

Vorlesung

# Effiziente Algorithmen und Komplexitätstheorie

Sommersemester 2008

Ingo Wegener

# Was bisher geschah...

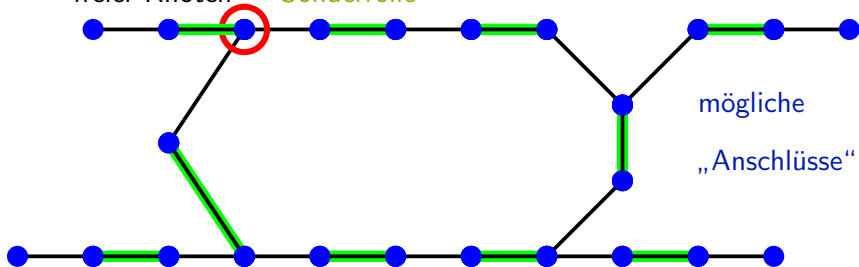
- starke Zusammenhangskomponenten ✓
- Matchings
  - Begriffe ✓
  - Greedy-Algorithmus, 2-Approximation ✓
  - $M$ -verbessernde Pfade ✓
  - Algorithmus von Hopcroft und Karp für bipartite Graphen:  
 $O(\sqrt{n} \cdot e)$  ✓
  - Wunsch Verallgemeinerung für allgemeine Graphen
  - Problem „falsche“  $M$ -verbessernde Pfade Blüten



# „Blüten“

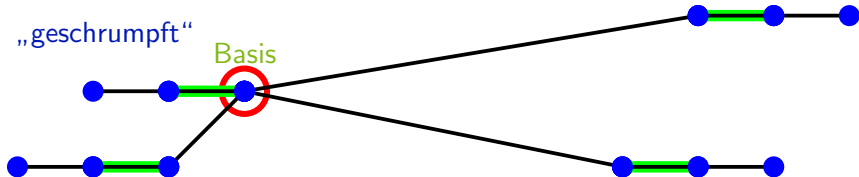
Ein Kreis ungerader Länge eine Blüte

freier Knoten — Sonderrolle



„geschrumpft“

Basis



# Vom Umgang mit „Blüten“

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

**Einsicht** „nur“ Berechnung kürzester knotendisjunkter  $M$ -verbessernder Pfade muss angepasst werden

## Definition 3.14 (für Kanten $\{s, t\} \in E$ )

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

- $\{s, t\}$  mit  $\min\{\text{geradeTiefe}(s) + \text{geradeTiefe}(t), \text{ungeradeTiefe}(s) + \text{ungeradeTiefe}(t)\} < \infty$  heißt **Brücke**.
- für Brücke  $\{s, t\}$ :  $\text{Gewicht}(\{s, t\}) := \min\{\text{geradeTiefe}(s) + \text{geradeTiefe}(t), \text{ungeradeTiefe}(s) + \text{ungeradeTiefe}(t)\} + 1$

# Die Suche nach Brücken

## Plan für eine Phase

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

## 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 Kanten 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“.

# Ein Matching-Algorithmus für allgemeine Graphen

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

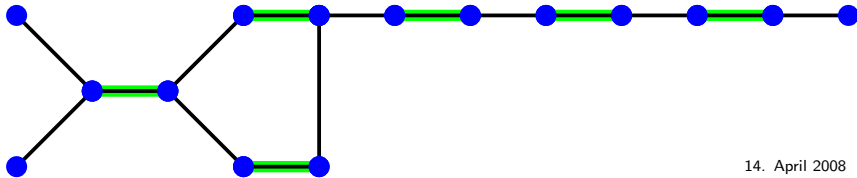
## Resultat

### 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  $|E| = e$ .

Warum so kompliziert?

Können Blüten nicht vorab „geschrumpft“ werden?



# Noch mehr Begriffe

## Definition

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

- Zwei durch eine Matchingkante verbundene Knoten heißen **Partner**.
- Ist  $v$  äußerer Knoten ( $\text{geradeTiefe}(v) < \text{ungeradeTiefe}(v)$ ) und besetzt, so ist sein Partner  $v'$  **Vorgänger** von  $v$ . Ist für Knoten  $v$  und  $v'$ ,  $v$  innerer besetzter Knoten und  $\text{ungeradeTiefe}(v) = \text{geradeTiefe}(v') + 1$ , so ist  $v'$  **Vorgänger** von  $v$ . Dann ist jeweils auch  $v$  **Nachfolger** von  $v'$ .
- Ist  $v$  innerer Knoten und  $v'$  äußerer Knoten,  $\{v, v'\}$  frei und  $\text{geradeTiefe}(v') > \text{ungeradeTiefe}(v)$ , so ist  $v'$  **Anomalie** für  $v$ .

