Lecture IV - Matching

[1] Department of Mathematics and Systems Analysis,
Systems Analysis Laboratory, Aalto University, Finland

January 8, 2024

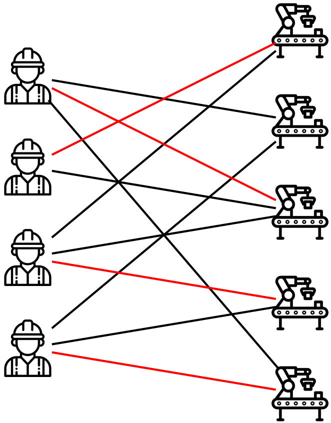
Aalto University

Previously on..

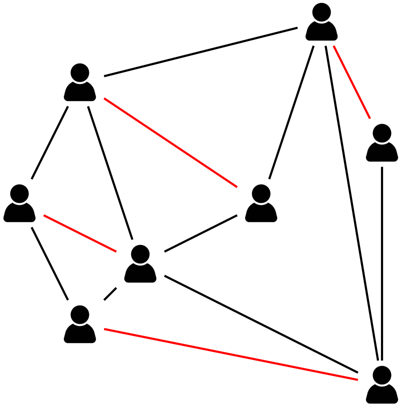
PREVIOUSLY ON...

- Shortest Path: Dijkstra;
- Minimum Spanning Tree: Prim and Kruskal

Motivation



assigning jobs to workers

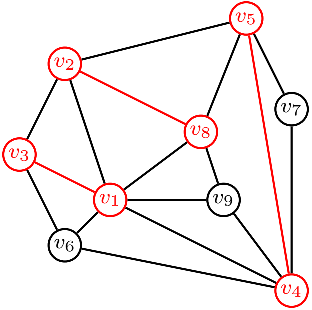


finding pairs for homework assignment

Definition

- $G = (V, E)$ undirected graph
- $M \subset E$ is called *matching* if all $e \in M$ are pairwise disjoint, i.e., if the endpoints are all different
- $M \subset E$ is a *maximum matching* in G if M is a matching with highest cardinality, i.e.,

$$|M'| \leq |M| \quad \text{for all matchings } M'$$



IP-formulation

$$
\begin{aligned}
\max \quad & \sum_{e \in E} x_e \\
\text{s.t.} \quad & \sum_{e \in \delta(v)} x_e \leq 1, \qquad v \in V \\
& x_e \in \{0,1\}, \quad e \in E
\end{aligned}
$$

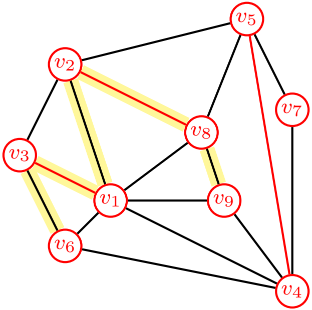
- incident edges of $v \in V$:

$$\delta(v) = \{e \in E \colon e = \{v, w\}\}$$

M-augmenting paths

- $G = (V, E)$ undirected graph, $M \subseteq E$ matching
- $v \in V$ is *covered* by M if $v \in e$ for some $e \in M$
- $v \in V$ is *exposed* by M if $v \notin e$ for all $e \in M$
- *M-alternating path P*: edges $E(P)$ are alternately in M and not in M (or not in M and in M)
- *M-augmenting path P*: M-alternating path with first and last vertex exposed

M-augmenting paths have odd number of edges



Berge's Theorem

Theorem (Petersen (1891), Berge (1957))

Let G be a graph with some matching M. Then M is maximum if and only if there is no M-augmenting path.

Proof idea $\Rightarrow$:

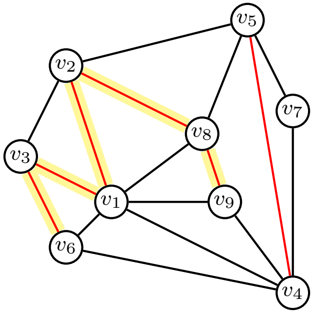
By contraposition: Let $P = (v_0, e_1, \ldots, e_k, v_k)$ be an M-augmenting path.

