Algorithm Theory

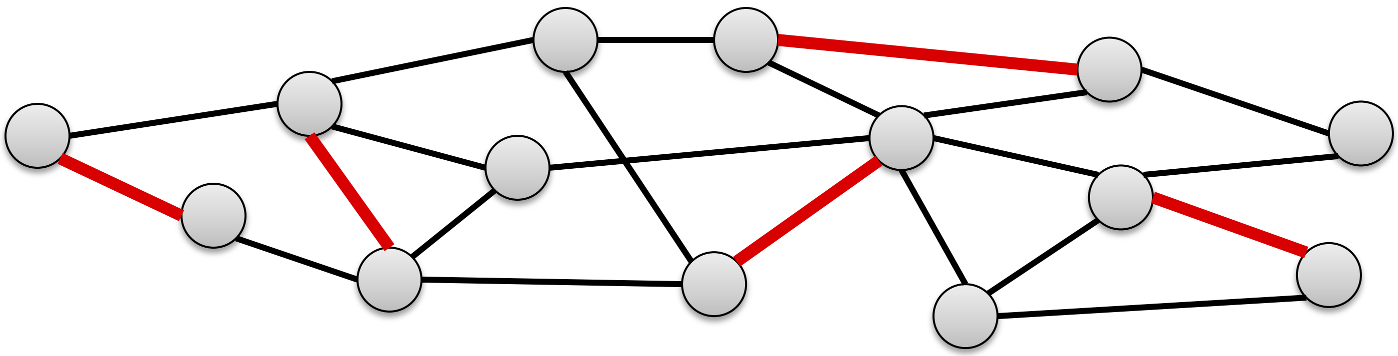
Chapter 6
Graph Algorithms

Maximum Matching

Fabian Kuhn

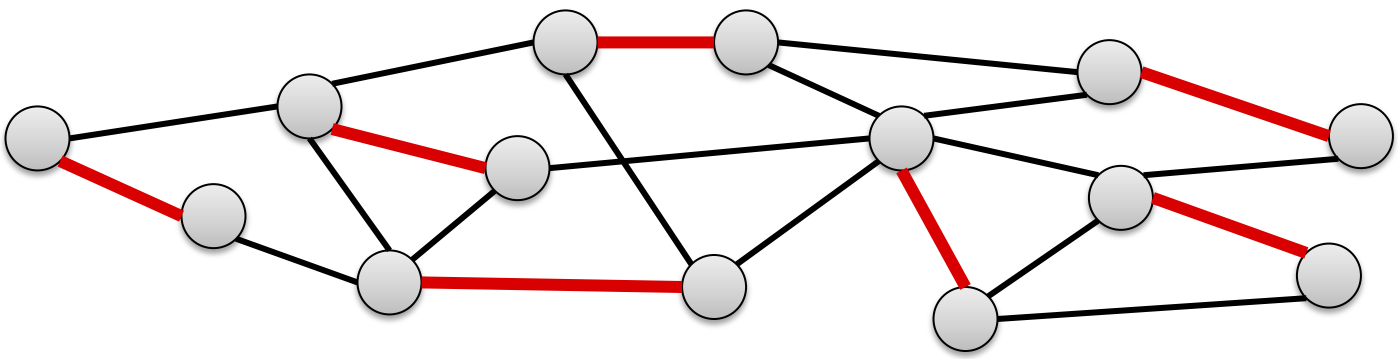
Matching

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



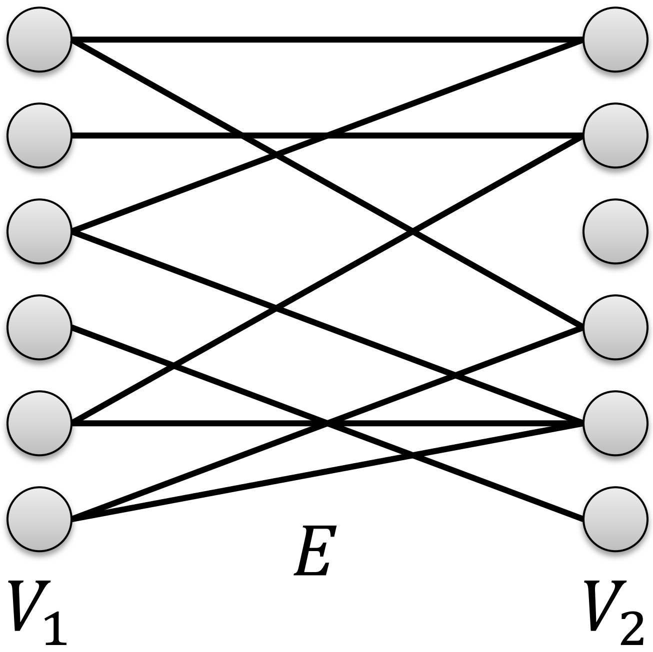
Perfect Matching: Matching of size $^n/_2$ (every node is matched)

