

# Fortgeschrittene Netzwerk- und Graph-Algorithmen

Prof. Dr. Hanjo Täubig

Lehrstuhl für Effiziente Algorithmen  
(Prof. Dr. Ernst W. Mayr)  
Institut für Informatik  
Technische Universität München

Wintersemester 2010/11



# Cycle-type normal cactus representations

## Definition

In einem Kaktus  $\mathcal{R}$  nennen wir einen Kreis mit  $h$  Knoten einen  $h$ -Kreis.

Einen Knoten, der zu genau  $k$  Kreisen gehört, nennt man  $k$ -Verzweigungsknoten.

Eine Kaktus-Repräsentation heißt *normal*, wenn sie keinen leeren 2-Verzweigungsknoten enthält, der zu einem 2-Kreis gehört.

Eine Kaktus-Repräsentation heißt *cycle-type*, wenn sie keine leere 3-Verzweigung enthält.

Eine *cycle-type normal cactus representation* nennt man auch kurz **CNCR**.

# CNCRs

## Lemma (Nagamochi/Kameda)

*Angenommen es gibt für eine Teilmenge  $\mathcal{C}' \subseteq \mathcal{C}(G)$  der MinCuts eines Graphen  $G$  eine Kaktus-Repräsentation  $(\mathcal{R}, \varphi)$ . Dann gilt:*

- *Es gibt eine CNCR für  $\mathcal{C}'$ .*
- *Die CNCR für  $\mathcal{C}'$  ist eindeutig.*
- *Jede CNCR für  $\mathcal{C}'$  hat höchstens  $|V(G)|$  leere Knoten.*
- *$(\mathcal{R}, \varphi)$  kann in eine CNCR für  $\mathcal{C}'$  konvertiert werden in Zeit und Platz linear in der Größe von  $(\mathcal{R}, \varphi)$ .*

## Eigenschaften von Kaktus-Repräsentationen

- Angenommen wir haben zwei Kaktus-Repräsentationen  $(\mathcal{R}_1, \varphi_1)$  und  $(\mathcal{R}_2, \varphi_2)$  für Teilmengen  $\mathcal{C}_1, \mathcal{C}_2$  von  $\mathcal{C}(G)$ .
- Nagamochi/Kameda haben gezeigt, dass es bei Existenz von Knoten  $z_1 \in V(\mathcal{R}_1)$  und  $z_2 \in V(\mathcal{R}_2)$  mit

$$\varphi_1^{-1}(z_1) \cup \varphi_2^{-1}(z_2) = V(G)$$

eine Kaktus-Repräsentation  $(\mathcal{R}, \varphi)$  für  $\mathcal{C}_1 \cup \mathcal{C}_2$  gibt, wo man  $\mathcal{R}$  durch Identifikation von Knoten  $z_1$  mit Knoten  $z_2$  (zum neuen Knoten  $z$ ) erhält und die

Abbildung  $\varphi : V(G) \rightarrow V(\mathcal{R}_1) \cup V(\mathcal{R}_2) \cup \{z\} \setminus \{z_1, z_2\}$  wie folgt definiert ist:

$$\begin{aligned} \varphi^{-1}(z) &= \varphi_1^{-1}(z_1) \cap \varphi_2^{-1}(z_2) \\ \varphi^{-1}(x_1) &= \varphi_1^{-1}(x_1) \text{ für alle Knoten } x_1 \in V(\mathcal{R}_1) \setminus z_1 \\ \varphi^{-1}(x_2) &= \varphi_2^{-1}(x_2) \text{ für alle Knoten } x_2 \in V(\mathcal{R}_2) \setminus z_2 \end{aligned}$$

# Eigenschaften von Kaktus-Repräsentationen

- Die so definierte Repräsentation bezeichnen wir mit  $(\mathcal{R}_1, \varphi_1) \oplus (\mathcal{R}_2, \varphi_2) = (\mathcal{R}, \varphi)$  und die entsprechenden Knoten  $z_1, z_2$  heißen **verbundene Knoten**.
- Der neue Knoten  $z$  ist immer ein Artikulationsknoten in  $\mathcal{R}$ .  
Er ist genau dann leer in  $(\mathcal{R}, \varphi)$ , wenn  $\varphi_1^{-1}(z_1) \cap \varphi_2^{-1}(z_2) = \emptyset$  gilt.

