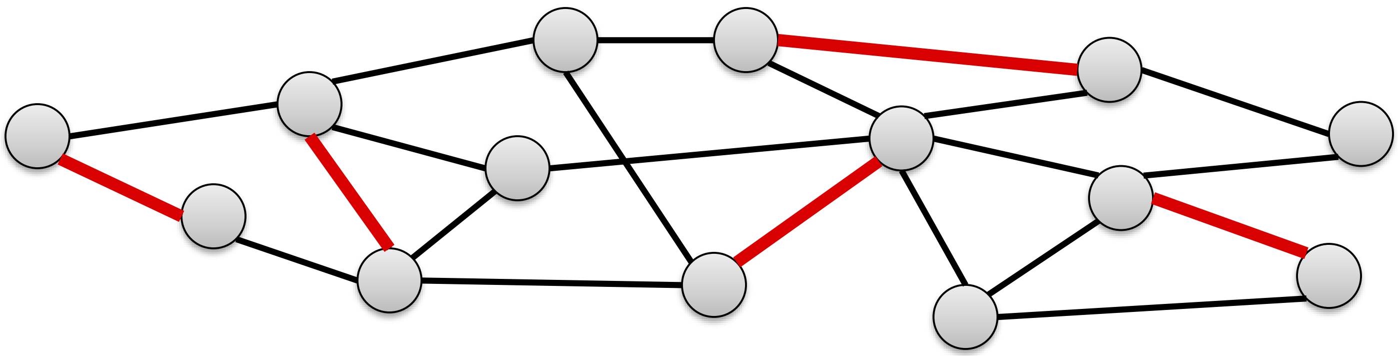
Chapter 6
Graph Algorithms

Algorithm Theory
WS 2018/19

Fabian Kuhn

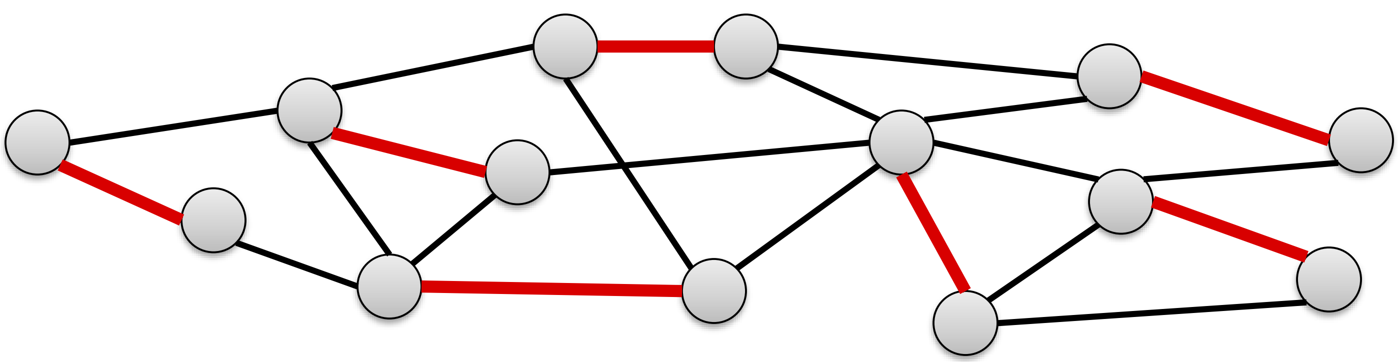
Matching

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)

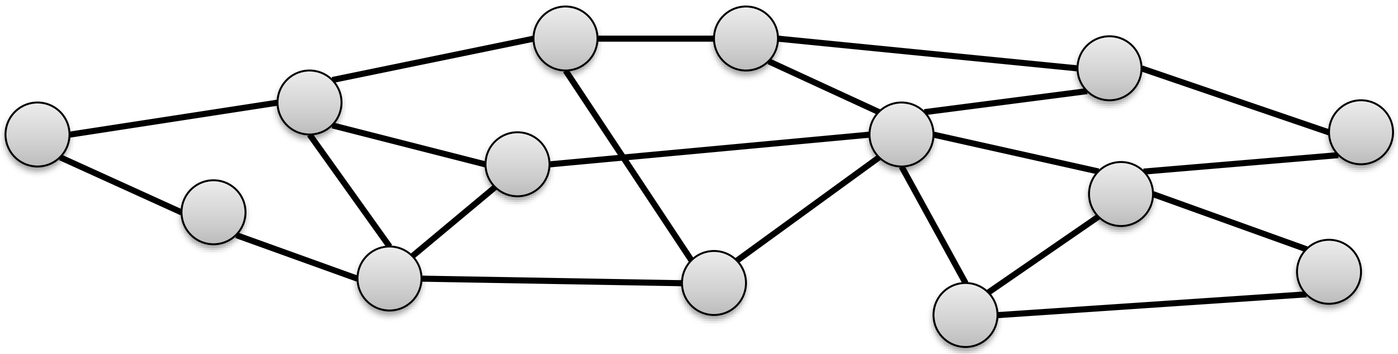
What About General Graphs

- Can we efficiently compute a maximum matching if G is not bipartite?