Bipartite Graph

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

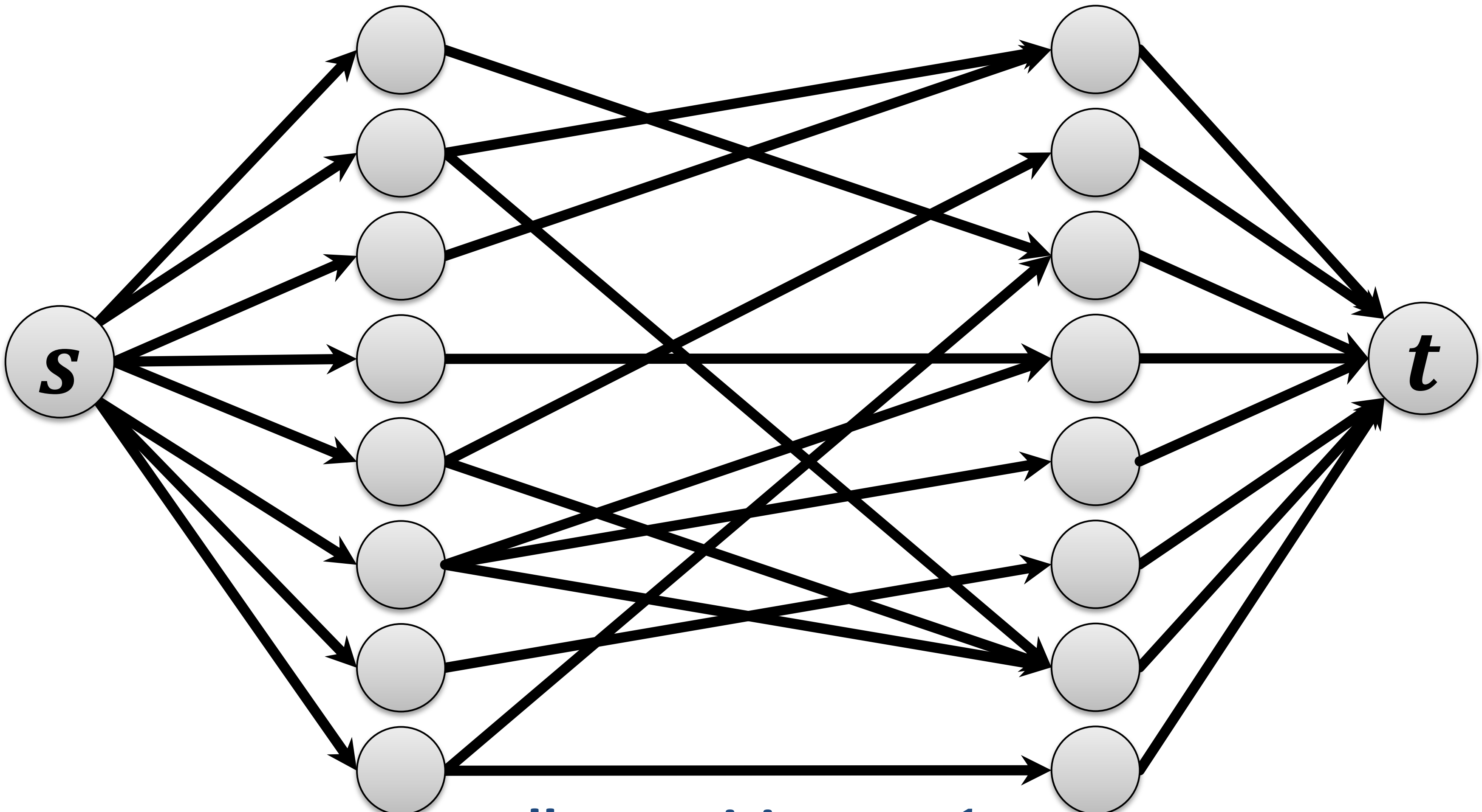
$$|\{u, v\} \cap V_1| = 1.$$

- Thus, edges are only between the two parts



Reducing to Maximum Flow

- Like edge-disjoint paths…



all capacities are 1

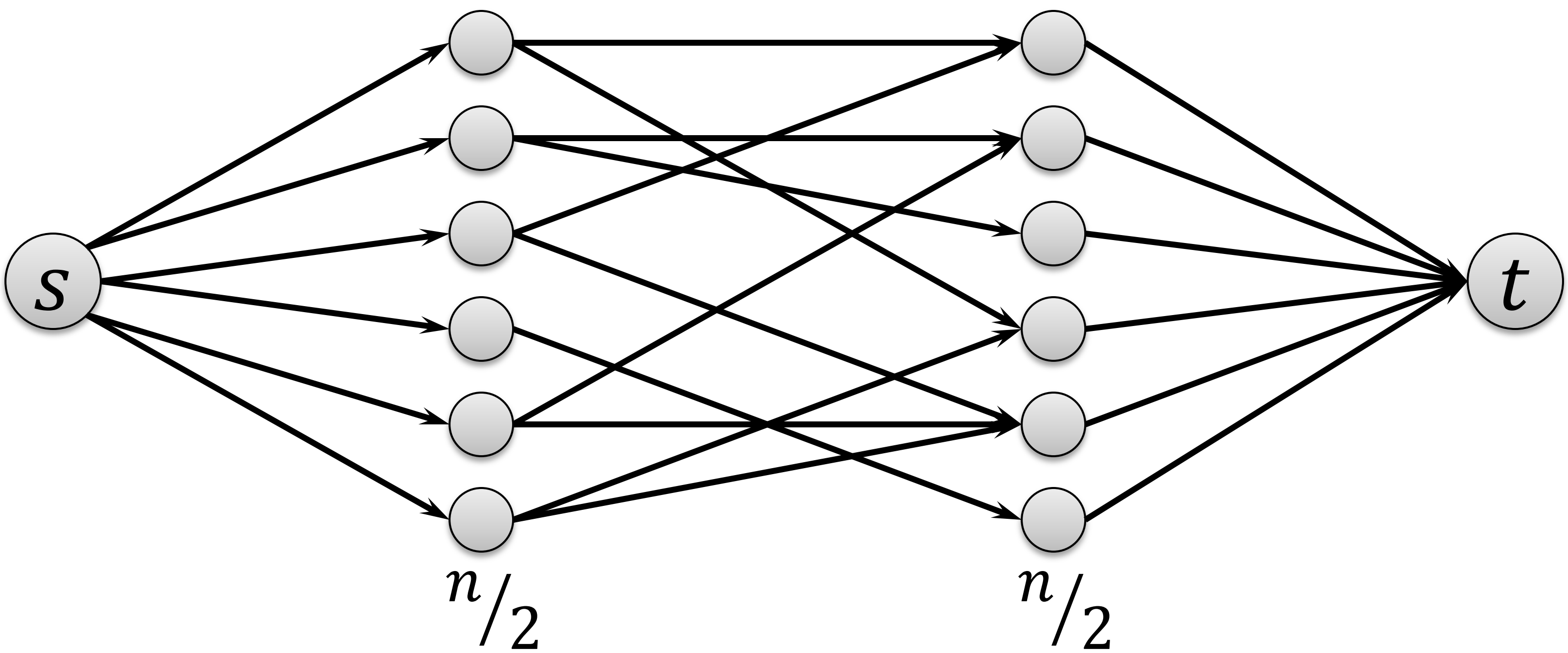
Running Time of Max. Bipartite Matching

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

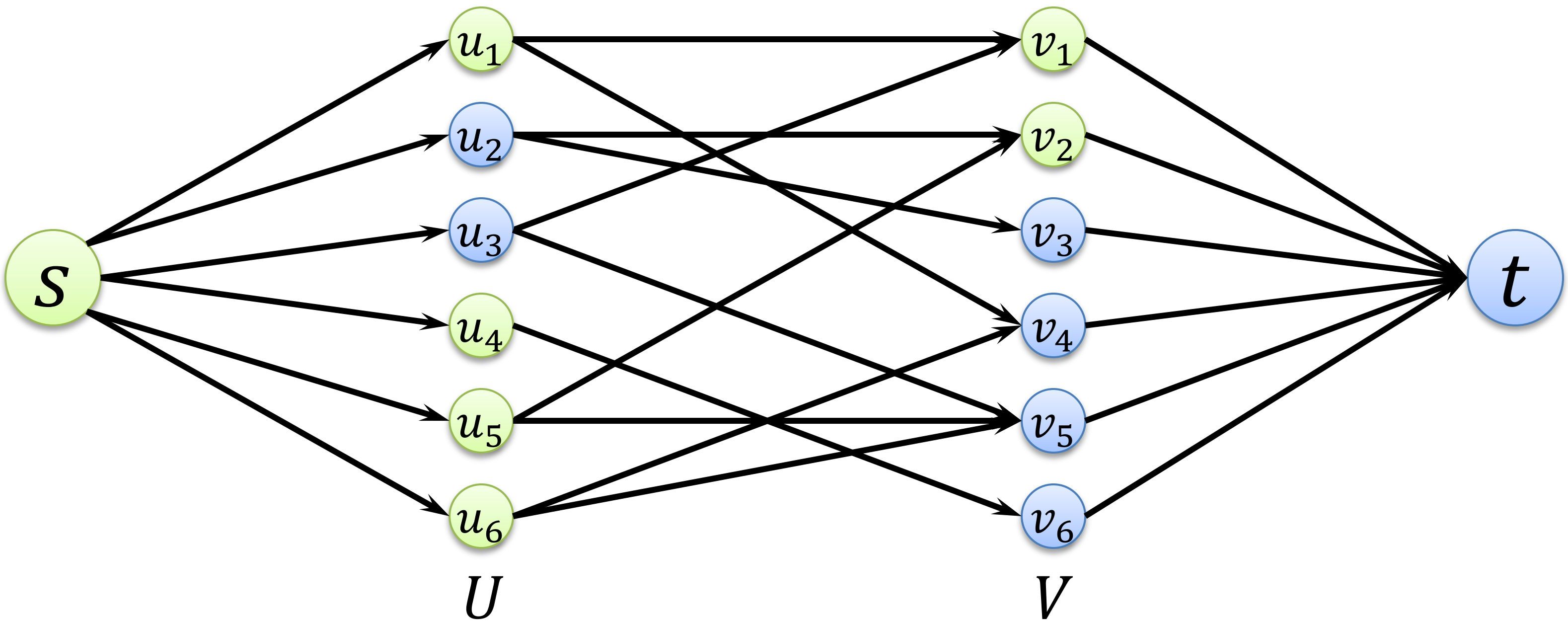
- The problem can be reduced to a maximum flow problem on a flow network with $O(m)$ edges and all capacities $= 1$
- The Ford-Fulkerson algorithm solves the maximum flow problem in time $O(m \cdot C)$, where C is the value of the maximum flow (i.e., $C = |M^*|$).
- A maximum matching M^* has size $|M^*|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