# Der Algorithmus von Micali und Vazirani (1980)

## Rahmenalgorithmus

1.  $M := \emptyset$
2. Repeat
3.     Für alle  $v \in V$
4.         geradeTiefe( $v$ ) := ungeradeTiefe( $v$ ) :=  $\infty$
5.         Blüte( $v$ ) := - *{\* Name der Blüte, zu der  $v$  gehört \*}*
6.         Vorgänger( $v$ ) := Nachfolger( $v$ ) := Anomalien( $v$ ) :=  $\emptyset$
7.         num( $v$ ) := 0 *{\* Anzahl nicht gelöschter Vorgänger \*}*
8.         gelöscht( $v$ ) := besucht( $v$ ) := links( $v$ ) := rechts( $v$ ) := false
9.     Für alle  $\{u, v\} \in E$
10.         benutzt( $\{u, v\}$ ) := besucht( $\{u, v\}$ ) := false
11.     Für  $i \in \{1, 2, \dots, |V|\}$  *{\* Suchlevel \*}*
12.         Kandidaten( $i$ ) :=  $\emptyset$
13.         Brücken( $i$ ) :=  $\emptyset$
14.     VerbesserungGefunden := Suche()
15. Until VerbesserungGefunden = false

## Suche()

1.  $i := 0$ ; Verbesserung := false; Für alle  $v \in \{v' \in V \mid v' \text{ frei}\}$
2.     geradeTiefe( $v$ ) := 0; Füge  $v$  in Kandidaten(0) ein.
3. While Kandidaten( $i$ )  $\neq \emptyset$  und Verbesserung = false
4.     If  $i$  gerade
5.         Für alle  $v \in \text{Kandidaten}(i)$
6.             Für alle Nachbarn  $u$  von  $v$  mit gelöscht( $u$ ) = benutzt( $u$ ) = false,  $\{u, v\}$  frei
7.                 If geradeTiefe( $u$ )  $< \infty$
8.                     Then Füge  $\{u, v\}$  in Brücken( $((\text{geradeTiefe}(u) + \text{geradeTiefe}(v))/2)$ ) ein
9.                     Else
10.                         If ungeradeTiefe( $u$ ) =  $\infty$  Then ungeradeTiefe( $u$ ) =  $i + 1$
11.                         If ungeradeTiefe( $u$ ) =  $i + 1$
12.                         Then
13.                             num( $u$ ) := num( $u$ ) + 1; Füge  $u$  in Nachfolger( $v$ ) ein.
14.                             Füge  $v$  in Vorgänger( $u$ ) ein; Füge  $u$  in Kandidaten( $i + 1$ ) ein.
15.                             If ungeradeTiefe( $u$ )  $< i$  Then Füge  $v$  in Anomalien( $u$ ) ein.

## Suche() — Teil 2

16. Else *{\* i ungerade \*}*
17.     Für alle  $v \in \text{Kandidaten}(i)$  mit  $\text{Blüte}(v) = -$
18.          $u := \text{Partner von } v$  *{\* u ist besetzt \*}*
19.         If  $\text{ungeradeTiefe}(u) < \infty$
20.             Then Füge  $\{u, v\}$  in  
                $\text{Brücken}((\text{ungeradeTiefe}(u) + \text{ungeradeTiefe}(v))/2)$  ein.
21.         If  $\text{geradeTiefe}(u) = \infty$
22.             Then
23.                  $\text{Nachfolger}(v) := \{u\}$ ;  $\text{Vorgänger}(u) := \{v\}$ ;  $\text{num}(u) := 1$
24.                  $\text{geradeTiefe}(u) := i + 1$ ; Füge  $u$  in  $\text{Kandidaten}(i + 1)$  ein.
25.     Für alle  $\{s, t\} \in \text{Brücken}(i)$
26.         If  $\text{gelöscht}(s) = \text{gelöscht}(t) = \text{false}$
27.             Then  $\text{Verbesserung} := \text{BlüteOderVerbesserung}(\{s, t\})$
28.      $i := i + 1$
29. Exit mit Rückgabe von  $\text{Verbesserung}$