- How good is a maximal matching?
  - A matching that cannot be extended…

- **Vertex Cover:** set $S \subseteq V$ of nodes such that

$$\forall\{u,v\} \in E, \qquad \{u,v\} \cap S \neq \emptyset.$$



- A vertex cover covers all edges by incident nodes

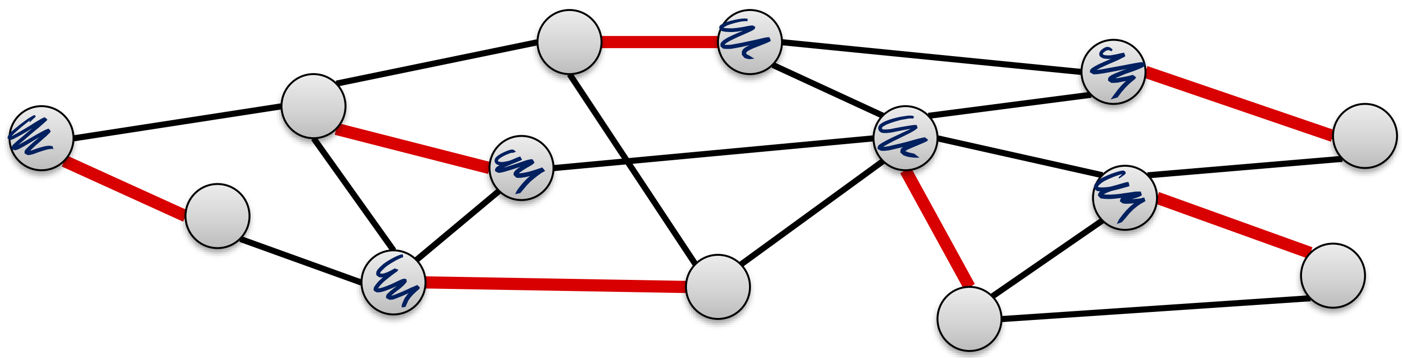
Vertex Cover vs Matching

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



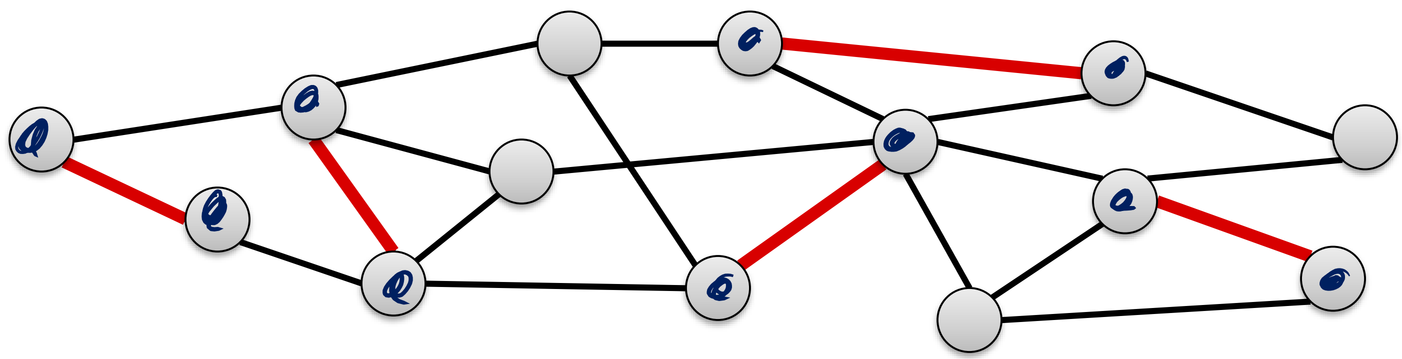
Vertex Cover vs Matching

Consider a matching M and a vertex cover S

Claim: If M is maximal and S is minimum, $|S| \leq 2|M|$

Proof:

- M is maximal: for every edge $\{u, v\} \in E$, either u or v (or both) are matched



- Every edge $e \in E$ is "covered" by at least one matching edge
- Thus, the set of the nodes of all matching edges gives a vertex cover S of size $|S| = 2|M|$.

Maximal Matching Approximation

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

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

Proof:

- Let S^* be a minimum vertex cover:

$$|M^*| \leq |S^*| \leq 2|M|$$

Theorem: The set of all matched nodes of a maximal matching M is a vertex cover S of size at most twice the size of a min. vertex cover.

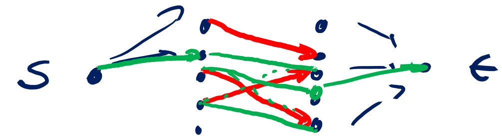
Proof:

