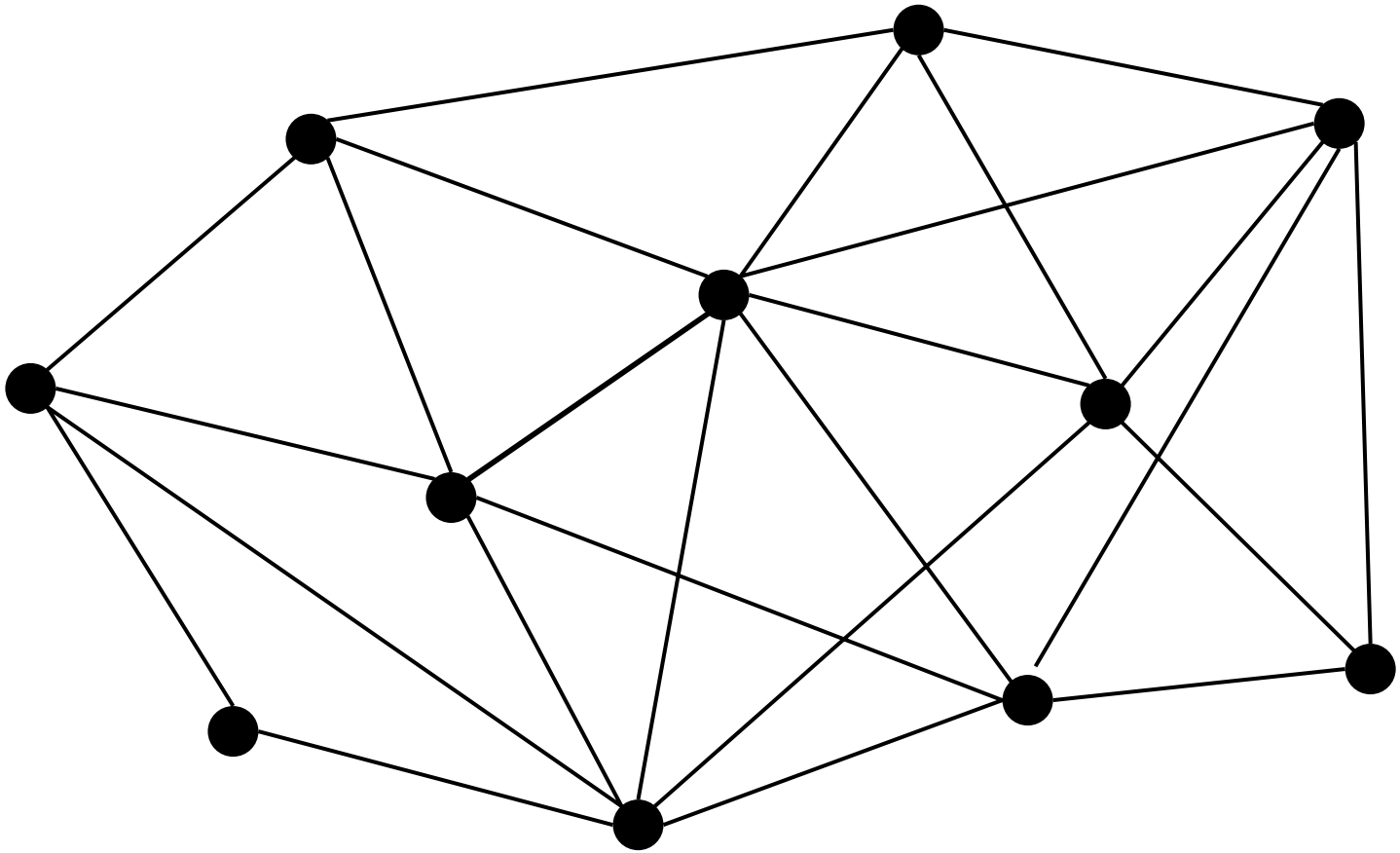
Algorithmentheorie

17 – Minimale Schnitte

Prof. Dr. S. Albers

1. Minimale Schnitte



Minimale Schnitte

Eingabe: Ungerichteter Graph $G = (V,E)$

Ausgabe: $V_1, V_2, \subseteq V$, so dass $V_1 \cup V_2 = V$, $V_1 \cap V_2 = \emptyset$ und die Anzahl der Kanten zwischen V_1 und V_2 so klein wie möglich ist.

