Kapitel 1.6

Matchings

Matching-Algorithmen

Der Satz von Hall (Heiratssatz)

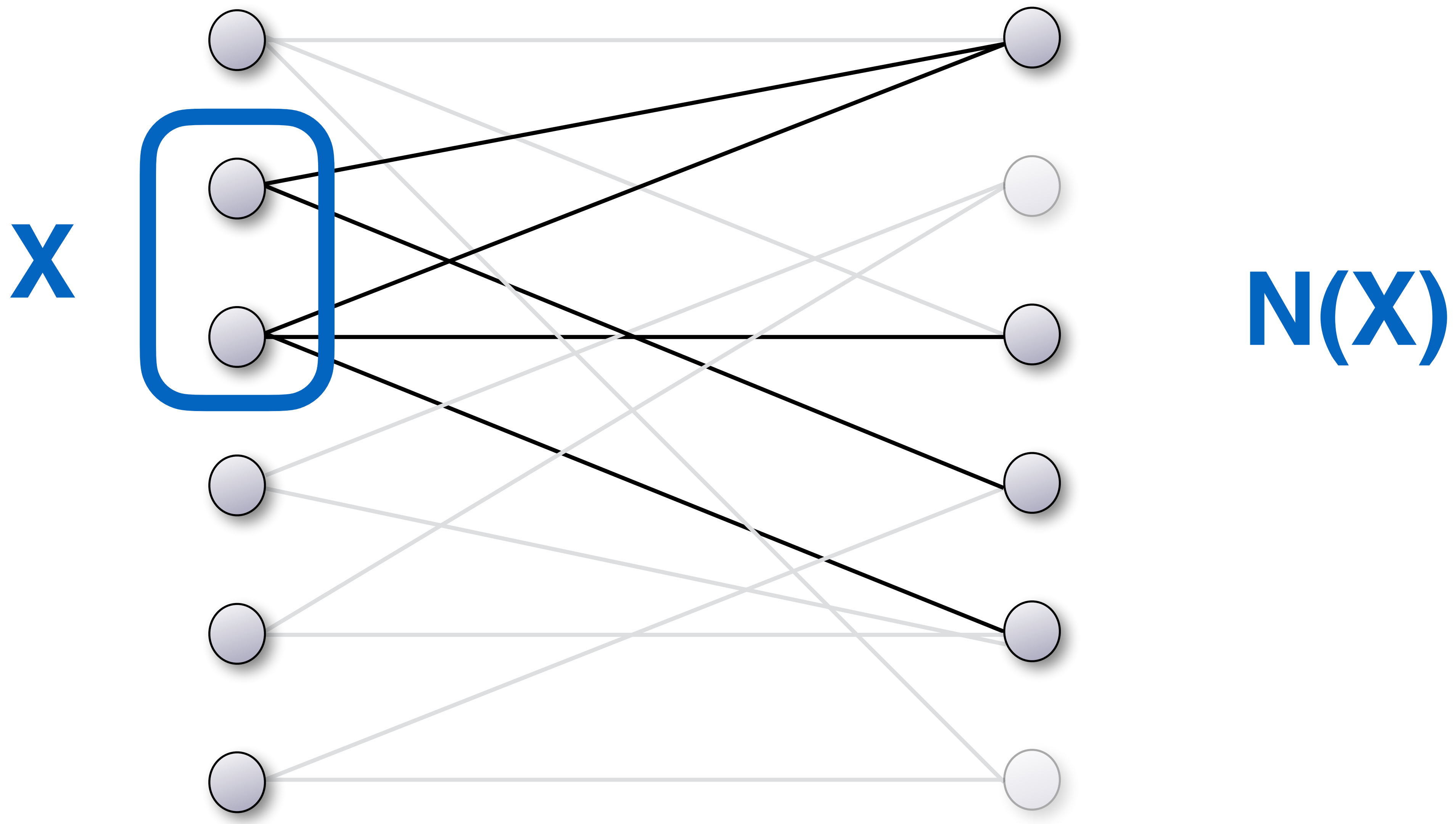
Der Satz von Hall (Heiratssatz)

Philip Hall
(1904-1982)

Satz: (Hall, Heiratssatz)

Ein bipartiter graph $G=(A \uplus B, E)$ enthält ein Matching M der Kardinalität $|M|=|A|$ **gdw**

$$\forall\, X \subseteq A \;:\; |X| \leq |N(X)|$$



muss für **jede** Teilmenge gelten

„$\Longrightarrow$“ ✓

„$\Longleftarrow$“ !