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

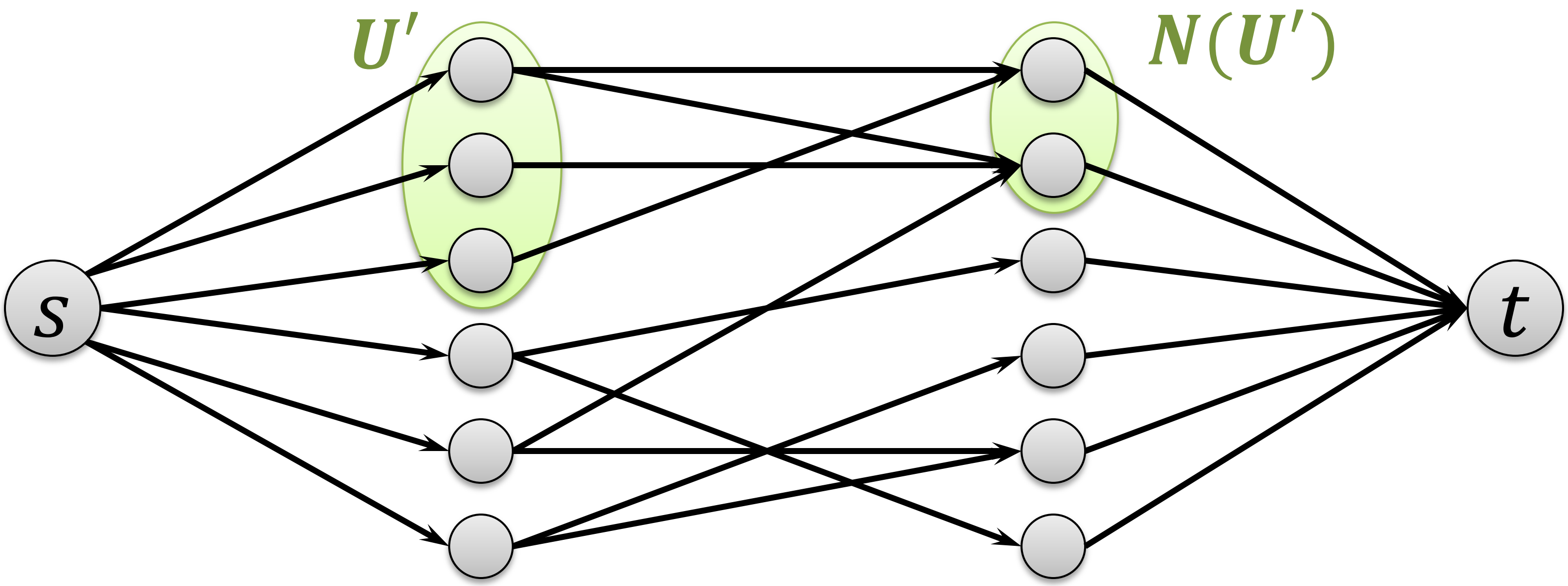
Theorem: A bipartite graph $G = (U \cup V, E)$ for which $|U| = |V|$ has a perfect matching if and only if

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

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

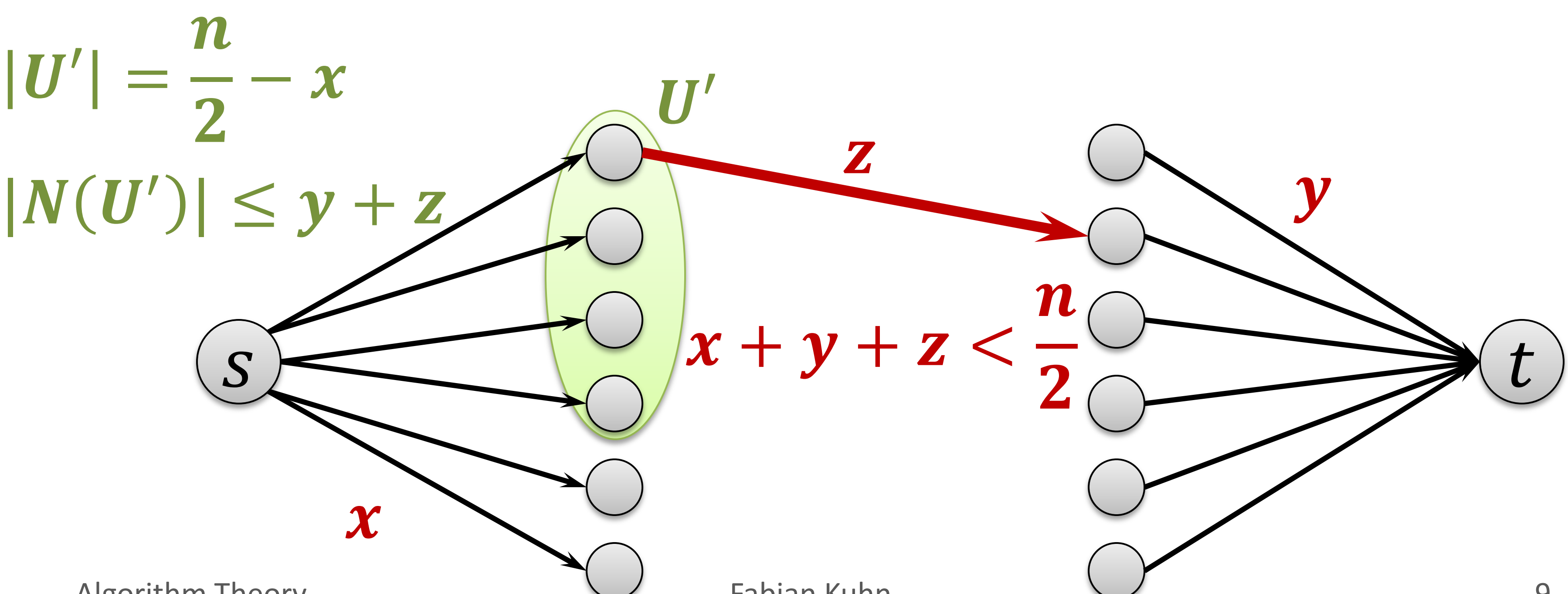
Theorem: A bipartite graph $G = (U \cup V, E)$ for which $|U| = |V|$ has a perfect matching if and only if

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

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 $< n$

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

$$\Downarrow$$

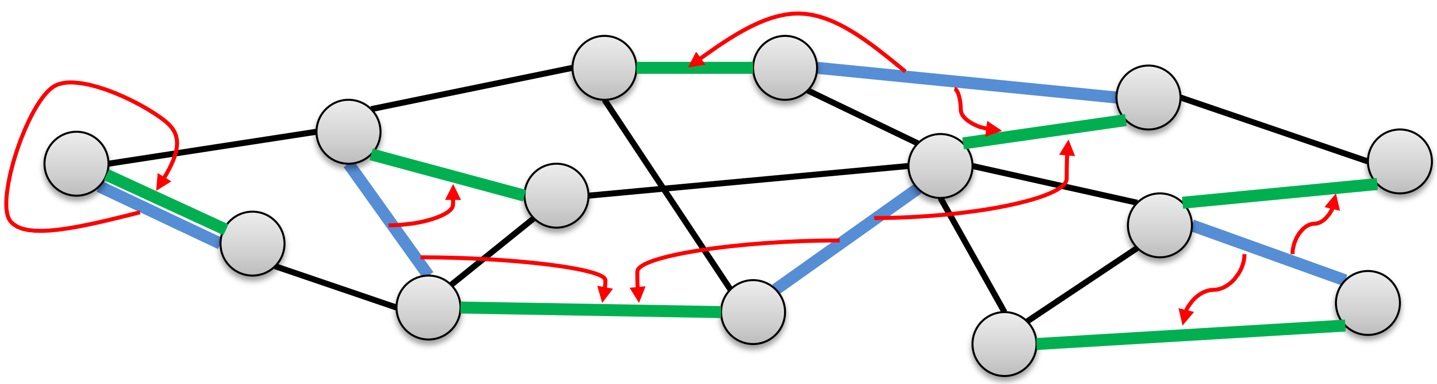
$$|U'| = \frac{n}{2} - x \Rightarrow y + z < |U'|$$

$$\Downarrow$$

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

What About General Graphs

- Can we efficiently compute a maximum matching if G is not bipartite?
- How good is a maximal matching?
  - A matching that cannot be extended…
- **Compare the size of a maximal and a maximum matching**



- Each maximal matching edge is adjacent to ≤ 2 maximum matching edges

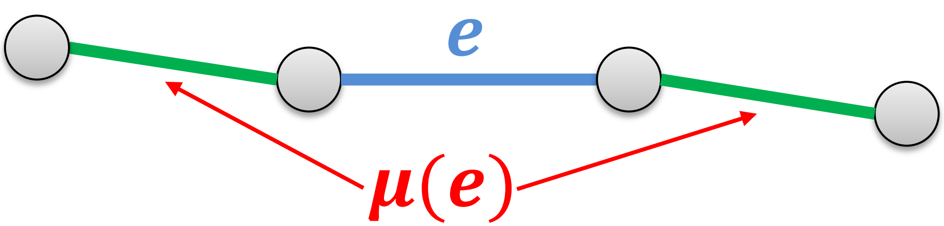
Maximal vs. Maximum Matching

Theorem: For any maximal matching M and any maximum matching M^*, it holds that

$$|M| \geq \frac{|M^*|}{2}.$$

Proof:

- For each edge $e \in M$, let $\mu(e) \subseteq M^*$ be the adjacent edges in M^*



$\forall e \in M : |\mu(e)| \leq 2$

- Every edge in M^* is adjacent to some edge of M:

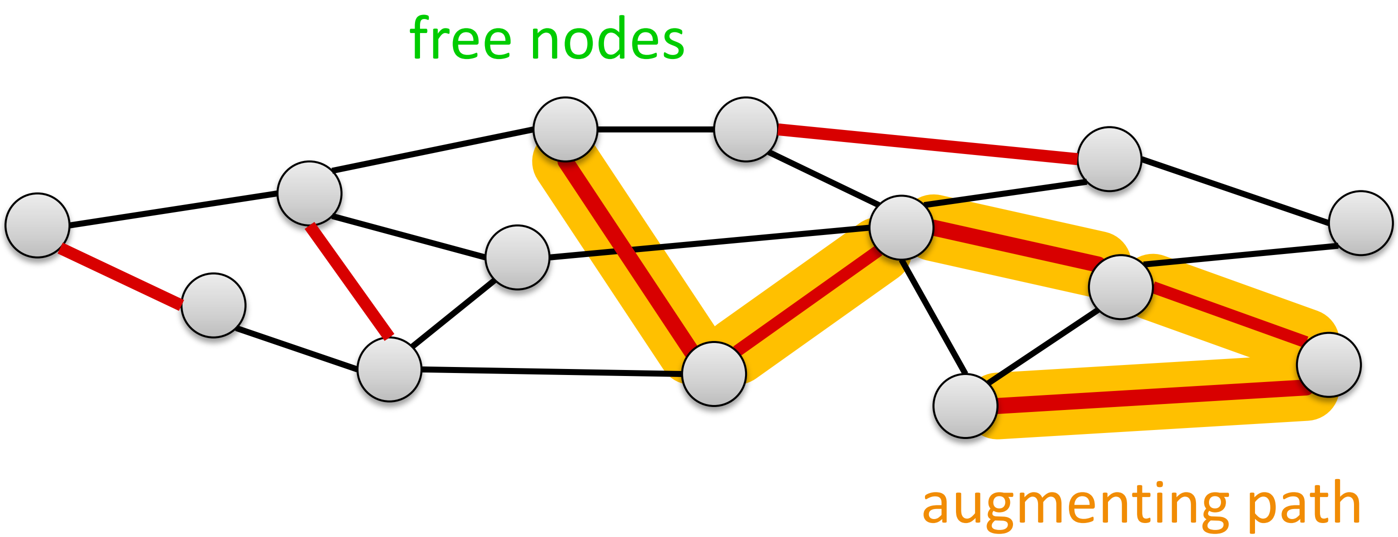
$$|M^*| = \left|\bigcup_{e \in M} \mu(e)\right| \leq \sum_{e \in M} |\mu(e)| \leq 2|M|.$$

Augmenting Paths

Consider a matching M of a graph $G = (V, E)$:

- A node $v \in V$ is called **free** iff it is not matched

Augmenting Path: A (odd-length) path that starts and ends at a free node and visits edges in $E \setminus M$ and edges in M alternatingly.



- Matching M can be improved using an augmenting path by switching the role of each edge along the path

Existence of Augmenting Paths

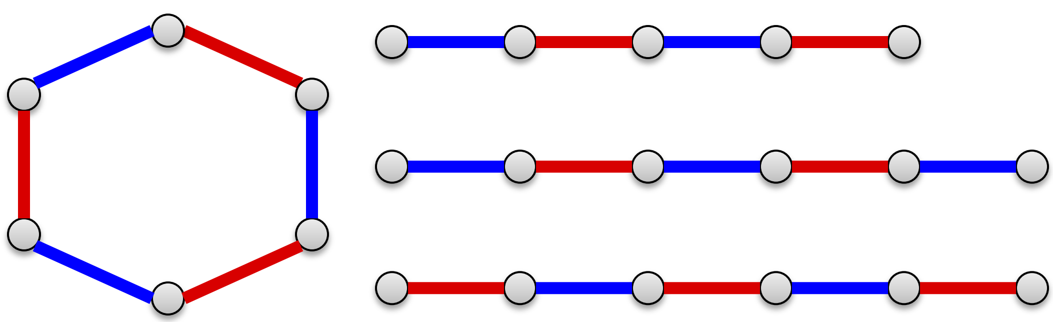
Theorem: A matching M of $G = (V, E)$ is maximum if and only if there is no augmenting path.

Proof:

- Consider non-max. matching M and max. matching M^* and define

$$F := M \setminus M^*, \qquad F^* := M^* \setminus M$$

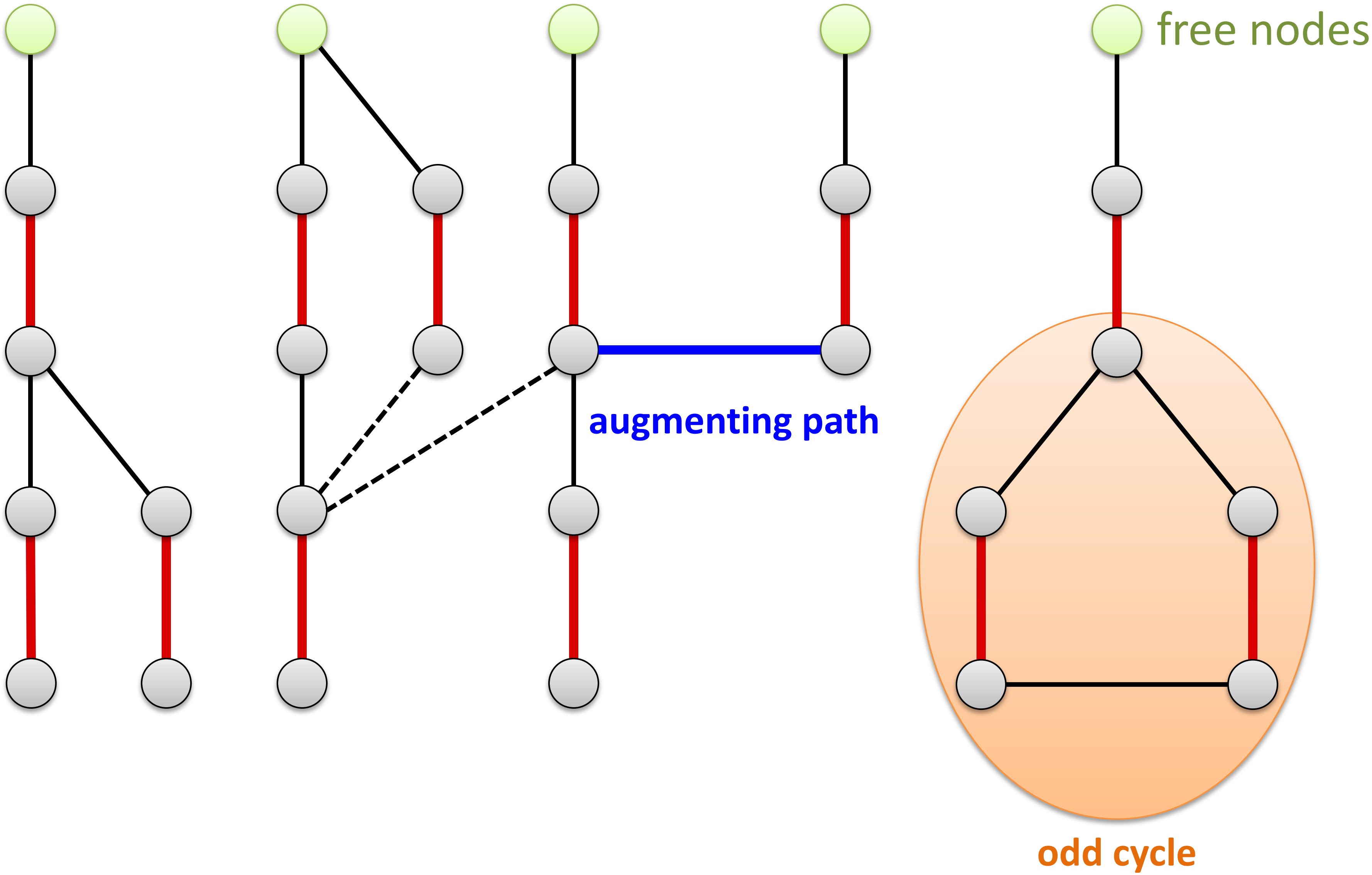
- Note that $F \cap F^* = \emptyset$ and $|F| < |F^*|$
- Each node $v \in V$ is incident to at most one edge in both F and F^*
- $F \cup F^*$ induces even cycles and paths