$c_{min}(G)$ = # Kanten eines minimalen Schnitts von G

Ein Schnitt wird auch oft durch die Kanten zwischen V_1, V_2 repäsentiert

Gewichtetes Problem: Kante e hat Gewicht $w(e)$.
Bestimme Schnitt minimalen Gewichts.

Reduktion auf Flussproblem: Für alle Paare $s, t \in V$ berechne maximalen (s,t)- Fluss. Zeit $O(n^5)$

2. Randomisierter Algorithmus

Multigraph: Zwischen zwei Knoten können mehrere Kanten existieren.

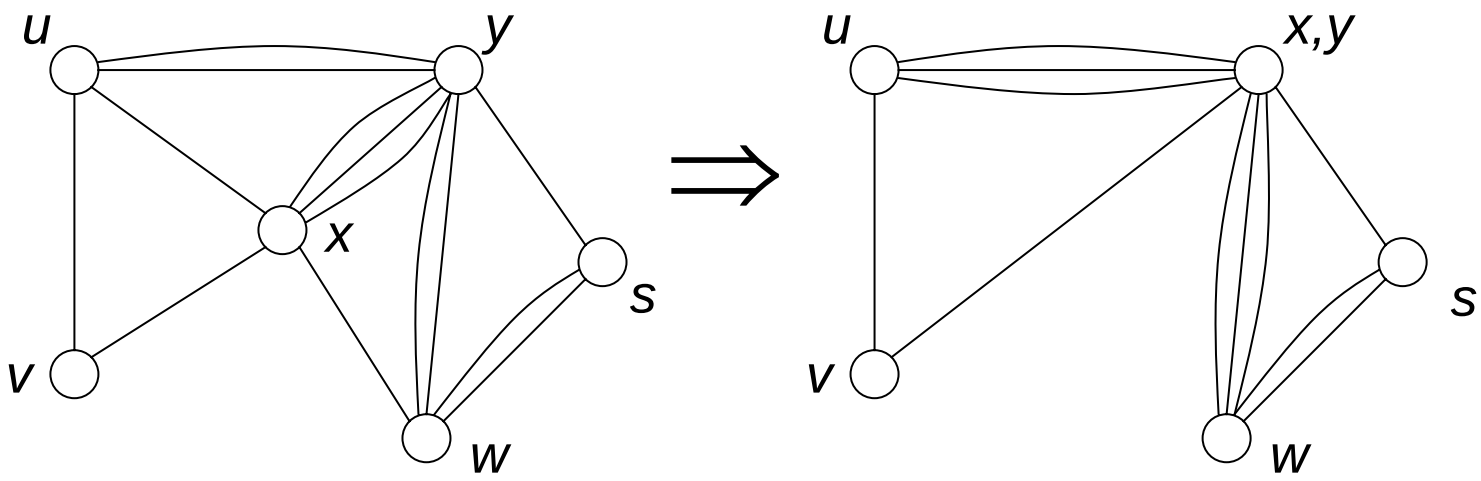
Grundlegende Operation: Kontraktion einer Kante $e = \{x,y\}$.

