

# Kapitel 3

## Matching in Graphen

Effiziente Algorithmen, SS 2018

Professor Dr. Petra Mutzel

Dipl.-Inform. Andre Droschinsky

VO 3/4 am 17./19. April 2018

## 3.1 Maximale Matchings in Graphen

### Definition (Matching)

Sei  $G = (V, E)$  ein ungerichteter Graph (ohne Schleifen).

Eine Kantenmenge  $M \subseteq E$  heißt **Matching** (oder **Paarung**)

falls gilt:

Für alle Paare  $e, e' \in M$  mit  $e = (u, v), e' = (u', v') \in M$  gilt  
 $\{u, v\} \cap \{u', v'\} = \emptyset$ .

Also: jeder Knoten darf zu höchstens einer  $M$ -Kante inzident sein.

Wir suchen ein **maximales Matching** (**Maximum Matching**, maximale Kardinalität).

# Bipartite Graphen

wichtige Graphenklasse

## Definition (Bipartiter Graph)

Sei  $G = (V, E)$  ein gerichteter oder ungerichteter Graph.

$G$  ist **bipartit**, wenn die Knotenmenge in zwei Mengen  $V_1$  und  $V_2$  geteilt werden kann, so dass alle Kanten zwischen  $V_1$  und  $V_2$  verlaufen, d.h. für alle  $e = (u, v)$  gilt:  
 $(u \in V_1 \text{ und } v \in V_2)$  oder  $(u \in V_2 \text{ und } v \in V_1)$ .

## Beobachtung

Es gilt:  $G$  ist bipartit  $\Leftrightarrow G$  enthält keine ungeraden Kreise (im ungerichteten Sinne).

# Anwendungen von Matching

Matching-Probleme fallen in die Klasse der Zuordnungsprobleme

Viele Varianten: perfektes Matching, maximales Matching, maximal gewichtetes Matching, ...

- Travelling Salesman Problem: Christofides-Heuristik
- Chinese Postman Problem (kürzester Zyklus in Graphen, der jede Kante mindestens einmal durchfährt, z.B. Briefträger, Müllabfuhr)
- Maximaler Schnitt in planaren Graphen (meine Diplomarbeit)
- Steganographie (Verstecken geheimer Informationen in Bildern), ...

# Anwendungen von Matching in bipartiten Graphen

## Klasse der Zuordnungsprobleme

- Zuordnung von Medizinstudierenden zu Krankenhäusern in den USA (Heiratsproblem, Prioritäten)
- Zuordnung von Studierenden zu Übungsgruppen (gewichtetes perfektes Matching)
- Zuordnung von Lehrveranstaltungen zu Räumen (maximales Matching)
- Preisfindung bei Auktionen (maximalen Gesamtgewinn)
- Satellitenkommunikation (Zeitschlitz-Zuordnungsproblem), ...

Voraussetzung: Graph ohne Schleifen und ohne Mehrfachkanten

# Zwei bahnbrechende Aufsätze - der erste 1965:

## PATHS, TREES, AND FLOWERS

JACK EDMONDS

**1. Introduction.** A *graph*  $G$  for purposes here is a finite set of elements called *vertices* and a finite set of elements called *edges* such that each edge *meets* exactly two vertices, called the *end-points* of the edge. An edge is said to *join* its end-points.

A *matching* in  $G$  is a subset of its edges such that no two meet the same vertex. We describe an efficient algorithm for finding in a given graph a matching of maximum cardinality. This problem was posed and partly solved by C. Berge; see Sections 3.7 and 3.8.

Maximum matching is an aspect of a topic, treated in books on graph theory, which has developed during the last 75 years through the work of about a dozen authors. In particular, W. T. Tutte (8) characterized graphs which do not contain a *perfect* matching, or *1-factor* as he calls it—that is a set of edges with exactly one member meeting each vertex. His theorem prompted attempts at finding an efficient construction for perfect matchings.

# Zwei bahnbrechende Aufsätze - der zweite:

JOURNAL OF RESEARCH of the National Bureau of Standards—B. Mathematics and Mathematical Physics  
Vol. 69B, Nos. 1 and 2, January–June 1965

## Maximum Matching and a Polyhedron With 0,1-Vertices<sup>1</sup>

Jack Edmonds

(December 1, 1964)

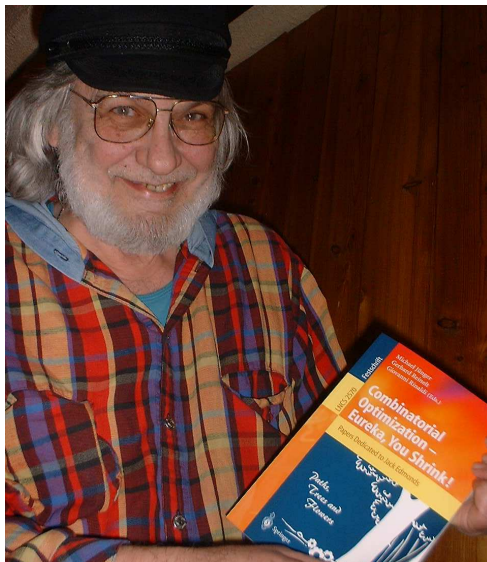
A matching in a graph  $G$  is a subset of edges in  $G$  such that no two meet the same node in  $G$ . The convex polyhedron  $C$  is characterized, where the extreme points of  $C$  correspond to the matchings in  $G$ . Where each edge of  $G$  carries a real numerical weight, an efficient algorithm is described for finding a matching in  $G$  with maximum weight-sum.