- by definition: v_0, v_k exposed

$\Rightarrow$ $|E(P) \setminus M| = |E(P) \cap M| + 1$

$\Rightarrow$ $M' = (M \setminus E(P)) \cup (E(P) \setminus M)$ is matching with $|M'| = |M| + 1$

$\Rightarrow$ M not maximum



Lemma

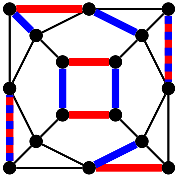
Lemma

Let G be a graph with two matchings M, M'. Let $G' = (V, E' = M \Delta M')$, with symmetric difference

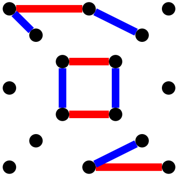
$$M \Delta M' = (M \cup M') \setminus (M \cap M').$$

Then the connected components of G' are

- *isolated vertices*
- *cycles C with $|E(C)| \in 2\mathbb{N}$ where edges in C are alternately in M and M'*
- *paths $P = (v_0, e_1, \ldots, e_k, v_k)$ where edges are alternately in M and M'*



graph G



graph G'

Aalto University

Combinatorial Optimization

Berge's Theorem

Proof idea:
M, M' matchings:

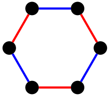
$$|\{e \in M : v \in e\}| \leq 1, v \in V$$
$$|\{e \in M' : v \in e\}| \leq 1, v \in V$$
$$\Rightarrow |\{e \in E' : v \in e\}| \leq 2, v \in V$$

If $g_{G'}(v) = |\{e \in E' : v \in e\}| = 2$:
$\exists! e \in M : v \in e$ and $\exists! e \in M' : v \in e$.

Berge's Theorem

- isolated vertices $v \rightsquigarrow g_{G'}(v) = 0$

- cycles C with $|E(C)| \in 2\mathbb{N} \rightsquigarrow g_{G'}(v) = 2$



- paths $P = (v_0, e_1, \ldots, e_k, v_k) \rightsquigarrow g_{G'}(v_0) = 0 = g_{G'}(v_k) = 1,\ g_{G'}(v_i) = 2,$ $1 \leq i \leq k-1$

Berge's Theorem

Aalto University

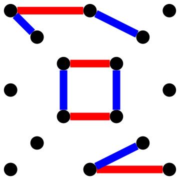
Combinatorial Optimization

Theorem (Petersen (1891), Berge (1957))

Let G be a graph with some matching M. Then M is maximum if and only if there is no M-augmenting path.

Proof idea $\Leftarrow$:

By contraposition: Let M' be a matching with $|M'| > |M|$. Construct G'.



graph G'

$$|M'| > |M| \Rightarrow |E' \cap M'| > |E' \cap M|$$
$$\Rightarrow \exists P = (v_0, e_1, \ldots, e_k, v_k) \text{ with } e_1 \in M', e_k \in M'$$
$$\Rightarrow v_0, v_k \text{ exposed by } M$$
$$\Rightarrow P\ M\text{-augmenting path}$$

General algorithm

Algorithm: MAXIMUM MATCHING

Input: undirected graph $G = (V, E)$

Output: maximum matching M

```
set M = ∅
while there exists M-augmenting path in G do
    choose M-augmenting path P
    set M = (M \ E(P)) ∪ (E(P) \ M)
return M
```

