

SS 2009

# Effiziente Algorithmen und Datenstrukturen II

Riko Jacob

Fakultät für Informatik  
TU München

<http://www14.in.tum.de/lehre/2009SS/ea/>

Sommersemester 2009

# Kapitel 0 Organisatorisches

- Vorlesungen:
  - 4SWS
    - Dienstag, 10:15-11:45, MI HS 2
    - Donnerstag, 10:15-11:45, MI 00.13.009A
    - Wahlpflichtvorlesung im Gebiet Algorithmen (Theoretische Informatik, Informatik III), Bioinformatik
- Übung:
  - 2SWS Zentralübung: Zeit und Ort wird noch festgelegt
  - Übungsleitung: Matthias Baumgart
- Umfang:
  - 4V+2ZÜ, 8 ECTS-Punkte
- Sprechstunde:
  - nach Vereinbarung

- Übungsleitung:
  - Matthias Baumgart, MI 03.09.060 ([baumgart@in.tum.de](mailto:baumgart@in.tum.de))  
Sprechstunde: Montag, 10:30Uhr und nach Vereinbarung per Email
- Sekretariat:
  - Frau Lissner, MI 03.09.052 ([lissner@in.tum.de](mailto:lissner@in.tum.de))




- Übungsaufgaben und Klausur:
  - Ausgabe jeweils am Dienstag in der Vorlesung bzw. auf der Webseite der Vorlesung
  - Abgabe eine Woche später vor der Vorlesung
  - Besprechung in der Zentralübung
- Klausur:
  - Klausur/mndl. Prüfung
  - bei der Klausur sind *keine* Hilfsmittel außer einem handbeschriebenen DIN-A4-Blatt zugelassen
  - Leistungsnachweis: 40% der erreichbaren Hausaufgabenpunkte, erfolgreiche Teilnahme an Klausur/mndl. Prüfung
  - vorauss. 10 Übungsblätter, jedes 10 Punkte

- Vorkenntnisse:
  - Einführung in die Informatik 1/2
  - Diskrete Strukturen (DS, DWT)
  - Grundlagen: Algorithmen und Datenstrukturen (GAD)
  - Effiziente Algorithmen und Datenstrukturen
- Weiterführende Vorlesungen:
  - Randomisierte Algorithmen
  - Komplexitätstheorie
  - Internetalgorithmik
  - ...
- Webseite:

<http://wwwmayr.in.tum.de/lehre/2009SS/ea/>

- ① Flüsse in Netzwerken
- ② String und Pattern Matching
- ③ Textkompression
- ④ Scheduling

# 1. Literatur

-  Alfred V. Aho, John E. Hopcroft, Jeffrey D. Ullman:  
*The design and analysis of computer algorithms*,  
Addison-Wesley Publishing Company: Reading (MA), 1974
-  Ravindra K. Ahuja, Thomas L. Magnanti, James B. Orlin:  
*Network flows — Theory, algorithms, and applications*,  
Prentice-Hall: Englewood Cliffs, NJ, 1993
-  Thomas H. Cormen, Charles E. Leiserson, Ron L. Rivest,  
Clifford Stein:  
*Introduction to algorithms*,  
McGraw-Hill, 1990



Donald E. Knuth:

*The art of computer programming. Vol. 1: Fundamental algorithms,*

3. Auflage, Addison-Wesley Publishing Company: Reading (MA), 1997



Volker Heun:

*Grundlegende Algorithmen: Einführung in den Entwurf und die Analyse effizienter Algorithmen,*

2. Aufl., Vieweg: Braunschweig-Wiesbaden, 2003



Christos H. Papadimitriou, Kenneth Steiglitz:

*Combinatorial optimization: Algorithms and complexity,*  
Prentice-Hall, Englewood Cliffs, NJ, 1982





Robert E. Tarjan:

*Data Structures and Network Algorithms*,  
CBMS-NSF Regional Conference Series in Applied  
Mathematics, SIAM, Philadelphia, PA, 1983



Steven S. Skiena:

*The algorithm design manual*,  
Springer-Verlag: Berlin-Heidelberg-New York, 1998



Sven O. Krumke, Hartmut Noltemeier:

*Graphentheoretische Konzepte und Algorithmen*,  
Teubner, Wiesbaden, 2005

Weitere Originalarbeiten und Texte werden im Verlauf der  
Vorlesung angegeben.