- Let S^* be a minimum vertex cover

$|S| \leq 2|S^*|$

$$|S| = 2|M| \leq 2|S^*|$$

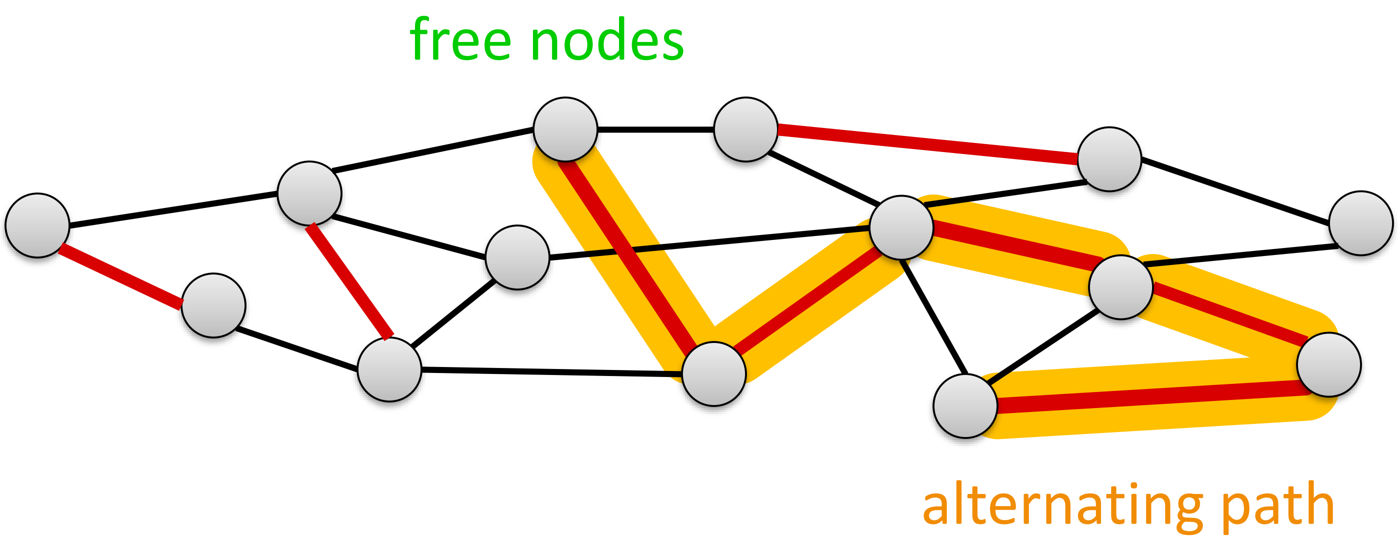
Augmenting Paths



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

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

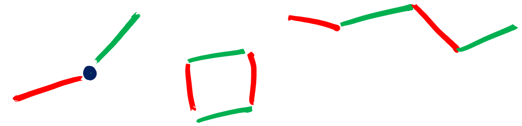
Augmenting Path: A (odd-length) path that starts and ends at a free node and visits edges in $E \setminus M$ and edges in M alternatingly.



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