# Kritische Kanten und $(s, t)$ -MC-Partition

Eine Kante  $e = (s, t)$  heißt *kritisch*, falls  $c_G(e) > 0$  und  $\lambda_G(s, t) = \lambda_G$ .

## Lemma (Karzanov/Timofeev)

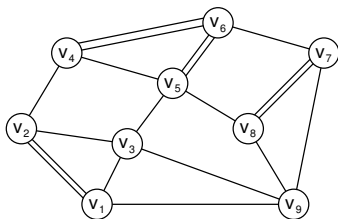
*Für jede kritische Kante  $e = (s, t)$  sind beliebige MinCuts, die  $s$  und  $t$  separieren, keine Crossing Cuts.*

*D.h. es gibt eine geordnete Partition  $(V_1, \dots, V_r)$  von  $V(G)$ , so dass die Menge der Cuts der Form  $\{V_1 \cup V_2 \cup \dots \cup V_i, V_{i+1} \cup \dots \cup V_r\}$  gleich der Menge der MinCuts in  $\mathcal{C}(G)$  ist, die  $s$  und  $t$  separieren.*

Solch eine geordnete Partition nennt man  **$(s, t)$  minimum cut o-partition** oder kurz  **$(s, t)$ -MC-Partition**.

# Beispiel für $(s, t)$ -MC-Partition

Beispiel:



Die  $(s, t)$ -MC-Partition für  $(s, t) = (v_1, v_9)$  ist  $\{V_1 = \{v_1\}, V_2 = \{v_2\}, V_3 = \{v_3\}, V_4 = \{v_4, v_5, v_6\}, V_5 = \{v_7, v_8\}, V_6 = \{v_9\}\}$ .

# Berechnung der $(s, t)$ -MC-Partition

## Lemma

*Sei  $(s, t)$  eine kritische Kante in einem Graphen.*

*Wenn ein beliebiger Maximum Flow zwischen  $s$  und  $t$  gegeben ist, kann die  $(s, t)$ -MC-Partition in Zeit und Platz  $\mathcal{O}(m + n)$  berechnet werden.*

(Beweis: Karzanov/Timofeev, Naor/Vazirani)



## Mit Partition $\pi$ kompatible und unteilbare Cuts

Sei  $\pi$  eine Partition  $\{V_1, V_2, \dots, V_r\}$  oder eine geordnete Partition  $(V_1, V_2, \dots, V_r)$  von  $V(G)$ .

Wir nennen einen Cut  $\{X, \bar{X}\}$  **kompatibel mit  $\pi$** , falls

$$X = \bigcup_{i \in I} V_i \text{ für ein } I \subset \{1, 2, \dots, r\}$$

Wir nennen einen Cut  $\{X, \bar{X}\}$  **unteilbar mit  $\pi$** , falls

$$X \subset V_i \text{ für ein } i \in \{1, 2, \dots, r\}$$

Ein Cut  $\{X, \bar{X}\}$  kreuzt eine Partition  $\{V_1, V_2, \dots, V_r\}$  oder eine geordnete Partition  $(V_1, V_2, \dots, V_r)$ , falls ein  $V_i$  existiert, so dass keine der Teilmengen  $X \cap V_i$ ,  $X \setminus V_i$  und  $V_i \setminus X$  leer ist.

