - $d_v < 0$: node has a supply of $-d_v$ and is therefore a source
  - $d_v = 0$: node is neither a source nor a sink

Flow: Function $f: E \to \mathbb{R}_{\geq 0}$ satisfying

- *Capacity Conditions*: $\forall e \in E$: $\boldsymbol{\ell_e \leq f(e) \leq c_e}$
- *Demand Conditions*: $\forall v \in V$: $f^{\text{in}}(v) - f^{\text{out}}(v) = d_v$

Objective: Does a flow f satisfying all conditions exist?
If yes, find such a flow f.

Solution Idea

- Define initial circulation $f_0(e) = \ell_e$
  Satisfies capacity constraints: $\forall e \in E: \ell_e \leq f_0(e) \leq c_e$

- Define

$$L_v := f_0^{\text{in}}(v) - f_0^{\text{out}}(v) = \sum_{e \text{ into } v} \ell_e - \sum_{e \text{ out of } v} \ell_e$$

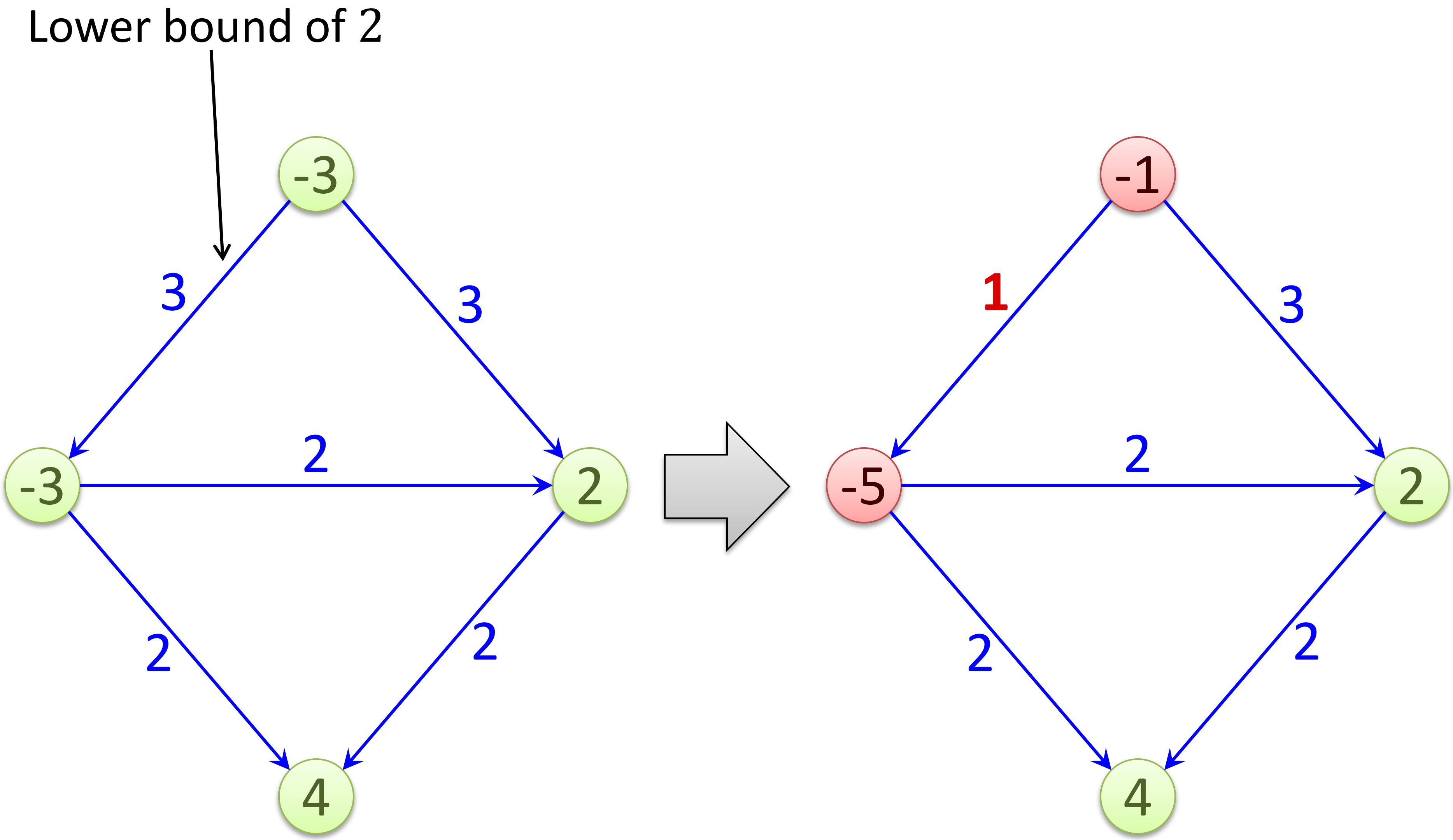
- If $L_v = d_v$, demand condition is satisfied at v by f_0, otherwise, we need to superimpose another circulation f_1 such that

$$d'_v := f_1^{\text{in}}(v) - f_1^{\text{out}}(v) = d_v - L_v$$

- Remaining capacity of edge e: $c'_e := c_e - \ell_e$

- We get a circulation problem with new demands d'_v, new capacities c'_e, and no lower bounds

Eliminating a Lower Bound: Example

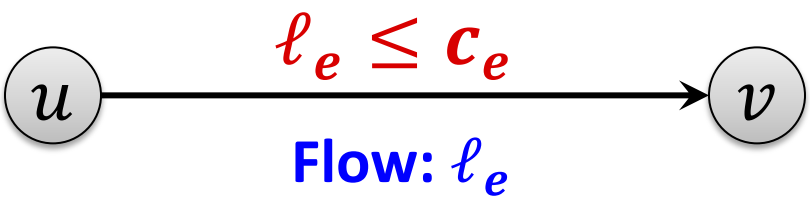


Reduce to Problem Without Lower Bounds

Graph $G = (V, E)$:

- Capacity: For each edge $e \in E$: $\ell_e \leq f(e) \leq c_e$
- Demand: For each node $v \in V$: $f^{\text{in}}(v) - f^{\text{out}}(v) = d_v$

Model lower bounds with supplies & demands:



Create Network G' (without lower bounds):

- For each edge $e \in E$: $c'_e = c_e - \ell_e$
- For each node $v \in V$: $d'_v = d_v - L_v$

Circulation: Demands and Lower Bounds

Theorem: There is a feasible circulation in G (with lower bounds) if and only if there is feasible circulation in G' (without lower bounds).