### Section 1

An algorithm is described for optimally pairing a finite set of objects. That is, given a real numerical weight for each unordered pair of objects in a set  $Y$ , to select a family of mutually disjoint pairs the sum of whose weights is maximum. The well-known optimum assignment problem [5]<sup>2</sup> is the special case where  $Y$  partitions into two sets  $A$  and  $B$  such that

inequalities. In particular, we prove a theorem analogous to one of G. Birkhoff [1] and J. von Neuman [5] which says that the extreme points of the convex set of doubly stochastic matrices (order  $n$  by  $n$ ) are the permutation matrices (order  $n$  by  $n$ ). That theorem and the Hungarian method are based on König's theorem about matchings in bipartite graphs. Our work is related to results on graphs due to Tutte [4].

# Aussois 2002: Jack Edmonds freut sich





## Köln 2004: Jack & Kathie mit Pauline & Paul



## 3.2 Matchings in bipartiten Graphen

Wie berechnet man maximale Matchings in bipartiten Graphen?

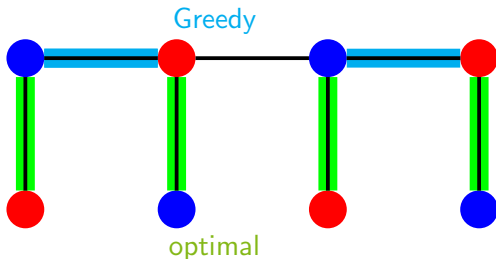
Idee Greedy

Starte mit  $\emptyset$  und füge so lange eine Kante hinzu,  
bis keine Kante mehr hinzufügbare ist.

### Algorithmus 3.4 (Greedy Matching-Algorithmus)

1.  $M := \emptyset$
2. If  $\exists e \in E: M \cup e$  ist Matching  
Then  $M := M \cup e$ ; Weiter bei 2.
3. Ausgabe  $M$

# Matchings in bipartiten Graphen



## Theorem 3.5

Sei  $G = (V, E)$  ein ungerichteter Graph,  $M_{\text{opt}}$  ein maximales Matching in  $G$ .

Der Greedyalgorithmus berechnet ein Matching  $M_{\text{greedy}}$  mit  $|M_{\text{greedy}}| \geq |M_{\text{opt}}| / 2$ .

Sogar für bipartite Graphen ist  $|M_{\text{greedy}}| = |M_{\text{opt}}| / 2$  möglich. ✓

# Beweis der Approximationsgüte

## Beweis.

$V_{\text{greedy}}$ : Menge der zu  $M_{\text{greedy}}$  inzidenten Knoten

klar  $|V_{\text{greedy}}| = 2 |M_{\text{greedy}}|$

Jede Kante  $e \in M_{\text{opt}}$  ist zu einem Knoten in  $V_{\text{greedy}}$  inzident ,  
sonst wäre  $e$  noch hinzu gewählt worden.

Aus der Disjunktheit der Matchingkanten folgt also:

also  $|M_{\text{opt}}| \leq |V_{\text{greedy}}| = 2 |M_{\text{greedy}}|$

also  $|M_{\text{greedy}}| \geq |M_{\text{opt}}| / 2$



Wir setzen ab jetzt voraus: Graph zusammenhängend

(also  $n - 1 \leq |E| \leq \binom{n}{2}$ )

sonst unabhängig auf Zusammenhangskomponenten

# Zentrale Begriffe

## Definition 3.6 (Matching-Begriffe)

$G = (V, E)$  ungerichteter Graph,  $M \subseteq E$  Matching auf  $G$ .

- $e \in M$  heißt **Matching-Kante** ( $M$ -Kante).
- $e \notin M$  heißt **freie Kante**.
- $v \in V$  inzident zu Matching-Kante  $e \in M$  heißt **besetzt**.
- $v \in V$  nicht besetzt heißt **frei**.

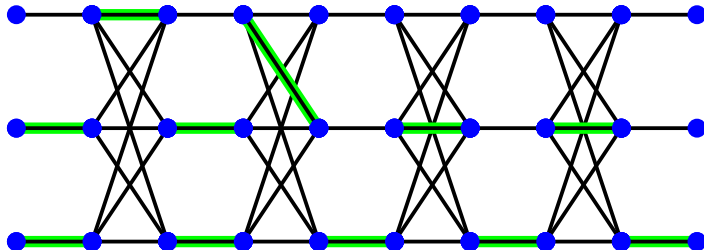
# Zentrale Begriffe

## Definition 3.6 ff (Alternierende und $M$ -verbessernde Pfade)

$G = (V, E)$  ungerichteter Graph,  $M \subseteq E$  Matching auf  $G$ .

- Ein **Pfad**  $P$  der Länge  $k$  ist eine Folge  $(v_0, e_1, v_1, e_2, \dots, e_k, v_k)$  von abwechselnd Knoten und Kanten aus  $G$  mit  $e_i = (v_{i-1}, v_i)$  für  $i = 1, \dots, k$ .
- Man schreibt auch:  $P = (v_0, v_1, \dots, v_k)$ .
- Ein **Weg** ist ein Pfad in dem alle Knoten verschieden sind.
- Ein nicht-leerer Pfad  $P = (v_1, v_2, \dots, v_k)$  mit  $k \geq 2$  bei dem sich  $M$ -Kanten und freie Kanten **abwechseln** heißt  **$M$ -alternierend**.
- Ein **kreisfreier**  $M$ -alternierender Pfad  $P = (v_1, \dots, v_k)$ ,  $k \geq 2$ , mit  $v_1$  und  $v_k$  frei heißt  **$M$ -verbessernd** (oder  **$M$ -augmentierend**).