# Kapitel VII Flüsse in Netzwerken

## 1. Grundlagen

### Flussnetzwerk

Graph  $G = (V, E)$ ,      Quelle  $s \in V$ ,      Senke  $t \in V$ ,  
 $E \subseteq V \setminus \{t\} \times V \setminus \{s\}$

### Fluss

$f: E \rightarrow \mathbb{R}_0^+$  (oder  $\mathbb{N}_0$ )

Flusserhaltung: Für alle  $v \in V \setminus \{s, t\}$ :

$$\sum_{u: (u,v) \in E} f(u,v) = \sum_{w: (v,w) \in E} f(v,w)$$

Flusswert:

$$|f| := \sum_{u: (s,u) \in E} f(s,u)$$

## Übungsaufgabe

Es gilt

$$|f| = \sum_{u: (u,t) \in E} f(u, t)$$

## 2. Schnitte

Schnitt in  $G = (V, E)$ :  $S \subset V$ ,  $s \in S$ ,  $t \notin S$

$$\delta^+(S) = \{(u, v) \in E \mid u \in S, v \notin S\}$$

$$\delta^-(S) = \{(u, v) \in E \mid u \notin S, v \in S\}$$

$$f^+(S) = f(\delta^+(S)) = \sum_{e \in \delta^+(S)} f(e)$$

$$f^-(S) = f(\delta^-(S))$$

Exzess / Überschuss:  $f(S) = f^+(S) - f^-(S)$

Hinweis Übung

## Kapazitäten in Graph $G = (V, E)$

$$c: E \rightarrow \mathbb{R}_0^+$$

Ein Fluss  $f$  ist zulässig  $\iff f(u, v) \leq c(u, v)$  für alle  $(u, v) \in E$

## Kapazität eines Schnittes $S$

$$c(S) = \sum_{e \in \delta^+(S)} c(e)$$

## Lemma 1 (Schnitt beschränkt Fluss)

*Für jeden zulässigen Fluss  $f$  und jeden Schnitt  $S$  in  $G$  gilt:*

$$|f| \leq c(S)$$

### 3. Min-Cut-Max-Flow-Theorem

#### Theorem 2 (Min-Cut-Max-Flow)

Für jedes Flussnetzwerk  $G = (V, E, s, t)$  mit Kapazitäten  $c$  gilt

$$\max_{f \text{ ist Fluss in } G} |f| = \min_{S \text{ ist Schnitt in } G} c(S)$$

## Residualnetz, selbe Knotenmenge

$$c_f(u, v) = c(u, v) - f(u, v) + f(v, u)$$

$$G_f = (V, \{(u, v) \mid c_f(u, v) > 0\})$$

## Augmentierender Pfad

ist ein Pfad ohne Kantenwiederholung  $p = (s = v_0, v_1, \dots, v_k = t)$   
in  $G_f$

$c_p = \min c_f(v_i, v_{i+1}) > 0$  seine Kapazität.

### Definition 3 (Addition von Flüssen)

$$f + f'(u, v) := \max\{0, f(u, v) - f(v, u) + f'(u, v) - f'(v, u)\}$$

### Lemma 4

*Sei  $f$  zulässiger Fluss in  $G$  und  $f'$  ein zulässiger Fluss in  $G_f$ . Dann ist  $f + f'$  zulässiger Fluss in  $G$ .*



## Theorem 5 (Max Flow Min Cut)

Für ein Flussnetzwerk  $G$  mit Kapazitäten  $c$  und einen Fluss  $f$  sind äquivalent:

- 1  $f$  ist maximaler Fluss
- 2  $G_f$  hat keinen augmentierenden Pfad
- 3  $|f| = c(S)$  für einen Schnitt  $S$

### Beweis:

$\neg 2 \Rightarrow \neg 1$ : ein augmentierender Pfad kann den Fluss vergrößern

$2 \Rightarrow 3$ :  $S$  ist die  $s$ -Komponente von  $G_f$

$\neg 1 \Rightarrow \neg 3$ : Ein größerer Fluss widerspricht Lemma 1



## 4. Ford-Fulkerson-Algorithmus



L.R. Ford, Jr., D.R. Fulkerson:

*Maximal flow through a network.*

Can. J. Math. **8** pp. 399–404, 1956

### Algorithmus

Wiederhole solange möglich:

Vergrößere Fluss entlang irgendeines augmentierenden Pfads

### Theorem 6

*Für jedes Flussnetzwerk mit integralen Kapazitäten existiert ein integraler maximaler Fluss  $f^*$ .*

*Ein solcher kann in  $O(|f^*||E|)$  Zeit gefunden werden.*

## Theorem 7 (Satz von Hall, Heiratssatz)

*Sei  $G = (U \cup V, E)$  ein bipartiter Graph.*

*Es gibt genau dann ein  $U$ -perfektes Matching in  $G$  wenn für alle  $U' \subseteq U$  gilt*

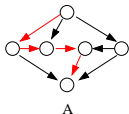
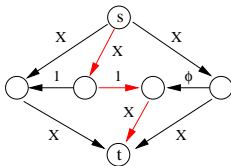
$$|N(U')| \geq |U'|$$