- Given circulation f' in G', $f(e) = f'(e) + \ell_e$ is circulation in G
  - The capacity constraints are satisfied because $f'(e) \leq c_e - \ell_e$
  - Demand conditions:

$$f^{\text{in}}(v) - f^{\text{out}}(v) = \sum_{e \text{ into } v} \big(\ell_e + f'(e)\big) - \sum_{e \text{ out of } v} \big(\ell_e + f'(e)\big)$$
$$= L_v + (d_v - L_v) = d_v$$

- Given circulation f in G, $f'(e) = f(e) - \ell_e$ is circulation in G'
  - The capacity constraints are satisfied because $\ell_e \leq f(e) \leq c_e$
  - Demand conditions:

$$f'^{\text{in}}(v) - f'^{\text{out}}(v) = \sum_{e \text{ into } v} \big(f(e) - \ell_e\big) - \sum_{e \text{ out of } v} \big(f(e) - \ell_e\big)$$
$$= d_v - L_v$$

Integrality

Theorem: Consider a circulation problem with integral capacities, flow lower bounds, and node demands. If the problem is feasible, then it also has an integral solution.

Proof:

- Graph G' has only integral capacities and demands
- Thus, the flow network used in the reduction to solve circulation with demands and no lower bounds has only integral capacities
- The theorem now follows because a max flow problem with integral capacities also has an optimal integral solution

- It also follows that with the max flow algorithms we studied, we get an integral feasible circulation solution.

Matrix Rounding

- **Given:** $p \times q$ matrix $D = \{d_{i,j}\}$ of real numbers
- **row *i* sum:** $a_i = \sum_j d_{i,j}$, **column *j* sum:** $b_j = \sum_i d_{i,j}$
- **Goal:** Round each $d_{i,j}$, as well as a_i and b_j up or down to the next integer so that the sum of rounded elements in each row (column) equals the rounded row (column) sum
- **Original application:** publishing census data

Example:

3.14	6.80	7.30	17.24
9.60	2.40	0.70	12.70
3.60	1.20	6.50	11.30
16.34	10.40	14.50	

original data

3	7	7	17
10	2	1	13
3	1	7	11
16	10	15	

possible rounding

Matrix Rounding

Theorem: For any matrix, there exists a feasible rounding.

Remark: Just rounding to the nearest integer doesn't work

0.35	0.35	0.35	1.05
0.55	0.55	0.55	1.65
0.90	0.90	0.90	

original data

0	0	0	0
1	1	1	3
1	1	1	

rounding to nearest integer

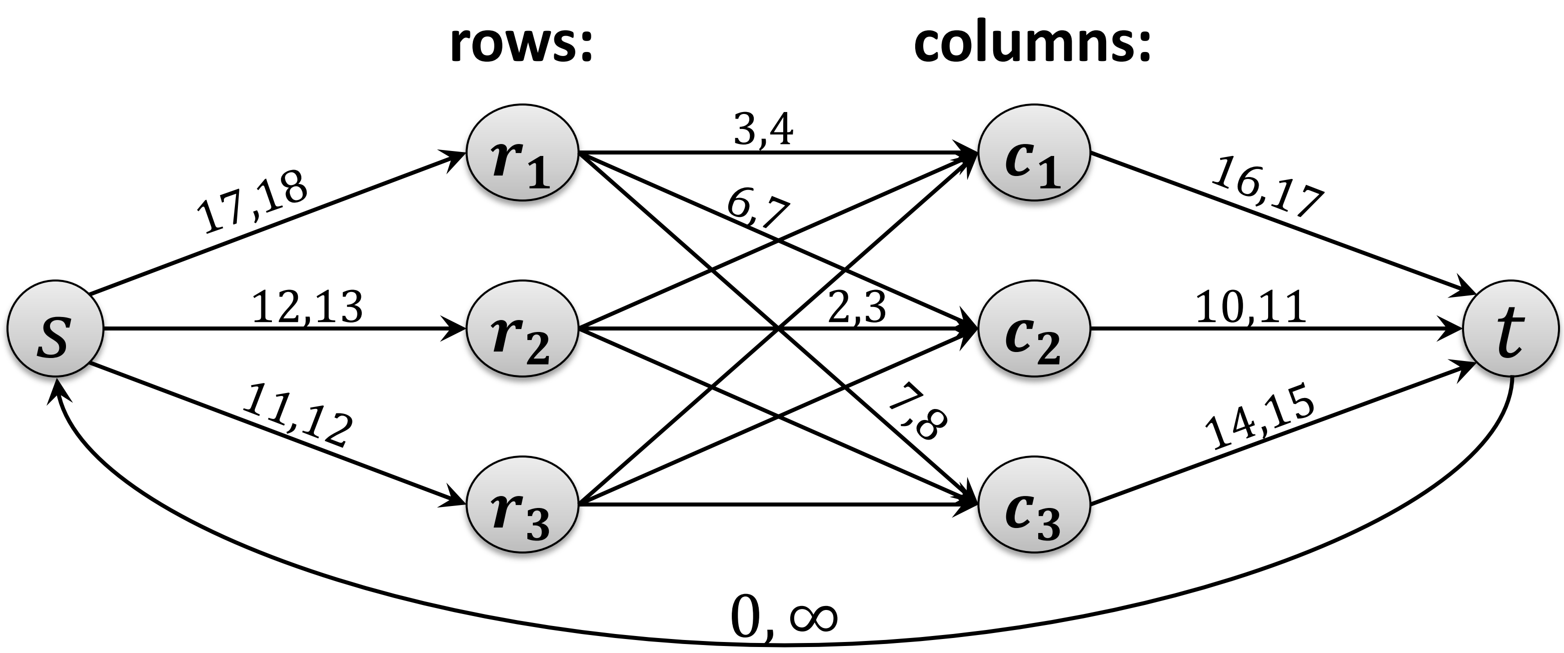
0	0	1	1
1	1	0	2
1	1	1	

feasible rounding

Reduction to Circulation

3.14	6.80	7.30	17.24
9.60	2.40	0.70	12.70
3.60	1.20	6.50	11.30
16.34	10.40	14.50	

Matrix elements and row/column sums give a feasible circulation that satisfies all lower bound, capacity, and demand constraints



