Advanced Algorithms

Nicole Megow (Universität Bremen)

SoSe 2025

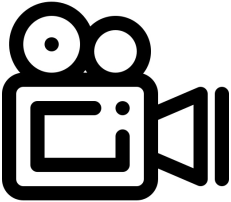
Weighted Bipartite Matchings

Lecture 1

Recording of this Lecture

This lecture will be recorded

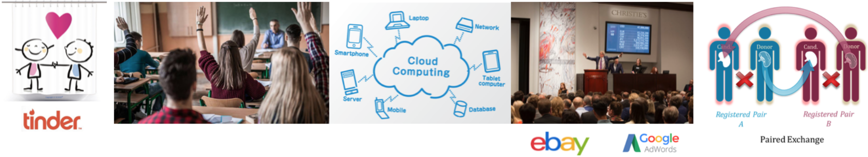
- Recording only of the lecturers by themselves.
- If there are questions from the audience, please make a clear signal if the microphone shall be muted.
- Our goal is to record the lecture, but it is no guarantee that each lecture will be recorded.



Matchings

Matching = Assignment, Pairing

Examples: dating-apps, assignment of students to schools/universities, allocation of resources in cloud computing, auctions, cross-over kidney exchange



Typical Questions:

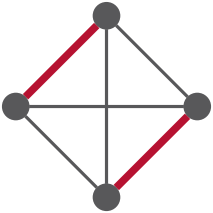
- which matchings are optimal?
  - w.r.t. number of found pairs, cost function, or fairness
- how do we find optimum matchings?

Matching

Definition

Let $G = (V, E)$ be a graph.

- A matching is a subset $M \subseteq E$ of edges in G with the property that no two edges in M have a common vertex.
- A vertex $v \in V$ is covered by M if there is a $u \in V$ such that $(v, u) \in M$.
- A matching M is perfect if M covers all vertices of G.



Maximum Matchings and Bipartite Graphs

Definition

Let $G = (V, E)$ be a graph.

- A matching M is maxim**al** if for every $e \in E \setminus M$ the edge set $M \cup \{e\}$ is not a matching.
- A matching M ist maxim**um** if $|M| \geq |M'|$, for all matchings M' in G.

Definitions for arbitrary graphs. **Today**: bipartite graphs.

Recall: An undirected graph $G = (V, E)$ is called bipartite, if the vertex set V can be partitioned into two sets L and R, such that there are no edges $e \in E$ with both endpoints in the same set.

Maximum Cardinality Matchings in bipartite Graphs

The Maximum Matching Problem

Maximum Matching Problem

Input: a (bipartite) graph $G = (V, E)$.
Task: a maximum matching M in G, i.e., a matching M, such that for all matchings M' in G it holds: $|M| \geq |M'|$.

Observation:

- For every matching M in $G = (V, E)$ it holds: $|M| \leq |V|/2$.
- If M is a perfect matching, we have: $|M| = |V|/2$.

Theorem

In bipartite graphs we can find a maximum matching in polynomial time.

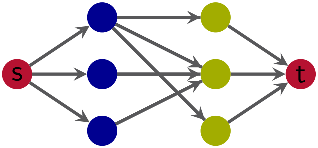
We already saw two versions.

Reduction to a Max-Flow Problem

Construction

From bipartite graph $G = (L \cup R, E)$ construct a network G':

- Set $G' = G$. Direct all edges from L to R.
- Add vertex s (source) and t (sink/target) as well as the edges (s, u) for all $u \in L$ and (v, t) for all $v \in R$.
- Set $c(e) = 1$ for all $e \in E'$.



$G' = (L \cup R \cup \{s, t\}, E', s, t, c)$

with $E' = \vec{E} \cup \{(s, u)\}_{u \in L} \cup \{(v, t)\}_{v \in R}$

and $c(e) = 1, e \in E'$

Theorem

G has a Matching of size $k \Leftrightarrow G'$ has a flow of value k.

Solving the Matching Problem

1. Construction of the graph G' in time $\mathcal{O}(n + m)$.
2. Computation of max flow with Ford-Fulkerson in time $\mathcal{O}(n \cdot m)$, since max flow has value at most $n/2$
3. From construction we can compute the max matching from the max flow.

Theorem

The Maximum Matching Problem can be solved in time$\mathcal{O}(n \cdot m)$ (via reduction to max flow).

Question:

- Can we also solve the matching problem directly?
- Also in edge-weighted bipartite graphs?
- Also in non-bipartite graphs?

M-alternating and M-augmenting Path

Let M be a matching in a (not necessarily bipartite) graph $G = (V, E)$. An M-alternating path in G is a path W in G, that alternatingly contains matching and non-matching edges.

A vertex is called exposed w.r.t. M if it is not incident to an edge of M

An M-alternating path W is called M-augmenting if both its endpoints are exposed. Then, $|E(W)|$ is odd.