## 5. Konvergenzprobleme

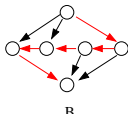
### Irrationale Kapazitäten

Es gibt ein Flussnetzwerk und eine Folge von augmentierenden Pfaden, so dass für die Folge  $f_i$  von Flüssen gilt

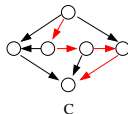
- $G_{f_i}$  ist zusammenhängend
- $|f_i|$  konvergiert, aber nicht gegen den Maximalwert



A



B



C

## 6. Edmonds-Karp Heuristik

### Heuristik 1

Wähle den augmentierenden Pfad mit dem größten Wert

### Theorem 8

*Im Fall von ganzzahligen Kapazitäten benötigt Heuristik 1  $|E| \ln |f^*|$  Iterationen und  $O(|E|^2 \log |V| \ln |f^*|)$  Laufzeit um den optimalen Fluss  $f^*$  zu finden.*

### Lemma 9

*Ein maximaler augmentierender Pfad kann (wie ein maximaler spannender Baum) in  $O(|E| \log |V|)$  berechnet werden.*

## Lemma 10

Für jeden Fluss  $f$  hat der maximale augmentierende Pfad  $p$  einen Wert  $c_p \geq (|f^*| - |f|)/|E|$

## Lemma 11

Seien  $f, f'$  zulässige Flüsse in  $G$ ,  $|f'| \geq |f|$ . Dann ist  $f' - f$  zulässiger Fluss in  $G_f$ .

## Lemma 12

Sei  $G' = (V, E, s, t, c)$  ein Residualnetzwerk,  $q$  ein Kapazitäts-maximaler augmentierender Pfad,  $c_q$  dessen Kapazität. Sei  $h$  ein maximaler Fluss in  $G'$ . Dann gilt  $c_q \geq |h|/|E|$

## Heuristik 2

Wähle den augmentierenden Pfad mit den wenigsten Kanten

### Definition 13

$\text{dist}_i(v) :=$  Kantendistanz in  $G_{f_i}$  von  $s$  nach  $v$  ( $+\infty$  erlaubt)

### Lemma 14

*Für jeden Knoten  $v \in V$  gilt  $\text{dist}_{i+1}(v) \geq \text{dist}_i(v)$ .*

### Lemma 15

*Während der Abarbeitung der Heuristik 2 kann jede Kante  $(u, v)$  höchstens  $|V|/2$  mal vom residualen Graphen verschwinden.*

### Lemma 16

*Heuristik 2 benötigt  $|E| \cdot |V|/2$  Iterationen und eine Laufzeit von  $O(|V| \cdot |E|^2)$ .*

## 7. Blockierende Flüsse

### 7.1 Dinits' Algorithmus

#### Definition 17 (geschichtet)

Ein Flussnetzwerk ist geschichtet, falls es eine Partition  $V = V_0 \cup V_1 \cup \dots \cup V_k$  mit  $V_0 = \{s\}$  und  $V_k = \{t\}$  der Knoten gibt, so dass für alle Kanten  $(u, v)$  gilt  $u \in V_i, v \in V_{i+1}$  für ein  $i$ .

#### Lemma 18

Für  $v_i \in V_i$  gilt  $\text{dist}(s, v_i) \geq i$ ,  $\text{dist}(v_i, t) \geq k - i$ .

#### Definition 19 ( $L$ )

Die Schichtung  $L$  eines Flussnetzwerkes  $G$  definiert  $V_i = \{v \mid \text{dist}_G(s, v) = i\}$  und entfernt alle Kanten, die nicht der Schichtungsbedingung genügen.

So entsteht  $L_f$  aus dem Residualnetzwerk  $G_f$ .



## Definition 20 (Blocking Flow)

Ein Fluss  $f$  in einem geschichteten Flussnetzwerk  $L$  ist blockierend, falls jeder kürzeste  $s - t$  Pfad eine saturierte Kante ( $f(u, v) = c(u, v)$ ) enthält.

## Dinitz' Algorithmus

- $f_0 = 0$
- Wiederhole  $f_{i+1} = f_i +$  blockierender Fluss in  $L_{f_i}$  solange  $G_{f_i}$  augmentierende Pfade enthält.

## Lemma 21

*Ein blockierender Fluss kann in  $O(|V||E|)$  Zeit gefunden werden.*

## Lemma 22

$$\text{dist}_{i+1}(s, t) \geq \text{dist}_i(s, t) + 1$$

## 7.2 0-1-Netzwerke

### Definition 23 (0-1 Netzwerk)

Ein Flussnetzwerk mit  $c(e) = 1$  für alle Kanten heißt 0-1 Netzwerk.