Der Satz von Hall (Heiratssatz)

Satz: (Hall, Heiratssatz)
Ein bipartiter graph $G=(A \uplus B, E)$ enthält ein
Matching M der Kardinalität $|M|=|A|$ **gdw**
$\forall\, X \subseteq A \;:\; |X| \leq |N(X)|$


Philip Hall
(1904-1982)

Beweis: **Induktion über a = |A|**

Induktionsverankerung: **a = 1:** ✓

Induktionsschritt: Wir müssen zeigen:

Satz gilt für alle bipartiten Graphen mit $|A| \leq a-1$ $\Rightarrow$ *Satz gilt für alle bipartiten Graphen mit* $|A| = a$

“a-1 $\Rightarrow$ a“

Der Satz von Hall (Heiratssatz)

Philip Hall
(1904-1982)

Satz: (Hall, Heiratssatz)

Ein bipartiter graph $G=(A \uplus B, E)$ enthält ein Matching M der Kardinalität $|M|=|A|$ **gdw**

$$\forall X \subseteq A \;:\; |X| \leq |N(X)| \qquad (*)$$

Beweis: "a-1 $\Rightarrow$ a" Betrachte *beliebigen* Graphen mit $|A|=a$:

1.Fall: $\forall \emptyset \neq X \subsetneq A \;:\; |X| < |N(X)|$

- Wähle beliebige Kante $\{x,y\}$ und lösche x, y und alle inzidenten Kanten.
- Zeige dass der verbleibende Graph die Bedingung (*) erfüllt.

2.Fall: $\exists \emptyset \neq X_0 \subsetneq A \;:\; |X_0| = |N(X_0)|$

- Betrachte die beiden durch
  $X_0 \cup N(X_0)$ bzw
  $A \setminus X_0 \cup B \setminus N(X_0)$
  induzierten Graphen
- Zeige dass beide Graphen die Bedingung (*) erfüllen.

Der Satz von Hall (Heiratssatz)

Satz: (Hall, Heiratssatz)

Ein bipartiter Graph $G=(A \uplus B, E)$ enthält ein Matching M der Kardinalität $|M|=|A|$ **gdw**

$\forall\, X \subseteq A \ : \ |X| \leq |N(X)|$

Philip Hall
(1904-1982)

Korollar: (Frobenius)

Für alle k gilt: jeder k-reguläre bipartite Graph enthält ein perfektes Matching.

Ferdinand Georg Frobenius
(1849 – 1917)

Der Satz von Hall (Heiratssatz)

Theorem: (Hall, 1935)

Ein bipartiter graph $G=(A \uplus B, E)$ enthält ein Matching M der Kardinalität $|M|=|A|$ **gdw**