Jeder Cut, der  $\pi$  nicht kreuzt, ist entweder kompatibel oder unteilbar bezüglich  $\pi$ . Wir bezeichnen die entsprechenden Mengen von MinCuts mit  $\mathcal{C}_{\text{comp}}(\pi)$  und  $\mathcal{C}_{\text{indv}}(\pi)$

## Mit Partition $\pi_{(s,t)}$ kompatible und unteilbare Cuts

### Lemma

Sei  $(s, t)$  eine kritische Kante in einem Graph  $G$  und  $\pi_{(s,t)}$  die  $(s, t)$ -MC-Partition über  $\mathcal{C}(G)$ . Dann ist jeder MinCut  $\{X, \bar{X}\} \in \mathcal{C}(G)$  entweder kompatibel oder unteilbar bezüglich  $\pi_{(s,t)}$ , d.h.

$$\mathcal{C}(G) = \mathcal{C}_{\text{comp}}(\pi_{(s,t)}) \cup \mathcal{C}_{\text{indv}}(\pi_{(s,t)}).$$

Man beachte, dass  $\mathcal{C}_{\text{comp}}(\pi_{(s,t)})$  einen MinCut enthalten kann, der  $s$  und  $t$  nicht separiert.

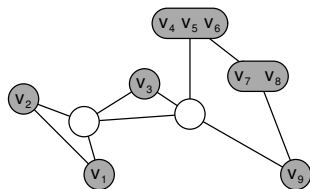
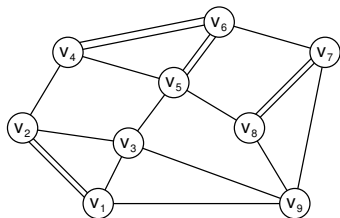
### Satz (Nagamochi/Kameda)

Sei  $(s, t)$  eine kritische Kante eines Graphen  $G$  und  $\pi_{(s,t)}$  die  $(s, t)$ -MC-Partition.

Dann existiert eine Kaktus-Repräsentation  $(\mathcal{R}_{(s,t)}, \varphi_{(s,t)})$  für alle MinCuts in  $\mathcal{C}_{\text{comp}}(\pi_{(s,t)})$ , die man  $(s, t)$ -Kaktus-Repräsentation nennt.

Weiterhin kann für gegebenes  $\pi_{(s,t)}$  die cycle-type normal  $(s, t)$ -cactus representation  $((s, t)$ -CNCR) in  $\mathcal{O}(m + n)$  Zeit und Platz konstruiert werden.

# Mit Partition $\pi_{(s,t)}$ kompatible und unteilbare Cuts



**Abbildung:** Cycle-type normal  $(s, t)$ -cactus representation mit  $(s, t) = (v_1, v_9)$  über  $\mathcal{C}(G)$

# Berechnungsgrundlage

Grundlage für den NNI-Algorithmus:

## Satz

*In einem Graphen  $G$  kann man eine Kante  $E = (s, t)$  mit  $c_G(e) > 0$ , die die folgenden zwei Bedingungen erfüllt, in Zeit  $\mathcal{O}(m + n \log n)$  und Platz  $\mathcal{O}(m + n)$  berechnen:*

- 1  $\lambda_G(s, t)$  kann in Zeit  $\mathcal{O}(m + n \log n)$  und Platz  $\mathcal{O}(m + n)$  berechnet werden.*
- 2 Wenn  $\lambda_G(s, t) = \lambda_G$ , dann kann die  $(s, t)$ -CNCR in Zeit  $\mathcal{O}(m + n \log n)$  und Platz  $\mathcal{O}(m + n)$  berechnet werden.*

# Maximum Adjacency Ordering

## Definition

Eine totale Ordnung  $v_1, v_2, \dots, v_n$  aller Knoten in  $V(G)$  bezeichnet man als **Maximum Adjacency Ordering (MAO)**, falls für alle  $i, j$  mit  $1 \leq i \leq j \leq n - 1$  gilt:

$$w(\{v_1, v_2, \dots, v_{i-1}\}, v_i) \geq w(\{v_1, v_2, \dots, v_{i-1}\}, v_j)$$

Verbal: Die Anbindung des Knotens  $v_i$  an seine Vorgänger  $v_1, v_2, \dots, v_{i-1}$  ist mindestens so groß wie die Anbindung jedes einzelnen nachfolgenden Knotens  $v_{i+1}, \dots, v_n$  an die Vorgänger von  $v_i$ .

