Kapitel 10

Schnittprobleme

Effiziente Algorithmen, SS 2018

Professor Dr. Petra Mutzel

VO 18/20 am 21./28. Juni 2018

Übersicht

I. Effiziente Graphalgorithmen

2. Starke Zusammenhangskomponenten
3. Matching-Probleme
4. Maximale Flussprobleme
5. Amortisierte Analyse
6. Minimale Schnitte

II. Approximationsalgorithmen

7. Rucksackproblem, Bin Packing Problem
8. Traveling Salesman Problem
9. Erfüllbarkeitsprobleme
10. Schnittprobleme

Design-Techniken im Verlauf der Vorlesung

1. Die Greedy-Methode: Rucksackproblem (Kap. 7)
2. Dynamische Programmierung: Rucksackproblem (FPTAS) (Kap. 7)
3. Inkrementelle Algorithmen für Partitionsprobleme: Bin Packing Problem (Kap. 7)
4. Spezielle, problemabhängige Verfahren: Traveling Salesman Problem (Kap. 8)
5. Zufalls-basierte Verfahren (Kap. 9, Kap. 10)
6. LP-basierte Verfahren: MaxSAT (Kap. 9)
7. Lokale Suchverfahren: Max Cut (Kap. 10)

Schnittprobleme

Eingabe: gewichteter, ungerichteter Graph $G = (V, E)$
Kantengewichte $w(e) \in \mathbb{Q}$
(ungewichteter Graph $\widehat{=}$ alle Kantengewichte $= 1$)

zulässige Lösung: Schnitt V_1, V_2
mit $V_1 \cap V_2 = \emptyset$ und $V_1 \cup V_2 = V$

Bewertung: $w(V_1, V_2) = \sum\limits_{e \in \{\{u,v\} \mid u \in V_1, v \in V_2\}} w(e)$

Max Cut Ziel finde Schnitt mit maximalem Wert

Min Cut $w(e) \geq 0$ Ziel finde Schnitt mit minimalem Wert
mit $V_1 \neq \emptyset$ und $V_2 \neq \emptyset$

Über Max Cut

bekannt

- zugehöriges Entscheidungsproblem NP-vollständig (auch für ungewichteten Fall) (Karp 1972)
- Max Cut hat eine $1{,}14$-Approximation (Goemans und Williamson 1994)
- Max Cut hat keine $1{,}06$-Approximation, wenn P $\neq$ NP (Håstad 1996)

jetzt einfacher Lokale-Suche Ansatz und
sehr einfacher randomisierter Algorithmus
für ungewichtete, ungerichtete Graphen,
schnell, sehr leicht zu implementieren,
(erwartete) Approximationsgüte ≤ 2

10.1 Lokale Suche für ungewichtetes Max Cut

Grundidee der Lokalen Suche

1. Finde eine zulässige Ausgangslösung.
2. Finde eine bessere „Nachbarlösung“, so lange es geht.

Dazu ist es notwendig, problemabhängige Nachbarschaften zu definieren.

Die Strategie führt zu einem „lokalen Optimum“ in dem Sinne, dass es keine bessere „Nachbarlösung“ gibt.

Lokale Suche für ungewichtetes Max Cut

Anfangslösung einfach, z.B. $V_1 = \emptyset$, $V_2 = V$

Nachbarschaft $\mathcal{N}$ bezüglich (V_1, V_2):

Alle Partitionen (V_{1k}, V_{2k}) für $k = 1, 2, \ldots, |V|$ mit

1. Falls $v_k \in V_1$, so ist $V_{1k} = V_1 \setminus \{v_k\}$ und $V_{2k} = V_2 \cup \{v_k\}$,
2. Falls $v_k \in V_2$, so ist $V_{1k} = V_1 \cup \{v_k\}$ und $V_{2k} = V_2 \setminus \{v_k\}$.

Das nennt man 1-Austausch (englisch 1-Exchange).

Analyse des 1-Austausch für Max Cut

Theorem (Lokale Suche Max Cut)

Für eine ungewichtete Max Cut Instanz G mit Optimalwert OPT sei $m_{\mathcal{N}}(G)$ der Lösungswert eines lokalen Optimums bezüglich $\mathcal{N}$. Dann gilt

$$\frac{\mathrm{OPT}}{m_{\mathcal{N}}(G)} \leq 2.$$

Analyse des 1-Austausch für Max Cut

Sei $m := |E|$ und V_1, V_2 die beiden zu $m_{\mathcal{N}}$ korrespondierenden Knotenmengen.

Wegen $\mathrm{OPT} \leq m$ reicht es, zu zeigen, dass $2m_{\mathcal{N}}(G) \geq m$.