# Beispiel zu Matching-Begriffen



Bemerkung: Ein  $M$ -verbessernder Pfad ist also ein (besonderer) Weg.

# $M$ -verbessernde Pfade

## Theorem 3.7 (Berge, 1957)

Sei  $G$  beliebiger Graph mit Matching  $M$ . Es gilt:  
 $M$  maximal  $\Leftrightarrow$  Es gibt keinen  $M$ -verbessernden Pfad.

### Beweis.

Wir beweisen

$M$  nicht maximal  $\Leftrightarrow$  Es gibt  $M$ -verbessernden Pfad.

„ $\Leftarrow$ “: Sei  $P = (v_1, \dots, v_k)$   $M$ -verbessernder Pfad.

**Beobachtung**  $k$  gerade (also  $k = 2j$  mit  $j \in \mathbb{N}$ )

Mache  $M$ -Kanten  $\{v_2, v_3\}, \{v_4, v_5\}, \dots, \{v_{2j-2}, v_{2j-1}\}$  zu freien Kanten.

Mache freie Kanten  $\{v_1, v_2\}, \{v_3, v_4\}, \dots, \{v_{2j-1}, v_{2j}\}$  zu  $M$ -Kanten.

**Beobachtung** Das ist möglich:  $M$  danach noch Matching.

**Beobachtung**  $M$  wächst dadurch um 1.



# Beweis der Gegenrichtung

„ $M$  nicht maximal  $\Rightarrow \exists M$ -verbessernden Pfad“

Sei  $M$  nicht maximal.

$\exists M': |M'| > |M|$

Betrachte  $M \oplus M'$

$= \{e \mid e \in M \wedge e \notin M'\} \cup \{e \mid e \notin M \wedge e \in M'\}$

**Beobachtung** in  $M \oplus M'$  alle Knotengrade  $\leq 2$

**also**  $M \oplus M'$  zerfällt in disjunkte Pfade und Kreise

**immer**  $M$ -Kante und  $M'$ -Kante abwechselnd

**also** alle Kreise haben gerade Länge

$|M'| > |M| \Rightarrow \exists$  Pfad  $P$  mit mehr  $M'$ - als  $M$ -Kanten

$P$  ist  $M$ -verbessernd



**Anwendung** einfacher Matching-Algorithmus

# Einfacher Matching-Algorithmus für bipartite Graphen

Ab jetzt:  $G = (V, E)$  bipartit

## Algorithmus

- ①  $M := \emptyset$
- ② Berechne  $M$ -verbessernden Pfad  $P$ .
- ③ If kein  $P$  gefunden, Then Exit mit Ausgabe  $M$ .
- ④  $M := M \oplus P$  (Augmentiere  $M$ )
- ⑤ Weiter bei 2.

klar terminiert, weil  $\leq |V|/2$  Iterationen

also Korrektheit klar ✓

Aber auch effizient?

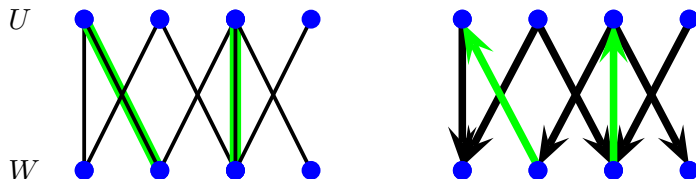
## $M$ -verbessernde Pfade finden

Sei  $G = (U \uplus W, E)$  bipartiter Graph,  $M$  Matching auf  $G$ .

Wir „richten“  $G$ :

$G_M = (U \uplus W, E_M)$  gerichteter Graph mit

- $(u, w) \in E_M$  für  $\{u, w\} \in E \setminus M$ ,  $u \in U$ ,  $w \in W$
- $(w, u) \in E_M$  für  $\{u, w\} \in E \cap M$ ,  $u \in U$ ,  $w \in W$



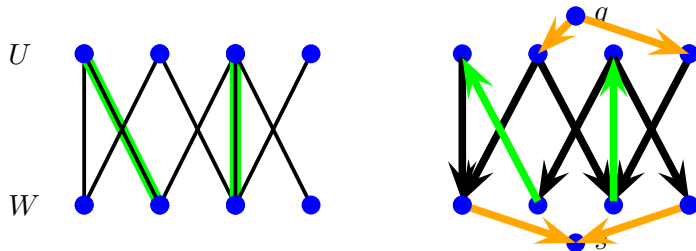
### Beobachtungen

- jeder gerichtete Weg in  $G_M$  ist  $M$ -alternierender Pfad in  $G$
- jeder gerichtete Weg in  $G_M$  von freiem  $U$ -Knoten zu freiem  $W$ -Knoten ist  $M$ -verbessernder Pfad in  $G$

## $M$ -verbessernde Pfade finden

$G_M = (U \uplus W, E_M)$  gerichteter Graph mit

- $(u, w) \in E_M$  für  $\{u, w\} \in E \setminus M$ ,  $u \in U$ ,  $w \in W$
- $(w, u) \in E_M$  für  $\{u, w\} \in E \cap M$ ,  $u \in U$ ,  $w \in W$



zusätzlich einfügen

- Knoten  $q, s$
- alle Kanten  $(q, u)$  mit  $u \in U$  frei
- alle Kanten  $(w, s)$  mit  $w \in W$  frei