# Maximum Adjacency Ordering

## Lemma

*Ein MAO eines Graphen  $G$  kann in Zeit  $\mathcal{O}(m + n \log n)$  und Platz  $\mathcal{O}(m + n)$  berechnet werden.*

*Für ein MAO  $v_1, v_2, \dots, v_n$  in  $G$  gilt für die letzten beiden Knoten  $v_{n-1}, v_n$ , dass  $\lambda_G(v_{n-1}, v_n) = w(\{v_n\}, V_G \setminus \{v_n\})$ .*

# Beweis des letzten Satzes

## Beweis.

- Berechne zuerst ein MAO  $v_1, v_2, \dots, v_n$  von  $G$ .
- Wähle den Knoten  $v_p$  mit höchstem Index  $p$ , so dass  $v_p$  und  $v_n$  durch eine Kante (mit pos. Gewicht) verbunden sind.
- Sei  $s = v_n$  und  $t = v_p$ .
- Man beachte, dass  $v_1, v_2, \dots, v_p, v_n$  ein MAO in dem Graphen  $G'$  ist, der aus  $G$  entsteht, wenn man die Knoten  $v_{p+1}, \dots, v_{n-1}$  löscht.
- D.h. nach dem letzten Lemma gilt:  
$$\lambda_G(s, t) \geq \lambda_{G'}(s, t) = w_{G'}(\{s\}, V(G') \setminus \{s\}) = w_G(\{s\}, V(G) \setminus \{s\})$$
- Da  $\lambda_G(s, t) \leq w_G(\{s\}, V(G) \setminus \{s\})$ , folgt der 1. Teil des Satzes.

# Beweis des letzten Satzes

## Beweis.

- Annahme:  $\lambda_G(s, t) = \lambda_G$  (wie im 2. Teil des Satzes)
  - Ein Maximum Flow zwischen den letzten beiden Knoten eines MAO kann in  $\mathcal{O}(m)$  Zeit und Platz berechnet werden (Arikati/Mehlhorn).
- ⇒ MaxFlow zwischen  $s$  und  $t$  kann in  $\mathcal{O}(m)$  gefunden werden.
- Für kritische Kante  $(s, t)$  mit gegebenem MaxFlow zwischen  $s$  und  $t$  kann nach Lemma die  $(s, t)$ -MC-Partition in  $\mathcal{O}(m + n)$  Zeit und Platz bestimmt werden.
  - Nach vorhergehendem Satz existiert für die kritische Kante  $(s, t)$  und die  $(s, t)$ -MC-Partition  $\pi_{(s,t)}$  eine Kaktus-Repräsentation für alle MinCuts in  $\mathcal{C}_{\text{comp}}(\pi_{(s,t)})$ , genannt  $(s, t)$ -Kaktus-Repräsentation, deren CNCR in  $\mathcal{O}(m + n)$  Zeit und Platz konstruiert werden kann.
  - Die cycle-type normal  $(s, t)$ -cactus representation kann in Zeit  $\mathcal{O}(m)$  aus dem MaxFlow berechnet werden.