Ersetze x,y durch einen Metaknoten z

Für $v \notin \{x,y\}$ ersetze $\{v,x\}$ durch $\{v,z\}$

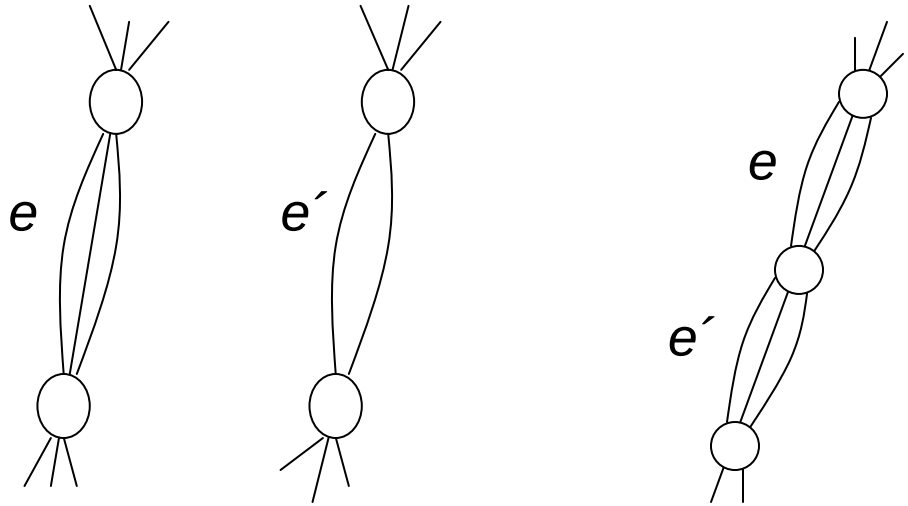
ersetze $\{v,y\}$ durch $\{v,z\}$

→ $G \setminus \{e\}$



Kontraktion

Die Reihenfolge von Kontraktionen spielt keine Rolle.



Eine Kontraktion kann in Zeit $O(n)$ implementiert werden.

- Für jeden Metaknoten speichere die Anzahl der Kanten zu anderen Knoten.
- Für jeden Eingabeknoten speichere den Namen des Metaknotens.

Algorithmus *Kontraktion*

Algorithmus *Kontraktion*;

1. $H \leftarrow G$;
2. **while** H hat mehr als zwei Knoten **do**
3. Wähle zufällig eine Kante e aus H;
4. $H \leftarrow H \setminus \{e\}$;
5. **endwhile**;
6. Seien V_1, V_2 die Knotenmengen, die durch die letzten beiden Knoten von H repräsentiert werden.

Laufzeit: $O(n^2)$

Eigenschaften

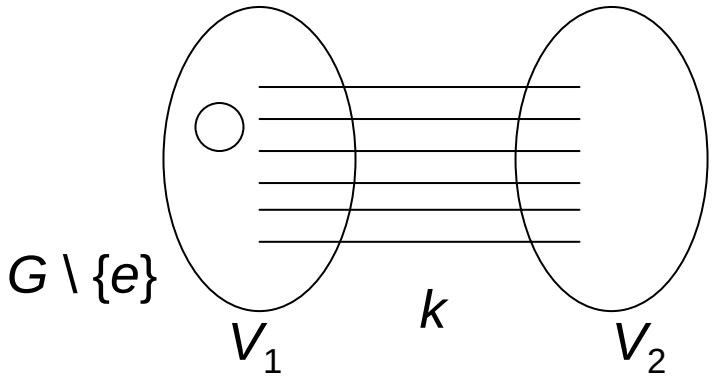
Lemma 1 : Partition V_1, V_2 ist Ergebnis des Algorithmus *Kontraktion* gdw keine Kante zwischen V_1 und V_2 kontrahiert wird.

Lemma 2: Ist $c_{min}(G) = k$, dann haben alle Knoten Grad $\geq k$ und G hat $\geq nk/2$ Kanten.

Eigenschaften

Lemma 3: Für jede Kante e in G gilt $c_{min}(G) \leq c_{min}(G \setminus \{e\})$

Beweis: Partition V_1, V_2 von $G \setminus \{e\}$ mit k Kanten ist auch Partition vom G mit k Kanten.



Erfolgswahrscheinlichkeit

Satz 1: Sei C ein minimaler Schnitt in G.
Kontraktion gibt C mit Wahrscheinlichkeit $\geq 2/n^2$ aus.

Beweis: Sei $c_{min}(G) = k$.
Betrachte i-te Iteration der while-Schleife.
H hat $n_i = n - i + 1$ Knoten.
Angenommen, die ersten $i - 1$ Iterationen kontrah. keine Kante von C.
C ist Schnitt von H, und wegen Lemma 3 gilt $c_{min}(H) = k$.
Wegen Lemma 2 hat H mindestens $n_i k/2$ Kanten.