### Lemma 24

*Ein blockierender Fluss in einem geschichteten 0-1 Netzwerk kann in  $O(|E|)$  Zeit gefunden werden.*

### Lemma 25

*Sei  $G$  ein 0-1 Flussnetzwerk und  $f$  ein zulässiger, ganzzahliger Fluss in  $G$ . Dann kann  $G_f$  als 0-1 Netzwerk mit  $|E|$  Kanten dargestellt werden.*

## Lemma 26

Sei  $G$  ein 0-1 Netzwerk,  $f^*$  ein maximaler Fluss darin,  $M = |f^*|$ , und  $h = \text{dist}(s, t)$ . Dann gilt

$$h \leq \frac{|E|}{M}$$

## Lemma 27

Für 0-1 Netzwerke benötigt Dinits' Algorithmus  $O\left(|E|^{\frac{3}{2}}\right)$  Zeit.

## Definition 28 (Typ 1)

Ein 0-1 Netzwerk ist vom Typ 1, falls es keine parallelen Kanten hat. Es ist vom Typ 1a, falls jede Kante höchstens eine parallele hat.

## Lemma 29

*Sei  $G$  ein 0-1 Netzwerk vom Typ 1,  $f$  ein gültiger Fluss in  $G$ . Dann ist  $G_f$  ein 0-1 Netzwerk vom Typ 1a.*

## Lemma 30

*Sei  $G$  ein geschichtetes 0-1 Netzwerk vom Typ 1a,  $M = |f^*|$  der maximale Flusswert,  $h = \text{dist}(s, t)$ . Dann gilt  $h \leq \frac{2|V|}{\sqrt{M/2}} + 1$ .*

## Lemma 31

*Für 0-1 Netzwerke vom Typ 1 benötigt Dinits' Algorithmus  $O\left(|V|^{\frac{2}{3}} \cdot |E|\right)$  Zeit.*

## Definition 32 (Typ 2)

Ein 0-1 Netzwerk ist vom Typ 2, falls jeder Knoten entweder nur eine eingehende, oder nur eine ausgehende Kante hat.

## Lemma 33

*Sei  $G$  ein 0-1 Netzwerk vom Typ 2,  $f$  ein ganzzahliger gültiger Fluss in  $G$ .*

*Dann ist  $G_f$  ein 0-1 Netzwerk vom Typ 2.*

## Lemma 34 (Übung)

*Sei  $G$  ein geschichtetes 0-1 Netzwerk vom Typ 2,  $M = |f^*|$  der maximale Flusswert,  $h = \text{dist}(s, t)$ . Dann gilt  $h \leq \frac{|V|-2}{M} + 1$ .*

## Lemma 35

*Für 0-1 Netzwerke vom Typ 2 benötigt Dinits' Algorithmus  $O\left(|V|^{\frac{1}{2}} \cdot |E|\right)$  Zeit.*

## 8. Erweiterungen und Spezialfälle

### 8.1 Netzwerke mit unteren und oberen Schranken



Shimon Even:

*Networks with upper and lower bounds.*

Graph Algorithms, section 5.3, Computer Science Press:  
Rockville, MD, 1979

#### Lemma 36

*Für ein Flussnetzwerk  $G$  mit oberen und unteren Fluss-Schranken kann mit der Berechnung eines maximalen Flusses auf einem Graphen  $G'$  mit  $|V'| = |V| + 2$  und  $|E'| = |E| + 2|V| + 2$  gültiger Fluss in  $G$  gefunden werden, falls ein solcher existiert.*

## 9. Preflow Algorithmen

### 9.1 Der Malhotra-Pramodh Kumar-Maheshwari (MPM++) Algorithmus

#### Weitere Literatur:



Robert Endre Tarjan:

*A simple version of Karzanov's blocking flow algorithm.*

*Oper. Res. Lett.* **2** pp. 265–268, 1984

#### Struktur des MPM++ Algorithmus

Berechne blockierenden Fluss in  $O(|V|^2)$  Zeit, also einen maximalen Fluss in  $O(|V|^3)$ .

## Definition 37 (Preflow)

Sei  $G = (V, A, s, t, c)$  ein Flussnetzwerk.  $f: A \rightarrow \mathbb{R}_0^+$  (oder  $\mathbb{N}_0$ ) ist ein Preflow, falls  $f$  die Kapazitäten einhält und  $f_{\text{in}}(v) \geq f_{\text{out}}(v)$  für alle  $v \in V \setminus \{s, t\}$  gilt.

## Definition 38 (Blockierte Knoten)

Der Algorithmus nennt einige Knoten blockiert:  
Diese Knoten “haben genug” ausgehenden Fluss.  
Ein unblockierter Knoten kann blockiert werden, nicht umgekehrt.