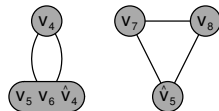
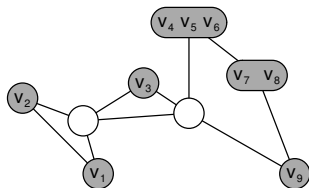
## Berechnung der Kaktus-Repräsentation

- Sei im folgenden  $G^*$  der Eingabe-Graph, für den eine Kaktus-Repräsentation berechnet werden soll.
- Die Konstruktion der Kaktus-Repräsentation von  $\mathcal{C}(G^*)$  erfolgt rekursiv auf der Basis des letzten Satzes.
- $G/X$  stellt den Graph dar, den man aus  $G$  erhält, wenn man alle Knoten in  $X$  zu einem einzigen Knoten kontrahiert.
- Sei  $\lambda = \lambda(G^*)$ .
- Wir wählen eine Kante  $(s, t)$  entsprechend dem letzten Satz.
- Falls  $\lambda_G(s, t) > \lambda$ , dann kontrahiere Knoten  $\{s, t\}$  (kein MinCut in  $G$  separiert  $s$  und  $t$ ).
- Falls  $\lambda_G(s, t) = \lambda$ , dann bestimme  $(s, t)$ -MC-Partition  $\pi_{(s,t)} = (V_1, \dots, V_r)$  und  $(s, t)$ -Kaktus-Repräsentation  $(\mathcal{R}_{(s,t)}, \varphi_{(s,t)})$  in  $G$ .
- Nach Lemma sind alle mit  $\pi_{(s,t)}$  kompatiblen MinCuts durch  $(\mathcal{R}_{(s,t)}, \varphi_{(s,t)})$  repräsentiert und jeder mit  $\pi_{(s,t)}$  unteilbare MinCut  $\{X, V(G) \setminus X\}$  erfüllt  $X \subset V_i$  für ein  $V_i \in \pi_{(s,t)}$

# Berechnung der Kaktus-Repräsentation

- Sei  $G_i = G/(V(G) \setminus V_i)$  der Graph, bei dem die Knoten  $V(G) \setminus V_i$  ( $i = 1, \dots, r$ ) zu einem Knoten  $\hat{v}_i$  kontrahiert wurden, wobei gilt:  
 $\lambda_{G_i} \geq \lambda_G$ .
- Annahme: für jedes  $G_i$  wurde eine Kaktus-Repräsentation  $(\mathcal{R}_i, \varphi_i)$  rekursiv berechnet, wobei  $(\mathcal{R}_i, \varphi_i)$  der triviale Kaktus sein soll falls  $\lambda_{G_i} > \lambda_G$ .
- Dann sind alle MinCuts repräsentiert in  $(\mathcal{R}_{(s,t)}, \varphi_{(s,t)})$ ,  
 $(\mathcal{R}_1, \varphi_1), \dots, (\mathcal{R}_r, \varphi_r)$ .
- Weiterhin können diese Repräsentationen zu einer einzigen vereinigt werden, indem die Knoten, die  $\hat{v}_i$  enthalten, als verbundene Knoten betrachtet werden.

# Berechnung der Kaktus-Repräsentation



Kaktus-Repräsentationen  $(\mathcal{R}_i, \varphi_i)$ , wobei sich in den Fällen  $i = \{1, 2, 3, 6\}$  der triviale Kaktus ergibt.

Auf dem Bild sieht man  $(\mathcal{R}_4, \varphi_4)$  und  $(\mathcal{R}_5, \varphi_5)$ .

## Berechnung der Kaktus-Repräsentation

- Wenn  $\text{CACTUS}(G', V^{\text{old}})$  für einen Graph  $G'$  während der Ausführung von  $\text{CACTUS}(G'', V^{\text{old}})$  für einen Graph  $G''$  aufgerufen wird, nennen wir  $G'$  ein Kind von  $G''$  und  $G''$  den Vater von  $G'$ .
- Die Vater-Kind-Beziehung induziert einen Baum  $\mathcal{T}$ , der am Eingabe-Graph  $G^*$  gewurzelt ist (Aufrufbaum).
- Bemerkung: Falls  $w(V_i, V(G) \setminus V_i) = \lambda$ , dann bleibt der Cut  $\{V_i, \hat{v}_i\}$  ein MinCut in einem Kind  $G_i$ , obwohl der Cut schon in seinem Vater  $G$  erkannt wurde.
- Derselbe MinCut kann in einem Nachfolger von  $G_i$  sein.
- Ein MinCut ist alt in einem Graph  $G'$  (also wurde schon in einem Vorfahren entdeckt), nur wenn ein einzelner Knoten  $v$  von  $V(G') \setminus \{v\}$  separiert wird.
- D.h. wir können testen, ob die  $(s, t)$ -Kaktus-Repräsentation neue MinCuts enthält, indem wir die Knoten  $v \in V(G')$  als 'alt' markieren, falls  $v$  ein kontrahierter Knoten  $\hat{v}_i$  im Vorgänger ist.