augmenting path for M

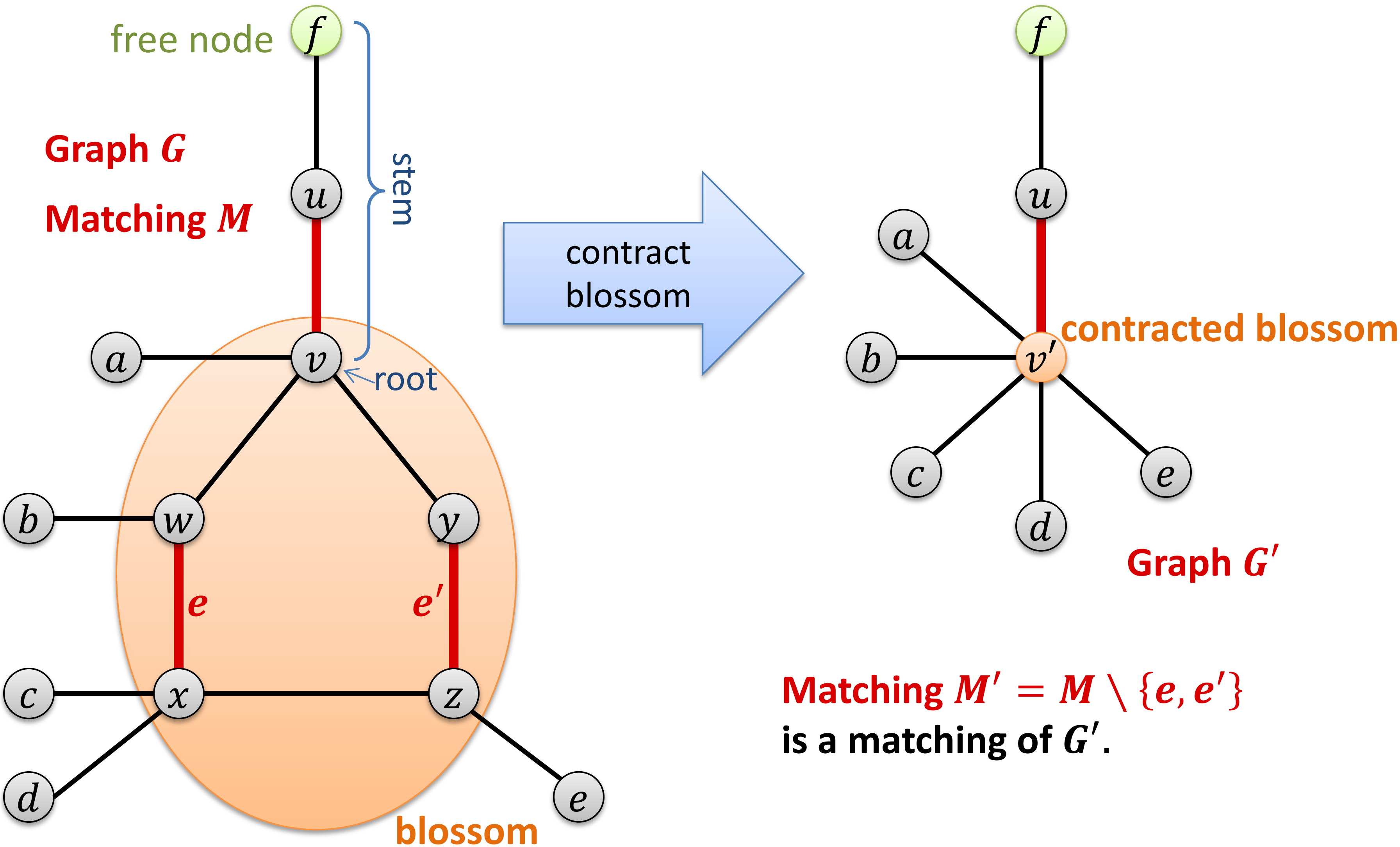
augmenting path for M^* (cannot exist)

Finding Augmenting Paths



Blossoms

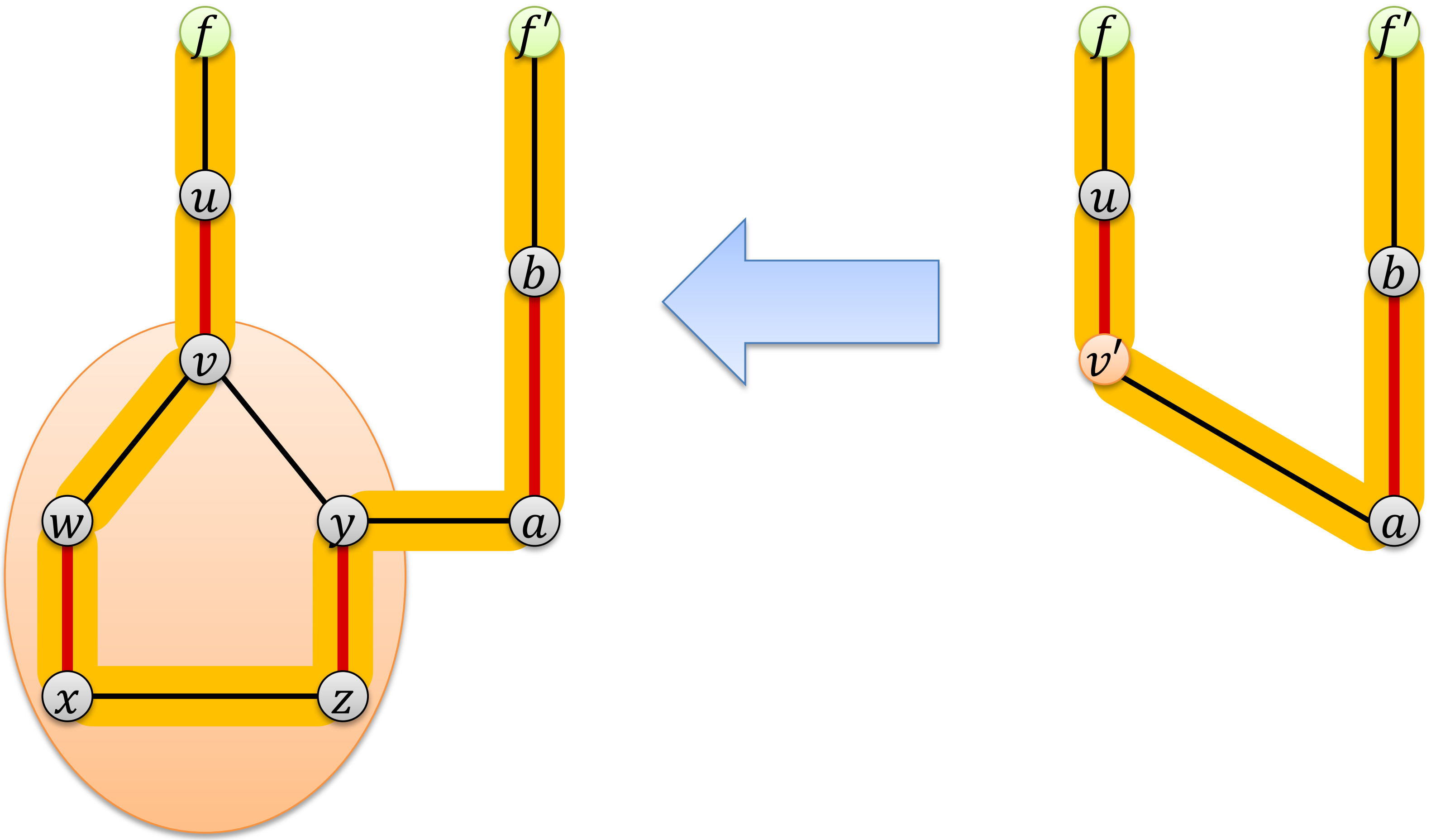
- If we find an odd cycle…



Matching $M' = M \setminus \{e, e'\}$ is a matching of G'.

Contracting Blossoms

Lemma: Graph G has an augmenting path w.r.t. matching M iff G' has an augmenting path w.r.t. matching M'.