Universität Bremen

Berge's Theorem

Idea: Increase a given matching M to M' by „edge-flip“ on M-augmenting path W.

We call this the symmetric difference:

$$M' = M \Delta E(W) := (M \cup E(W)) \setminus (M \cap E(W))$$
$$= M \setminus E(W) \cup E(W) \setminus M$$

Berge's Theorem

Idea: Increase a given matching M to M' by „edge-flip" on M-augmenting path W.

We call this the symmetric difference:

$$M' = M \Delta E(W) := (M \cup E(W)) \setminus (M \cap E(W))$$
$$= M \setminus E(W) \cup E(W) \setminus M$$

Berge's Theorem

Idea: Increase a given matching M to M' by „edge-flip“ on M-augmenting path W.

We call this the symmetric difference:

$$
\begin{aligned}
M' = M \Delta E(W) &:= (M \cup E(W)) \setminus (M \cap E(W)) \\
&= M \setminus E(W) \cup E(W) \setminus M
\end{aligned}
$$

Observation: M' is a matching and $|M'| = |M| + 1$.

Berge's Theorem

Idea: Increase a given matching M to M' by „edge-flip“ on M-augmenting path W.

We call this the symmetric difference:

$$
\begin{aligned}
M' = M \Delta E(W) &:= (M \cup E(W)) \setminus (M \cap E(W)) \\
&= M \setminus E(W) \cup E(W) \setminus M
\end{aligned}
$$

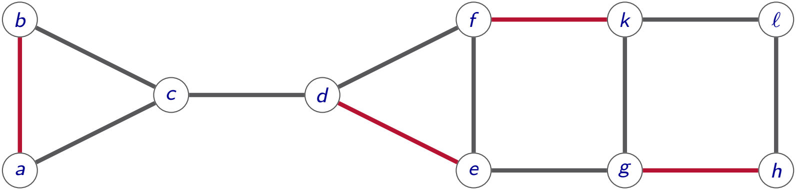
Observation: M' is a matching and $|M'| = |M| + 1$.

Theorem (Berge 1957)

A matching M in an arbitrary graph is maximum if and only if there is no M-augmenting path.

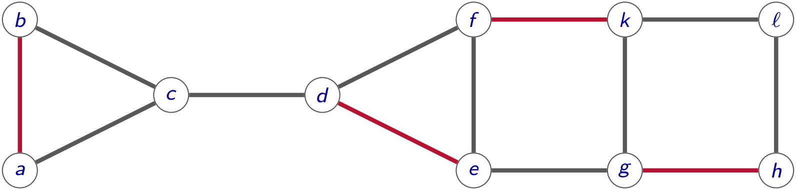
Example

Consider the following graph G with matching M:



Example

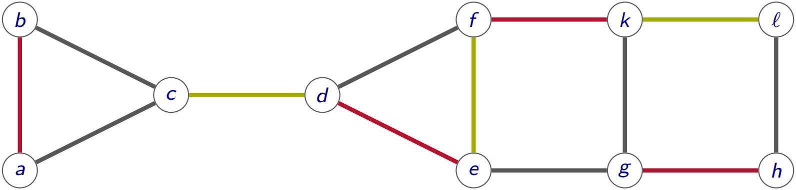
Consider the following graph G with matching M:



- Exposed vertices w.r.t. M: c, ℓ

Example

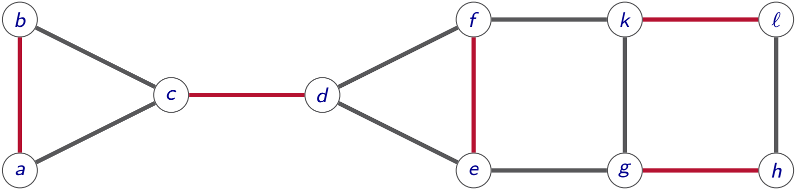
Consider the following graph G with matching M:



- Exposed vertices w.r.t. M: c, ℓ
- M-augmenting path: (c, d, e, f, k, ℓ)

Example

Consider the following graph G with matching M:



- Exposed vertices w.r.t. M: c, ℓ
- M-augmenting path: (c, d, e, f, k, ℓ)
- We get matching M' with $|M'| = |M| + 1$.

Universität Bremen

Augmenting paths in bipartite graphs

How do we find M-augmenting paths? In general not easy.

Idea for bipartite graphs:
- Direct non-matching edges from $L \to R$ and matching edges from $R \to L$.
- Add source s with edges towards exposed vertices of L.

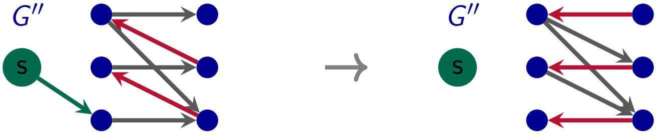
Observation: M-augmenting paths in G are directed paths in G'' with exposed endvertices.