Für $W \subseteq V$ sei $E(W) = \{(u,v) \in E \mid u,v \in W\}$ und
$\delta(W) = \{(u,v) \in E \mid u \in W \text{ und } v \notin W\}$,

$$\begin{aligned} m_1 &= |E(V_1)|, \\ m_2 &= |E(V_2)|. \end{aligned}$$

Dann gilt

$$(*) \quad m = m_1 + m_2 + m_{\mathcal{N}}(G).$$

Analyse des 1-Austausch für Max Cut

(V_1, V_2) ist ein lokales Optimum mit Wert $m_{\mathcal{N}}(G)$ und V_{1k}, V_{2k} seien die Nachbarpartitionen. Für $v_i \in V$ sei

$$\begin{aligned} m_{1i} &= |\{v \mid v \in V_1, (v, v_i) \in E\}|, \\ m_{2i} &= |\{v \mid v \in V_2, (v, v_i) \in E\}|. \end{aligned}$$

$$\Longrightarrow \quad (\forall k) \begin{cases} |\delta(V_{1k})\,| \leq m_{\mathcal{N}}(G) \\ |\delta(V_{2k})\,| \leq m_{\mathcal{N}}(G) \end{cases}$$

$$\Longrightarrow \quad \left\{ \begin{array}{l} (\forall v_i \in V_1)\; m_{1i} - m_{2i} \leq 0 \\ (\forall v_j \in V_2)\; m_{2j} - m_{1j} \leq 0 \end{array} \right\} \geq \text{ so viele Nachbarn auf der anderen Seite}$$

$$\Longrightarrow \quad \begin{cases} \sum_{v_i \in V_1} (m_{1i} - m_{2i}) = 2m_1 - m_{\mathcal{N}}(G) \leq 0 \\ \sum_{v_j \in V_2} (m_{2j} - m_{1j}) = 2m_2 - m_{\mathcal{N}}(G) \leq 0 \end{cases}$$

$$\Longrightarrow \quad m_1 + m_2 - m_{\mathcal{N}}(G) \leq 0$$

Analyse des 1-Austausch für Max Cut

Wir haben also: $m_1 + m_2 - m_{\mathcal{N}}(G) \leq 0$

sowie von vorhin $(*)$ $m = m_1 + m_2 + m_{\mathcal{N}}(G)$.

Zusammen ergibt das

$$\Longrightarrow \quad m - 2m_{\mathcal{N}}(G) \leq 0$$

$$\Longrightarrow \quad 2m_{\mathcal{N}}(G) \geq m$$

$$\Longrightarrow \quad 2m_{\mathcal{N}}(G) \geq OPT$$

$\square$

10.2 Einfacher randomisierter Max Cut Algorithmus

Algorithmus 10.1 (Einfacher randomisierter Max Cut)

1. $V_1 := V_2 := \emptyset$
2. Für $v \in V$
3. Mit W'keit $1/2$ setze $V_1 := V_1 \cup \{v\}$
   sonst setze $V_2 := V_2 \cup \{v\}$
4. Ausgabe V_1, V_2

Theorem 10.2

Algorithmus 10.1 berechnet für einen ungewichteten Graph $G = (V, E)$ in Zeit $O(|V|)$ einen Schnitt, der im Erwartungswert $|E|/2$ Kanten schneidet und erwartete Güte ≤ 2 hat.

Zum Beweis von Theorem 10.2

Beobachtung Laufzeit $O(|V|)$ ✓

Beobachtung erwartete Güte ≤ 2
folgt aus $\mathsf{E}(\#\text{geschnittene Kanten}) = |E|/2$
weil OPT $\leq |E|$

noch offen $\mathsf{E}(\#\text{geschnittene Kanten}) = \mathsf{E}(w(V_1, V_2)) = |E|/2$

Definiere Indikatorvariablen X_e für $e \in E$

$$\text{mit } X_e := \begin{cases} 1 & \text{falls } |e \cap V_1| = |e \cap V_2| = 1 \\ 0 & \text{sonst} \end{cases}$$

Beobachtung $w(V_1, V_2) = \sum_{e \in E} X_e$

Analyse von Algorithmus 10.1

Wir haben Indikatorvariablen X_e für $e \in E$

mit $X_e := \begin{cases} 1 & \text{falls } |e \cap V_1| = |e \cap V_2| = 1 \\ 0 & \text{sonst} \end{cases}$

und damit $w(V_1, V_2) = \sum_{e \in E} X_e$

$$\begin{aligned} \mathsf{E}\left(w(V_1, V_2)\right) &= \mathsf{E}\left(\sum_{e \in E} X_e\right) \\ &= \sum_{e \in E} \mathsf{E}\left(X_e\right) \\ &= \sum_{e \in E} \mathrm{Prob}\left(X_e = 1\right) \\ &= |E| \cdot \mathrm{Prob}\left(X_e = 1\right), \text{ für eine Kante } e = (u, v) \end{aligned}$$

Erwartete Anzahl geschnittener Kanten

Wir haben

$$
\begin{aligned}
& \mathsf{E}\left(w(V_1, V_2)\right) \\
= \;& |E| \cdot \mathsf{Prob}\left(X_e = 1\right), \text{ für eine Kante } e = (u, v) \\
= \;& |E| \cdot \mathsf{Prob}\left(((u \in V_1) \wedge (v \in V_2)) \vee ((u \in V_2) \wedge (v \in V_1))\right) \\
= \;& |E| \cdot \left(\mathsf{Prob}\left((u \in V_1) \wedge (v \in V_2)\right) + \mathsf{Prob}\left((u \in V_2) \wedge (v \in V_1)\right)\right) \\
= \;& |E| \cdot \left(\mathsf{Prob}\left(u \in V_1\right) \cdot \mathsf{Prob}\left(v \in V_2\right) + \mathsf{Prob}\left(u \in V_2\right) \cdot \mathsf{Prob}\left(v \in V_1\right)\right) \\
= \;& |E| \cdot \left(\frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2}\right) = |E| \cdot \left(\frac{1}{4} + \frac{1}{4}\right) = \frac{|E|}{2}
\end{aligned}
$$