Prob[i-te Iterationen kontrahiert eine Kante von C] $\leq 2/n_i$

Prob[i-te Iterationen kontrahiert keine Kante von C] $\geq 1 - 2/n_i$

Erfolgswahrscheinlichkeit

Prob[*C* wird ausgegeben]

$$\geq \prod_{i=1}^{n-2}\left(1-\frac{2}{n_i}\right) = \prod_{i=1}^{n-2}\frac{n-i-1}{n-i+1} = \prod_{j=3}^{n}\frac{j-2}{j} = \frac{1\cdots(n-2)}{3\cdots n}$$

$$= \frac{2}{n(n-1)} \geq \frac{2}{n^2}$$

Erhöhung der Erfolgswahrscheinlichkeit

Wiederhole *Kontraktion* $dn^2 \ln n$ Male und wähle den kleinsten Schnitt.

$$\text{Prob}\left[C \text{ wird nicht gefunden}\right] \leq \left(1 - \frac{2}{n^2}\right)^{n^2 d \ln n} \leq e^{-2d \ln n} = n^{-2d}$$

3. Verbesserte Laufzeit

Lemma 4: Sei *C* ein minimaler Schnitt. Bricht K*ontraktion* ab, wenn genau *t* Knoten übrig sind, dann gilt

$$\text{Prob}[\text{keine Kante von } C \text{ wird kontrahiert}] \geq \frac{t(t-1)}{n(n-1)}.$$

Beweis:

$$\prod_{i=1}^{n-t}\left(1-\frac{2}{n_i}\right) = \prod_{i=1}^{n-t}\frac{n-i-1}{n-i+1} = \prod_{j=t+1}^{n}\frac{j-2}{j} = \frac{(t-1)\cdots(n-2)}{(t+1)\cdots n}$$

$$= \frac{(t-1)t}{n(n-1)}$$

Algorithmus *Fast-Cut*

Eingabe: Multigraph $G = (V, E)$.

Ausgabe: Eine Zerlegung $V = V_1 \cup V_2$ bzw. zugehörige Kantenmenge.

1. $n \leftarrow |V|$
2. **if** $n \leq 6$ **then**
3. Berechne minimalen Schnitt durch vollständige Enumeration;
4. **else**
5. $t \leftarrow \left\lceil 1 + n/\sqrt{2} \right\rceil$ $/* \frac{t(t-1)}{n(n-1)} \geq \frac{1}{2}$
6. Wende *Kontraktion* zweimal an, so dass jeweils genau t Knoten übrig sind. Seien H_1 und H_2 die resultierenden Graphen;
7. Wende *Fast-Cut* rekursiv auf H_1 und H_2 an;
8. Gib den kleineren Schnitt aus;
9. **endif;**

Algorithmus *Fast-Cut*

Satz 2: *Fast-Cut* hat Laufzeit $O(n^2 \log n)$.

Beweis: *Kontraktion* hat Laufzeit $O(n^2)$.

$$T(n) = 2T\left(\left\lceil 1 + n/\sqrt{2} \right\rceil\right) + O(n^2)$$

Erfolgswahrscheinlichkeit

Satz 3: *Fast-Cut* findet minimalen Schnitt mit WSK $\geq \Omega(1/\log n)$.

Beweis: Sei k die Anzahl der Kanten eines minimalen Schnitts C.
Fast-Cut liefert C zurück, wenn

- während der Kontraktion auf H_1 oder H_2 keine Kante aus C kontrahiert wird und
- *Fast-Cut* auf H_i angewendet C liefert.

$P(n)$ = Prob[*Fast-Cut* findet minimalen Schnitt C in Graphen mit n Knoten]

Erfolgswahrscheinlichkeit

$$P(n) = 1 - \text{Prob}[\textit{Fast} - \textit{Cut} \text{ findet } C \text{ in keinem der beiden Versuche}]$$

$$= 1 - \prod_{i=1,2} \text{Prob}[\textit{Fast} - \textit{Cut} \text{ findet } C \text{ nicht im Versuch bzgl. } H_i]$$

$$= 1 - \prod_{i=1,2} (1 - \text{Prob}[\textit{Fast} - \textit{Cut} \text{ findet } C \text{ im Versuch bzgl. } H_i])$$

$$= 1 - \left(1 - \frac{1}{2} P(t)\right)^2$$

Erfolgswahrscheinlichkeit

$p(k)$ = untere Schranke für P, wenn es k Rekursionsstufen gibt

$$p(k+1) = 1 - \left(1 - \frac{1}{2}\,p(k)\right)^2 = p(k) - \frac{p(k)^2}{4}$$

$$p(0) = 1$$

Wir zeigen, dass $p(k) \geq 1/d$ genau $p(k+1) \geq 1/(d+1)$ impliziert.