all demands $d_v = 0$

Matrix Rounding

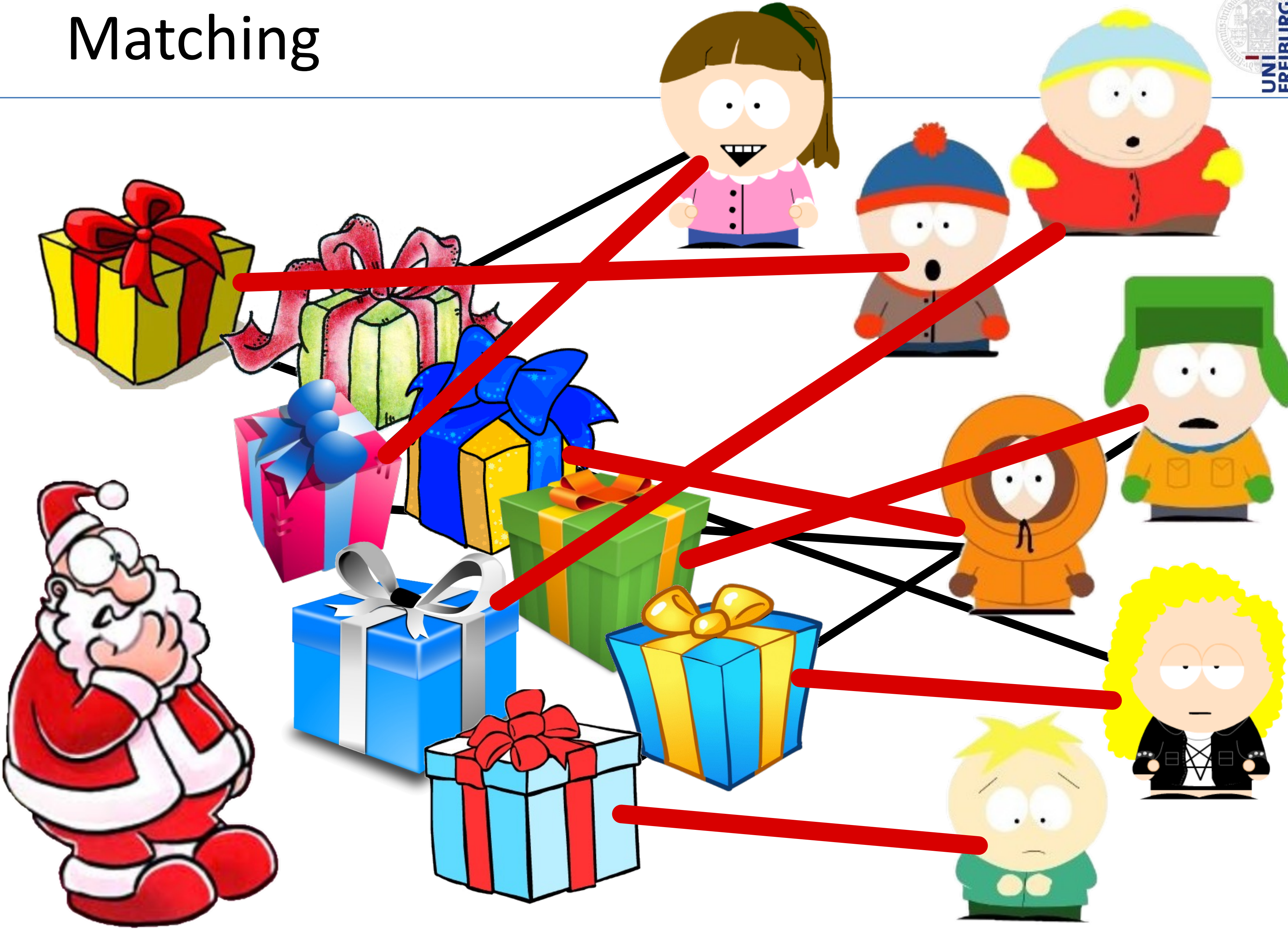
Theorem: For any matrix, there exists a feasible rounding.

Proof:

- The matrix entries $d_{i,j}$ and the row and column sums a_i and b_j give a feasible circulation for the constructed network
- Every feasible circulation gives matrix entries with corresponding row and column sums (follows from demand constraints)
- Because all demands, capacities, and flow lower bounds are integral, there is an integral solution to the circulation problem