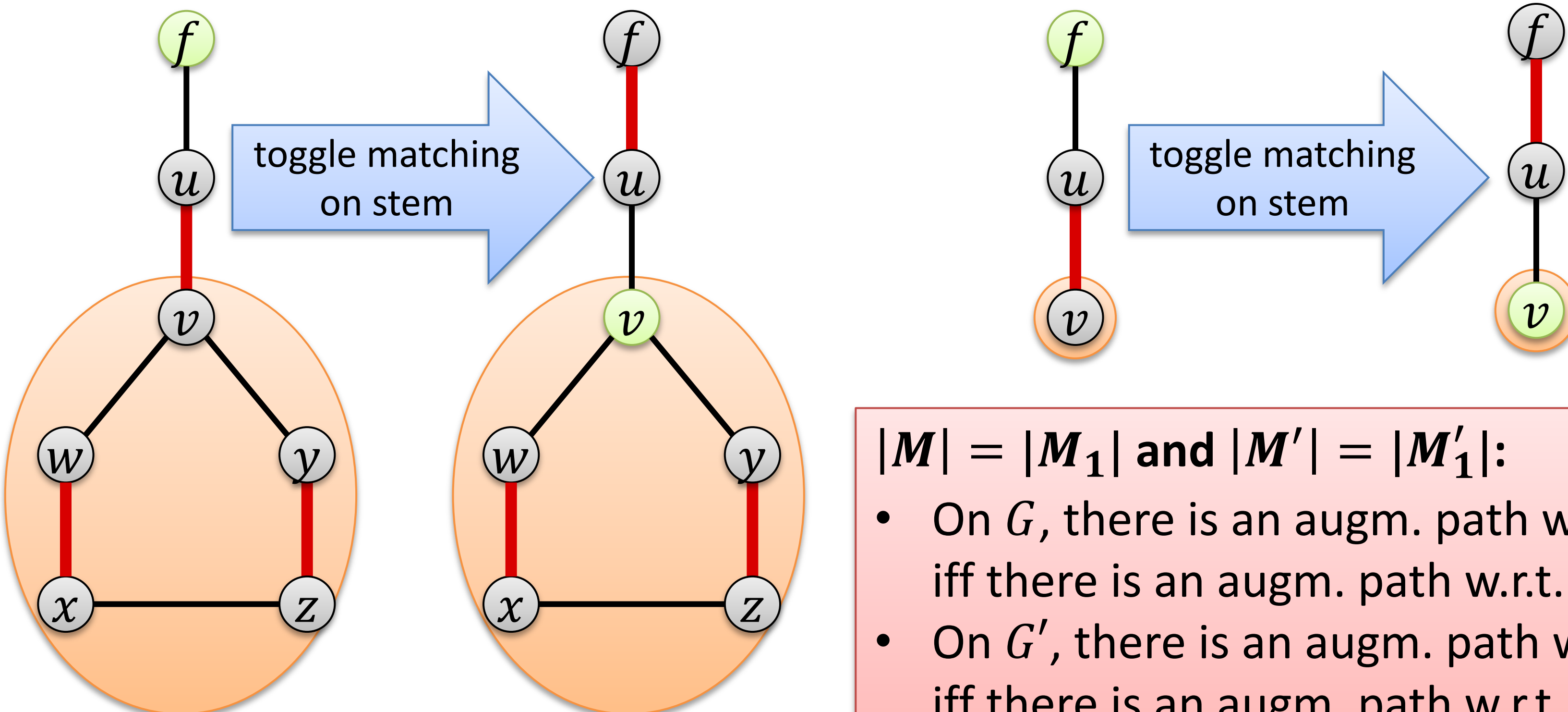


Also: The matching M can be computed efficiently from M'.

Contracting Blossoms

Lemma: Graph G has an augmenting path w.r.t. matching M iff G' has an augmenting path w.r.t. matching M'.

- Obtain matchings M_1 / $M_1{}'$ on G / G' by toggling matching on stem



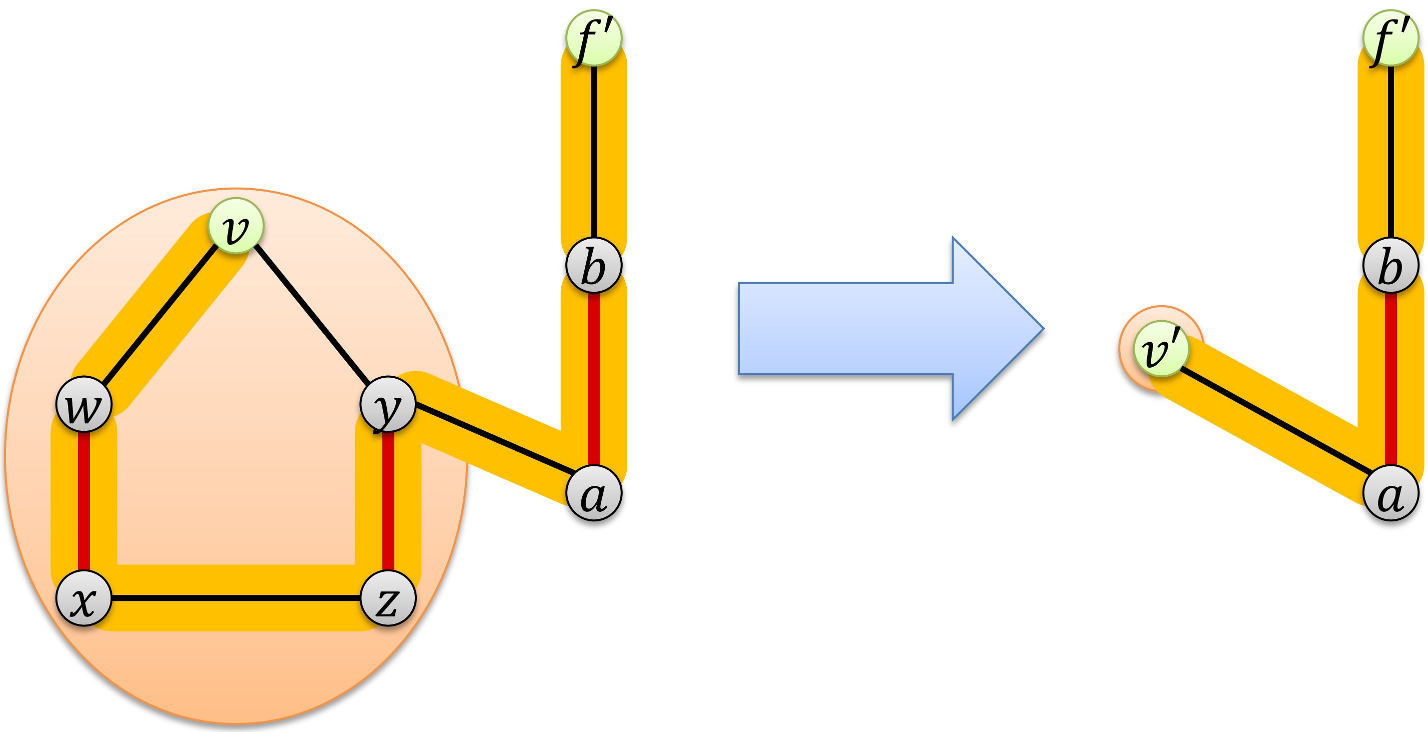
$|M| = |M_1|$ and $|M'| = |M_1'|$:

- On G, there is an augm. path w.r.t. M iff there is an augm. path w.r.t. M_1
- On G', there is an augm. path w.r.t. M' iff there is an augm. path w.r.t. M_1'
- We can w.l.o.g. assume that the root of the stem is a free node.

Contracting Blossoms

Lemma: Graph G has an augmenting path w.r.t. matching M iff G' has an augmenting path w.r.t. matching M'.

- If the root of the blossom is free, any augmenting path w.r.t. M_1 that contains nodes of the blossom can be turned into an augmenting path that ends at the root of the blossom and consists of a part inside the blossom and a part outside it.



Edmond's Blossom Algorithm

Algorithm Sketch:

1. Build a tree for each free node
2. Starting from an explored node u at even distance from a free node f in the tree of f, explore some unexplored edge $\{u, v\}$:

   1. If v is an unexplored node, v is matched to some neighbor w: add w to the tree (w is now explored)

   2. If v is explored and in the same tree:
      at odd distance from root → ignore and move on
      at even distance from root → **blossom found**

   3. If v is explored and in another tree
      at odd distance from root → ignore and move on
      at even distance from root → **augmenting path found**

Running Time

Finding a Blossom: Restart search on smaller graph

Finding an Augmenting Path: Improve matching

Theorem: The algorithm can be implemented in time $O(mn^2)$.