Jeder gerichtete  $(q, s)$ -Weg ist  $M$ -verbessernder Pfad

# Ausformulierter Algorithmus

## Algorithmus 3.8 (einfacher Matching-Algorithmus)

1.  $M := \emptyset$
2. Konstruiere gerichteten Graphen  $G' = (U \uplus W, E')$  mit  
 $E' = \{(u, w) \mid u \in U, w \in W, \{u, w\} \notin M\}$   
 $\cup \{(w, u) \mid u \in U, w \in W, \{u, w\} \in M\}$  mit neuen Knoten  $q, s$   
 und allen Kanten  $(q, u)$  mit  $u \in U$  frei sowie allen Kanten  
 $(w, s)$  mit  $w \in W$  frei.
3. Suche mit Breitensuche einen gerichteten Weg von  $q$  nach  $s$ .  
 Sei  $P$  dieser Weg.
4. If  $P$  gefunden  
 Then  $M := M \oplus P$ ; Weiter bei 2.  
 Else Ausgabe  $M$ .

# Analyse zu Algorithmus 3.8

## Theorem 3.9

Algorithmus 3.8 berechnet in Zeit  $O(ne) = O(n^3)$  ein maximales Matching für einen bipartiten Graphen  $G = (U \uplus W, E)$  mit  $|U \uplus W| = n$  und  $|E| = e$ .

Beweis.

**Laufzeit**    Ein Breitensuche-Aufruf geht in  $O(e)$   
                   Augmentierung eines  $M$ -verbessernden Pfades:  $O(e)$   
                   Konstruktion/Update des gerichteten Graphen:  $O(e)$   
                   maximal  $\leq n/2$  Iterationen

**Korrektheit** alle gefundenen gerichteten Pfade „frei  $\rightsquigarrow$  frei“  
 $\Leftrightarrow M$ -verbessernde Pfade

**klar**        Breitensuche findet solche Pfade     $\square$

Geht es nicht schneller?

# Was bisher geschah. . .

## Matchings

- Greedy-Algorithmus mit 2-Approximation
- Einfacher Algorithmus mit Laufzeit  $O(n \cdot e) = O(n^3)$  für bipartite Graphen

Wunsch schnellerer Algorithmus

dazu Struktureinsichten

## Ideen zur Verbesserung

**bekannt** eine Graph-Traversierung geht in Zeit  $O(e)$

**bisher** eine Graph-Traversierung für Matchingverbesserung **um 1**

**Wunsch** Matchingverbesserungen „in größeren Sprüngen“

Wie geht das?

**klar** gleichzeitige „Addition“  
von  $k$  **knotendisjunkten**  $M$ -verbessernder Pfade  
ist **möglich** und verbessert **um  $k$**

**Idee** Finde in einer Graph-Traversierung möglichst viele  
knotendisjunkte  $M$ -verbessernde Pfade.

**intuitiv klar** kurze  $M$ -verbessernde Pfade eher günstig



# Algorithmus von Hopcroft und Karp

Wir zeigen zunächst eine Mindestanzahl an knotendisjunkten  $M$ -verbessernden Pfaden:

## Lemma:

Sei  $G = (V, E)$  beliebiger Graph, seien  $M, N \subseteq E$  Matchings auf  $G$ . Falls  $|N| > |M|$ , dann enthält  $(M \oplus N)$  mindestens  $|N| - |M|$  knotendisjunkte  $M$ -verbessernde Pfade.

## Beweis.

Betrachte  $M \oplus N$ . Mit den Beobachtungen aus Beweis zu Theorem 3.7 folgt: Jede Zusammenhangskomponente ist entweder (a) ein  $M$ -alternierender Kreis gerader Länge oder (b) ein  $M$ -alternierender Pfad. Jeder  $M$ -verbessernde Pfad enthält höchstens eine Kante aus  $N$  mehr als aus  $M$ .

$\Rightarrow$  Es muss mindestens  $s := |N| - |M|$  knotendisjunkte  $M$ -verbessernde Pfade geben.

# Algorithmus von Hopcroft und Karp

## Lemma 3.10: Eigenschaften kürzester Pfade

Sei  $G = (V, E)$  beliebiger Graph,  $M \subseteq E$  Matching auf  $G$ ,  
 $P$  ein kürzester  $M$ -verbessernder Pfad,  
 $P'$  ein  $(M \oplus P)$ -verbessernder Pfad.  
 Dann gilt:  $|P'| \geq |P| + |P \cap P'|$ .

Erinnerung:  $M \oplus P = \{e \mid e \in M \wedge e \notin P\} \cup \{e \mid e \notin M \wedge e \in P\}$

Beweis.