$$\forall A' \subseteq A \ : \ |A'| \leq |N(A')|$$

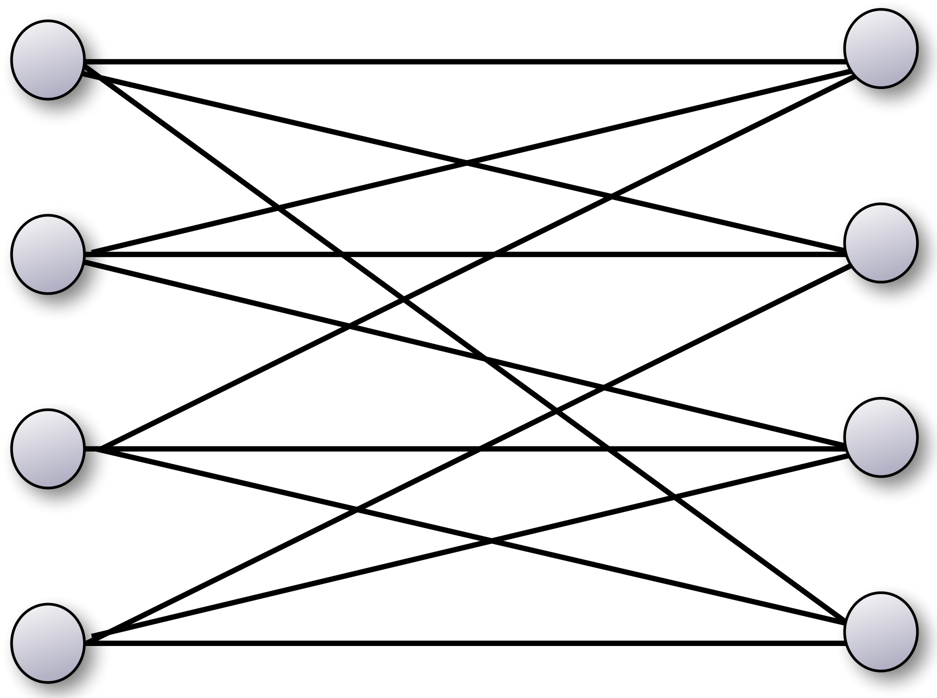
Philip Hall
(1904-1982)

Korollar: (Frobenius, 1917)

Für alle k gilt: jeder k-reguläre bipartite graph enthält ein perfektes Matching.

Es gilt sogar: Graph ist Vereinigung von perfekten Matchings.

Ferdinand Georg Frobenius
(1849 – 1917)



Der Satz von Hall (Heiratssatz)

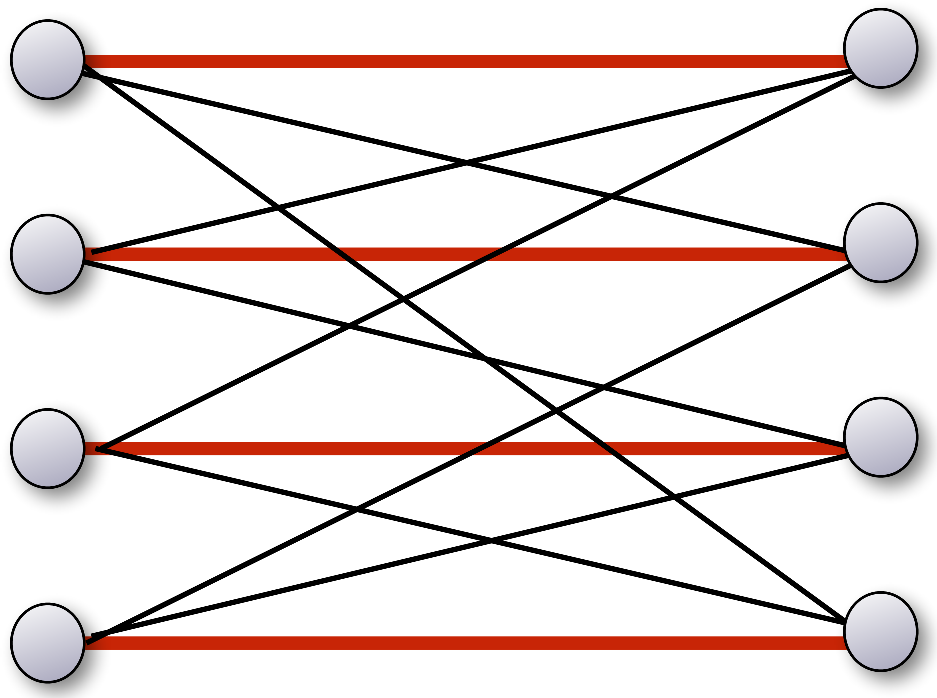
Theorem: (Hall, 1935)
Ein bipartiter graph $G=(A \uplus B, E)$ enthält ein Matching M der Kardinalität $|M|=|A|$ **gdw**
$\forall\, A' \subseteq A \;:\; |A'| \le |N(A')|$


Philip Hall
(1904-1982)

Korollar: (Frobenius, 1917)
Für alle k gilt: jeder k-reguläre bipartite graph enthält ein perfektes Matching.