□

10.3 Randomisierter Min-Cut Algorithmus

Min Cut
gewichtete Graphen mit $w(e) \geq 0$
Ziel: Minimierung von $w(V_1, V_2)$, wobei $V_1 \neq \emptyset$ und $V_2 \neq \emptyset$

Erinnerung
Berechnung minimaler Schnitt in $O(n^3)$

hier und jetzt
einfache randomisierte Algorithmen
mit W'keit sehr dicht an 1 minimaler Schnitt
sehr schnell $\rightarrow O\left(n^2 \log^3 n\right)$
Idee von Karger [1993]; Karger und Stein [1996]

Voraussetzung
G zusammenhängend
sonst Lösung mit $w(V_1, V_2) = 0$ optimal

Min Cut - Idee

Idee Iterative Kontraktion:
Wiederhole:
Wähle Kante $e = u, v$ und kontrahiere sie
Solange bis nur noch zwei Knoten übrig sind
Diese definieren den Schnitt

Kontraktion von 2 Knoten u, v: (vgl. Kapitel 6)

- Verschmelze die beiden Knoten zu neuem Knoten z
- Eine eventuelle Kante $\{u, v\}$ verschwindet
- Alle Kanten $\{x, u\}$ und $\{x, v\}$ werden durch neue Kanten $\{x, z\}$ ersetzt
- Mehrfachkanten werden zu einer Kante verschmolzen

Min Cut durch Kontraktion

Idee

1. Repeat
2. Wähle $u \neq v \in V$ geeignet.
3. $G := \texttt{Kontraktion}(G, u, v)$
4. Until $|V| = 2$
5. Gib den durch die beiden Knoten definierten Schnitt aus.

klar Implementierung von „geeignet“ entscheidend

Heuristik Kanten mit großem Gewicht kontrahieren

klar „groß“ ist kontextabhängig

Ein einfacher randomisierter Kontraktions-Algorithmus

Algorithmus 10.3 (Randomisierte Kontraktion von Karger)

1. Repeat
2. Wähle $e = \{u, v\}$ mit W'keit $w(e)/\sum_{e' \in E} w(e')$.
3. $G :=$ `Kontraktion`(G, u, v)
4. Until $|V| = 2$
5. Gib den durch die beiden Knoten definierten Schnitt aus.

Lemma 10.4

Algorithmus 10.3 hat auf einem Graphen $G = (V, E)$ die Laufzeit $O\left(|V|^2\right)$ und berechnet einen Schnitt in G.

Analyse von Randomisierter Kontraktion

Korrektheit

- initial zusammenhängend
- also Wahl von $e = \{u, v\}$ möglich
- Zusammenhang bleibt bei Kontraktion erhalten
- also Wahl von $e = \{u, v\}$ immer möglich
- also korrekt $\checkmark$

Laufzeit

- initial Summe der Kantengewichte in Zeit $O(|E|) = O\left(|V|^2\right)$
- $|V| - 2$ Aufrufe von `Kontraktion`
- je Aufruf Zeit $O(|V|)$, da nur Nachbarn zu durchlaufen sind
- Zufallsexperiment ebenfalls in $O(|V|)$ möglich
- also Gesamtzeit $O\left(|V|^2\right)$

$\square$

Min Cut: Struktureinsichten

Notation $w(E') := \sum_{e \in E'} w(e)$

$\mathsf{minCut}(G) := \min\{w(V_1, V_2) \mid V_1, V_2 \text{ Schnitt für } G\}$

$\texttt{Kontraktion}(G, u, v) := G$ nach Kontraktion von u, v

Lemma 10.5

Für alle zusammenhängenden, gewichteten, ungerichteten Graphen $G = (V, E)$ gelten die folgenden Aussagen.

1. Ein Schnitt V_1, V_2 ist genau dann Ausgabe von Algorithmus 10.3, wenn nie eine Kante zwischen V_1 und V_2 kontrahiert wurde.
2. $w(E) \geq \mathsf{minCut}(G) \cdot |V| / 2$
3. $\forall e = \{u, v\} \in E$:
   $$\mathsf{minCut}(G) \leq \mathsf{minCut}(\texttt{Kontraktion}(G, u, v))$$

Beweis von Lemma 10.5 – Teil 1

zu zeigen Schnitt V_1, V_2 ist Ausgabe von Algorithmus 10.3
$\Leftrightarrow$ keine Kante zwischen V_1 und V_2 wird kontrahiert

Betrachte V_1, V_2 mit $V_1 \cap V_2 = \emptyset$ und $V_1 \cup V_2 = V$

Beobachtung keine Kante zwischen V_1 und V_2 kontrahiert
$\Rightarrow$ V_1 und V_2 am Ende getrennt
$\Rightarrow$ Ausgabe V_1, V_2