Da $p(0) = 1 \geq 1/1$, folgt $p(k) \geq 1/(k+1) = \Omega(1/k)$.
Wir haben $O(\log n)$ Rekursionsstufen, also $P(n) = \Omega(1/\log n)$.

Erfolgswahrscheinlichkeit

$f(x) = x - x^2/4$ ist monoton steigend in [0,1]

Daher implizieren $p(k) \geq 1/d$ und $p(k) \in [0,1]$

$$
\begin{aligned}
p(k+1) &\geq \frac{1}{d} - \frac{1}{4d^2} \\
&= \frac{d+1}{d+1} \cdot \frac{4d-1}{4d^2} \\
&= \frac{1}{d+1} \frac{4d^2+3d-1}{4d^2} \\
&\geq \frac{1}{d+1}
\end{aligned}
$$

Erhöhung der Erfolgswahrscheinlichkeit

Wiederhole *Fast-Cut* $d \ln^2 n$ Male und nimm den kleinsten Schnitt.

$$\text{Prob}[\text{C wird nicht gefunden}] \leq \left(1 - \frac{c}{\ln n}\right)^{d \ln^2 n} \leq e^{-cd \ln n} = n^{-cd}$$

Laufzeit: $O(n^2 \log^3 n)$

4. Minimale gewichtete Schnitte

Gewichtetes Problem: Kante e in $G = (V,E)$ hat Gewicht $w(e) \geq 0$.

Ein minimaler (s,t)-Schnitt für $s, t \in V$ ist ein Schnitt $V_s, V_t \subseteq V$ mit $V_s \cup V_t = V$, $V_s \cap V_t = \emptyset$ und $s \in V_s, t \in V_t$ minimalen Gewichts.
Dieses Gewicht wird mit $c_{min}(G, s, t)$ bezeichnet.

Gewichtete minimale Schnitte

$G \setminus \{x,y\}$ = Graph, wenn x, y kontrahiert sind.

Gewichte von Mehrfachkanten addieren sich.

Lemma 5: Seien $s, t \in V$. Dann gilt

$$c_{min}(G) = \min\{c_{min}(G,s,t), c_{min}(G \setminus \{s,t\})\}.$$

Deterministischer Algorithmus

Algorithmus *Irgendein-(s,t)-Schnitt;*

Eingabe: Graph *G*

Ausgabe: *s*, *t* (minimaler (*s*,*t*)-Schnitt ist implizit gegeben)

1. $A \leftarrow$ {beliebiger Knoten aus V};
2. **while** $A \neq V$ **do**
3. Füge den Knoten $v \in V - A$ zu A hinzu, für den $w(v,A)$ maximal ist;
4. **endwhile;**
5. Sei *s* der zweitletzte und *t* der letzte zu *A* hinzugefügte Knoten;

$w(v,A)$ = Gesamtgewicht der Kanten zwischen v und Knoten in A

Deterministischer Algorithmus

Algorithmus *Minimaler-Schnitt*;

1. *Min* ← ∞; *N* ← |*V* |;
2. **while** *N* ≥ 2 **do**
3. Wende *Irgendein-(s,t)-Schnitt* an und erhalte *s, t* und einen Schnitt *C* mit Gewicht *W*;
4. **if** *W* < Min **then** speichere *C*; Min ← *W*; **endif**;
5. Kontrahiere *s* und *t*; *N* ← *N* – 1;
6. **endwhile;**
7. Liefere *Min* und den zuletzt gespeicherten Schnitt zurück;

Analyse

Satz 4: Die von *Irgendein-(s,t)-Schnitt* berechneten $V_t = \{t\}$ und $V_s = V - \{t\}$ sind ein minimaler (s,t)-Schnitt.

Beweis:

Nummeriere Knoten von 1 bis n so, dass Knoten i der in der i-ten Iteration zu A hinzugefügte Knoten ist.