Es gilt sogar: Graph ist Vereinigung von perfekten Matchings.


Ferdinand Georg Frobenius
(1849 – 1917)



Der Satz von Hall (Heiratssatz)

Theorem: (Hall, 1935)

Ein bipartiter graph $G=(A \uplus B, E)$ enthält ein Matching M der Kardinalität $|M|=|A|$ **gdw**

$\forall A' \subseteq A : |A'| \leq |N(A')|$

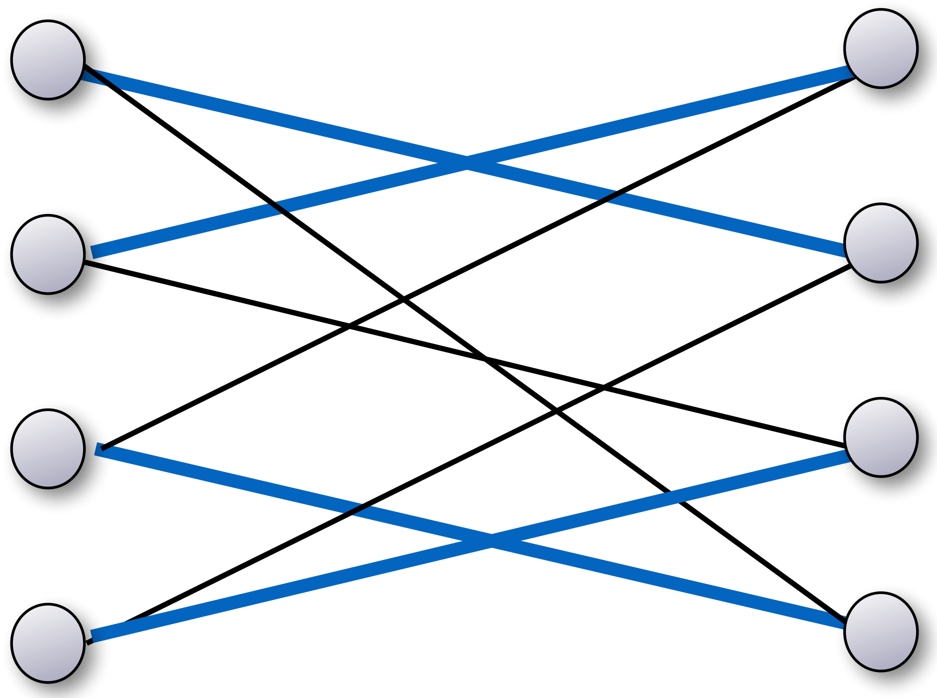
Philip Hall
(1904-1982)

Korollar: (Frobenius, 1917)

Für alle k gilt: jeder k-reguläre bipartite graph enthält ein perfektes Matching.

Es gilt sogar: Graph ist Vereinigung von perfekten Matchings.

Ferdinand Georg Frobenius
(1849 – 1917)



Der Satz von Hall (Heiratssatz)

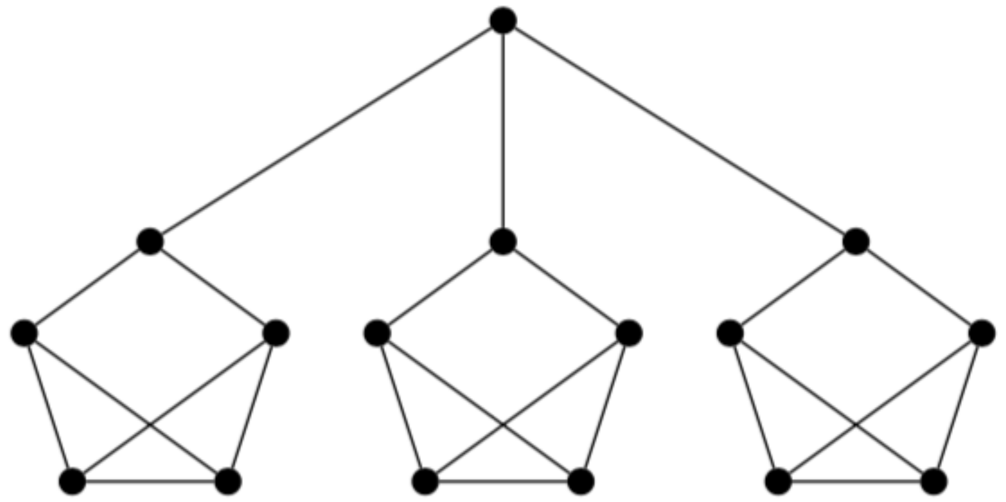
Satz: (Hall, Heiratssatz)
Ein bipartiter Graph $G=(A \uplus B, E)$ enthält ein Matching M der Kardinalität $|M|=|A|$ **gdw** $\forall X \subseteq A : |X| \le |N(X)|$