- DFS to find augmenting path or blossom: $O(m)$
- Needs to be repeated each time, when a blossom is found
  - Contraction of blossom reduces number of nodes by at least 2
  - Number of repetitions is $\leq n/2$
- In time $O(mn)$, we can find an augmenting path, if there is one and improve a given non-maximum matching
- Maximum matching has size $\leq n/2$

Matching Algorithms

We have seen:

- $O(mn)$ time alg. to compute a max. matching in *bipartite graphs*
- $O(mn^2)$ time alg. to compute a max. matching in *general graphs*

Better algorithms:

- Best known running time (bipartite and general gr.): $O(m\sqrt{n})$

Weighted matching:

- Edges have weight, find a matching of **maximum total weight**
- The problem can also be solved optimally in polynomial time, both in bipartite graphs and in general graphs
  - Algorithms use maximum matching in unweighted graphs as subroutine

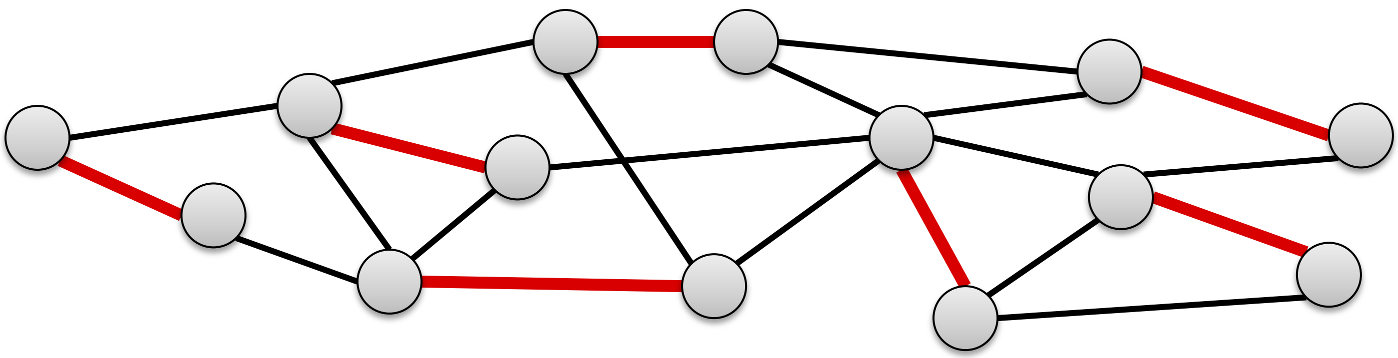
Vertex Cover vs Matching

Consider a matching M and a vertex cover S

Claim: $|M| \leq |S|$

Proof:

- At least one node of every edge $\{u, v\} \in M$ is in S
- Needs to be a different node for different edges from M

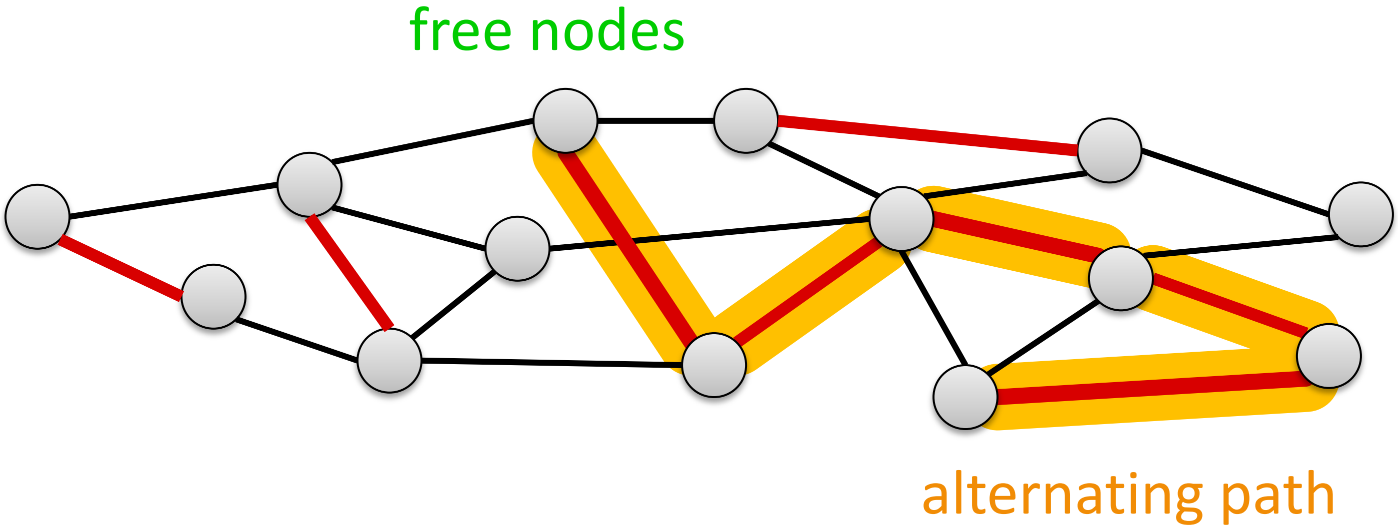


Augmenting Paths

Consider a matching M of a graph $G = (V, E)$:

- A node $v \in V$ is called **free** iff it is not matched

Augmenting Path: A (odd-length) path that starts and ends at a free node and visits edges in $E \setminus M$ and edges in M alternatingly.



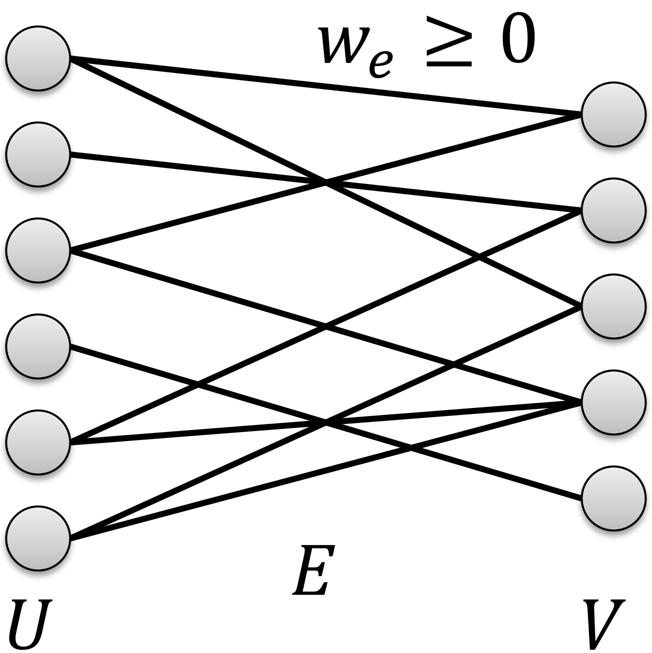
- Matching M can be improved using an augmenting path by switching the role of each edge along the path

Maximum Weight Bipartite Matching

- Let's again go back to bipartite graphs…

Given: Bipartite graph $G = (U \dot{\cup} V, E)$ with edge weights $w_e \geq 0$

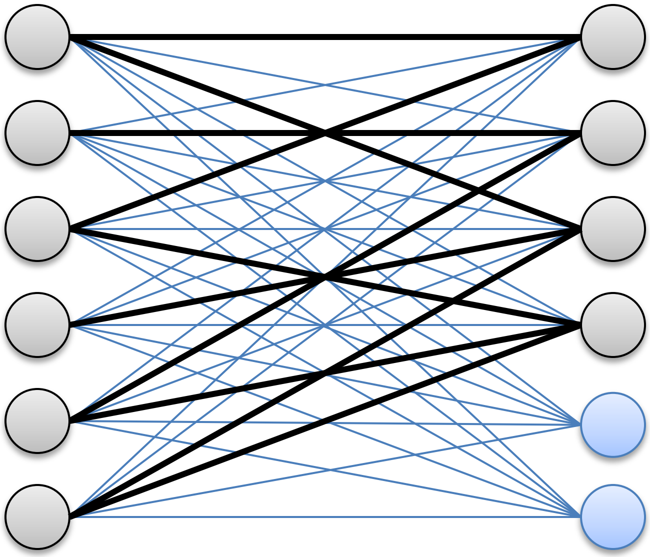
Goal: Find a matching M of maximum total weight



Minimum Weight Perfect Matching

Claim: Max weight bipartite matching is equivalent to finding a minimum weight perfect matching in a complete bipartite graph.