## BlüteOderVerbesserung( $\{s, t\}$ )

1. If Blüte( $s$ ) = Blüte( $t$ ) und Blüte( $s$ )  $\neq -$  Then Exit.
2. If Blüte( $s$ )  $\neq -$  Then  $v_L := \text{Basis}(\text{Blüte}(s))$  Else  $v_L := s$
3. If Blüte( $r$ )  $\neq -$  Then  $v_R := \text{Basis}(\text{Blüte}(r))$  Else  $v_R := r$
4. links( $v_L$ ) := true; rechts( $v_R$ ) := true; DCV := -; Schranke :=  $v_R$
5. While ( $v_L$  nicht frei oder  $v_R$  nicht frei) und BlüteGefunden = false
6.     Tiefe $_L := \min\{\text{geradeTiefe}(v_L), \text{ungeradeTiefe}(v_L)\}$
7.     Tiefe $_R := \min\{\text{geradeTiefe}(v_R), \text{ungeradeTiefe}(v_R)\}$
8.     If Tiefe $_L \geq$  Tiefe $_R$   
        Then BlüteGefunden := DFSLinks()  
        Else BlüteGefunden := DFSRechts()
9. If BlüteGefunden = false Then
10.     $P_L := \text{FindePfad}(s, v_L, -)$ ;  $P_R := \text{FindePfad}(t, v_R, -)$ ;
11.    Verbessere Matching mit Pfad  $P := P_L, \{s, t\}, P_R$ .
12.    Repeat
13.       Entferne  $y$  aus  $P$ ; gelöscht( $y$ ) := true;
14.       Für alle  $z \in \text{Nachfolger}(y)$  mit gelöscht( $z$ ) = false
15.           num( $z$ ) := num( $z$ ) - 1
16.           If num( $z$ ) = 0 Then Füge  $z$  zu  $P$  hinzu
17.    Until  $P$  leer.
18.    Exit mit Rückgabe von true;

BlüteOderVerbesserung( $\{s, t\}$ ) — Teil 2

19. Else *{\* Blüte gefunden \*}*
20.   rechts(DCV) := false;
21.   Erzeuge Blüte  $B$  aus allen in diesem Aufruf von BlüteOderVerbesserung  
als links oder rechts markierten Knoten, Blüte( $v$ ) :=  $B$  für diese  $v$ .  
Speichere  $r$  und  $s$  als Spitzen von  $B$ .
22.   Basis( $B$ ) := DCV
23.   Für alle  $y \in B$
24.     If  $\min\{\text{geradeTiefe}(y), \text{ungeradeTiefe}(y)\}$  gerade
25.     Then  $\text{ungeradeTiefe}(y) := 2i + 1 - \text{geradeTiefe}(y)$ ;
26.     Else
27.        $\text{geradeTiefe}(y) := 2i + 1 - \text{ungeradeTiefe}(y)$ ;
28.       Füge  $y$  in Kandidaten( $\text{ungeradeTiefe}(y)$ ) ein.
29.       Für alle  $z \in \text{Anomalien}(y)$
30.          $j := (\text{geradeTiefe}(y) + \text{geradeTiefe}(z))/2$
31.         Füge  $\{y, z\}$  in Brücken( $j$ ) ein; benutzt( $\{y, z\}$ ) := true

## DFSLinks()

1. While  $\exists u \in \text{Vorgänger}(v_L)$   
mit  $\text{benutzt}(\{u, v_L\}) = \text{gelöscht}(u) = \text{false}$
2.      $\text{benutzt}(\{u, v_L\}) := \text{true}$
3.     If  $\text{Blüte}(u) \neq -$  Then  $u := \text{Basis}(\text{Blüte}(u))$
4.     If  $\text{links}(u) = \text{rechts}(u) = \text{false}$  Then
5.          $\text{links}(u) := \text{true}; \text{Elter}(u) := v_L; v_L := u$
6.         Exit mit Rückgabe von  $\text{false}$  *{\* keine Blüte \*}*  
*{\* u schon links oder rechts  $\rightsquigarrow$  Backtracking \*}*
7.     If  $v_L = s$  Then Exit mit Rückgabe von  $\text{true}$  *{\* Blüte \*}*
8.     Else  $v_L := \text{Elter}(v_L)$ ; Exit mit Rückgabe von  $\text{false}$  *{\* Backtracking \*}*

## DFSRechts()

1. While  $\exists u \in \text{Vorgänger}(v_R)$   
mit benutzt( $\{u, v_R\}$ ) = gelöscht( $u$ ) = false
2.     benutzt( $\{u, v_R\}$ ) := true
3.     If Blüte( $u$ )  $\neq -$  Then  $u := \text{Basis}(\text{Blüte}(u))$
4.     If links( $u$ ) = rechts( $u$ ) = false Then
5.         rechts( $u$ ) := true; Elter( $u$ ) :=  $v_R$ ;  $v_R := u$
6.         Exit mit Rückgabe von false *{\* keine Blüte \*}*  
*{\* u schon links oder rechts  $\rightsquigarrow$  Backtracking \*}*
7.     Else If  $u = v_L$  Then DCV :=  $u$
8.     If  $v_R = \text{Schranke}$  Then
9.          $v_R := \text{DCV}$ ; Schranke := DCV;  
links( $v_R$ ) := false; rechts( $v_R$ ) := true;
10.      $v_L := \text{Elter}(v_L)$  *{\* Backtracking links \*}*
11.     Else  $v_R := \text{Elter}(v_R)$  *{\* Backtracking rechts \*}*
12.     Exit mit Rückgabe von false