# NNI-Alg. für Kaktus-Repräsentation

## Algorithmus 11 : CACTUS( $G, V^{\text{old}}$ )

**Input** : Graph  $G$ , Teilmenge  $V^{\text{old}} \subset V(G)$

**Output** : Kaktus-Repr.  $(\mathcal{R}, \varphi)$  für eine Menge  $\mathcal{C}'$  von MinCuts, so dass  
 $\mathcal{C}(G) \setminus \{\{\bar{v}, V(G) \setminus \{\bar{v}\}\} \mid \bar{v} \in V^{\text{old}}\} \subseteq \mathcal{C}' \subseteq \mathcal{C}(G)$

**if**  $|V(G)| = 1$  **then return** *Trivial-Kaktus*  $(\mathcal{R}, \varphi)$ ;

**else**

Wähle Kante  $e = (s, t) \in E(G)$  mit  $w_G(e) > 0$ ;

**if**  $\lambda_G(s, t) > \lambda$  **oder**  $(s, t)$ -Kaktus Repr.  $(\mathcal{R}_{(s,t)}, \varphi_{(s,t)})$  *repräsentiert keinen anderen Cut außer die folgenden*:  $\{\bar{v}, V(G) \setminus \{\bar{v}\}\}, \bar{v} \in V^{\text{old}}$  **then**

$G = G/\{s, t\}; \quad V^{\text{old}} = V^{\text{old}} \setminus \{s, t\};$   
**return** CACTUS( $G, V^{\text{old}}$ )

**else**

**foreach**  $V_i$  *in der*  $(s, t)$ -MC-Partition  $\pi_{(s,t)} = (V_1, \dots, V_r)$  **do**

$G_i = G/(V(G) \setminus V_i)$ , wobei  $\bar{v}_i$  den Knoten bezeichnet, der durch die Kontraktion von  $V(G) \setminus V_i$  entsteht;

$V_i^{\text{old}} = (V^{\text{old}} \cap V_i) \cup \{\bar{v}_i\};$

$(\mathcal{R}_i, \varphi_i) = \text{CACTUS}(G_i, V_i^{\text{old}})$

$(\mathcal{R}, \varphi) = (\mathcal{R}_{(s,t)}, \varphi_{(s,t)}) \oplus (\mathcal{R}_1, \varphi_1) \oplus \dots \oplus (\mathcal{R}_r, \varphi_r);$

**return**  $(\mathcal{R}, \varphi)$  *nach Konvertierung in CNCR*

# NNI-Alg. für Kaktus-Repräsentation

---

## Algorithmus 12 : CONSTRUCT

---

**Input** : Graph  $G^*$

**Output** : CNCR  $(\mathcal{R}, \varphi)$  für  $\mathcal{C}(G)$

Compute  $\lambda = \lambda(G)$ ;

$V^{\text{old}} = \emptyset$ ;

$(\mathcal{R}, \varphi) = \text{CACTUS}(G^*, V^{\text{old}})$

---

## Satz

*Der vorgestellte Algorithmus berechnet eine Kaktus-Repräsentation für alle Minimum Cuts in Zeit  $\mathcal{O}(mn + n^2 \log n)$  und Platz  $\mathcal{O}(m + n)$ .*