Beobachtung Kante zwischen V_1 und V_2 kontrahiert
$\Rightarrow$ $u \in V_1$ und $v \in V_2$ werden verschmolzen
$\Rightarrow$ V_1, V_2 nicht Ausgabe von Algorithmus 10.3

$\checkmark$

Beweis von Lemma 10.5 – Teil 2

zu zeigen $w(E) \geq \text{minCut}(G) \cdot |V|/2$

Definiere für $v \in V$ sei $d_w(v) := \sum_{e=\{v,\cdot\} \in E} w(e)$

Beobachtung 1: $\forall v \in V : d_w(v) \geq \text{minCut}(G)$

denn $d_w(v) \Rightarrow$ Schnitt V_1, V_2 mit $V_1 := \{v\}, V_2 := V \setminus V_1$
mit $w(V_1, V_2) = d_w(v)$

Beobachtung 2: $\sum_{v \in V} d_w(v) = 2w(E)$
$(w(\{u, v\})$ in $d_w(v), d_w(u))$

insgesamt $w(E) = \frac{1}{2} \sum_{v \in V} d_w(v) \geq \frac{1}{2} \cdot |V| \cdot \text{minCut}(G)$ ✓

Beweis von Lemma 10.5 – Teil 3

zu zeigen $\forall e = \{u, v\} \in E$:
$\text{minCut}(G) \leq \text{minCut}(\texttt{Kontraktion}(G, u, v))$

Beobachtung durch Kontraktion können Schnitte verloren gehen aber keine hinzukommen

also Optimum kann höchstens schlechter werden

entspricht $\text{minCut}(G) \leq \text{minCut}(\texttt{Kontraktion}(G, u, v))$

□

Über „Randomisierte Kontraktion“

Theorem 10.6

Für einen zusammenhängenden, gewichteten ungerichteten Graph $G = (V, E)$ sei V_1, V_2 ein minimaler Schnitt; $n := |V|$.
Algorithmus 10.3 berechnet in Zeit $O(n^2)$ mit Wahrscheinlichkeit mindestens $2/n^2$ den Schnitt V_1, V_2.

Beweis.

schon gesehen Laufzeit (Lemma 10.4)$\checkmark$

Definiere $G_{i+1} = (V_{i+1}, E_{i+1})$ Graph nach i-ter Kontraktion

Betrachte Folge $G = G_1, G_2, G_3, \ldots, G_{n-1}$
mit $n_i := |V_i| = |V| - i + 1$ für $i = 1, \ldots, |V|$
$(n := |V| = n_1)$

Über die Kontraktion passender Kanten

Notation $E' := \{\{u, v\} \mid u \in V_1, v \in V_2\}$

Definiere Ereignis A_i: in den ersten $i-1$ Kontraktionen keine Kante aus E' betroffen

Beobachtung Wenn A_i gilt, dann $\text{minCut}(G) = \text{minCut}(G_i)$

Beobachtung $\text{minCut}(G) = \text{minCut}(G_i)$
$\Rightarrow w(E_i) \geq \text{minCut}(G_i) \cdot |V_i|/2 = \text{minCut}(G) \cdot |V_i|/2$
(Lemma 10.5)

damit

$$\text{Prob}\left(\text{Algorithmus 10.3 berechnet } V_1, V_2\right)$$
$$= \prod_{i=1}^{n-2} \text{Prob}\left(i\text{-te Kontraktion betrifft keine Kante aus } E' \mid A_i\right)$$

Kontraktion von Kanten aus E'

Wir brauchen

Prob $(i$-te Kontraktion betrifft keine Kante aus $E' \mid A_i)$

Beobachtung Gegenwahrscheinlichkeit leichter zu berechnen

$$
\begin{aligned}
& \text{Prob}\left(i\text{-Kontraktion betrifft Kante aus } E' \mid A_i\right) \\
= \quad & \sum_{e \in E'} \text{Prob}\left(i\text{-te Kontraktion betrifft } e \mid A_i\right) \\
= \quad & \sum_{e \in E'} \frac{w(e)}{w(E_i)} = \frac{w(E')}{w(E_i)} = \frac{\text{minCut}(G)}{w(E_i)} \\
\overset{Lem.\ 10.5}{\leq} \quad & \frac{\text{minCut}(G)}{\text{minCut}(G) \cdot |V_i|/2} = \frac{2}{|V| - i + 1} = \frac{2}{n - i + 1}
\end{aligned}
$$

Beweis von Theorem 10.6

Wir haben Prob (Algorithmus 10.3 berechnet V_1, V_2)

$$= \prod_{i=1}^{n-2} \text{Prob}\,(i\text{-te Kontr. betrifft nicht } E' \mid A_i)$$

und Prob (i-Kontraktion betrifft $E' \mid A_i$) $= \frac{2}{n-i+1}$

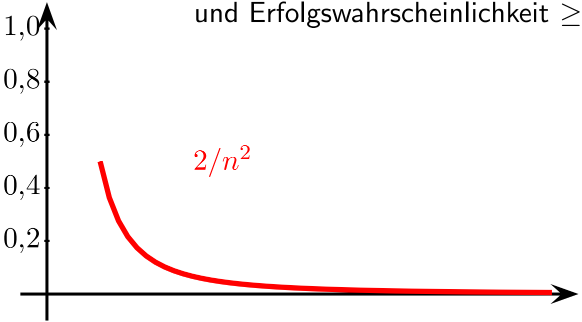
also

Prob (Algorithmus 10.3 berechnet V_1, V_2)