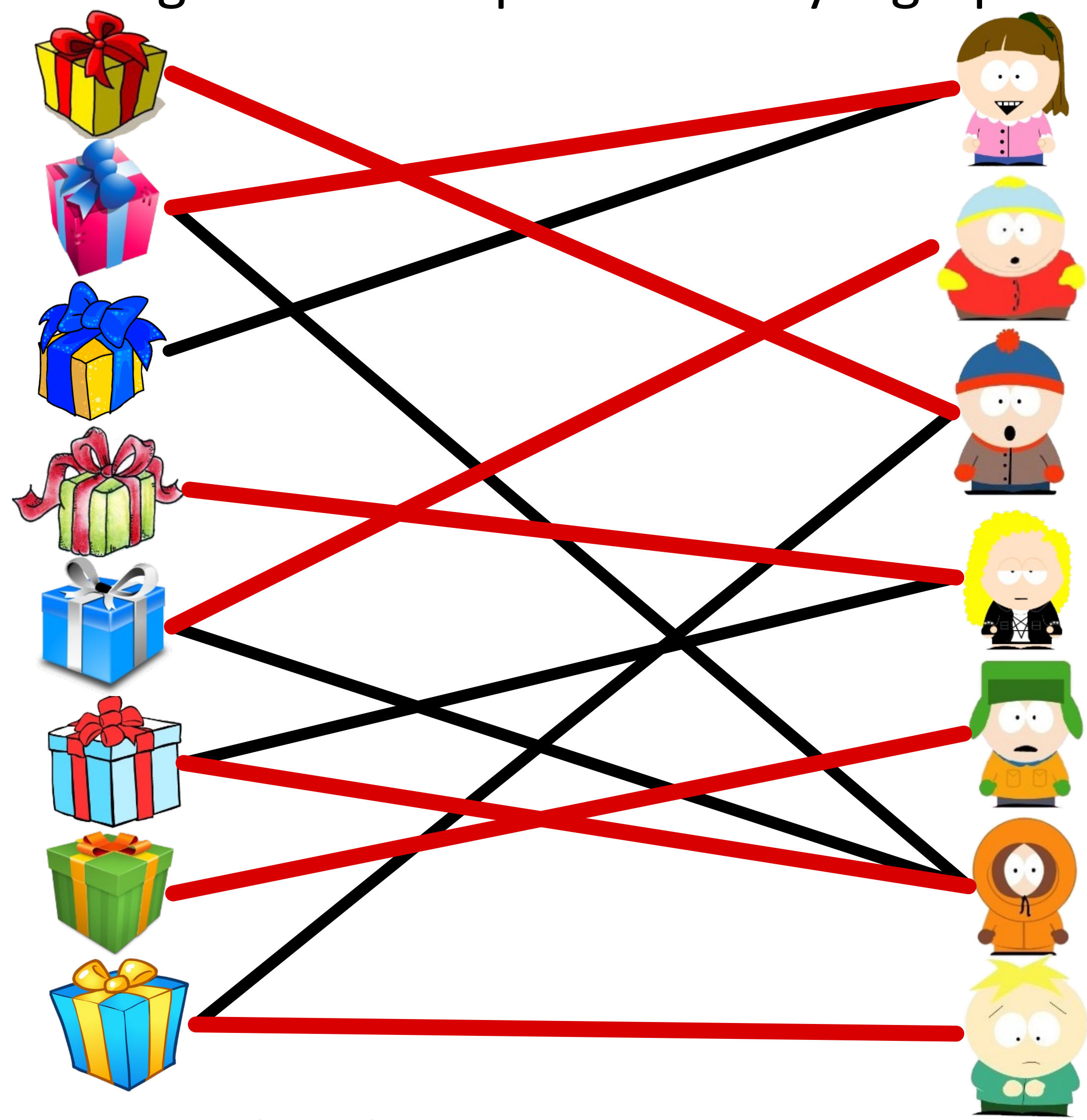
→ **gives a feasible rounding!**

Matching



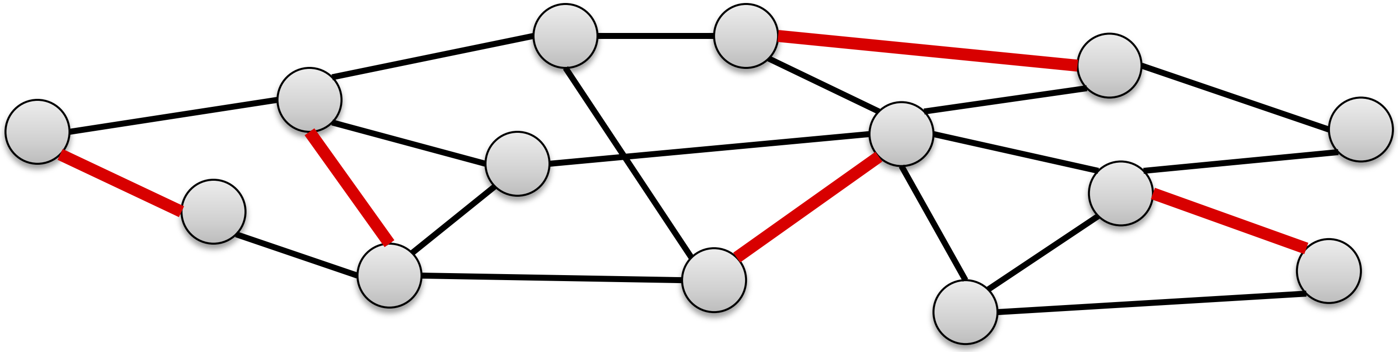
Gifts-Children Graph

- Which child likes which gift can be represented by a graph



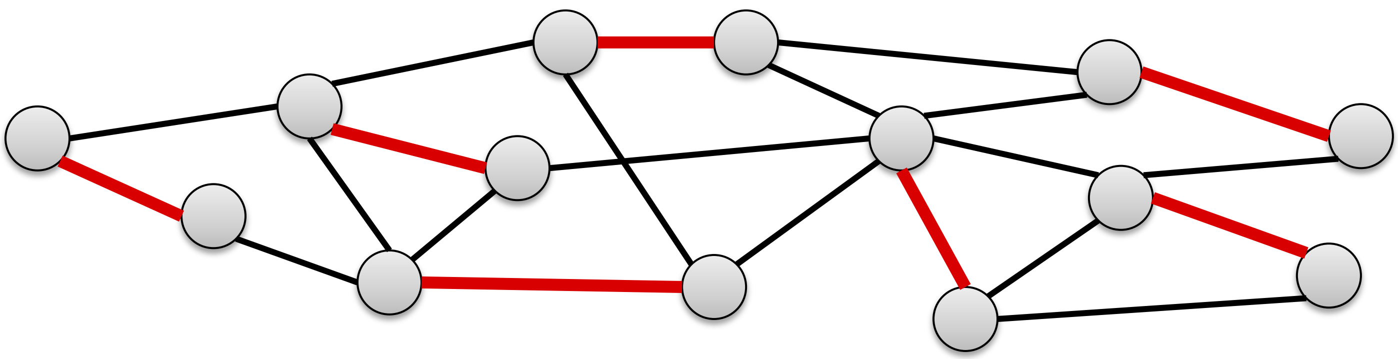
Matching

Matching: Set of pairwise non-incident edges



Maximal Matching: A matching s.t. no more edges can be added

Maximum Matching: A matching of maximum possible size



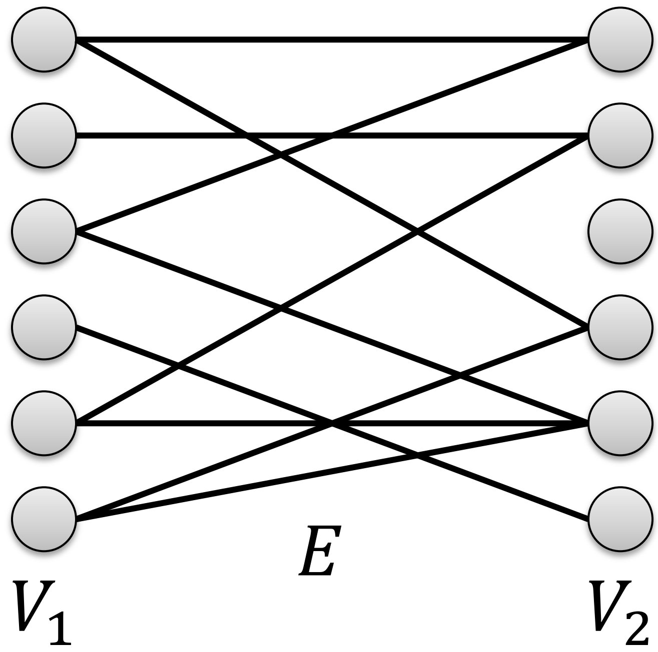
Perfect Matching: Matching of size $^n/_2$ (every node is matched)