## FindePfad( $h, l, B$ )

1. If  $h = l$  Then  $P := h$ ; Exit mit Rückgabe von  $P$
2.  $v := h$ ; Repeat
3.     While  $\nexists u \in \text{Vorgänger}(v)$  mit  $\text{benutzt}(\{u, v\}) = \text{false}$
4.          $v := \text{Elter}(v)$
5.     If  $\text{Blüte}(v) = B$  oder  $\text{Blüte}(v) = -$
6.         Then
  - Wähle  $u \in \text{Vorgänger}(v)$  mit  $\text{benutzt}(\{u, v\}) = \text{false}$ ;  
 $\text{benutzt}(\{u, v\}) := \text{true}$
7.         Else  $u := \text{Basis}(\text{Blüte}(v))$
8.     If  $\text{besucht}(u) = \text{false}$  und  $\text{gelöscht}(u) = \text{false}$  und  
 $\min\{\text{ungeradeTiefe}(u), \text{geradeTiefe}(u)\} > \min\{\text{ungeradeTiefe}(l), \text{geradeTiefe}(l)\}$   
 und  $\text{links}(u) = \text{links}(h)$  und  $\text{rechts}(u) = \text{rechts}(h)$
9.         Then  $\text{besucht}(u) := \text{true}$ ;  $\text{Elter}(u) := v$ ;  $v := u$
10.     Until  $u = l$
11. Sei  $P = x_1, x_2, \dots, x_m$  Pfad über Elter-Zeiger von  $h$  nach  $l$ .
12. Für alle  $j \in \{1, 2, \dots, m-1\}$
13.     If  $\text{Blüte}(x_j) \neq B$  und  $\text{Blüte}(x_j) \neq -$
14.         Then Ersetze  $x_j$  und  $x_{j+1}$  durch Ausgaben von  $\text{Öffne}(x_j)$
15. Exit mit Rückgabe von  $P$

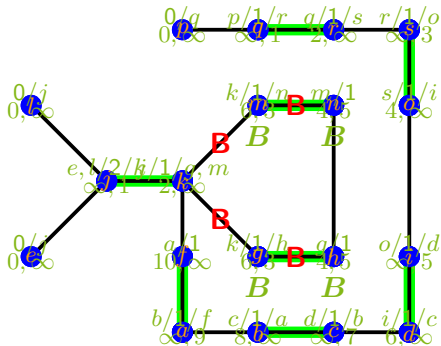
# Öffne( $x$ )

1.  $B := \text{Blüte}(x)$ ;  $b := \text{Basis}(B)$
2. If  $\min\{\text{ungeradeTiefe}(x), \text{geradeTiefe}(x)\}$  gerade
3. Then Exit mit Rückgabe  $\text{FindePfad}(x, b, B)$
4.  $p_L :=$  linke Spitze von  $B$ ;  $p_R :=$  rechte Spitze von  $B$
5. If  $\text{links}(x) = \text{true}$
6. Then  $P := \text{FindePfad}(p_L, x, B), \text{FindePfad}(p_R, b, B)$
7. Else  $P := \text{FindePfad}(p_R, x, B), \text{FindePfad}(p_L, b, B)$
8. Exit mit Rückgabe von  $P$

# Ein ganz kleines Beispiel (unkomplett)

$i = 10$

Kandidaten(10) :  $f$



# Algorithmus von Micali und Vazirani

Ideen noch einmal zusammengefasst

- **Breitensuche** parallel in allen freien Knoten startend
- benutzte **Wege merken**, bei Brücken stoppen
- mit balancierter **Tiefensuche**  $M$ -verbessernden Pfad oder Blüte suchen, dabei **nur Wege aus Breitensuche verwenden**
- Blüten **markieren**, Zeiger auf Basis an jedem Knoten ablegen
- kürzestes  $M$ -verbessernde Pfade ggf. in Blüten entfalten
- bei  $M$ -verbesserndem Pfad in Tiefe  $i$ , Tiefe  $i + 1$  **nicht mehr beginnen**