$$\begin{aligned}
&= \prod_{i=1}^{n-2} \text{Prob}\left(i\text{-te Kontraktion betrifft keine Kante aus } E' \mid A_i\right) \\
&= \prod_{i=1}^{n-2} \left(1 - \frac{2}{n-i+1}\right) = \prod_{i=1}^{n-2} \frac{n-i-1}{n-i+1} \\
&= \frac{(n-2)\cdots 3 \cdot 2 \cdot 1}{n(n-1)(n-2)\cdots 4 \cdot 3} = \frac{2}{n(n-1)} \geq \frac{2}{n^2} \qquad \square
\end{aligned}$$

Algorithmus „Randomisierte Kontraktion“ im Rückblick

Wir haben Laufzeit $O(n^2)$ (prima)
und Erfolgswahrscheinlichkeit $\geq 2/n^2$ (Mist!)



Also Randomisierte Kontraktion nutzlos?

Idee Probability Amplification

Probability Amplification

Lemma 10.7

Gibt man für einen Graph $G = (V, E)$, $n := |V|$, einen Schnitt mit kleinstem Wert unter r Ausgaben von unabhängigen Wiederholungen von Algorithmus 10.3 aus, so ist der ausgegebene Schnitt mit Wahrscheinlichkeit mindestens $1 - \left(1 - 2/n^2\right)^r \geq 1 - e^{-2r/n^2}$ minimal.

Beweis.

Theorem 10.6 je Durchlauf $\text{Prob}\,(\text{min. Schnitt}) \geq 2/n^2$

also je Durchlauf $\text{Prob}\,(\text{kein min. Schnitt}) \leq 1 - 2/n^2$

also $\text{Prob}\,(\text{in } r \text{ Durchläufen nie min. Schnitt}) \leq \left(1 - 2/n^2\right)^r$

also $\text{Prob}\,(\text{min. Schnitt in } r \text{ Durchläufen}) \geq 1 - \left(1 - 2/n^2\right)^r$

Beweis von Lemma 10.7

Wir haben

Prob (min. Schnitt in r Durchläufen) $\geq 1 - \left(1 - 2/n^2\right)^r$

bekannt: $1 + x \leq e^x$ für alle x

also

$$1 - 2/n^2 \leq e^{-2/n^2}$$
$$\Leftrightarrow \left(1 - 2/n^2\right)^r \leq e^{-2r/n^2}$$
$$\Leftrightarrow 1 - \left(1 - 2/n^2\right)^r \geq 1 - e^{-2r/n^2} \qquad \square$$

Nützliche Randomisierte Kontraktion

Korollar 10.8

Durch unabhängige Wiederholungen von Algorithmus 10.3 findet man für einen Graph $G = (V, E)$ mit $|V| = n$ und jede Konstante $c > 0$ in Zeit $O(n^4 \log n)$ einen minimalen Schnitt mit Wahrscheinlichkeit mindestens $1 - 1/n^{2c}$.

Beweis.

klar mit Lemma 10.7

Wähle $r := \lceil cn^2 \ln n \rceil$ Anzahl Wiederholungen

damit Erfolgsw'keit
$\geq 1 - e^{-2r/n^2} \geq 1 - e^{-(2cn^2 \ln n)/n^2} = 1 - 1/n^{2c}$

außerdem Gesamtlaufzeit $O\left(n^2 \cdot cn^2 \ln n\right) = O\left(n^4 \log n\right)$ □

Beobachtung schon mit $c = 1/2$ Erfolgsw'keit völlig ausreichend

Beobachtung Laufzeit $O\left(n^4 \log n\right)$ nicht überzeugend

10.4 Ein schnellerer Min Cut Algorithmus: Fast Min Cut

Laufzeit $n^4 \log n$ im Rückblick: Wieso brauchen wir so lange?

Zwei Faktoren

1. Laufzeit $\Theta(n^2)$ für Randomisierte Kontraktion
2. $\Theta(n^2 \log n)$ Wiederholungen

Beobachtung $\Theta(n^2)$ für Schnitt im Graph prima

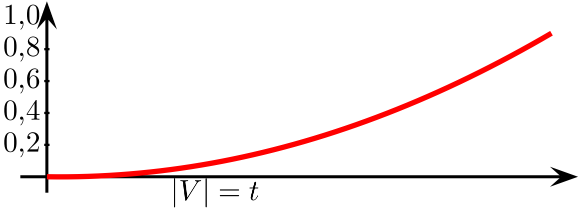
Wie können wir die Anzahl der Wiederholungen senken?

Wie können wir die Erfolgsw'keit je Durchgang verbessern?

Erfolgswahrscheinlichkeit erhöhen

Lemma 10.9

Bricht man Algorithmus 10.3 auf einem Graph $G = (V, E)$ mit $|V| = n$ ab, wenn $|V| = t$ gilt (mit $t \in \{2, 3, \ldots, n\}$), so ist mit Wahrscheinlichkeit mindestens $t \cdot (t-1)/(n \cdot (n-1))$ noch keine Kante eines festen minimalen Schnitts V_1, V_2 kontrahiert.