Theorem. Breadth-First-Search BFS(G'', E'') finds exposed endvertices in $G'' \Leftrightarrow G$ has M-augmenting path.

Algorithm for bipartite graphs

Theorem. Breadth-First-Search BFS(G'', E'') finds exposed endvertices in $G'' \Leftrightarrow G$ has M-augmenting path.

Algorithm: As long as there exists an M-augmenting path in G (check via BFS in G''), update M and G''.



In each iteration we increase the size of the matching by 1 (terminates after at most $|M| \leq |V|/2$ iterations; per iteration one BFS in polynomial time).

Theorem

The algorithm finds in time $\mathcal{O}(n \cdot m)$ a maximum matching M in a bipartite graph G.

Weighted Bipartite Matchings

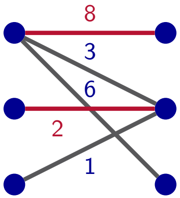
Weighted Bipartite Matchings

So far: Matchings with binary preferences (acceptable/not acceptable)

More general: Difference in preferences reflected by acceptance of paying/receiving different prices/values, if matched.

Example: AdWord Allocation for internet search engines

Companies bid money on keywords to place advertisements. The search engine computes a matching from bidders to keywords to maximize their profit.



Value of a matching:
Sum of the prices, that the companies pay for the matching; here: 8+2=10.

Maximum weight bipartite matching

Maximum weight bipartite matching

Input: A (bipartite) weighted graph $G = (V, E, c)$.
Task: Find a maximum-weight matching M in G, i.e., a matching M, such that for all matchings M' in G we have: $c(M) \geq c(M')$, where $c(M) = \sum_{e \in M} c(e)$.

- How do we compute a maximum weight matching in a bipartite graph?
- How can we use the techniques from maximum cardinality bipartite matching?

We use the following notation:

- A matching M is called extreme if $c(M) \geq c(M')$ for every matching M' in G with $|M| = |M'|$.

Augmenting paths in weighted bipartite graphs

We use an algorithm similar to before:

Idea for weighted bipartite graphs:

- Start with an empty matching $M = \emptyset$.
- Given a matching M, do the following:
- Direct non-matching edges e from $L \to R$ and assign a weight of $-c_e$.
- Direct matching edges e from $R \to L$ and assign a weight of c_e.
- Add source s with edges directed towards exposed vertices of L of cost 0.

Lemma. Let P be a shortest path from s to an exposed vertex of R. If M is extreme, then also $M \Delta P$ is extreme.

Algorithm for maximum weight bipartite matching

Full algorithm for weighted bipartite matching:

- Start with an empty matching $M = \emptyset$.
- Given a matching M, construct directed graph and compute a shortest path from s to any exposed vertex of R.
- Among all computed matchings, output the one with maximum weight.

Theorem (Kuhn 1955, Munkres 1957)

We can compute a maximum-weight matching in a bipartite graph in time $O(|V|^2 \cdot |E|)$. (Hungarian Method)

Using Dijkstra, we can improve this:

Theorem (Kuhn 1955, Munkres 1957)

We can compute a maximum-weight matching in a bipartite graph in time $O(|V| \cdot (|E| + |V| \log(|V|)))$.

Algorithm for maximum weight bipartite matching

- Actually, we can stop as soon as new matching M' has no larger weight than M, that is, no s-R_M path in D_M has negative length, where R_M is the set of exposed vertices from R.
- We obtain the following improved running-time. Its proof will be an exercise.

Theorem.

We can compute a maximum-weight matching in a bipartite graph in time $O(n' \cdot (|E| + |V| \log(|V|)))$, where n' is the minimum size of a maximum-weight matching.

The Assignment Problem

A closely related problem is the assignment problem, the problem of finding a minimum-weight perfect matching in an edge-weighted (bipartite) graph G:

Minimum-cost bipartite matching

Input: A (bipartite) weighted graph $G = (V, E, c)$.
Task: Find a minimum-weight perfect matching M in G.

Adapting our algorithm from above yields a polynomial time algorithm for finding an optimum solution for the assignment problem. Proving this is an exercise.

Theorem.

We can compute a minimum-weight perfect matching in a bipartite graph in time $O(|V| \cdot (|E| + |V| \log(|V|)))$.

Recap and Outlook

- Maximum matchings in bipartite graphs
  - Reduction to max flows
  - Optimality criteria, Berge's Theorem
  - Augmenting path algorithm
- Weighted matchings in bipartite graphs
- Next Lectures:
  - Maximum matchings in non-bipartite graphs
  - Outlook: Maximum-weight matchings in non-bipartite graphs
  - Stable matchings