## Invariante

Für jeden blockierten Knoten  $v$  gilt:  
Alle Pfade von  $v$  nach  $t$  enthalten eine saturierte Kante



## Definition 39 (Balanciere unblockiertes $v$ : Increase $(v, w)$ )

Falls  $w$  nicht blockiert ist:

$$f(v, w) + = \min\{c(v, w) - f(v, w), f_{\text{in}}(v) - f_{\text{out}}(v)\}$$

## Definition 40 (Balanciere blockiertes $v$ : Decrease $(u, v)$ )

$$f(u, v) - = \min\{f(u, v), f_{\text{in}}(v) - f_{\text{out}}(v)\}$$

## Algorithmus: Increase Flow

- In topologischer Ordnung:
  - Balanciere unblockierten Knoten  $v$  mittels Increase  $(v, w)$   
 $\Rightarrow v$  ist balanciert oder blockiert

## Algorithmus: Decrease Flow

- In umgekehrter topologischer Ordnung:
  - Balanciere blockierten Knoten  $v$  mittels Decrease  $(u, v)$   
 $\Rightarrow v$  ist balanciert

## Algorithmus: Blocking Flow in geschichtetem Netzwerk (MPM++)

Saturiere Kanten der Form  $(s, v)$ ;  $s$  ist blockiert.

Solange es einen unblockierten und unbalancierten Knoten gibt:

- Increase Flow (unblockierte Knoten balanciert oder blockiert)
- Decrease Flow (alle blockierte Knoten balanciert)

## Korrektheit

Falls der Algorithmus terminiert, hat er einen gültigen, blockierenden Fluss gefunden.

## Laufzeit

Der Algorithmus terminiert nach höchstens  $|V|$  Iterationen.

Es wird höchstens  $|V|^2$  mal versucht einen Knoten zu balancieren.

Der Fluss auf einer Kante  $e$  wird zunächst größer, dann kleiner.

Es gibt höchstens  $2|E| + |V|^2$  Increase oder Decrease Schritte.

## Theorem 41 (Laufzeit MPM++)

*Der Algorithmus MPM++ berechnet in  $O(|V|^2)$  Zeit einen blockierenden Fluss.*

*Somit erreicht Dinits' Algorithmus eine Laufzeit von  $O(|V|^3)$ .*

## 9.2 Push/Relabel-Algorithmus von Goldberg-Tarjan



Andrew V. Goldberg, Robert E. Tarjan:

*A new approach to the maximum-flow problem.*

J. ACM **35** pp. 921–924, ACM Press: New York, 1988

### Definition 42 (Distanzmarkierung)

$d: V \rightarrow \mathbb{N}$  ist D. in  $G$  bezüglich Preflow  $f$  falls gilt

- $d(t) = 0$
- für alle  $(u, v) \in G_f$  gilt  $d(u) \leq d(v) + 1$   
bei Gleichheit heißt  $(u, v)$  zulässig.

### Lemma 43

Für einen Pfad  $p$  in  $G_f$  von  $v$  nach  $t$  und eine Distanzmarkierung  $d$  gilt  $d(v) \leq |p|$ .

Falls  $d(s) \geq |V|$  und  $f$  ist Fluss, dann existiert kein Pfad von  $s$  nach  $t$  und  $f$  ist maximaler Fluss.

## Push-Relabel $u$ (unbalanciert)

- Falls zulässige Kante  $(u, v)$  existiert:  
 $f(u, v) + = \min c(u, v) - f(u, v), f_{\text{in}}(u) - f_{\text{out}}(u)$
- sonst: Erhöhe  $d(u) := 1 + \min\{d(v) \mid (u, v) \in G_f\}$

## Generischer Push-Relabel Algorithmus

- 1 für alle  $(u, v) \in E$ : setze  $f(u, v) = 0$
- 2 für alle  $v$ :  $d(v) = \text{dist}_{G_f}(v, t)$
- 3  $d(s) = n, f(s, u) = c(s, u)$  für alle passenden  $u$
- 4 solange es einen unbalancierten Knoten gibt:
- 5 wähle einen solchen,  $u$
- 6 Push-Relabel( $u$ )

## Lemma 44 (Pfad zu $s$ )

*Sei  $f$  Preflow im P-R Algorithmus,  $v$  ein aktiver Knoten.  
Dann existiert in  $G_f$  ein Weg von  $v$  nach  $s$ .*

## Lemma 45 ( $d < 2n$ )

*In P-R ist immer  $d(v) \leq 2n - 1$ .  
Jeder Knoten wird höchstens  $2n - 1$  mal "erhöht".  
Insgesamt finden  $O(n^2)$  Markenerhöhungen statt.*

## Definition 46 ((nicht-) sättigender Flussschub)

- $\delta = c(u, v) - f(u, v)$ : sättigend
- $\delta = f_{\text{in}}(u) - f_{\text{out}}(u)$ : nicht-sättigend