Philip Hall
(1904-1982)

Korollar: (Frobenius)
Für alle k gilt: jeder k-reguläre bipartite Graph enthält ein perfektes Matching.

Ferdinand Georg Frobenius
(1849 – 1917)

Beachte: bipartit ist wichtig!



Satz: In 2^k-regulären bipartiten Graphen kann man in Zeit $O(|E|)$ ein perfektes Matching bestimmen.

Man *kann* zeigen:

Satz: In k-regulären bipartiten Graphen kann man in Zeit $O(|E|)$ ein perfektes Matching bestimmen.

Augmentierende Pfade:

In bipartiten Graphen kann man in Zeit $O(|V|\,|E|)$ ein perfektes Matching bestimmen.

für bipartite Graphen

$O(|V|^{1/2} \cdot |E|)$ Hopcroft-Karp (ungewichtet)

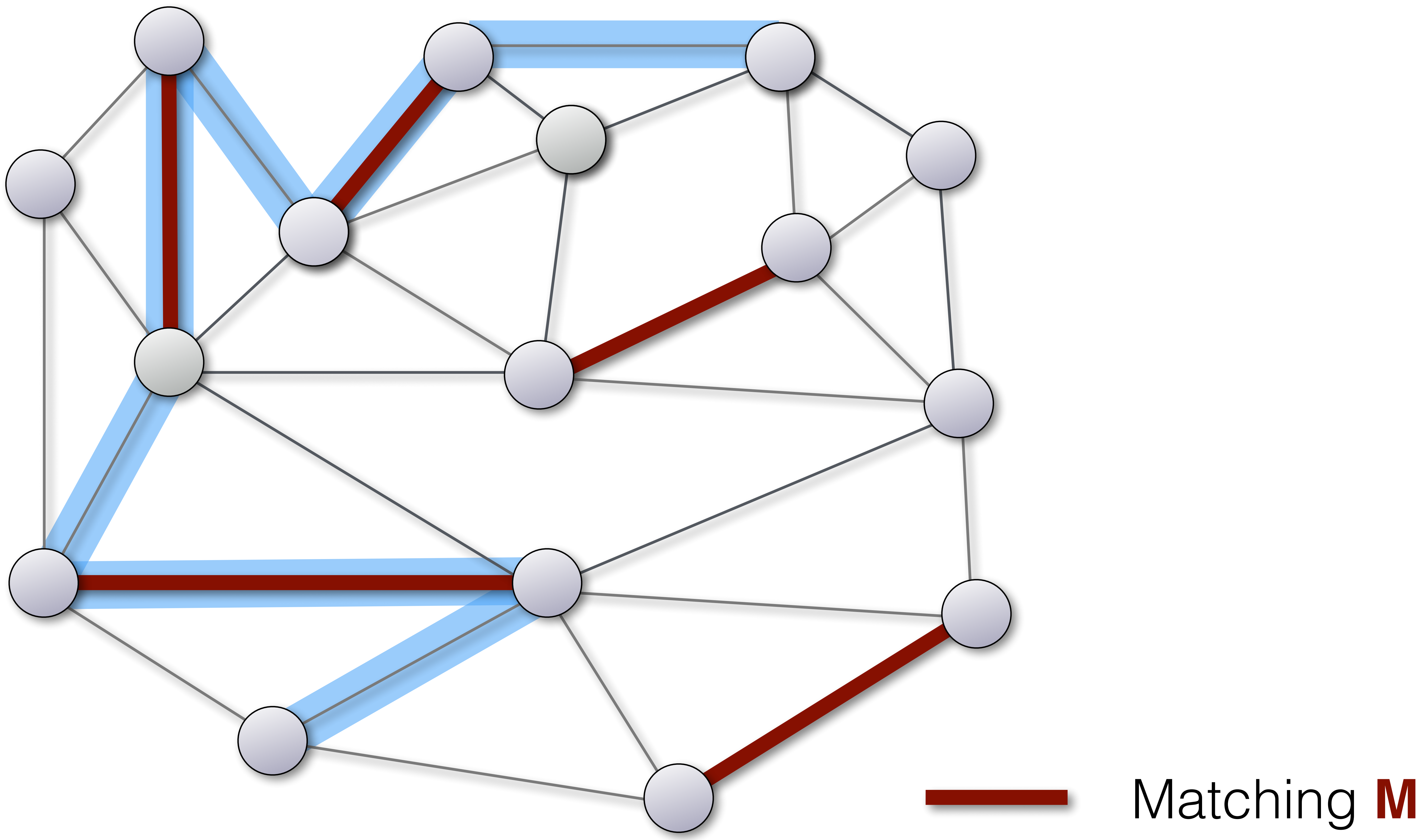
$O(|E|^{1+o(1)})$ (mit polynominellen Gewichte):