1. Turn into maximum weight perfect matching
   - add dummy nodes to get two equal-sized sides
   - add edges of weight 0 to make graph complete bipartite
2. Replace weights: $w'_e \coloneqq \max_f\{w_f\} - w_e$



A Dual Problem

Dual problem of the min. weight perfect matching problem

- Assign a variable $a_u \geq 0$ to each node $u \in U$
  and a variable $b_v \geq 0$ to each node $v \in V$
- **Condition:** for every edge $(\boldsymbol{u}, \boldsymbol{v}) \in \boldsymbol{U} \times \boldsymbol{V}$: $\boldsymbol{a_u + b_v \leq w_{uv}}$

Claim: For every perfect matching $M \in U \times V$, it holds that

$$\sum_{(u,v) \in M} w_{uv} \geq \sum_{u \in U} a_u + \sum_{v \in V} b_v$$

Optimality Condition

Slack: For each edge (u, v) and dual values a_u, b_v, we define

$$\boldsymbol{s_{uv}} := w_{uv} - a_u - b_v \geq 0$$

Claim: A perfect matching M is optimal if for all $(u, v) \in M$, $s_{uv} = 0$

- **Goal:** Find a dual solution a_u, b_v and a perfect matching such that the complementary slackness condition is satisfied!
  - i.e., for every matching edge (u, v), we want $s_{uv} = 0$
  - We then know that the matching is optimal!

Algorithm Overview

- Start with any feasible dual solution a_u, b_v
  - i.e., solution satisfies that for all (u, v): $w_{uv} \geq a_u + b_v$
- Let E_0 be the edges for which $s_{uv} = 0$
  - Recall that $s_{uv} = w_{uv} - a_u - b_v$
- Compute maximum cardinality matching M of E_0

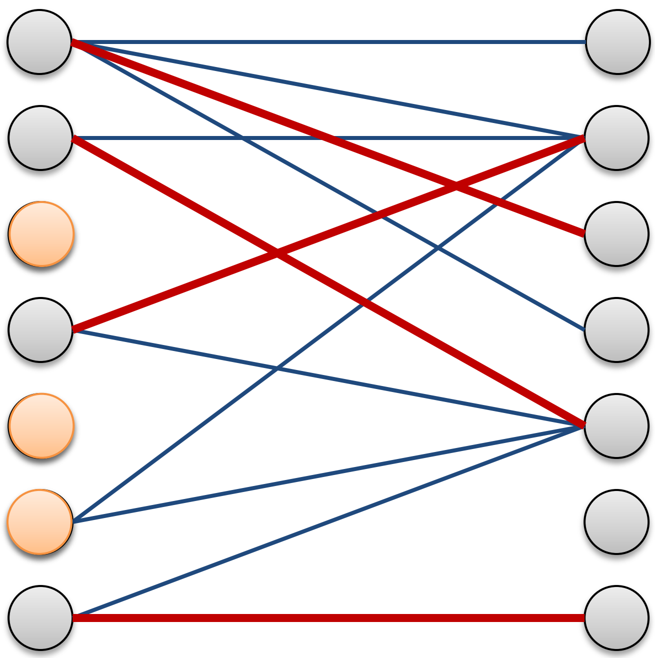
Observation: All edges (u, v) of the matching M satisfy $s_{uv} = 0$

- If M is a perfect matching, we are done
- If M is not a perfect matching, dual solution can be improved
  - We will look at this next…

Marked Nodes

Define set of marked nodes L:

- Set of nodes which can be reached on alternating paths on edges in E_0 starting from unmatched nodes in U



edges E_0 with $s_{uv} = 0$

optimal matching M

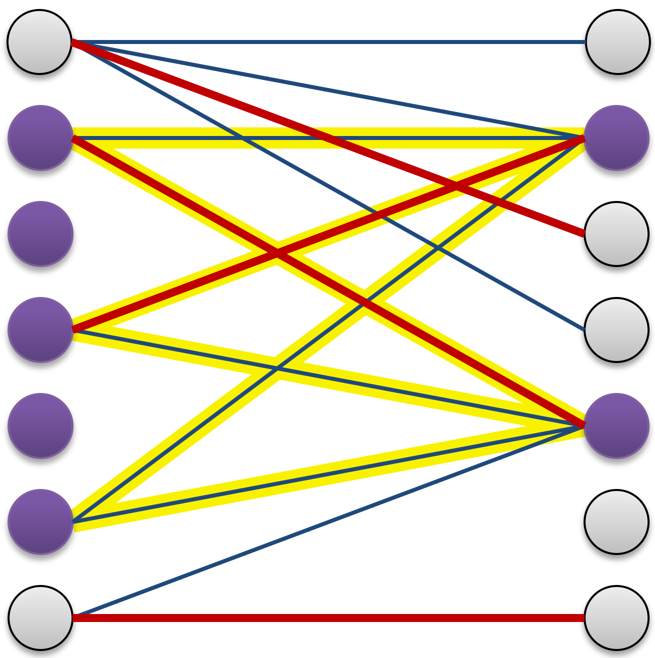
L_0: unmatched nodes in U

L: all nodes that can be reached on alternating paths starting from L_0

Marked Nodes

Define set of marked nodes L:

- Set of nodes which can be reached on alternating paths on edges in E_0 starting from unmatched nodes in U



edges E_0 with $s_{uv} = 0$

optimal matching M

L_0: unmatched nodes in U

L: all nodes that can be reached on alternating paths starting from L_0

Marked Nodes – Vertex Cover

Lemma:

a) There are no E_0-edges between $U \cap L$ and $V \setminus L$

b) The set $(U \setminus L) \cup (V \cap L)$ is a vertex cover of size $|M|$ of the graph induced by E_0

Improved Dual Solution

Recall: all edges (u, v) between $U \cap L$ and $V \setminus L$ have $s_{uv} > 0$

New dual solution:

$$\delta := \min_{u \in U \cap L,\, v \in V \setminus \mathrm{L}} \{s_{uv}\}$$

$$a'_u := \begin{cases} a_u, & \text{if } u \in U \setminus \mathrm{L} \\ a_u + \delta, & \text{if } u \in U \cap L \end{cases}$$

$$b'_v := \begin{cases} b_v, & \text{if } v \in V \setminus \mathrm{L} \\ a_v - \delta, & \text{if } v \in V \cap L \end{cases}$$

Claim: New dual solution is feasible (all s_{uv} remain ≥ 0)

Improved Dual Solution

Lemma: Obj. value of the dual solution grows by $\delta\left(\frac{n}{2} - |M|\right)$.

Proof:

$$\delta \coloneqq \min_{u \in U \cap L,\, v \in V \setminus \mathrm{L}} \{w_{uv}\}, \qquad a'_u \coloneqq \begin{cases} a_u, & \text{if } u \in U \setminus \mathrm{L} \\ a_u + \delta, & \text{if } u \in U \cap L \end{cases}, \qquad b'_v \coloneqq \begin{cases} b_v, & \text{if } v \in V \setminus \mathrm{L} \\ a_v - \delta, & \text{if } v \in V \cap L \end{cases}$$

Termination

Some terminology

- Old dual solution: $a_u, \quad b_v, \quad s_{uv} \coloneqq w_{uv} - a_u - b_v$
- New dual solution: $a'_u, \quad b'_v, \quad s'_{uv} \coloneqq w_{uv} - a'_u - b'_v$
- $E_0 \coloneqq \{(u,v) : s_{uv} = 0\}, \quad E'_0 \coloneqq \{(u,v) : s'_{uv} = 0\}$
- M, M' : max. cardinality matchings of graphs ind. By $E_0,\ E'_0$

Claim: We can always guarantee that $\boldsymbol{M \subseteq M'}$.

Termination

Lemma: The algorithm terminates in at most $O(n^2)$ iterations.

Proof:

- Each iteration: $|M'| > |M|$ or $M' = M$ and $|V \cap L'| > |V \cap L|$

Min. Weight Perfect Matching: Summary

Theorem: A minimum weight perfect matching can be computed in time $O(n^4)$.

- First dual solution: e.g., $a_u = 0$, $b_v = \min_{u \in U} w_{uv}$
- Compute set E_0: $O(n^2)$
- Compute max. cardinality matching of graph induced by E_0
  - First iteration: $O(n^2) \cdot O(n) = O(n^3)$
  - Other iterations: $O(n^2) \cdot O(1 + |M'| - |M|)$

Matching Algorithms

We have seen:

- $O(mn)$ time alg. to compute a max. matching in *bipartite graphs*
- $O(mn^2)$ time alg. to compute a max. matching in *general graphs*

Better algorithms:

- Best known running time (bipartite and general gr.): $O(m\sqrt{n})$

Weighted matching:

- Edges have weight, find a matching of **maximum total weight**
- *Bipartite graphs*: polynomial-time primal-dual algorithm
- *General graphs*: can also be solved in polynomial time
  (Edmond's algorithms is used as blackbox)