

Vorlesung

# Effiziente Algorithmen und Komplexitätstheorie

Sommersemester 2008

Ingo Wegener



# Übungen — Organisatorisches

Regulär	montags 14–16	OH 16, 205	Wim oder Robin
	dienstags 10–12	OH 16, E07	Robin oder Wim
	mittwochs 10–12	OH 16, 205	Alexander Munteanu
	mittwochs 12–14	OH 14, 304	Alexander Munteanu

Freie Gruppenwahl.

Es wäre schön, wenn die Mittwochstermine stärker genutzt würden.

Fragegruppen	montags 16–18	OH 16, 212	Wim Martens
	mittwochs 16–18	OH 14, 305	Robin Nunkesser

# Was bisher geschah...

starke Zusammenhangskomponenten ✓

## Matchings

- Begriffe
- Greedy-Algorithmus, 2-Approximation
- $M$ -Verbessernde Pfade
- 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

# Über kürzeste $M$ -verbessernde Pfade

## Lemma 3.10

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.

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

**Beweis.**

**Betrachte**  $N := (M \oplus P) \oplus P'$ .

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

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

**Wir haben**  $M \oplus N$  enthält zwei knotendisjunkte  
 $M$ -verbessernde Pfade  $P_1, P_2$

**also**  $|M \oplus N| = |P \oplus P'| \geq |P_1| + |P_2|$

## 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 Pfad  $P_1, P_2$   
 $|M \oplus N| = |P \oplus P'| \geq |P_1| + |P_2|$

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

also  $|P \oplus P'| \geq 2|P|$

Beobachtung  $|P \oplus P'| = |P| + |P'| - |P \cap P'|$

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

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



# 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

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

**Betrachte**  $P_h$  mit  $i \leq h \leq j$  (also  $|P_h| = |P_i| = |P_j|$ )

**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 überhaupt?**

**klar** im Zweifel  $k = l - 1$  und Forderung über  $P_m$  leer

**Wir haben**  $(M_k \oplus P_k)$ -verbessernden Pfad  $P_l$

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

**Beobachtung** zu  $v$  inzidente Kante aus  $M_k \oplus P_k$   
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**  $\square$

# Der Algorithmus von Hopcroft und Karp (1971)

**Folgerung** aus Lemma 3.11    mehrere kürzeste  $M$ -verbessernde Pfade gut parallel „addierbar“

## 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  $G = (U \cup W, E)$  mit  $|U \cup W| = n$  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

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

## Länge kürzester $M$ -verbessernder Pfade

**Betrachte** Matching  $M$  und maximales Matching  $M_{\text{opt}}$   
( $|M| < |M_{\text{opt}}|$ )

**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

**Beobachtung**  $\forall i: \delta(C_i) := |E_i \cap M_{\text{opt}}| - |E_i \cap M| \in \{-1, 0, 1\}$

**klar**  $C_i$   $M$ -verbessernd  $\Leftrightarrow \delta(C_i) = 1$

**Beobachtung**  $\sum_i \delta(C_i) = |M_{\text{opt}}| - |M|$

**also**  $\geq |M_{\text{opt}}| - |M|$  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  $|M_{\text{opt}}| \leq n/2$

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     $O(\sqrt{|M_{\text{opt}}|}) = O(\sqrt{n})$  Runden in Phase 1

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

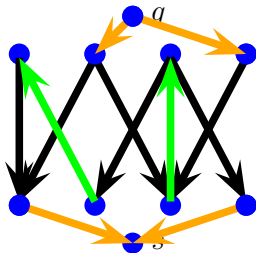
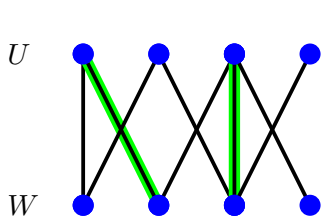
# Vorbereitung effiziente Durchführung einer Runde

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 Knoten  $(w, s)$  mit  $w \in W$  frei