Betrachte  $N := (M \oplus P) \oplus P'$  (neues Matching)

klar  $|N| = |M| + 2$

Beobachte  $M \oplus N = M \oplus ((M \oplus P) \oplus P') = P \oplus P'$

Aus vorigem Lemma folgt:  $M \oplus N$  enthält zwei knotendisjunkte  
 $M$ -verbessernde Pfade  $P_1, P_2$

# Beweis von Lemma 3.10

Wir haben Matching  $M$ ,  
 $P$  ein kürzester  $M$ -verbessernder Pfad  
 $P'$  ein  $(M \oplus P)$ -verbessernder Pfad  
 $N = (M \oplus P) \oplus P'$   
 $M \oplus N = P \oplus P'$  enthält zwei knotendisjunkte  
 $M$ -verbessernde Pfade  $P_1, P_2$   
 $|M \oplus N| = |P \oplus P'| \geq |P_1| + |P_2|$

klar  $|P| \leq |P_1|$  und  $|P| \leq |P_2|$

also  $|P \oplus P'| \geq |P_1| + |P_2| \geq 2|P|$

Beobachtung  $|P \oplus P'| = |P| + |P'| - 2|P \cap P'|$   
 also  $|P| + |P'| - |P \cap P'| \geq |P \oplus P'|$

zusammen  $|P| + |P'| - |P \cap P'| \geq 2|P|$

also  $|P'| \geq |P| + |P \cap P'|$



Wir haben also gezeigt:

### Lemma 3.10: Eigenschaften kürzester Pfade

Sei  $G = (V, E)$  beliebiger Graph,  $M \subseteq E$  Matching auf  $G$ ,  
 $P$  ein kürzester  $M$ -verbessernder Pfad,  
 $P'$  ein  $(M \oplus P)$ -verbessernder Pfad.  
 Dann  $|P'| \geq |P| + |P \cap P'|$

Daraus:

### Beobachtung für kürzeste Pfade

Sei  $G = (V, E)$  beliebiger Graph,  $M \subseteq E$  Matching auf  $G$ ,  
 $P$  ein kürzester  $M$ -verbessernder Pfad,  
 $P'$  ein  $(M \oplus P)$ -verbessernder Pfad.  
 Falls  $|P'| = |P|$ , dann sind die beiden Pfade kantendisjunkt.

# Eine Folge von Matchings

## Lemma 3.11

Sei  $G = (V, E)$  ungerichteter Graph,  $M_0 := \emptyset$ , für  $i \in \mathbb{N}_0$  sei  $M_{i+1} := M_i \oplus P_i$ , dabei  $P_i$  ein kürzester  $M_i$ -verbessernder Pfad. Für alle  $i, j \in \mathbb{N}_0$  gilt:

1.  $|P_i| \leq |P_{i+1}|$
2.  $|P_i| = |P_j|$  und  $i \neq j \Rightarrow P_i$  und  $P_j$  sind knotendisjunkt

Beweis.

1. **Beobachtung** folgt direkt aus Lemma 3.10 ✓
2. **durch Widerspruch** (Achtung: i.A. ist nicht  $j = i + 1$ )

**Annahme**  $|P_i| = |P_j|$  mit  $i \neq j$   
 und  $P_i$  und  $P_j$  **nicht** knotendisjunkt

## Beweis von Lemma 3.11

**Annahme**  $|P_i| = |P_j|$  mit  $i \neq j$  und  $P_i$  und  $P_j$  **nicht** knotendisjunkt

**Beobachtung** Alle  $P_h$  mit  $i \leq h \leq j$  haben  $|P_h| = |P_i| = |P_j|$

**Wähle**  $P_k$  und  $P_l$  mit  $i \leq k < l \leq j$  so, dass  
 $P_k$  und  $P_l$  **nicht** knotendisjunkt  
 und alle  $P_m$  mit  $k < m < l$  knotendisjunkt zu  $P_k$  und  $P_l$

**Geht das?** im Zweifel  $k = l - 1$  und Forderung über  $P_m$  leer

**Wir haben**  $(M_k \oplus P_k)$ -verbessernden Pfad  $P_l$   
 (da alle  $P_m$  mit  $k < m < l$  knotendisjunkt zu  $P_k$  und  $P_l$ )

**weil nicht knotendisjunkt**  $\exists v$  in  $P_k$  und  $P_l$

**Beobachtung**  $\exists e$  zu  $v$  inzidente Kante aus  $M_k \oplus P_k$   
 $e$  ist in  $P_k$  und  $P_l$  (sonst  $P_l$  nicht  $(M_k \oplus P_k)$ -verbessernd)  
**also**  $|P_k \cap P_l| \geq 1$

**also**  $|P_l| \geq |P_k| + |P_k \cap P_l| > |P_k|$  (Lemma 3.10) **Widerspruch**  $\square$

## Folgerung aus Lemma 3.11

Was haben wir gerade bewiesen?

### Lemma 3.11

Sei  $G = (V, E)$  ungerichteter Graph,  $M_0 := \emptyset$ , für  $i \in \mathbb{N}_0$  sei  $M_{i+1} := M_i \oplus P_i$ , dabei  $P_i$  ein kürzester  $M_i$ -verbessernder Pfad. Für alle  $i, j \in \mathbb{N}_0$  gilt:

1.  $|P_i| \leq |P_{i+1}|$
2.  $|P_i| = |P_j|$  und  $i \neq j \Rightarrow P_i$  und  $P_j$  sind **knotendisjunkt**

**Folgerung:** mehrere kürzeste  $M$ -verbessernde Pfade  
gut parallel „addierbar“

# Der Algorithmus von Hopcroft und Karp (1971)

## Algorithmus 3.12 (Hopcroft und Karp)

1.  $M := \emptyset$
2. Berechne eine maximale Menge kürzester knotendisjunkter  $M$ -verbessernder Pfade  $P_1, P_2, \dots, P_k$ .
3. If  $k \geq 1$   
 Then  $M := M \oplus P_1 \oplus P_2 \oplus \dots \oplus P_k$ . Weiter bei 2.  
 Else Ausgabe  $M$ .