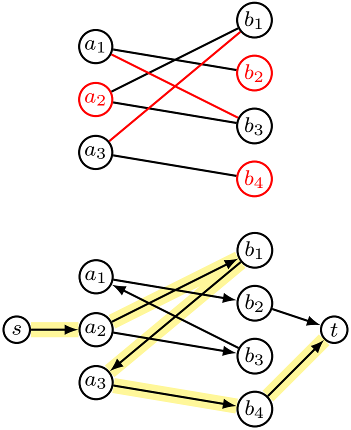
- at most $\frac{|V|}{2}$ iterations
- not obvious how to find an M-augmenting path
- for bipartite graphs: find s-t-path in auxiliary graph
- for general graphs: Edmond's blossom algorithm
  - complex algorithm
  - polynomial runtime

How to find M-alternating paths

- bipartite graph $G = (V, E)$ with
  - $V = A \cup B$, $A \cap B = \emptyset$
  - $E \subseteq \{\{a, b\} : a \in A, b \in B\}$
- for matching M construct auxiliary directed graph $G' = (V', E')$ with

$$
\begin{aligned}
V' =& V \cup \{s, t\}, \quad s, t \notin V \\
E' =& \{(b, a) : \{a, b\} \in M, a \in A, b \in B\} \\
& \cup \{(a, b) : \{a, b\} \in E \setminus M, a \in A, b \in B\} \\
& \cup \{(s, a) : a \text{ exposed}, a \in A\} \\
& \cup \{(b, t) : b \text{ exposed}, b \in B\}
\end{aligned}
$$

- $\exists$ M-augmenting path in G if and only if $\exists$ s-t-path in G'



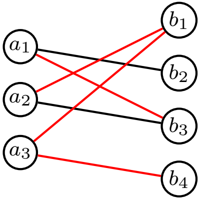
Maximum matching bipartite graphs – Example

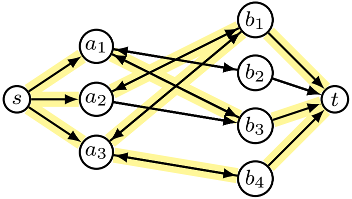
Algorithm: MAXIMUM MATCHING BIPARTITE GRAPHS

Input: undirected bipartite graph $G = (V, E)$

Output: maximum matching M

```
set M = ∅
construct G'
while there exists s-t-path in G' do
    choose s-t-path P
    set M = (M \ E(P)) ∪ (E(P) \ M)
    update G'
return M
```





Maximum matching bipartite graphs – Analysis

Aalto University

Algorithm: MAXIMUM MATCHING BIPARTITE GRAPHS

Input: undirected bipartite graph $G = (V, E)$

Output: maximum matching M

set $M = \emptyset$
construct G'
while *there exists s-t-path in* G' **do**
    choose s-t-path P
    set $M = (M \setminus E(P)) \cup (E(P) \setminus M)$
    update G'
return M

- $n = |V|, m = |E|$, no isolated nodes in G
- construction of G': $O(n+m)$
- at most $\frac{n}{2}$ iterations
  - finding P: $O(m)$
  - updating M: $O(n)$
  - updating G': $O(n)$

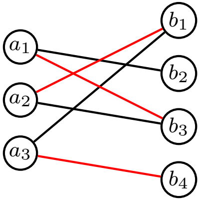
$\Rightarrow$ total runtime $O(nm)$

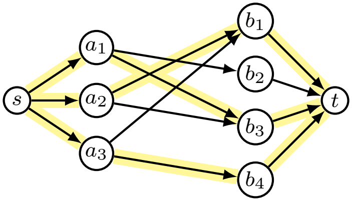
Connection to MaxFlow

- solve maximum matching as maximum flow problem
- construct auxiliary directed graph $G'' = (V'', E'')$ with

$$\begin{aligned} V'' =& V \cup \{s,t\}, \quad s,t \notin V \\ E'' =& \{(a,b) : \{a,b\} \in E, a \in A, b \in B\} \\ & \cup \{(s,a) : a \in A\} \\ & \cup \{(b,t) : b \in B\} \end{aligned}$$

  and capacity $u(e) = 1$ for all $e \in E''$
- G'' has maximal flow with value k if and only if G has maximum matching of cardinality k





Combinatorial Optimization


thank you