# Effiziente Implementierung einer Runde

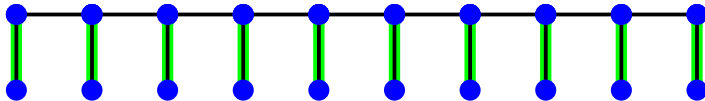
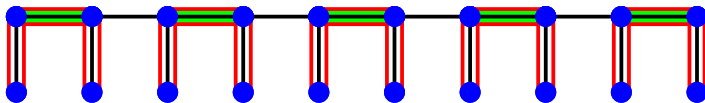
1. Berechne gerichteten Graphen mit Zusatzknoten  $q, s$  und Zusatzkanten  $\{(q, u) \mid u \in U \text{ frei}\}, \{(w, s) \mid w \in W \text{ frei}\}$ .
2. In einer Breitensuche, streiche alle Kanten, die nicht auf kürzesten  $q$ - $s$ -Wegen liegen.
3. In einer Breitensuche, finde und streiche alle kürzesten  $q$ - $s$ -Wege.
4. Verwende gefundene  $q$ - $s$ -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



# Matchings in allgemeinen Graphen

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

klar wollen maximale Matchings in allgemeinen Graphen

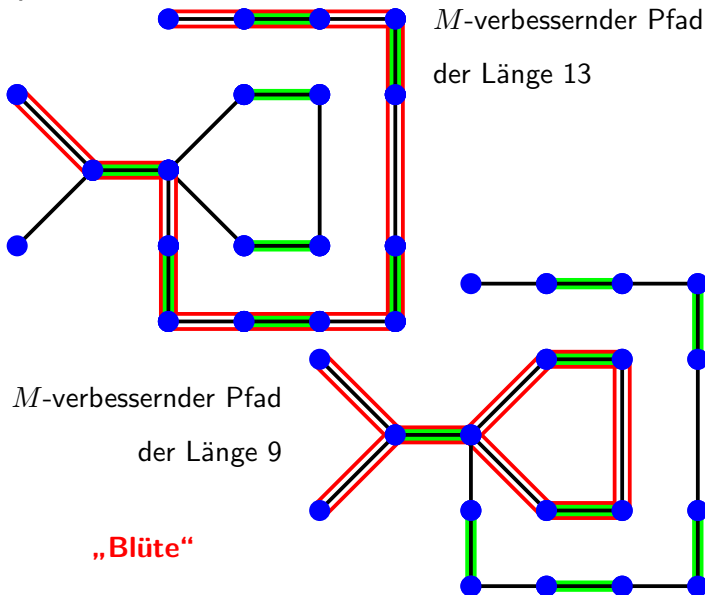
Geht das nicht genau so?

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

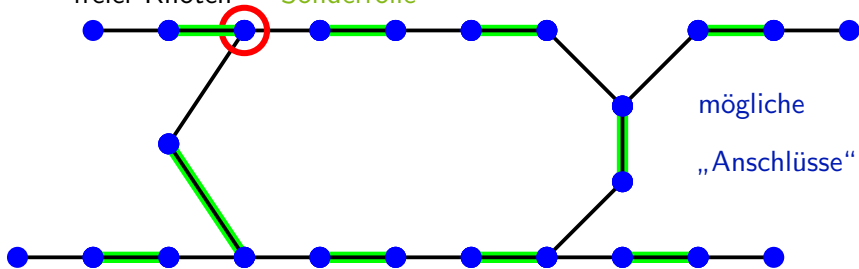
## Ein Beispiel



## „Blüten“

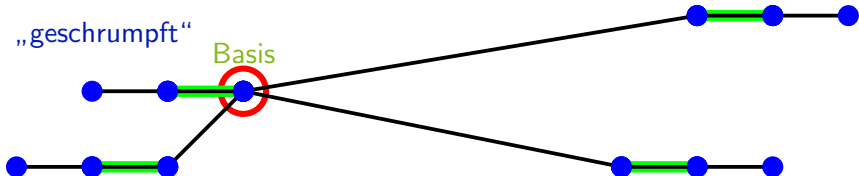
Ein Kreis ungerader Länge eine Blüte

freier Knoten — Sonderrolle



„geschrumpft“

Basis



# Vom Umgang mit „Blüten“

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

**Anmerkung** Kreise ungerader Länge  
gibt es in bipartiten Graphen nicht

**Können wir gefundene Kreise nicht einfach ignorieren?**

**Erinnerung** Graphen können exponentiell viele Kreise haben

**also** total naives Vorgehen geht nicht

# „Recycling“

**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

# Suche nach kürzesten $M$ -verbessernden Pfaden

zunächst nützliche Begriffe

## Definition 3.14 (für Knoten $v \in V$ )

Sei  $G = (V, E)$  Graph,  $M \subseteq E$  Matching,  $v \in V$  Knoten.