- anfangs Wahl einfach, Prob (wähle günstig) groß
- am Ende Wahl schwierig, Prob (wähle günstig) klein

Beweis von Lemma 10.9

Beweis.
analog zum Beweis von Theorem 10.6

$$
\begin{aligned}
& \text{Prob}\left(\text{Schnitt } V_1, V_2 \text{ noch möglich}\right) \\
\geq\ & \prod_{i=1}^{n-t}\left(1 - \frac{2}{n-i+1}\right) = \prod_{i=1}^{n-t}\frac{n-i-1}{n-i+1} \\
=\ & \frac{(n-2)\cdot(n-3)\cdots(t+1)\cdot t\cdot(t-1)}{n\cdot(n-1)\cdot(n-2)\cdot(n-3)\cdots(t+2)\cdot(t+1)} \\
=\ & \frac{t\cdot(t-1)}{n\cdot(n-1)}
\end{aligned}
$$

□

Struktureinsicht und Verbesserungsidee

Beobachtung

- anfangs Wahl einfach, Prob (wähle günstig) groß
- am Ende Wahl schwierig, Prob (wähle günstig) klein

Ideen

- reduziere Knotenanzahl rekursiv um konstanten Faktor
- berechne auf jeder Rekursionsebene zwei Schnitte
- $\rightsquigarrow$ Anzahl Versuche wächst exponentiell mit Rekursionstiefe
- löse sehr kleine Schnittprobleme optimal

Dauern exponentiell viele Versuche nicht zu lange?

Vergrößert das die Erfolgswahrscheinlichkeit signifikant?

Algorithmus Fast Cut

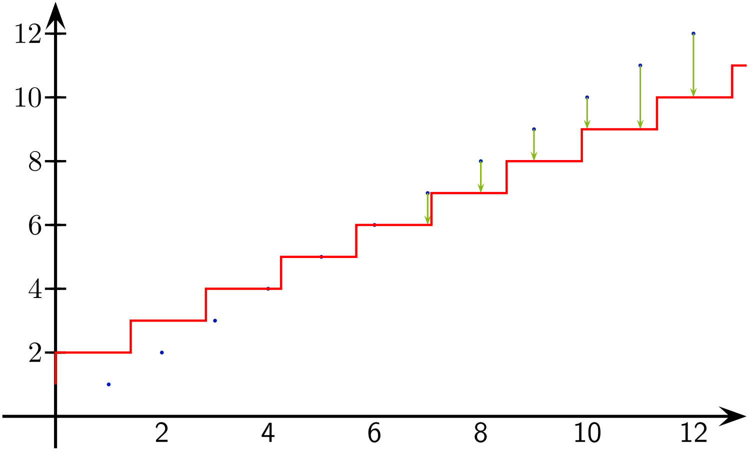
Algorithmus 10.10 von Karger und Stein [1996]

Eingabe zusammenhängender gewichteter Graph $G = (V, E)$
Ausgabe Schnitt V_1, V_2

1. If $|V| \leq 6$ Then
   Berechne minimalen Schnitt V_1, V_2.
   Ausgabe V_1, V_2
2. Else
3. $t := \left\lceil 1 + \frac{|V|}{\sqrt{2}} \right\rceil$
4. $H := \texttt{RandomisierteKontraktion}(G)$
   bis t Knoten übrig sind.
5. $H' := \texttt{RandomisierteKontraktion}(G)$
   bis t Knoten übrig sind.
6. $(V_1, V_2) := \texttt{FastCut}(H)$
7. $(V_1', V_2') := \texttt{FastCut}(H')$
8. Vergleiche (V_1, V_2) und (V_1', V_2'), gib kleineren Schnitt aus.

Über die Wahl des Rekursionsendes

Wieso Ende der Rekursion gerade bei $|V| \leq 6$?



$6 = \left\lceil 1 + \frac{6}{\sqrt{2}} \right\rceil$: größtes n, so dass $t = \left\lceil 1 + \frac{|V|}{\sqrt{2}} \right\rceil < n$ ist $n = 7$

Laufzeit von Fast Cut

Lemma 10.11

Algorithmus 10.10 hat auf einem Graph $G = (V, E)$ mit $|V| = n$ Laufzeit $O\big(n^2 \log n\big)$.

Beweis.