Bipartite Graph

Definition: A graph $G = (V, E)$ is called bipartite iff its node set can be partitioned into two parts $V = V_1 \dot\cup V_2$ such that for each edge $\{\mathrm{u}, \mathrm{v}\} \in E$,

$$|\{u, v\} \cap V_1| = 1.$$

- Thus, edges are only between the two parts



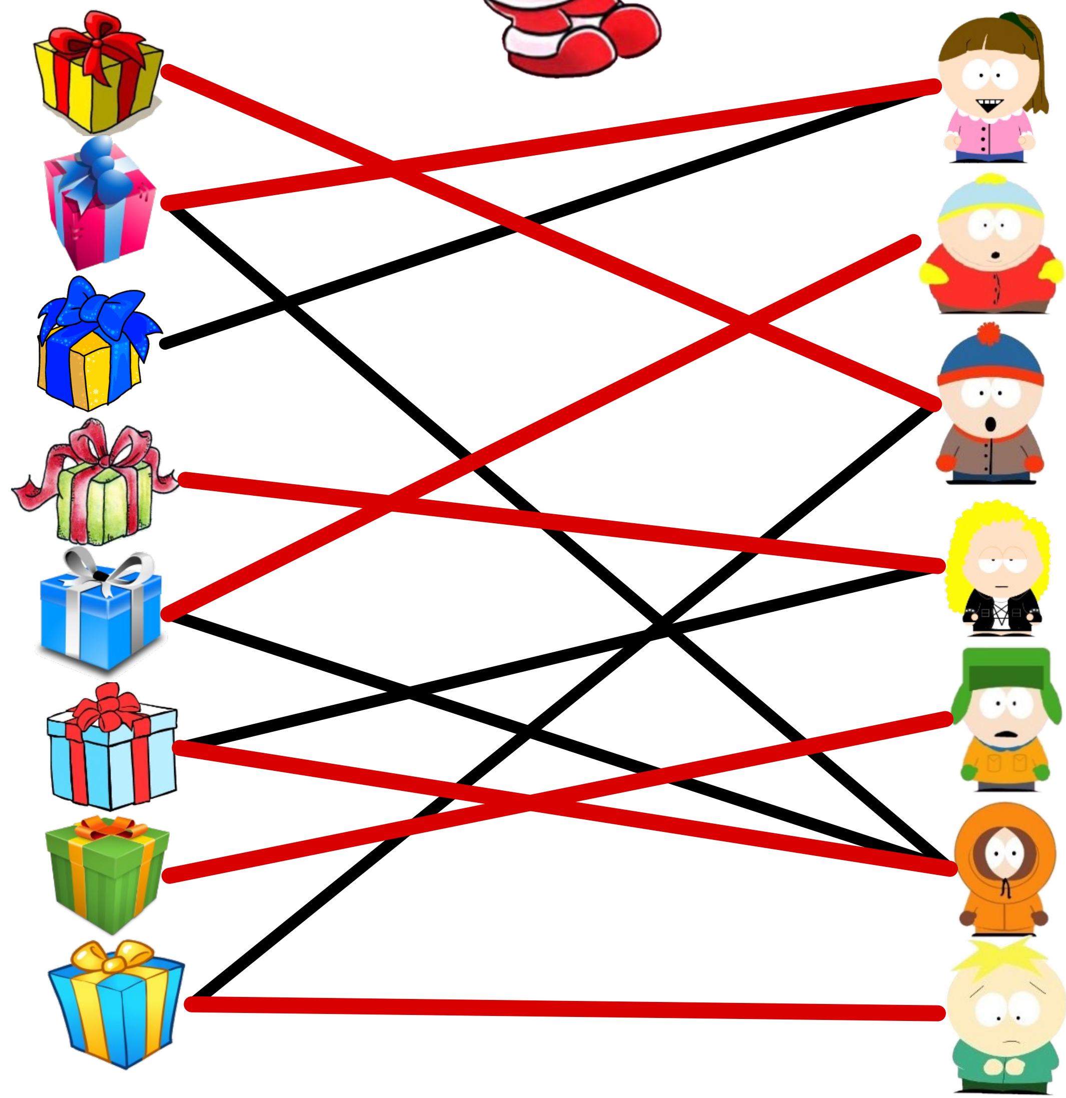
Santa's Problem

Maximum Matching in Bipartite Graphs:

Every child can get a gift iff there is a matching of size $\#\text{children}$

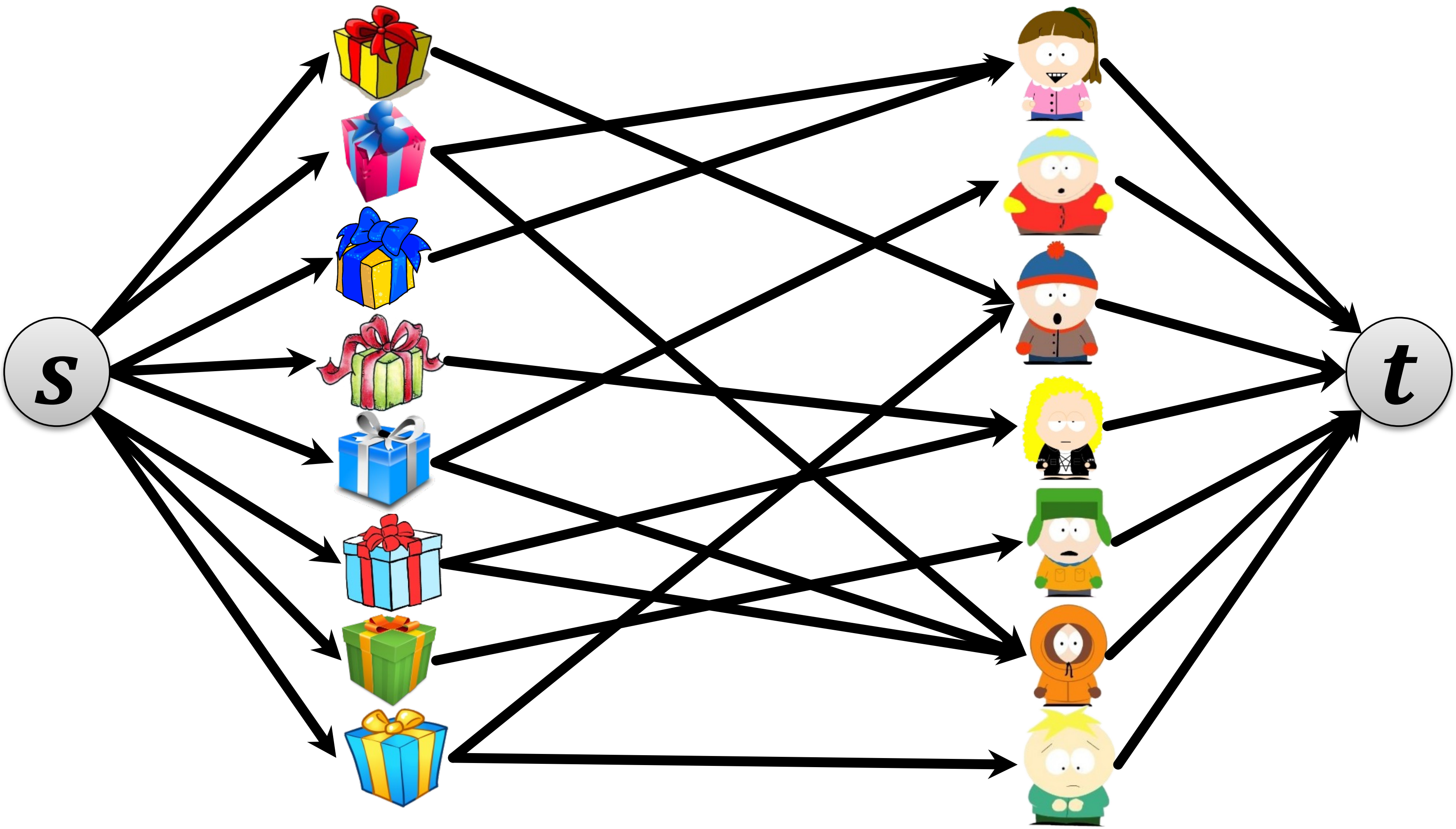
Clearly, every matching is at most as big

If $\#\text{children} = \#\text{gifts}$, there is a solution iff there is a perfect matching



Reducing to Maximum Flow

- Like edge-disjoint paths…



all capacities are 1

Reducing to Maximum Flow

Theorem: Every integer solution to the max flow problem on the constructed graph induces a maximum bipartite matching of G.

Proof:

1. An integer flow f of value $|f|$ induces a matching of size $|f|$
   – Left nodes (gifts) have incoming capacity 1
   – Right nodes (children) have outgoing capacity 1
   – Left and right nodes are incident to ≤ 1 edge e of G with $f(e) = 1$

2. A matching of size k implies a flow f of value $|f| = k$
   – For each edge $\{u, v\}$ of the matching:

$$f\big((s,u)\big) = f\big((u,v)\big) = f\big((v,t)\big) = 1$$

   – All other flow values are 0

Running Time of Max. Bipartite Matching

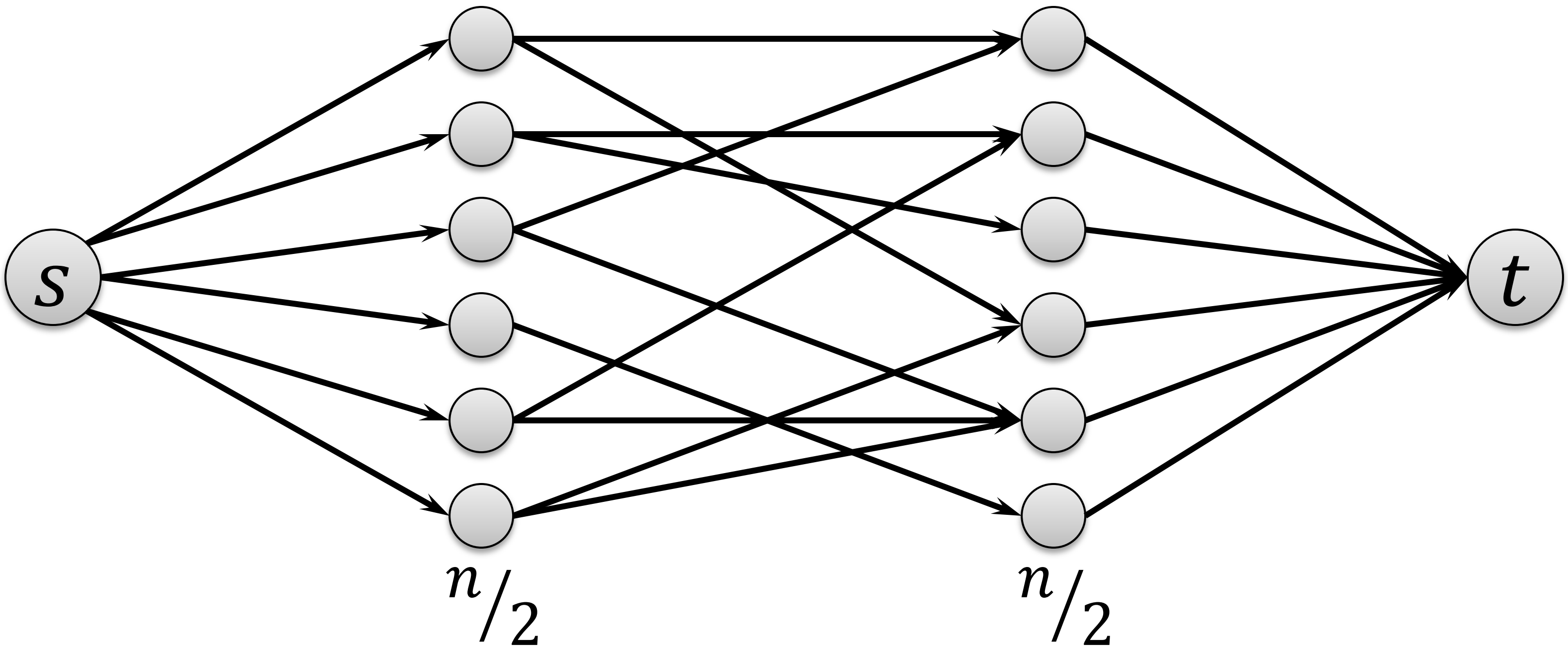
Theorem: A maximum matching M^* of a bipartite graph can be computed in time $O(m \cdot |M^*|) = O(m \cdot n)$.