## Lemma 47 (wenige sättigende Flussschübe)

*P-R führt  $O(nm)$  sättigende Flussschübe aus.*

## Lemma 48 (wenige nicht-sättigende Flussschübe)

*P-R führt  $O(n^2m)$  nicht-sättigende Flussschübe aus.*

$$\Phi := \sum_{v \text{ ist aktiv}} d(v)$$

## Theorem 49

*Der generische Push-Relabel Algorithmus benötigt  $O(n^2)$  Markenerhöhungen,  $O(nm)$  sättigende und  $O(n^2m)$  nicht-sättigende Flussschübe.*

## Highest-Label-Push-Relabel

Wähle aktives  $u$  mit höchster Marke (bis  $u$  balanciert ist)

## FIFO-Push-Relabel

Wähle aktives  $u$  am Kopf der FIFO-Schlange  $F$

Arbeite mit  $u$  bis dieses balanciert oder erhöht wird.

Neu aktivierte Knoten werden am Ende von  $F$  angefügt (auch  $u$ ).



## Theorem 50 (FIFO-Push-Relabel)

*Der FIFO-Push-Relabel Algorithmus benötigt  $O(n^2)$  Markenerhöhungen,  $O(nm)$  sättigende und  $O(n^3)$  nicht-sättigende Flussschübe.*

## Potential

$$\Phi := \max\{d(u) \mid u \text{ ist aktiv}\}$$

## parallele Implementierung

$O(n^2)$  Pulse-Phasen genügen.

# Implementierung Preflow Push

Ziel:  $\text{Push-Relabel}(u)$  ist amortisiert  $O(1)$  bzw.  $O(n^3)$  total

## Datenstruktur für $G_f$

Beide Sorten mögliche Pfeile doppelt verkettet, antiparallel verlinkt  
 $d(u)$ , Residualkapazität aller Pfeile und Überschuss aller Knoten  
Pfeil-Pointer "current"

## aktive Knoten gespeichert in

- doppelt verkettete Liste
- FIFO-Schlange
- Pro möglichem Markenwert eine d-v-Liste + höchste Marke  $k^*$

## 10. Der Skalierungsansatz von Ahuja-Orlin



Ravindra K. Ahuja, James B. Orlin:

*A fast and simple algorithm for the maximum flow problem.*  
Oper. Res. **37** pp. 748–759, Operations Research Society of America, 1989

Laufzeit  $O(nm + n^2 \log U)$

$\Delta$ -Phasen: P-R kleinste Marke:  $\Delta/2 < e \leq \Delta$ ; push-cap

Jeder nicht-sättigende push schickt  $\geq \Delta/2$

Kein “excess” über  $\Delta$  wird erzeugt

$\leq 8n^2$  nicht-sättigende pushes pro Skalierungsphase

## 11. Zerlegung eines Flusses

### Definition 51 ( $b$ -Fluss)

Flussüberschuss von  $v$ :

$$\text{exc}_f(v) := \sum_{u: (u,v) \in E} f(u,v) - \sum_{w: (v,w) \in E} f(v,w)$$

$f$  ist  $b$ -Fluss für Knotenbewertung  $b: V \rightarrow \mathbb{R}$ :  $\text{exc}_f(v) = b(v)$

Für  $b \equiv 0$  heisst  $f$  eine Strömung oder Zirkulation

Zulässigkeit: Schranken  $l, c$  mit  $0 \leq l(e) \leq f(e) \leq c(e)$

### Definition 52 (Weg- / Kreisströmung)

Sei  $P \subseteq E$  ein einfacher Weg oder Kreis,

$\delta > 0$  Flusswert/Strömungswert

$$f_{P,\delta}(e) = \begin{cases} \delta, & \text{falls } e \in P \\ 0, & \text{sonst} \end{cases}$$

## Theorem 53 (Flussdekompositionssatz)

*Jeder  $b$ -Fluss  $f$  lässt sich als Summe von  $|V| + |E|$  Wegflüssen und Kreisströmungen darstellen mit:*

- *Für jeden Wegfluss  $f_P$  ist  $P$  ein Weg von Knoten  $v$  mit  $b(v) < 0$  zu  $u$  mit  $b(u) > 0$*
- *es gibt höchstens  $m$  Kreisströmungen*

*Falls  $f$  ganzzahlig ist, gilt dies auch für die Wegflüsse und Kreisströmungen.*

## 12. Kostenminimale Flüsse

### Definition 54 (Flusskosten)

Sei  $k: E \rightarrow \mathbb{R}$  eine Kantenbewertung. Für einen  $b$ -Fluss  $f$  sind Flusskosten

$$k(f) = \sum_{e \in E} k(e) \cdot f(e)$$

### Definition 55 (Minimalkostenflussproblem)

**Eingabe** Gerichteter Graph  $G = (V, E)$ ,  
Kapazitäten  $c: E \rightarrow \mathbb{R}^+$ ,  
Überschüssen  $b: V \rightarrow \mathbb{R}$  und  
Flusskosten  $k: E \rightarrow \mathbb{R}$ .