für allgemeine Graphen (mit polynominellen Gewichte)

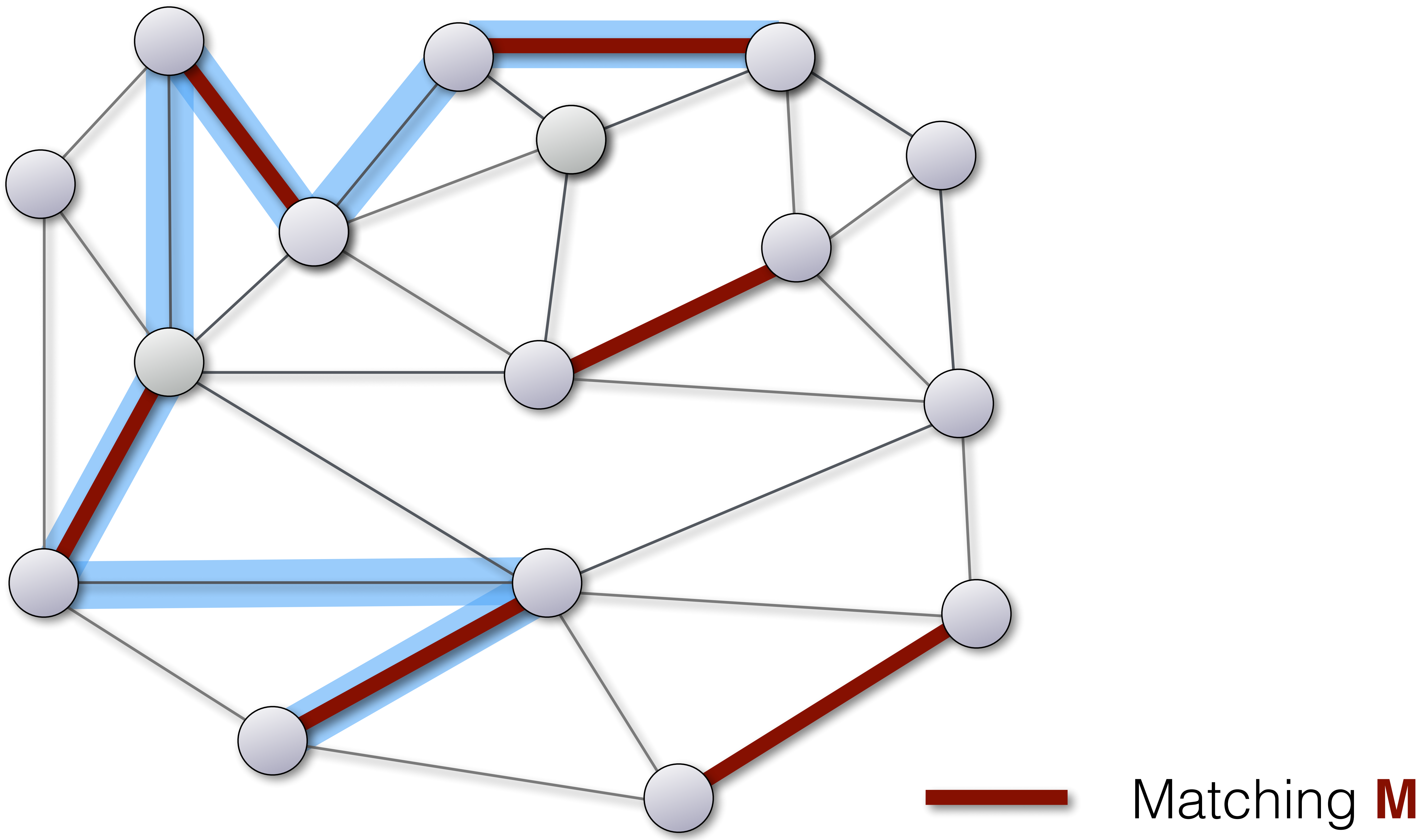
$O(|V|^{1/2} \cdot |E|)$ Micali-Vazirani (ungewichtet) / Gabow-Tarjan