Definiere $T(n) :=$ Laufzeit von Algo. 10.10 auf G mit n Knoten

Erinnerung Laufzeit von Algo 10.3 (Zeilen 4, 5) $O\big(n^2\big)$

Notationsvereinfachung $\leq n^2/2$

also für $n > 6$: $T(n) = 2 \cdot T\left(\left\lceil 1 + \frac{n}{\sqrt{2}} \right\rceil\right) + n^2$

für $n \leq 6$: $T(n) = O(1)$

Notationsvereinfachung $T(n) = 1$ für $n \leq 6$

Analyse der Rekursion

Sei $r :=$ Rekursionstiefe

Betrachte auftretende Knotenanzahlen

$$n_0 := n,\ n_1 := \left\lceil 1 + \frac{n_0}{\sqrt{2}} \right\rceil,\ n_2 := \left\lceil 1 + \frac{n_1}{\sqrt{2}} \right\rceil, \ldots,$$

$$n_r := \left\lceil 1 + \frac{n_{r-1}}{\sqrt{2}} \right\rceil \text{ mit } n_r \leq 6$$

damit

$$\begin{aligned} T(n) = T(n_0) &= 2T(n_1) + n_0^2 \\ &= 2\left(2T(n_2) + n_1^2\right) + n_0^2 \\ &= 4T(n_2) + 2n_1^2 + n_0^2 \\ &= 4(2T(n_3) + n_2^2) + 2n_1^2 + n_0^2 \\ &= 8T(n_3) + 4n_2^2 + 2n_1^2 + n_0^2 \\ &= \cdots \\ &= 2^r T(n_r) + \sum_{i=0}^{r-1} 2^i n_i^2 = 2^r + \sum_{i=0}^{r-1} 2^i n_i^2 \end{aligned}$$

Wie groß ist r? Wie entwickelt sich $2^i n_i^2$?

Laufzeitanalyse von Fast Cut

Wir haben $$T(n) = 2^r + \sum_{i=0}^{r-1} 2^i n_i^2$$
$$n_0 := n,\ n_{i+1} := \left\lceil 1 + \frac{n_i}{\sqrt{2}} \right\rceil$$

Wie groß ist r? Wie entwickelt sich $2^i n_i^2$?

Beobachtung $$\frac{n_{i+1}^2}{n_i^2} = \frac{\left\lceil 1+\frac{n_i}{\sqrt{2}} \right\rceil^2}{n_i^2} \geq \frac{\left(1+\frac{n_i}{\sqrt{2}}\right)^2}{n_i^2} = \frac{(n_i^2/2)+\sqrt{2}n_i+1}{n_i^2}$$
$$= \frac{1}{2} + \frac{\sqrt{2}}{n_i} + \frac{1}{n_i^2} > \frac{1}{2}$$

also $\left(2^{i+1} n_{i+1}^2\right) / \left(2^i n_i^2\right) > 2 \cdot (1/2) = 1$

also $2^i n_i^2$ streng monoton wachsend

Laufzeitanalyse von Fast Cut

Wir haben $T(n) = 2^r + \sum_{i=0}^{r-1} 2^i n_i^2$

$n_0 := n$, $n_{i+1} := \left\lceil 1 + \frac{n_i}{\sqrt{2}} \right\rceil$

$\frac{2^{i+1} n_i^2}{2^i n_i^2}$ streng monoton wachsend

also $2^i n_i^2$ maximal für $i = r$

also
$$\begin{aligned} T(n) &= 2^r + \sum_{i=0}^{r-1} 2^i n_i^2 \\ &\leq 2^r + r \cdot 2^r n_r^2 \leq 2^r + r \cdot 2^r \cdot 36 = O(r \cdot 2^r) \end{aligned}$$

noch offen Rekursionstiefe r

Plan Rekursionstiefe anhand von n_i bestimmen

Eine Abschätzung für n_i

Wir haben $n_0 := n$, $n_{i+1} := \left\lceil 1 + \frac{n_i}{\sqrt{2}} \right\rceil$

Notation $q := 1/\sqrt{2}$

Behauptung $n \cdot q^i + 2/(1-q) \geq n_i$

Beweis mit vollständiger Induktion über i

Induktionsanfang $i = 0$

Beobachtung $n \cdot q^0 + \frac{2}{1-q} = n + \frac{2}{1-q} > n = n_0$ ✓

Induktionsschritt

$$n \cdot q^{i+1} + \frac{2}{1-q} = q \cdot \left(n \cdot q^i + \frac{2}{q(1-q)} - \frac{2q}{q(1-q)} + \frac{2q}{q(1-q)} \right)$$
$$= q \cdot \left(n \cdot q^i + \frac{2(1-q)}{q(1-q)} + \frac{2}{1-q} \right) = q \cdot \left(n \cdot q^i + \frac{2}{q} + \frac{2}{1-q} \right)$$
$$\geq q \cdot \left(n_i + \frac{2}{q} \right) = 2 + n_i \cdot q \geq \lceil 1 + n_i \cdot q \rceil = n_{i+1}$$ ✓

Bestimmung der Rekursionstiefe

Wir haben $\quad n_i \leq n \cdot q^i + \frac{2}{1-q}$ mit $q = 1/\sqrt{2}$

Wir suchen $\quad$ kleinstes i mit $n_i < 7$

$$\begin{array}{lll} & n \cdot q^i + \frac{2}{1-q} & < 7 \\ \Leftrightarrow & n \cdot q^i & < 7 - \frac{2}{1-q} \\ \Leftrightarrow & \log(n) + i \log(q) & < \log\left(7 - \frac{2}{1-q}\right) \\ \Leftrightarrow & -\frac{i}{2} & < \log\left(7 - \frac{2}{1-q}\right) - \log n \\ \Leftrightarrow & i & > 2\log(n) - 2\log\left(7 - \frac{2}{1-q}\right) \end{array}$$

also $\quad r \leq 2\log(n) + 6$

Erinnerung $\quad T(n) = O(r \cdot 2^r)$

also $\quad T(n) = O\big((2\log n + 6)(2^{2\log n + 6})\big) = O\big(n^2 \log n\big)$ $\quad\square$

Erfolgswahrscheinlichkeit von Fast Cut

Lemma 10.12

Algorithmus 10.10 berechnet auf einem Graph $G = (V, E)$ mit $|V| = n$ einen minimalen Schnitt mit Wahrscheinlichkeit $\Omega(1/\log n)$.