**Gesucht** Ein  $b$ -Fluss  $f$  mit minimalen Flusskosten  $k(f)$ .

## Theorem 56 (Zerlegung der Differenz)

Seien  $f, f'$  beide bezüglich  $l$  (untere) und  $c$  (Kapazität) zulässige  $b$ -Flüsse in  $G$ .

Dann gibt es höchstens  $2m$  Kreisströmungen  $f_i$  (zulässig in  $G_f$ )

$$\text{mit } f' = f + \sum_i f_i$$

$$\text{und es gilt } k(f') = k(f) + \sum_i k(f_i)$$

## Definition 57 (Residualnetzwerk mit unteren Schranken)

$$c_f(+r) = c(r) - f(r), \quad k(+r) = k(r)$$

$$c_f(-r) = f(r) - l(r), \quad k(-r) = -k(r)$$

Wie Definition 3 und Lemma 11 über Differenz von Flüssen

## Theorem 58 (Kreis-Kriterium)

$f$  ist genau dann ein kostenminimaler  $b$ -Fluss, wenn das Residualnetzwerk  $G_f$  keinen Kreis mit negativer Länge bezüglich der Kostenfunktion  $k$  aufweist.

## Theorem 59 (Schranke endlicher Fluss)

Falls es einen kostenminimalen  $b$ -Fluss  $f$  gibt, dann gilt für alle  $e \in E$ :  $l(e) \leq f(e) \leq (|V| + |E|)(C + B)$  mit  $C = \max\{c(e) < \infty\}$  und  $B = \max\{|b(v)|\}$

## Algorithmus von Klein

- Suche einen zulässigen Fluss
- Wiederhole: Augmentiere entlang eines Kreises negativer Länge

Die Anzahl Iterationen für ganzzahlige  $k \geq 0, c, l, b$  ist  $O(CKm)$ .



## Kriterium für Kreise negativer Länge

In einem gerichteten Graphen  $G(V, E)$  mit Kantengewichten  $k: E \rightarrow \mathbb{R}$  gibt es genau dann keinen negativen Kreis, wenn es ein Potential  $p: V \rightarrow \mathbb{R}$  gibt, so dass alle reduzierten Kosten  $k'(u, v) = k(u, v) - p(u) + p(v)$  nicht negative sind.

## Lemma 60 (Dualitätskriterium mincostflow)

*Ein  $b$ -Fluss  $f$  ist genau dann minimal, wenn es für den Residualgraph  $G_f$  ein Potential  $p$  gibt, so dass alle Kanten in  $G_f$  nicht-negative reduzierte Kosten haben.*

## Normalisierungen

- $l \equiv 0$
- $k \geq 0$
- für alle  $u, v$  gibt es eine Kante mit  $c(u, v) = +\infty$
- $c(e) < +\infty$  oder  $c \equiv +\infty$

## Annahmen über Instanzen

- 1 es existiert eine zulässiger  $b$ -Fluss ( $\sum_{v \in V} b(v) = 0$ )
- 2 für alle  $u, v$  gibt es einen Weg über Kanten mit  $c(e) = +\infty$
- 3 es gilt  $l \equiv 0$  und  $k \geq 0$

## Definition 61 (Pseudofluss)

ist eine Kantenbewertung  $0 \leq f \leq c$ .

$\text{imbal}_f(v) := \text{exc}_f(v) - b(v)$ :

- übersättigt (surplus node)  $S_f$
- unterversorgt (deficit node)  $D_f$
- befriedigt / balanciert

## Definition 62 (Reduzierte Kosten Kriterium)

Ein  $b$ -Fluss  $f$  und ein Potential  $p$ , so dass alle Kanten im Residualgraph  $G_f$  nicht-negative reduzierte Kosten haben.

## Lemma 63

Sei  $f$  Pseudofluss,  $s \in V$  beliebig,  $G' \subseteq G_f$ , so dass alle Ecken von  $s$  aus erreichbar sind. Sei  $p$  ein Potential so dass alle reduzierten Kosten in  $G'$  nicht-negativ sind, und  $d(v)$  die kürzeste Wege Distanzen von  $s$  aus in  $G'$  bezüglich der reduzierten Kosten. Dann gilt

- Die reduzierten Kosten bezüglich  $p' = p + d$  sind nicht-negativ.
- Ist  $(u, v) \in G'$  auf dem kürzesten Weg, so sind die reduzierten Kosten bezüglich  $p'$  gleich 0.