$O(|V|^{2.373})$ mit Matrix-Multiplikation - Mucha, Sankowski (ungewichtet)

augmentierende Pfade



Ein **M-augmentierender Pfad P** ist ein Pfad, der abwechselnd Kanten aus **M** und nicht aus **M** enthält und der in von **M** nicht überdeckten Knoten beginnt und endet.

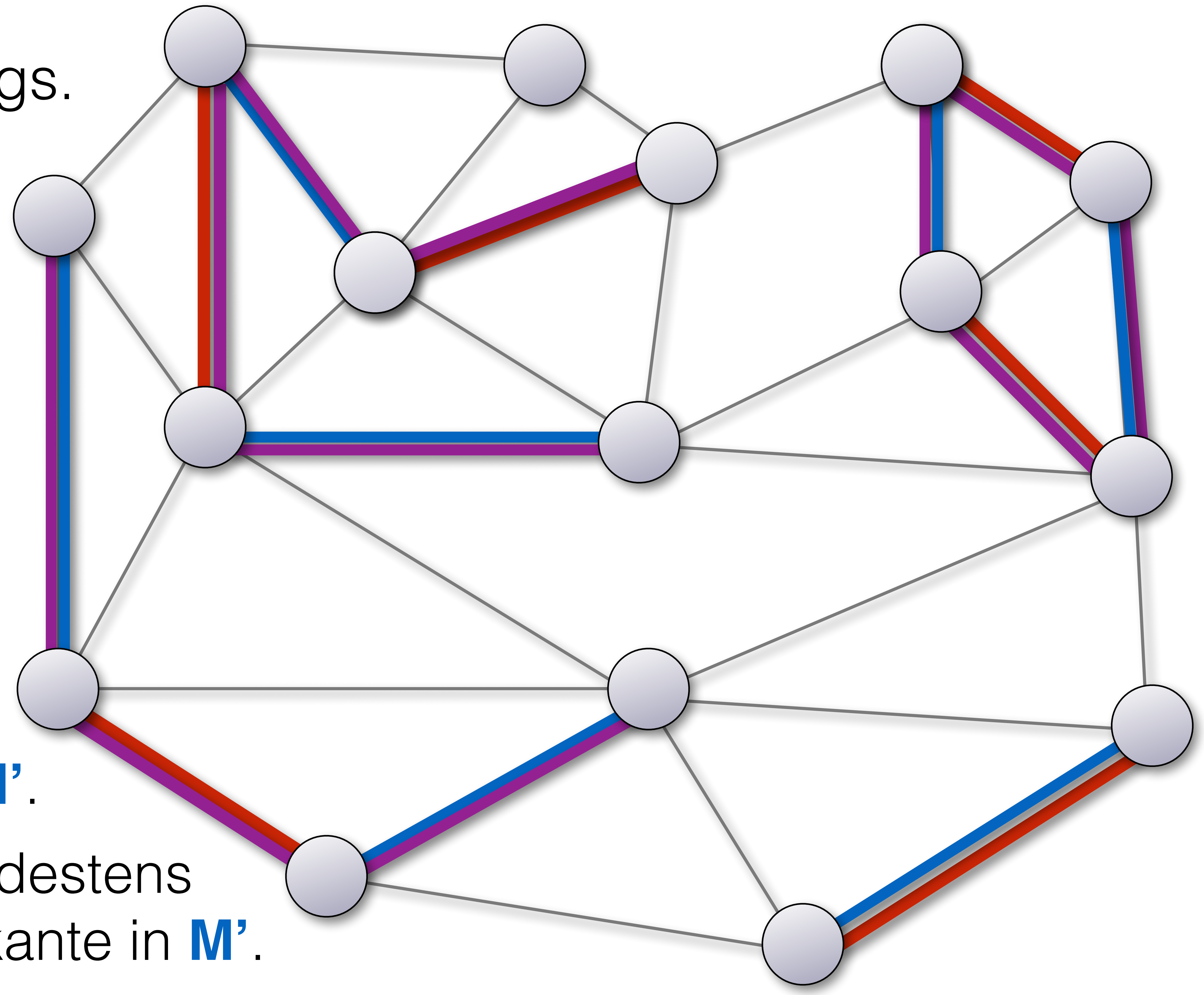


Ein **M-augmentierender Pfad P** ist ein Pfad, der abwechselnd Kanten aus **M** und nicht aus **M** enthält und der in von **M** nicht überdeckten Knoten beginnt und endet.

⇒ durch *Tauschen* entlang M können wir das Matching vergrössern:

M' := **M** ⊕ **P**

Seien **M**, **M’** beliebige Matchings.



Betrachte den Teilgraphen
mit Kantenmenge **M ⊕ M’**.

Beobachtungen:

- Jeder Knoten hat Grad ≤ 2.
- ⇒ Kollektion von Pfaden und Kreisen.
- Jeder Pfad/Kreis wechselt ab zwischen Kanten aus **M** und **M’**.
- Falls **|M|** < **|M’|**, so gibt es mindestens einen Pfad mit Start- und Endkante in **M’**.

Gedankenexperiment:

M nicht-maximales Matching (schon bekannt)
M’ maximales Matching (noch unbekannt)

Dann besitzt **M** einen augmentierenden Pfad!

augmentierende Pfade