- The problem can be reduced to a maximum flow problem on a flow network with $O(m)$ edges and all capacities $= 1$
- The Ford-Fulkerson algorithm solves the maximum flow problem in time $O(m \cdot C)$, where C is the value of the maximum flow (i.e., $C = |M^*|$).
- A maximum matching M^* has size $|M^*| \leq {}^{n}/_{2} = O(n)$.

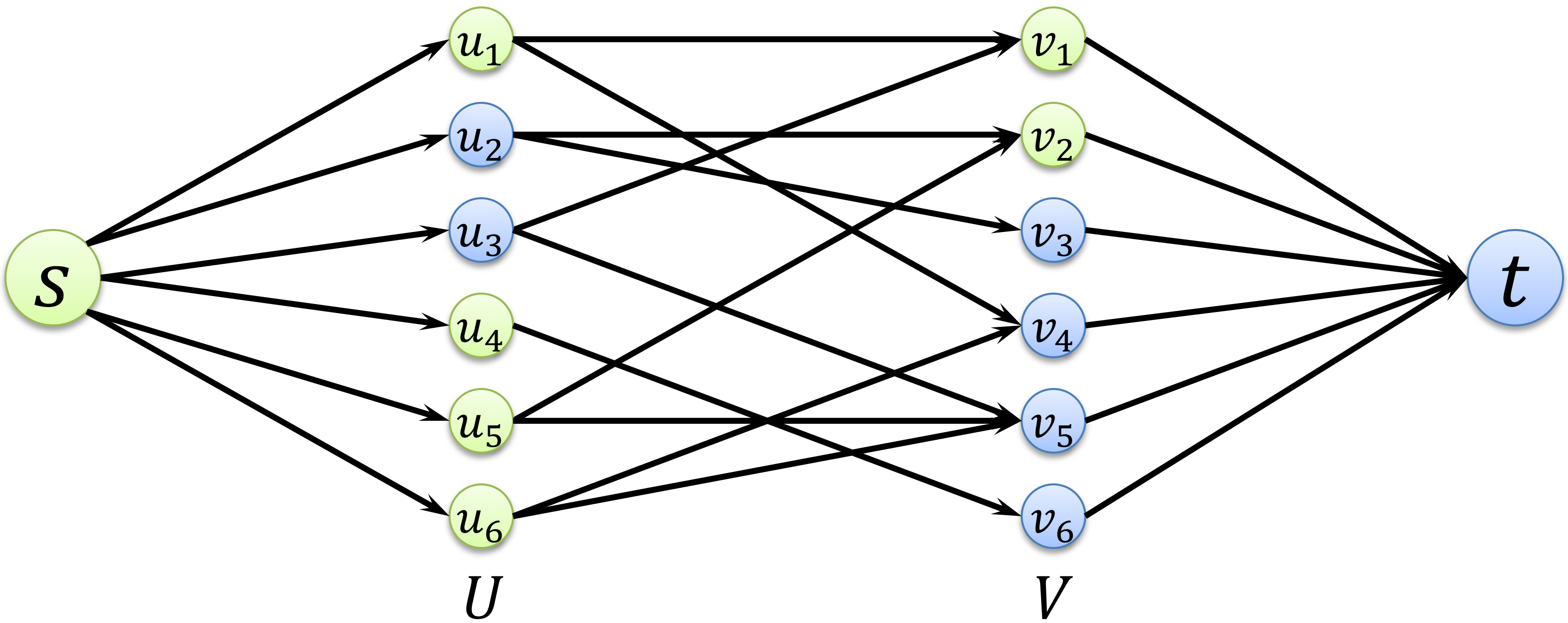
Perfect Matching?

- There can only be a perfect matching if both sides of the partition have size $n/2$.

- There is no perfect matching, iff there is an s-t cut of size $< n/2$ in the flow network.



s-t Cuts



Partition (A, B) of node set such that $s \in A$ and $t \in B$

- If $v_i \in A$: edge (v_i, t) is in cut (A, B)
- If $u_i \in B$: edge (s, u_i) is in cut (A, B)
- Otherwise (if $u_i \in A$, $v_i \in B$), all edges from u_i to some $v_j \in B$ are in cut (A, B)