Augmenting Paths

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

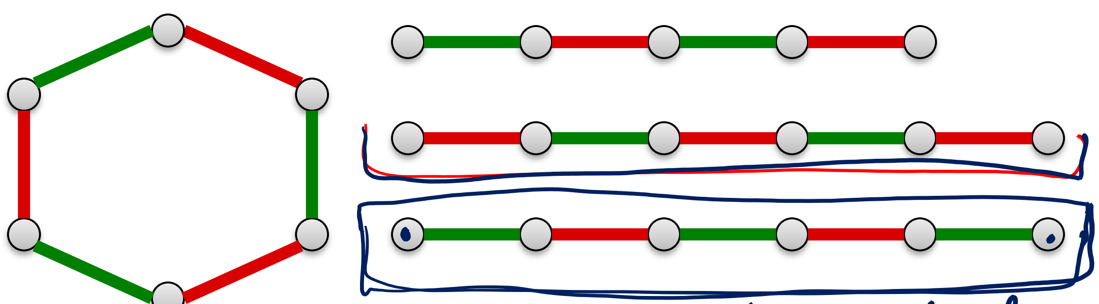


Proof: $\boxed{M \cap M^*}$

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

$$\color{red}{F := M \setminus M^*}, \qquad \color{green}{F^* := M^* \setminus M}$$

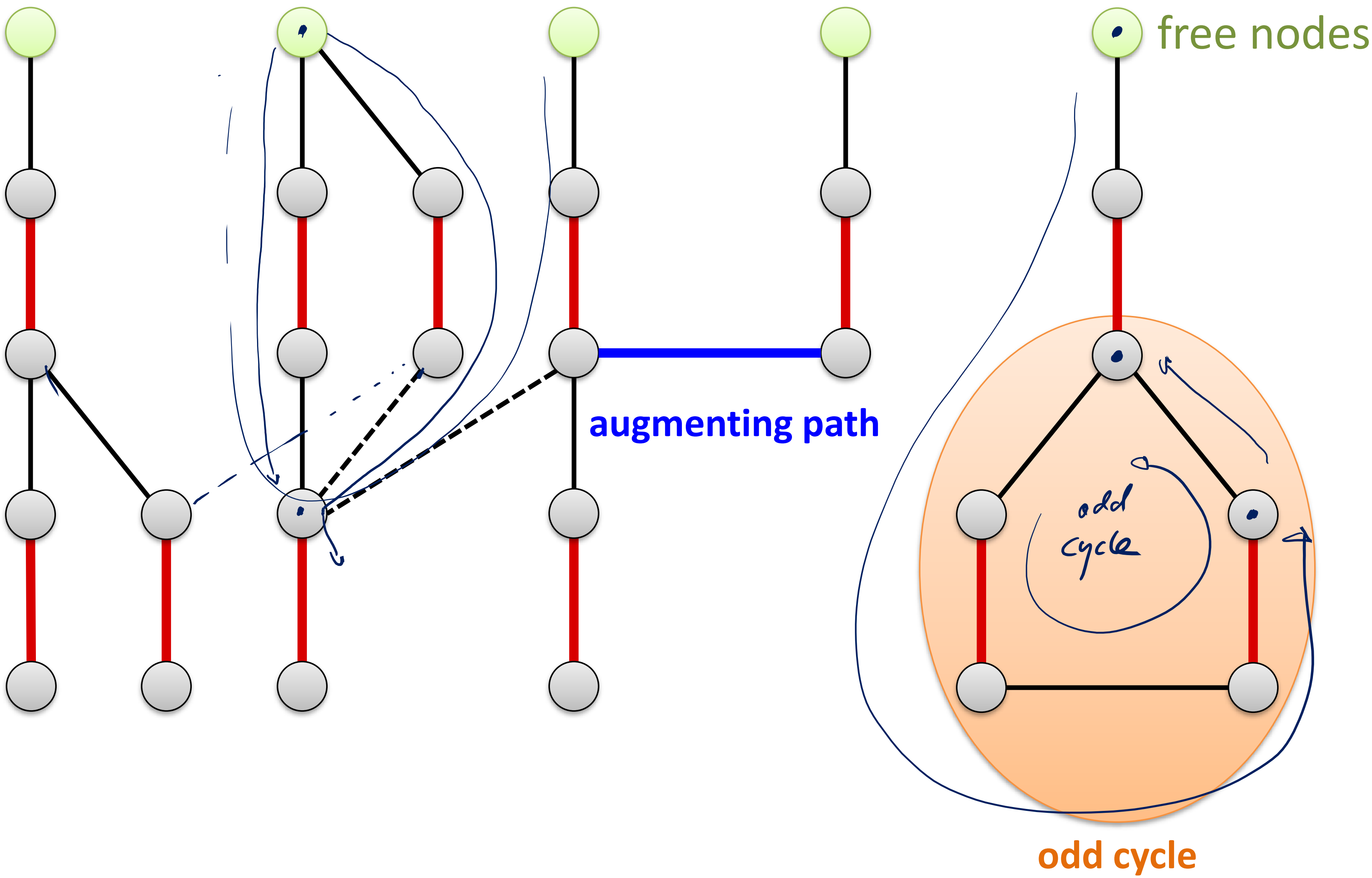
- Note that $F \cap F^* = \emptyset$ and $|F| < |F^*|$
- Each node $v \in V$ is incident to at most one edge in both F and F^*
- $F \cup F^*$ induces even cycles and paths