Satz (Berge, 1957): Jedes Matching, das nicht (kardinalitäts-) maximal ist, besitzt einen augmentierenden Pfad.

Algorithmus

Input: Graph $G = (V, E)$
Output: maximales Matching M

Starte mit $M = \varnothing$.
repeat

- Suche augmentierenden Pfad P.
- **if** kein solcher Pfad existiert **then return** M.
- **else** $M := M \oplus P$.

Wie?

Suchen/Finden eines augmentierenden Pfades:

- in bipartiten Graphen in Zeit $O(|V|+|E|)$. Mit BFS, sehen wir gleich.
- in allgemeinen Graphen in Zeit $O(|V| \cdot |E|)$. Blossom-Algorithmus von Edmond, deutlich technischer, sehen wir nicht.

BFS für alternierende Pfade:
Input: bipartiter Graph G = (A⊎B,E), Matching M
Output: (kürzester) augmentierender Pfad,
falls solche Pfade existieren

L_0 := {unüberdeckten Knoten aus A}
Markiere Knoten aus L_0 als besucht.

for i = 1 **to** n
if i ungerade **then**
L_i := {unbesuchte Nachbarn von L_{i-1}
via Kanten in E \ M}
else (falls i gerade)
L_i := {unbesuchte Nachbarn von L_{i-1}
via Kanten in M}
Markiere Knoten aus L_i als besucht.

if ein Knoten v in L_i ist nicht überdeckt
then return Pfad zu v (backtracking)