- $\text{geradeTiefe}(v) := \min\{\{l \mid (v_1, v_2, \dots, v_l, v) \text{ alternierender Pfad, } v_1 \text{ frei, } l \text{ gerade}\} \cup \{\infty\}\}$
- $\text{ungeradeTiefe}(v) := \min\{\{l \mid (v_1, v_2, \dots, v_l, v) \text{ alternierender Pfad, } v_1 \text{ frei, } l \text{ ungerade}\} \cup \{\infty\}\}$
- $\text{Tiefe}(v) := \min\{\text{geradeTiefe}(v), \text{ungeradeTiefe}(v)\}$
- Für  $v \in V$  mit  $\text{Tiefe}(v) = \text{geradeTiefe}(v) < \infty$  heißt  $v$  **äußerer Knoten**,  $\text{ungeradeTiefe}(v)$  heißt **andere Tiefe**.
- Für  $v \in V$  mit  $\text{Tiefe}(v) = \text{ungeradeTiefe}(v) < \infty$  heißt  $v$  **innerer Knoten**,  $\text{geradeTiefe}(v)$  heißt **andere Tiefe**.



# $M$ -verbessernde Pfade und Brücken

**klar**  $M$ -verbessernde Pfade starten und enden an freien Knoten, haben immer ungerade Länge

**also** jede Kante auf  $M$ -verbesserndem Pfad ist Brücke

Umgekehrt auch?

**Beobachtung** in Graphen ohne Blüten schon

**Beobachtung** dann sogar  $\text{Gewicht}(\{s, t\}) = |P|$

Plan für eine Phase

- Finde Brücken mit minimalem Gewicht.
- Baue entweder zu  $M$ -verbesserndem Pfad aus oder behandle Blüte.

# Die Suche nach Brücken

## etwas genauer

- Suche **gleichzeitig** mit Breitensuche startend von allen freien Knoten aus.
- **gleichzeitig** Finde alle Knoten der Tiefe  $i$ , bevor ein Knoten der Tiefe  $i + 1$  gefunden wird.
- Sobald Brücke in Tiefe  $i$  gefunden wird, zum Pfad ausbauen oder Blüte behandeln.
- Wenn Pfadausbau in Tiefe  $i$  erfolgreich, wird Tiefe  $i + 1$  nicht mehr betrachtet.
- Für effiziente Suche beschrittene Wege (Vorgänger- und Nachfolgerbeziehungen) an den Knoten speichern.

# Von der Brücke zum Pfad

## etwas genauer

- Suche von den beiden Knoten der Brücke aus mit Tiefensuche nach freien Knoten.
- Tiefensuche im tieferen Knoten je einen Schritt vorantreiben.
- Benutzte Knoten markieren, wenn Pfad gefunden, Pfadknoten markieren, markierte Knoten vermeiden.  $\rightsquigarrow$  disjunkte  $M$ -verbessernde Pfade
- Beschrifteten Weg an den Knoten speichern.
- Falls beide Tiefensuchen sich treffen, Alternative gleicher Tiefe suchen. Wenn keine Alternative vorhanden, Blüte gefunden.

# Über Blüten

**Erinnerung** ausgehend von einer Brücke  $\{s, t\}$

**Definition** **Blüte**

$:\Leftrightarrow \exists w \in V$ :  $w$  über Vorgänger von  $s$  und  $t$  erreichbar  
und kein anderer so erreichbarer Knoten hat gleiche Tiefe

**Betrachte** für eine Brücke  $\{s, t\}$  alle solche Knoten  $w$

**Definition** Unter diesen Knoten  $w$  ist  
der Knoten  $b$  mit max. Tiefe **Basis** der Blüte  $B$ .  
 $B$  enthält von Knoten, die noch zu keiner Blüte gehören,  
 $s$ ,  $t$  und über Vorgänger von  $s$  und  $t$  erreichbare Knoten,  
aber nicht  $b$  selbst.

**Blütenbehandlung**

- Stößt DFS auf eine Blüte,  
wird direkt bei ihrer Basis fortgesetzt.
- Bei Pfadkonstruktion wieder „entfalten“.