## Theorem 3.13

Algorithmus 3.12 berechnet für einen bipartiten Graphen  $G = (U \uplus W, E)$  mit  $|U \uplus W| = n$  und  $|E| = e$  ein maximales Matching in Zeit  $O(\sqrt{n} \cdot e) = O(n^{5/2})$ .



## Auf dem Weg zum Beweis von Theorem 3.13

**Korrektheit** nach Vorüberlegungen offensichtlich ✓

**Wir werden zeigen**

1. Jede Runde ist in Zeit  $O(e)$  durchführbar.
2. Es gibt  $O(\sqrt{n})$  Runden.

**dazu hilfreich** obere Schranke für  
Länge kürzester  $M$ -verbessernder Pfade

**Warum?**

**Wir wissen** Länge kürzester  $M$ -verbessernder Pfade  
wächst in jeder Runde (da gleich lange Pfade  
knotendisjunkt sind)

**also** kürzeste Pfade nicht lang  $\Rightarrow$  nicht viele Phasen

# $M$ -verbessernder Pfade: Anzahl und Länge

- Betrachte** Matching  $M$  und maximales Matching  $M_{\text{opt}}$   
 $(|M| \leq |M_{\text{opt}}|)$
- Betrachte**  $M \oplus M_{\text{opt}}$  **Zusammenhangskomponenten**  
 davon seien  $C_i = (V_i, E_i)$  ( $i \in \{1, 2, \dots\}$ )
- Erinnerung** alle  $C_i$  jeweils Kreise gerader Länge  
 oder einfache Pfade  
 $\Rightarrow$  es gibt  $\geq |M_{\text{opt}}| - |M|$  knotendisjunkte  
 $M$ -verbessernde Pfade

## $M$ -verbessernde Pfade: Anzahl und Länge

haben  $\geq |M_{\text{opt}}| - |M|$  kontendisjunkte  $M$ -verbessernde Pfade

darin  $\leq |M|$   $M$ -Kanten

Schubfachprinzip  $\exists$   $M$ -verbessernder Pfad

mit  $\leq \left\lfloor \frac{|M|}{|M_{\text{opt}}| - |M|} \right\rfloor$   $M$ -Kanten

klar  $M$ -Kanten und  $M_{\text{opt}}$ -Kanten alternieren

also Pfadlänge  $\leq 2 \left\lfloor \frac{|M|}{|M_{\text{opt}}| - |M|} \right\rfloor + 1$

Beobachtung kürzeste  $M$ -verbessernde Pfade **kurz**,  
wenn  $|M_{\text{opt}}| - |M|$  **groß**

Idee Ausnutzen zur Fallunterscheidung

1. Fall  $|M_{\text{opt}}| - |M|$  groß  $\rightsquigarrow$  nur kurze Pfade

2. Fall  $|M_{\text{opt}}| - |M|$  klein  $\rightsquigarrow$  nur wenige Runden

# Anzahl der Runden des Hopcroft-Karp-Algorithmus

Definiere zwei Phasen

Phase 1  $0 \leq |M| \leq \lfloor |M_{\text{opt}}| - \sqrt{|M_{\text{opt}}|} \rfloor$

Phase 2  $\lfloor |M_{\text{opt}}| - \sqrt{|M_{\text{opt}}|} \rfloor < |M| < |M_{\text{opt}}|$

Aber wir kennen  $|M_{\text{opt}}|$  doch gar nicht!

klar  $|M_{\text{opt}}|$  existiert, also wohldefiniert  
dient nur der Analyse, Algorithmus bleibt unverändert

zunächst Phase 2

klar  $|M_{\text{opt}}| \leq n/2$  (Def. Matching)

also  $\sqrt{|M_{\text{opt}}|} = O(\sqrt{n})$

also in Phase 2 nur  $O(\sqrt{n})$  Runden ✓

# Länge kürzester $M$ -verbessernder Pfade in Phase 1

Phase 1  $0 \leq |M| \leq \lfloor |M_{\text{opt}}| - \sqrt{|M_{\text{opt}}|} \rfloor$

Erinnerung Pfadlänge kürzester  $M$ -verbessernder Pfade  
 $\leq 2 \left\lfloor \frac{|M|}{|M_{\text{opt}}| - |M|} \right\rfloor + 1$

$$\begin{aligned}
 & 2 \left\lfloor \frac{|M|}{|M_{\text{opt}}| - |M|} \right\rfloor + 1 \leq 2 \left\lfloor \frac{\lfloor |M_{\text{opt}}| - \sqrt{|M_{\text{opt}}|} \rfloor}{|M_{\text{opt}}| - \lfloor |M_{\text{opt}}| - \sqrt{|M_{\text{opt}}|} \rfloor} \right\rfloor + 1 \\
 &= 2 \left\lfloor \frac{|M_{\text{opt}}| - \lceil \sqrt{|M_{\text{opt}}|} \rceil}{\lceil \sqrt{|M_{\text{opt}}|} \rceil} \right\rfloor + 1 \\
 &= 2 \left\lfloor \frac{|M_{\text{opt}}|}{\lceil \sqrt{|M_{\text{opt}}|} \rceil} - 1 \right\rfloor + 1 \\
 &\leq 2\sqrt{|M_{\text{opt}}|} + 1
 \end{aligned}$$