$s = n - 1$ und $t = n$

Sei C ein beliebiger (s,t)-Schnitt.

C_i = Kanten in C, die beide Endknoten in $\{1,...,i\}$ haben

$w(C_i)$ = Gesamtgewicht der Kanten in C_i

Analyse

Knoten *i* ist aktiv, wenn *i* und *i* – 1 auf verschiedenen Seiten von *C* liegen.

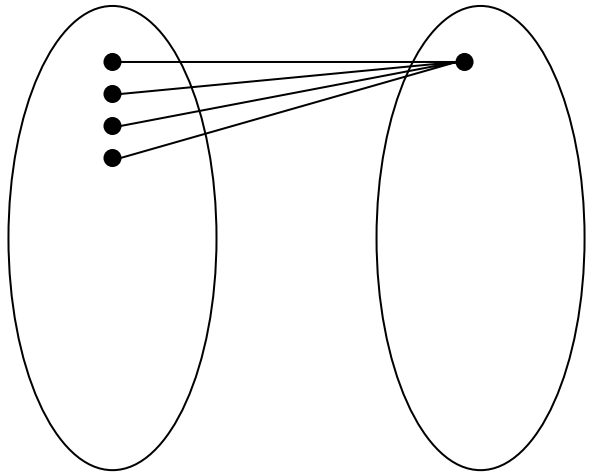
Behauptung: Für jeden aktiven Knoten *i* gilt $w(i, \{1, ..., i-1\}) \leq w(C_i)$.

n ist aktiv und daher $w(n, \{1, ..., n-1\}) \leq w(C)$.

Analyse

Behauptung: Für jeden aktiven Knoten i gilt $w(i,\{1,..,\ i-1\}) \leq w(C_i)$.

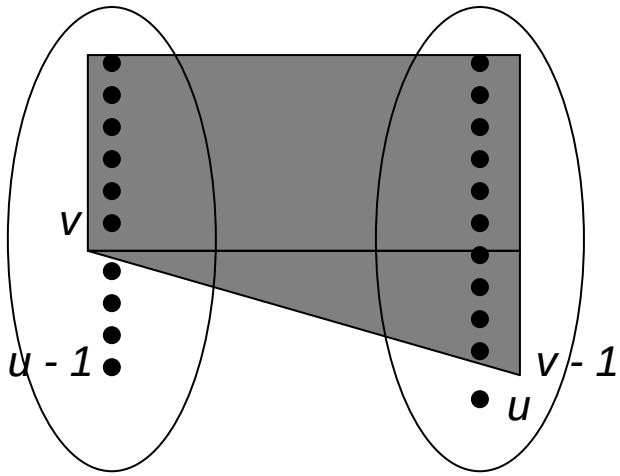
Beweis: Die Aussage gilt für den ersten aktiven Knoten.



Analyse

Angenommen, die Aussage gilt für aktiven Knoten *v* und der nächste aktive Knoten ist *u*.

$$
\begin{aligned}
w(u,\{1,\ldots,u-1\}) &= w(u,\{1,\ldots,v-1\}) + w(u,\{v,\ldots,u-1\}) \\
&\leq w(v,\{1,\ldots,v-1\}) + w(u,\{v,\ldots,u-1\}) \\
&\leq w(C_v) + w(u,\{v,\ldots u-1\}) \\
&\leq w(C_u)
\end{aligned}
$$



Analyse

Satz 5: *Minimaler-Schnitt* berechnet einen minimalen Schnitt in Zeit $O(en + n^2 \log n)$.

Beweis: Korrektheit: Induktion nach n.

Algorithmus arbeitet korrekt für Multigraphen mit $n = 2$ Knoten.

$n \rightarrow n + 1$: *Irgendein-(s,t)-Schnitt* berechnet s,t und c_{min} (G,s,t) korrekt.
MinimalerSchnitt berechnet c_{min} $(G \setminus \{s,t\})$ korrekt.

Irgendein-(s,t)-Schnitt hat Laufzeit $O(e + n \log n)$.

Verwalte Prioritätswarteschlange für $v \in V - A$ mit $key(v) = w(v,A)$.

n *DeleteMax*- und e *IncreaseKey*-Operationen.