Hall's Theorem

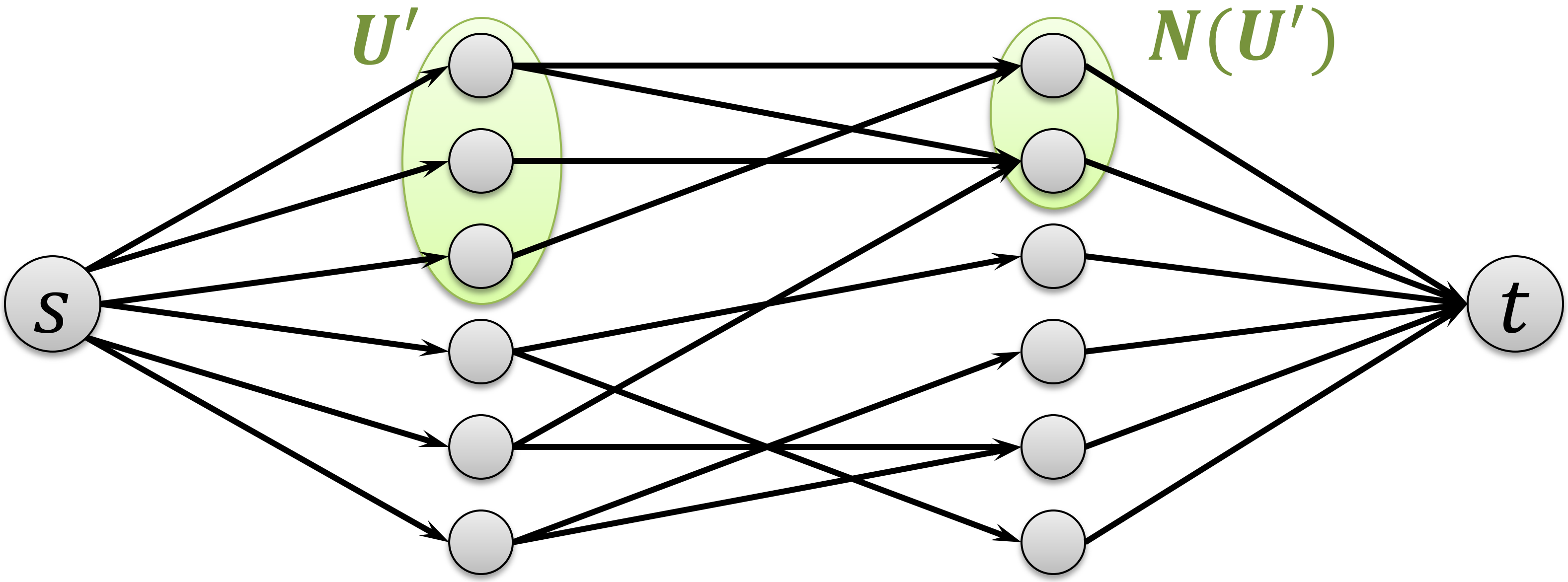
Theorem: A bipartite graph $G = (U \cup V, E)$ for which $|U| = |V|$ has a perfect matching if and only if

$$\forall \boldsymbol{U'} \subseteq \boldsymbol{U}: |\boldsymbol{N}(\boldsymbol{U'})| \geq |\boldsymbol{U'}|,$$

where $N(U') \subseteq V$ is the set of neighbors of nodes in U'.

Proof: No perfect matching $\Leftrightarrow$ some s-t cut has capacity $< n/2$

1. Assume there is U' for which $|N(U')| < |U'|$:



Hall's Theorem

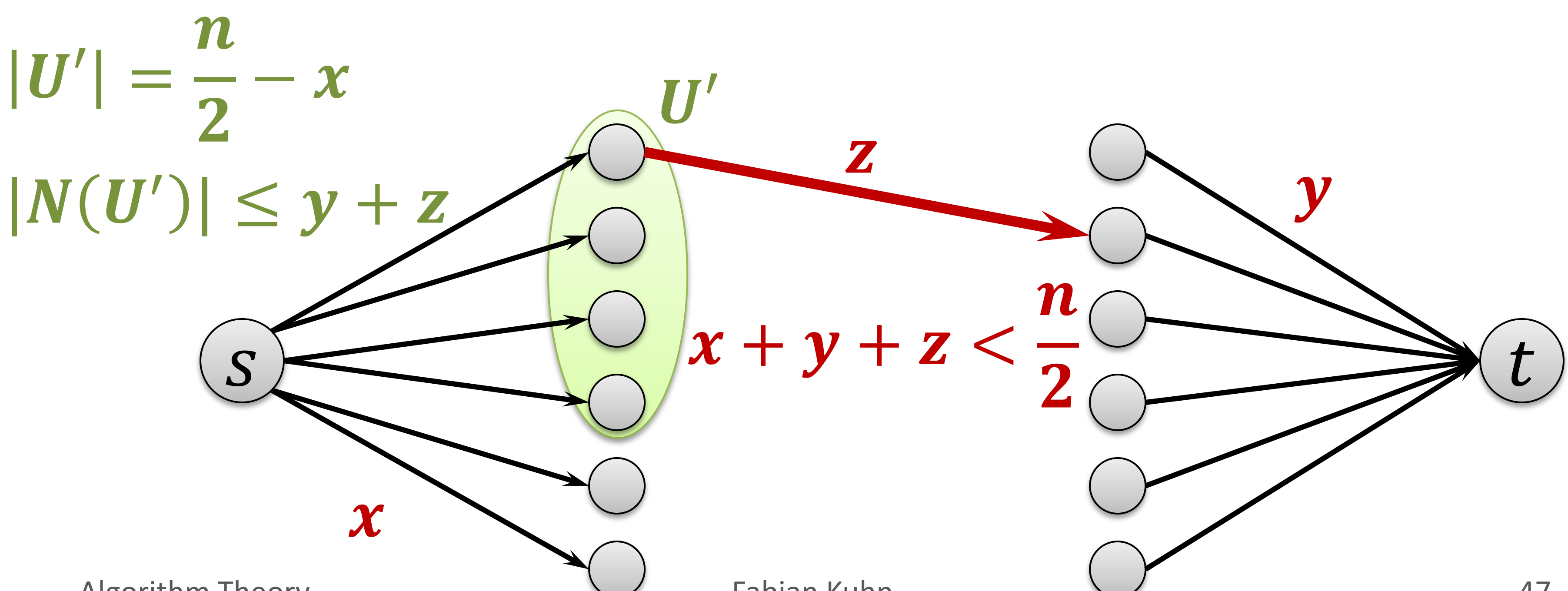
Theorem: A bipartite graph $G = (U \cup V, E)$ for which $|U| = |V|$ has a perfect matching if and only if

$$\forall U' \subseteq U : |N(U')| \geq |U'|,$$

where $N(U') \subseteq V$ is the set of neighbors of nodes in U'.

Proof: No perfect matching $\Leftrightarrow$ some s-t cut has capacity $< n/2$

2. Assume that there is a cut (A, B) of capacity $< n/2$



Hall's Theorem

Theorem: A bipartite graph $G = (U \cup V, E)$ for which $|U| = |V|$ has a perfect matching if and only if

$$\forall U' \subseteq U: |N(U')| \geq |U'|,$$

where $N(U') \subseteq V$ is the set of neighbors of nodes in U'.

Proof: No perfect matching $\Leftrightarrow$ some s-t cut has capacity $< n$

2. Assume that there is a cut (A, B) of capacity $< n$

$$x + y + z < \frac{n}{2} \quad \Rightarrow \quad y + z < \frac{n}{2} - x$$

$$\Downarrow$$

$$|U'| = \frac{n}{2} - x \quad \Rightarrow \quad y + z < |U'|$$

$$\Downarrow$$

$$|N(U')| \leq y + z \quad \Rightarrow \quad |N(U')| < |U'|$$