# Anzahl der Runden

- Wir haben
- $O(\sqrt{|M_{\text{opt}}|}) = O(\sqrt{n})$  Runden in Phase 2
  - Pfadlänge kürzester  $M$ -verbessernder Pfade  
 $= O(\sqrt{|M_{\text{opt}}|}) = O(\sqrt{n})$  in Phase 1

Erinnerung    Pfadlänge kürzester  $M$ -verbessernder Pfade  
 wächst um  $\geq 1$  in **jeder** Runde

also    nach  $O(\sqrt{|M_{\text{opt}}|}) = O(\sqrt{n})$  Runden ist Phase 1 beendet

also    insgesamt  $O(\sqrt{|M_{\text{opt}}|}) = O(\sqrt{n})$  Runden

wollen zeigen    Zeit  $O(\sqrt{n} \cdot e)$  insgesamt

jetzt genügt zu zeigen    Zeit  $O(e)$  je Runde

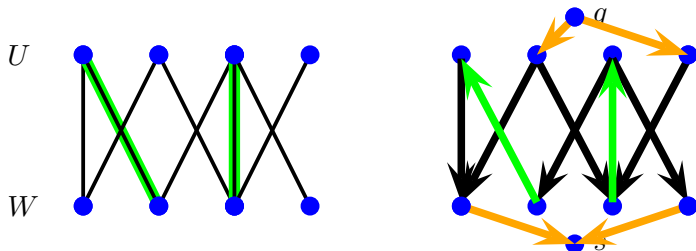
# Vorbereitung effiziente Durchführung einer Runde

Erinnerung  $G$  bipartit

Erinnerung Algorithmus 3.8 richten bipartiter Graphen

$G_M = (U \cup W, E_M)$  gerichteter Graph mit

- $(u, w) \in E_M$  für  $\{u, w\} \in E \setminus M$ ,  $u \in U$ ,  $w \in W$
- $(w, u) \in E_M$  für  $\{u, w\} \in E \cap M$ ,  $u \in U$ ,  $w \in W$



zusätzlich einfügen

- Knoten  $q, s$
- alle Kanten  $(q, u)$  mit  $u \in U$  frei
- alle Kanten  $(w, s)$  mit  $w \in W$  frei

# Effiziente Implementierung einer Runde

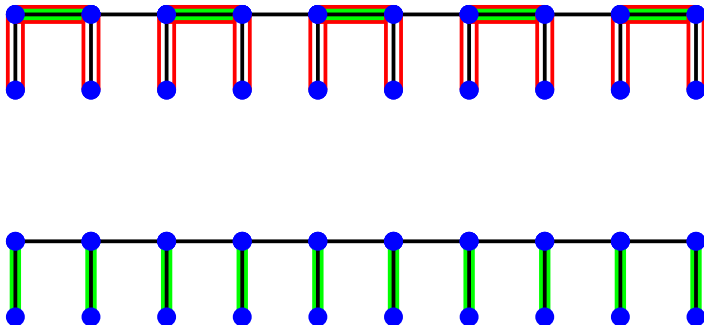
1. Berechne gerichteten Graphen mit Zusatzknoten  $q, s$  und Zusatzkanten  $\{(q, u) \mid u \in U \text{ frei}\}, \{(w, s) \mid w \in W \text{ frei}\}$ .
2. In einer Breitensuche, berechne die kürzesten Distanzen ( $\text{dist}()$ )-Werte bzgl.  $q$ ) der erreichten Knoten bis zum ersten Mal  $s$  erreicht wird; dann Stop BFS
3. In einer Tiefensuche, die nur Kanten  $(u, v)$  benutzt mit  $\text{dist}(v) - \text{dist}(u) = 1$ , extrahiere alle kürzesten  $q$ - $s$ -Wege, nach jedem extrahierten Weg  $P$  markiere Knoten auf  $P$  als nicht mehr benutzbar
4. Verwende alle gefundenen kürzesten  $q$ - $s$ -Wege als  $M$ -verbessernde Pfade.

klar jeder Schritt in Zeit  $O(n + e) = O(e)$  durchführbar

also insgesamt Zeit  $O(\sqrt{|M_{\text{opt}}|} \cdot e) = O(\sqrt{n} \cdot e) = O(n^{5/2})$



## Ein kleines Beispiel



## 3.2 Matchings in allgemeinen Graphen

wir haben maximale Matchings in bipartiten Graphen  
in Zeit  $O(n^{5/2})$

klar wollen maximale Matchings in allgemeinen Graphen

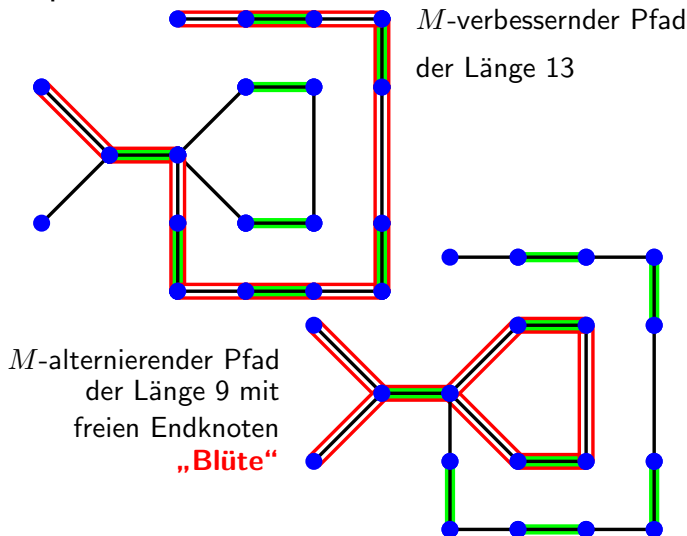
Geht das nicht genau so?

klar Algorithmus von Hopcroft und Karp nicht direkt übertragbar  
weil keine Mengen  $U \uplus W = V$  identifizierbar

Ist ein einfacher Matching-Algorithmus direkt übertragbar?