augmenting path for M^*
↳ cannot exist

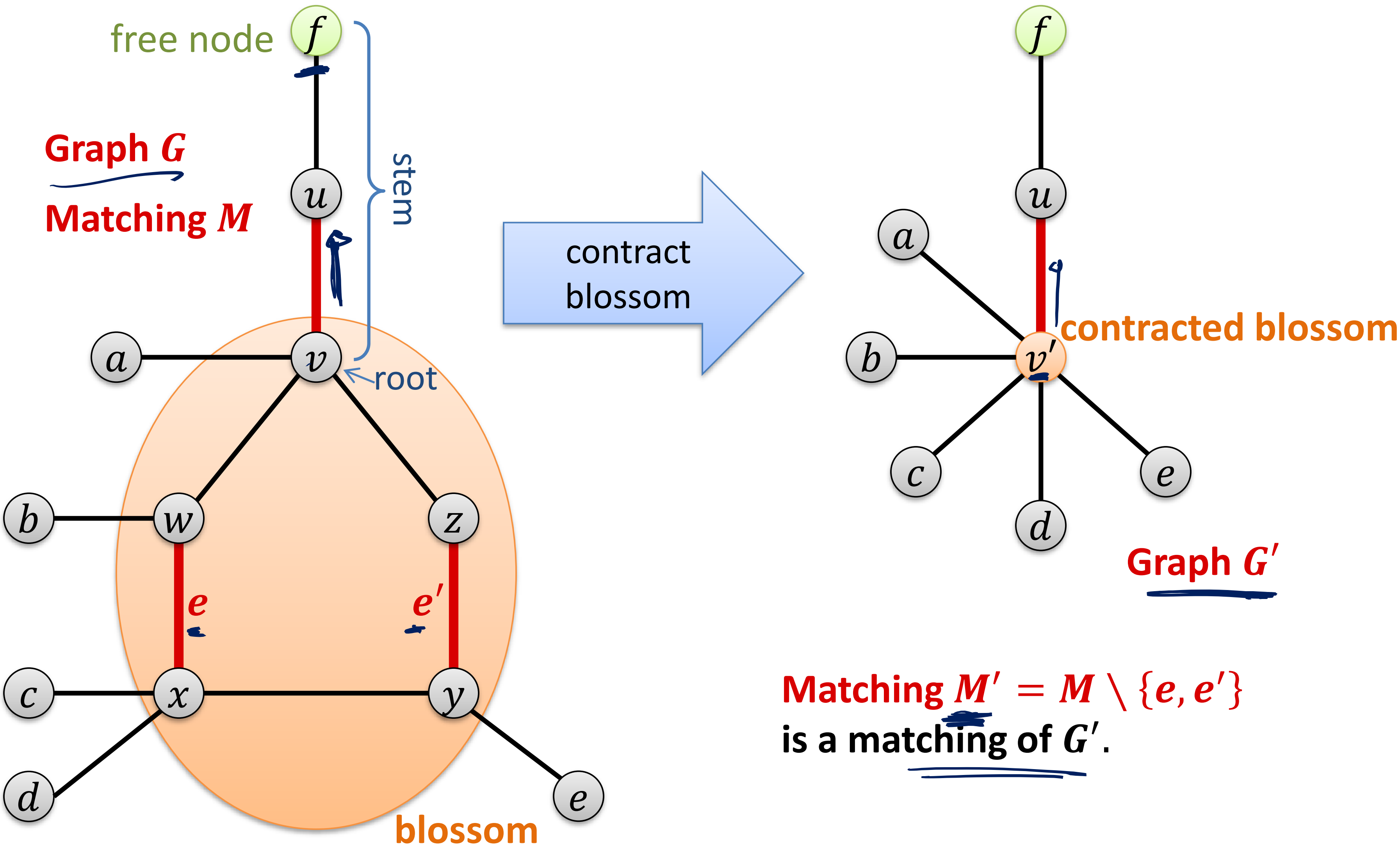
augmenting path for M

Finding Augmenting Paths



Blossoms

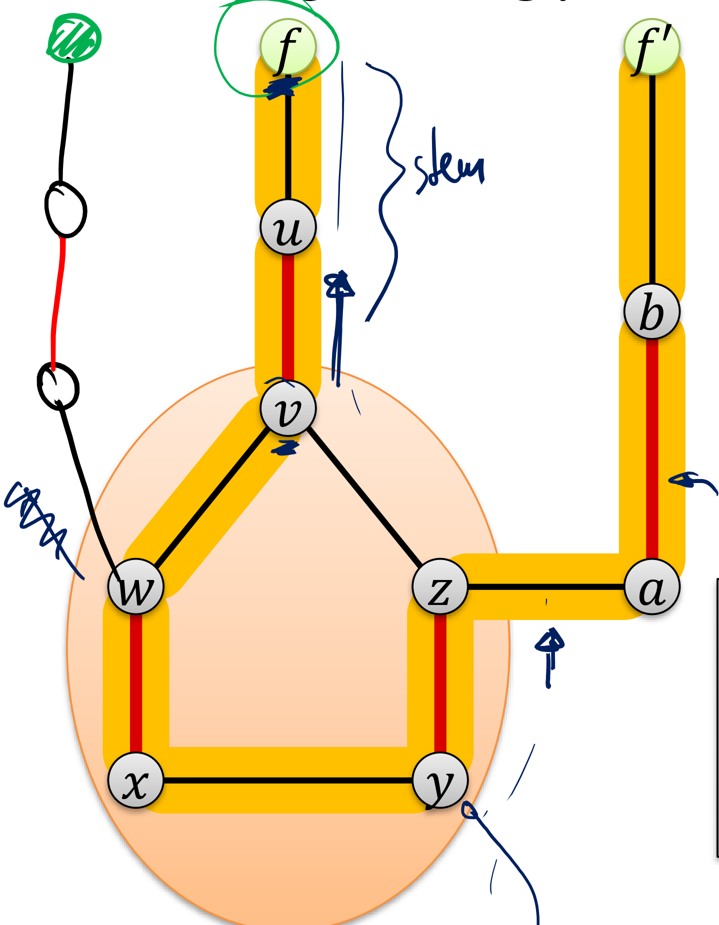
- If we find an odd cycle…

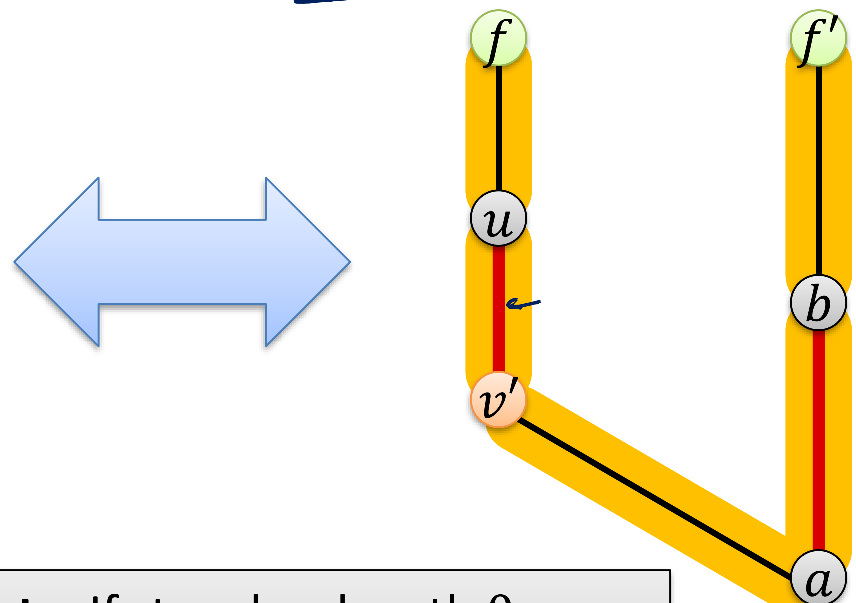


Matching $M' = M \setminus \{e, e'\}$
is a matching of G'.

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'





Note: If stem has length 0, root v of blossom is free and thus also the node v' is free in G'.

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

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** → contract blossom → recurse on smaller graph