# Successive Shortest Path Algorithmus

Setze  $f \equiv 0$  und  $p \equiv 0$ , berechne  $\text{imbal}_f, S_f, D_f$

Solange  $S_f \neq \emptyset$

- 1 Wähle  $s \in S_f, t \in D_f$ , berechne  $d(v)$  bezüglich  $s, p$
- 2 Sei  $P$  ein kürzester Weg von  $s$  nach  $t$
- 3 bestimme  $\varepsilon := \min\{\Delta = c_f(P), \text{imbal}_f(s), -\text{imbal}_f(t)\}$
- 4 Erhöhe  $f$  längs  $P$  um  $\varepsilon$ , aktualisiere  $p := p + d$
- 5 Aktualisiere  $G_f, S_f, D_f, \text{imbal}_f$

## Lemma 64

*Der Algorithmus berechnet für ganzzahlige Eingaben einen kostenminimalen  $b$ -Fluss.*

*Die Anzahl der Iterationen ist  $O(nB)$*

# Capacity Scaling Algorithm

Setze  $f \equiv 0$  und  $p \equiv 0$ , berechne  $\text{imbal}$ ,  $S_f, D_f, U = B + C$

Setze  $\Delta = 2^{\lceil \log U \rceil}$

Solange  $\Delta \geq 1$

- 1 Sättige Kanten mit negativen Kosten
- 2 Berechne  $S_f(\Delta), D_f(\Delta), G_f(\Delta)$
- 3 Solange beide  $S_f(\Delta), D_f(\Delta) \neq \emptyset$ 
  - Wähle  $s \in S_f(\Delta), t \in D_f(\Delta)$ , berechne  $d(v)$  bezüglich  $s, p$
  - Sei  $P$  ein kürzester Weg von  $s$  nach  $t$
  - Erhöhe  $f$  längs  $P$  um  $\Delta$ , aktualisiere  $p := p + d$
  - Aktualisiere  $G_f(\Delta), S_f(\Delta), D_f(\Delta)$

Setze  $\Delta := \Delta/2$

## Lemma 65

*Alg. ist korrekt, Anzahl Flusserhöhungen  $O((n + m) \log U)$ ,  
Gesamtlaufzeit  $O((m + n)(m + n \log n) \log U)$*

# Kapitel VIII Textsuche

## 1. Begriffe und Notation



Volker Heun:

*Grundlegende Algorithmen — Einführung in den Entwurf und die Analyse effizienter Algorithmen.*

p. 215, Vieweg Verlag: Braunschweig-Wiesbaden, 2003

## 2. Der Algorithmus von Knuth-Morris-Pratt



Volker Heun:

*Grundlegende Algorithmen — Einführung in den Entwurf und die Analyse effizienter Algorithmen.*

pp. 216–218, Vieweg Verlag: Braunschweig-Wiesbaden, 2003



Volker Heun:

*Grundlegende Algorithmen — Einführung in den Entwurf und die Analyse effizienter Algorithmen.*

pp. 218–221, Vieweg Verlag: Braunschweig-Wiesbaden, 2003



Donald E. Knuth, James H. Morris, Vaughan R. Pratt:

*Fast pattern matching in strings.*

SIAM J. Comput. **6** pp. 323–350, Society for Industrial and Applied Mathematics: Philadelphia, PA, 1977



### 3. Der Algorithmus von Boyer und Moore



Volker Heun:

*Grundlegende Algorithmen — Einführung in den Entwurf und die Analyse effizienter Algorithmen.*

pp. 221–224, Vieweg Verlag: Braunschweig-Wiesbaden, 2003

## Der Algorithmus von Boyer und Moore (Forts.):



Volker Heun:

*Grundlegende Algorithmen — Einführung in den Entwurf und die Analyse effizienter Algorithmen.*

pp. 224–228, Vieweg Verlag: Braunschweig-Wiesbaden, 2003

## Der Algorithmus von Boyer und Moore (Forts.):



Volker Heun:

*Grundlegende Algorithmen — Einführung in den Entwurf und die Analyse effizienter Algorithmen.*

pp. 228–230, Vieweg Verlag: Braunschweig-Wiesbaden, 2003



Robert S. Boyer, J. Strother Moore:

*A fast string searching algorithm.*

Comm. ACM **20** pp. 762–772, ACM Press: New York, 1988



Zvi Galil:

*On improving the worst case running time of the Boyer-Moore string matching algorithm.*

Comm. ACM **22** pp. 505–508, ACM Press: New York, 1988

## Einige weitere interessante Artikel:



Zvi Galil:

*String Matching in Real Time.*

J. ACM **28** pp. 134–149, ACM Press: New York, 1981



Zvi Galil, Joel Seiferas:

*Time-Space-Optimal String Matching.*

J. Comput. Syst. Sci. **26** pp. 280–294, Academic Press: New York-San Francisco-London-San Diego, 1983