1.  $M := \emptyset$
2. Finde  $M$ -verbessernden Pfad  $P$ .
3. If  $P$  gefunden, Then  $M := M \oplus P$ ; Weiter bei 2.
4. Ausgabe  $M$

## Ein Beispiel



# Vom Umgang mit „Blüten“

**Beobachtung** Kreise ungerader Länge können ein Problem sein

**Anmerkungen** Kürzeste alternierende Pfade  
mit 2 freien Endknoten sind in allgemeinen Graphen  
nicht automatisch  $M$ -verbessernd;  
In bipartiten Graphen hingegen schon,  
denn alle Kreise sind gerade

Können wir gefundene Kreise nicht einfach ignorieren?

**Erinnerung** Graphen können exponentiell viele Kreise haben

also total naives Vorgehen geht nicht

# Idee von Jack Edmonds (1965): „Heureka, you shrink“

Kontraktion der ungeraden Kreise („Blüten“) zu einem Knoten

Achtung: wiederholte Kontraktion führt zu „Blütenhierarchie“

Diese Idee führte zu dem ersten polynomiellen  
Matching-Algorithmus in allgemeinen Graphen

Laufzeit jedoch relativ hoch:  $O(n^4)$

# Idee von Micali und Vazirani (1980)

**Erinnerung** Algorithmus von Hopcroft und Karp (Algorithmus 3.12) löst das Problem in Zeit  $O(n^{5/2})$  für bipartite Graphen

Was können wir beibehalten? – Was müssen wir ändern?

Was ist mit

Theorem 3.7:  $M$  maximal  $\leftrightarrow \nexists M$ -verbessernder Pfad?

**Beobachtung** gilt in beliebigen Graphen ✓

Was ist mit

Lemma 3.10/3.11: kürzeste  $M$ -verbessernde Pfade wachsen, kürzeste  $M$ -verbessernde Pfade gleicher Länge sind disjunkt?

**Beobachtung** gilt in beliebigen Graphen ✓

# Hopcroft/Karp auf allgemeinen Graphen

wie gesehen    geht nicht

Aber vielleicht funktionieren Teile?

Einsicht    Grundgerüst funktioniert ✓

1.  $M := \emptyset$
2. Berechne eine maximale Menge kürzester knotendisjunkter  $M$ -verbessernder Pfade  $P_1, P_2, \dots, P_k$ .
3. If  $k \geq 1$   
Then  $M := M \oplus P_1 \oplus P_2 \oplus \dots \oplus P_k$ . Weiter bei 2.  
Else Ausgabe  $M$ .

Was ist mit der Anzahl der Runden?

Einsicht    Beweis funktioniert unverändert  
 $O(\sqrt{M_{\text{opt}}}) = O(\sqrt{n})$  Runden

also    nur effiziente Implementierung von Schritt 2 offen

# Ein Matching-Algorithmus für allgemeine Graphen

**Bemerkung** bei sorgfältiger Implementierung  
 $\rightsquigarrow$  Laufzeit  $O(e)$  je Runde

... aber wie das genau funktioniert gehört nicht zum Stoff

**Resultat**

## Theorem 3.16

Der Algorithmus von Micali und Vazirani berechnet in Zeit  $O(\sqrt{ne})$  ein maximales Matching in einem beliebigen Graphen  $G = (V, E)$  mit  $|V| = n$  und  $|E| = e$ .



# Bemerkungen zum Bipartiten Matching

- Der Algorithmus von Hopcroft-Karp ist bis heute für dünne Graphen der theoretisch beste.
- Alt, Blum, Mehlhorn und Paul stellten 1991 einen auf Netzwerkflüssen basierenden Algorithmus vor, der für dichte Graphen etwas besser ist (Laufzeit:  $O(n^{1.5} \sqrt{e/\log n})$ ).
- Experimentelle Vergleiche (teilweise veraltet) widersprechen sich teilweise bezüglich der praktisch besten Varianten.

⇒ **Mögliche Bachelorarbeiten:** Experimentelle Vergleiche verschiedener Varianten

## Literaturhinweise für Interessierte

- **Originalartikel:** John E. Hopcroft, Richard M. Karp: "An  $n^{5/2}$  Algorithm for Maximum Matchings in Bipartite Graphs", SIAM Journal on Computing 2 (4): 225–231, 1973
- **Originalartikel:** Helmut Alt, Norbert Blum, Kurt Mehlhorn, Markus Paul: "Computing a maximum cardinality matching in a bipartite graph in time", Information Processing Letters 37 (4): 237–240, 1991
- **Experimentelle Vergleiche:** Kurt Mehlhorn: "The Engineering of Some Bipartite Matching Programs", LNCS 1741, Springer, 1–3, 1999
- Boris V. Cherkassky, Andrew V. Goldberg, Paul Martin, J.C. Setubal, Joao C. Setubal, Jorge Stolfi: "Augment or push: A computational study of bipartite matching and unit capacity flow algorithms", ACM J. Exp. Algorithmics, vol. 3 (8), 1998