   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: Repeat on smaller graph

Finding an Augmenting Path: Improve matching

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

graph expl. to find augm. path or blossom : DFS traversal
time: $O(m)$

can only contract $O(n)$ blossoms until we find an augm. path

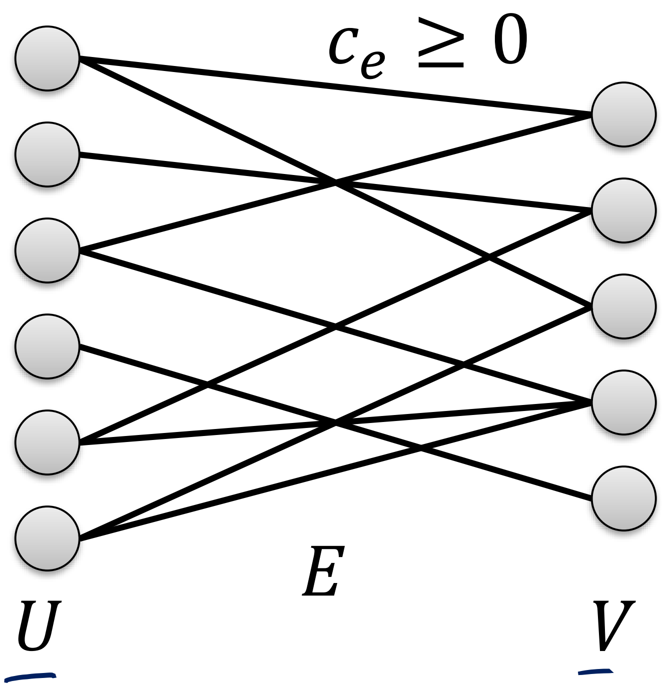
at most $n/2$ augm. paths

Maximum Weight Bipartite Matching

- Let's again go back to bipartite graphs…

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

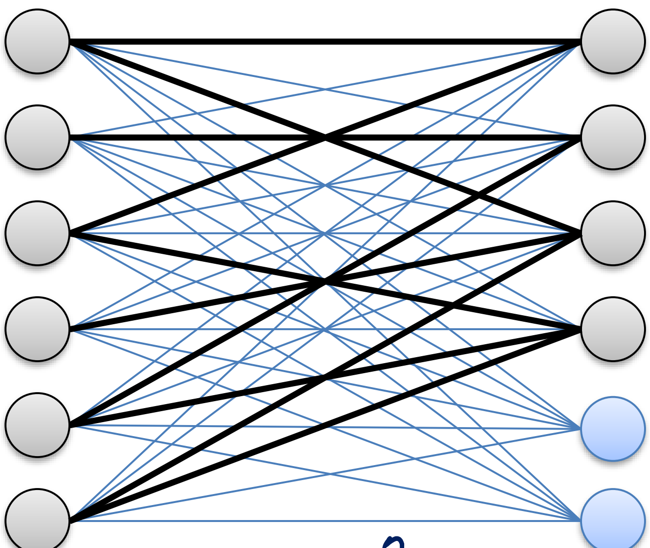
Goal: Find a matching M of maximum total weight



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: $c'_e := \max_f\{c_f\} - c_e$



As an Integer Linear Program

- We can formulate the problem as an integer linear program

Var. x_{uv} for every edge $(u, v) \in U \times V$ to encode matching M:

$$x_{uv} = \begin{cases} 1, & \text{if } \{u, v\} \in M \\ 0, & \text{if } \{u, v\} \notin M \end{cases}$$

Minimum Weight Perfect Matching

$$\min \sum_{(u,v)\in U\times V} c_{u,v} \cdot x_{u,v}$$

$$\forall u \in U: \quad \sum_{v \in V} x_{u,v} = 1$$

$$\forall v \in V: \quad \sum_{u \in U} x_{u,v} = 1$$

$$\boxed{\forall u,v: \quad x_{u,v} \in \{0,1\}}$$

Linear Programming (LP) Relaxation

Linear Program (LP)

- Continuous optimization problem on multiple variables with a linear objective function and a set of linear side constraints

LP Relaxation of Minimum Weight Perfect Matching

- Weight c_{uv} & variable x_{uv} for ever edge $(u, v) \in U \times V$

$$\min \sum_{(u,v)\in U\times V} c_{uv} \cdot x_{uv}$$

$$\text{s.t.}$$

$$\forall u \in U: \sum_{v\in V} x_{uv} = 1,$$

$$\forall v \in V: \sum_{u\in U} x_{uv} = 1$$

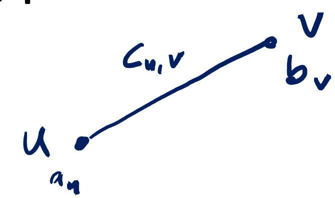
$$\forall u \in U, \forall v \in V: \; x_{uv} \geq 0$$

Dual Problem

- Every linear program has a dual linear program
  - The dual of a minimization problem is a maximization problem
  - Strong duality: primal LP and dual LP have the same objective value

In the case of the minimum weight perfect matching problem

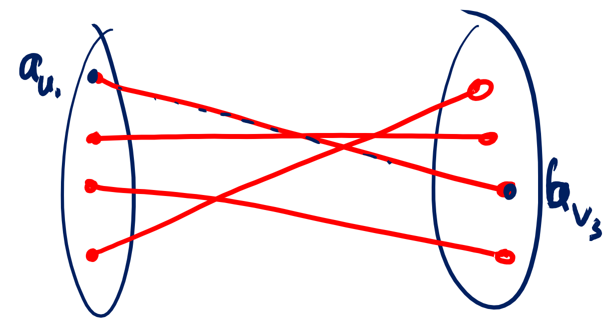
- Assign a variable $a_u \geq 0$ to each node $u \in U$
  and a variable $b_v \geq 0$ to each node $v \in V$



- **Condition:** for every edge $(\boldsymbol{u}, \boldsymbol{v}) \in \boldsymbol{U} \times \boldsymbol{V}$: $\boldsymbol{a_u + b_v \leq c_{uv}}$

- Given perfect matching M:

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



Dual Linear Program

- Variables $a_u \geq 0$ for $u \in U$ and $b_v \geq 0$ for $v \in V$

$$\max \sum_{u \in U} a_u + \sum_{v \in V} b_v$$

$s.t.$

$$\forall u \in U, \forall v \in V: \; \underline{a_u + b_v \leq c_{uv}}$$

- For every perfect matching M:

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

$=$ would imply that M is optimal!

would be sufficient to have

$\forall (u,v) \in M: \quad \underline{c_{u,v} = a_u + b_v}$

Complementary Slackness

$a_u + b_v \leq c_{u,v}$

- A perfect matching M is optimal if

$$\sum_{(u,v)\in M} c_{uv} \underset{\uparrow}{=} \sum_{u\in U} a_u + \sum_{v\in V} b_v$$

- In that case, for every $(u, v) \in M$

$$\boldsymbol{w_{uv}} \coloneqq c_{uv} - a_u - b_v = 0$$

  - In this case, M is also an optimal solution to the LP relaxation of the problem
  - Every optimal LP solution can be characterized by such a property, which is then generally referred to as complementary slackness

- **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 $w_{uv} = 0$
  - 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): $c_{uv} \geq a_u + b_v$

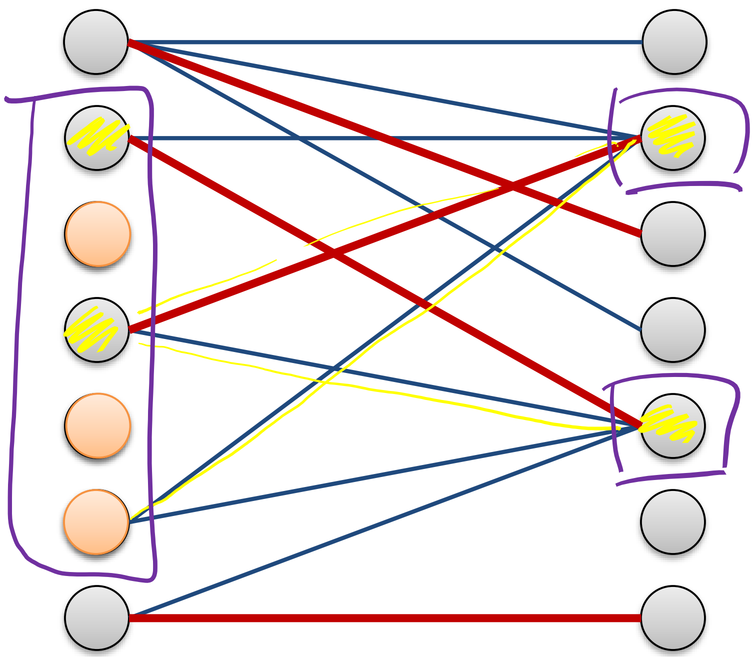
for example: $a_u = b_v = 0$

- Let E_0 be the edges for which $w_{uv} = 0$
  - Recall that $w_{uv} = c_{uv} - a_u - b_v$
- Compute maximum cardinality matching M of E_0
- All edges (u, v) of M satisfy $w_{uv} = 0$
  - Complementary slackness if satisfied
  - If M is a perfect matching, we are done
- If M is not a perfect matching, dual solution can be improved

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 $w_{uv} = 0$

optimal matching M

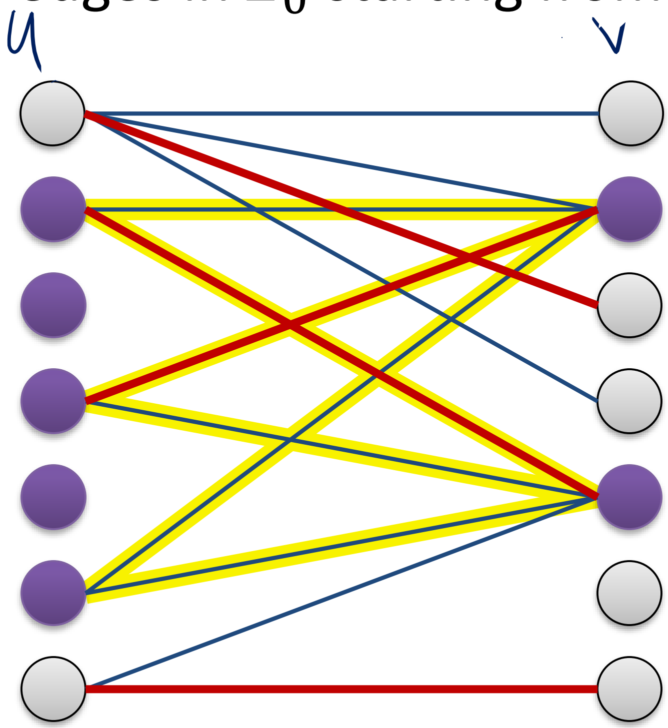
L_0: unmatched nodes in U

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 $w_{uv} = 0$

optimal matching M

L_0: unmatched nodes in U

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

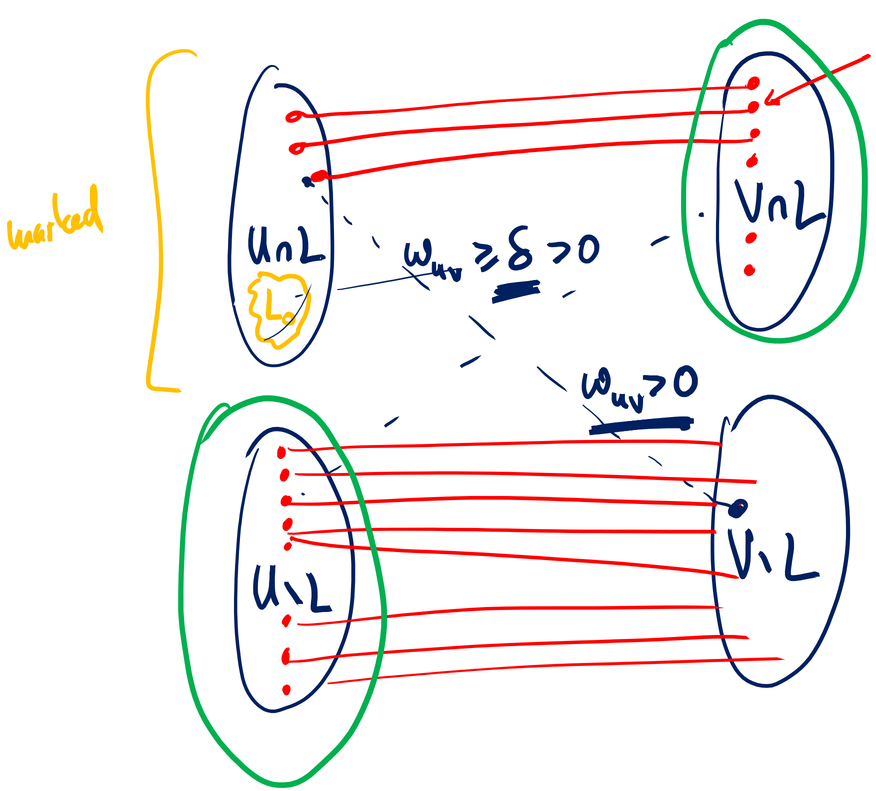
Marked Nodes – Vertex Cover

Lemma:

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

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

all node are matched



$$|M| = |U \setminus L| + |V \cap L|$$

Improved Dual Solution

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

New dual solution:

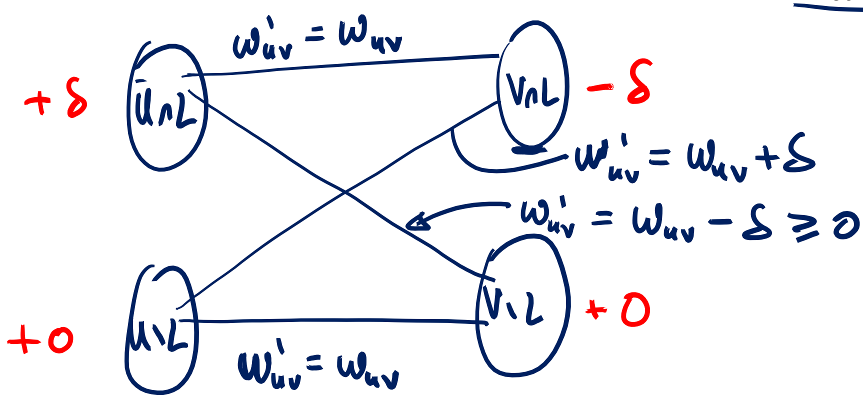
$$\delta := \min_{u \in U \cap L,\, v \in V \setminus L} \{w_{uv}\}$$

$w'_{uv} = c_{uv} - a'_u - b'_v$

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

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

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



Improved Dual Solution

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

Proof:

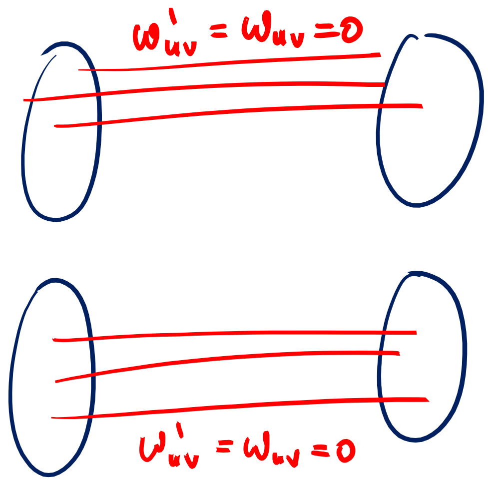
$$\delta \coloneqq \min_{u \in U \cap L,\, v \in V \setminus \mathrm{L}} \{w_{uv}\}, \qquad a'_u \coloneqq \begin{cases} a_u, & \text{if } u \in U \setminus \mathrm{L} \\ a_u + \delta, & \text{if } u \in U \cap L \end{cases}, \qquad b'_v \coloneqq \begin{cases} b_v, & \text{if } v \in V \setminus \mathrm{L}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

Termination

Some terminology

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

Claim: $|\boldsymbol{M'}| \geq |\boldsymbol{M}|$ and if $|M'| = |M|$, we can assume that $M = M'$.



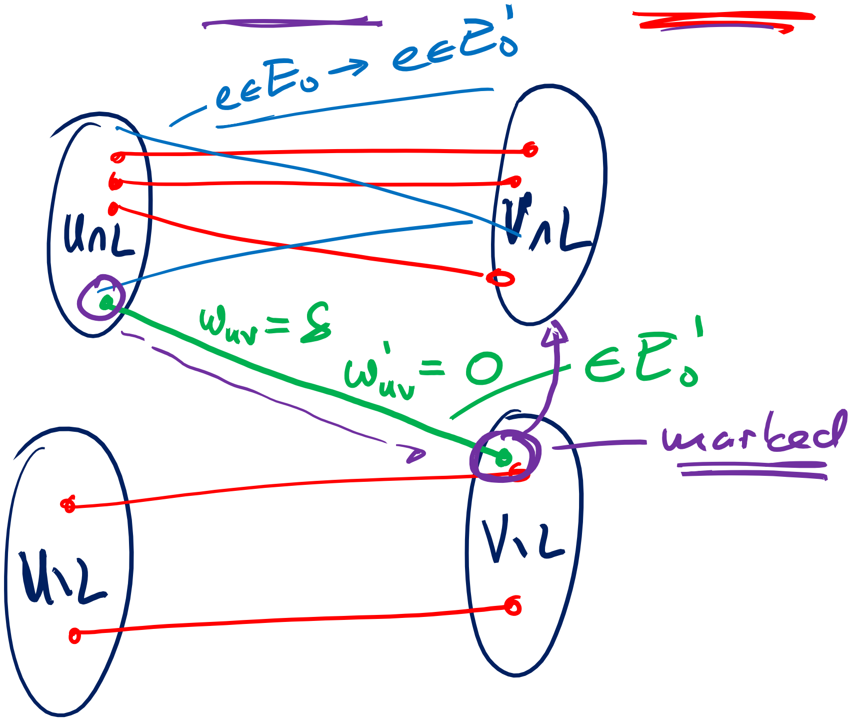
$M \subseteq E'_0$

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

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)