Beweis.

Sei V_1, V_2 ein optimaler Schnitt; klar: für $n \leq 6$ Erfolgsw'keit 1

Definiere Ereignisse

- $A(s)$ in H keine Kante zwischen V_1, V_2 kontrahiert (s ist Knotenanzahl in G)
- $A'(s)$ in H' keine Kante zwischen V_1, V_2 kontrahiert (s ist Knotenanzahl in G)
- $B(s)$ Fast Cut(H) liefert V_1, V_2 unter der Annahme, dass das möglich ist (s ist Knotenanzahl in H)
- $B'(s)$ Fast Cut(H') liefert V_1, V_2 unter der Annahme, dass das möglich ist (s ist Knotenanzahl in H')

Beweis von Lemma 10.12

Wir suchen untere Schranke für
Prob $(B(n)) =$ Prob $((A(n) \wedge B(t)) \vee (A'(n) \wedge B'(t)))$
mit $t = \left\lceil 1 + \frac{n}{\sqrt{2}} \right\rceil$

Beobachtungen

- $A(n)$ und $A'(n)$ unabhängig
- $B(t)$ und $B'(t)$ unabhängig
- $A(n)$ und $A'(n)$ gleichartig $\rightsquigarrow$ Prob $(A(n)) =$ Prob $(A'(n))$
- $B(n)$ und $B'(n)$ gleichartig $\rightsquigarrow$ Prob $(B(n)) =$ Prob $(B'(n))$
- $(A(n), B(n))$ und $(A'(n), B'(n))$ unabhängig

Analyse der Erfolgswahrscheinlichkeit von Fast Cut

Mit unseren Beobachtungen folgt

$$\begin{aligned}
\text{Prob}\,(B(n)) &= \text{Prob}\,\big((A(n) \wedge B(t)) \vee \big(A'(n) \wedge B'(t)\big)\big) \\
&= 1 - \text{Prob}\left(\overline{(A(n) \wedge B(t)) \vee (A'(n) \wedge B'(t))}\right) \\
&= 1 - \text{Prob}\left(\overline{A(n) \wedge B(t)} \wedge \overline{A'(n) \wedge B'(t)}\right) \\
&= 1 - \text{Prob}\left(\overline{A(n) \wedge B(t)}\right)^2 \\
&= 1 - (1 - \text{Prob}\,(A(n) \wedge B(t)))^2 \\
&= 1 - (1 - \text{Prob}\,(A(n)) \cdot \text{Prob}\,(B(t)))^2
\end{aligned}$$

Abschätzung

Wir haben $\text{Prob}\,(B(n)) = 1 - (1 - \text{Prob}\,(A(n)) \cdot \text{Prob}\,(B(t)))^2$

Erinnerung $\text{Prob}\,(A(n)) \geq \frac{\left\lceil 1+\frac{n}{\sqrt{2}}\right\rceil \cdot \left(\left\lceil 1+\frac{n}{\sqrt{2}}\right\rceil - 1\right)}{n \cdot (n-1)}$
aus Lemma 10.9

damit $\text{Prob}\,(A(n)) \geq \frac{1}{2}$ für alle n

also $\text{Prob}\,(B(n)) \geq 1 - \left(1 - \frac{\text{Prob}(B(t))}{2}\right)^2$

Noch eine Analyse einer Rekursion

Wir haben $\text{Prob}\,(B(n)) \geq 1 - \left(1 - \frac{\text{Prob}(B(t))}{2}\right)^2$

Definiere $p(i) := \text{Prob}\,(B(n_{r-i}))$

Beobachtung $p(0) = \text{Prob}\,(B(n_r)) = 1$

damit $p(i+1) \geq 1 - \left(1 - \frac{p(i)}{2}\right)^2 = p(i) - \frac{p(i)^2}{4}$

Behauptung $p(i) \geq \frac{1}{i+1}$ (nächste Folie)

dann $p(i) = \Omega(1/i)$

durch Einsetzen $\text{Prob}\,(B(n)) = \text{Prob}\,(B(n_0)) = \text{Prob}\,(B(n_{r-r}))$
$= p(r) = \Omega\left(\frac{1}{r}\right)$

Erinnerung $r = O(\log n)$

zusammen Erfolgswahrscheinlichkeit $\Omega\left(\frac{1}{\log n}\right)$

Abschätzung von $p(i)$

Wir haben $\quad p(i+1) \geq 1 - \left(1 - \frac{p(i)}{2}\right)^2 = p(i) - \frac{p(i)^2}{4}$

Behauptung $\quad p(i) \geq \frac{1}{i+1}$

Beweis durch vollständige Induktion über i:

IA: $i = 0 : p(0) = 1$

IS: $i \to i+1 :$

$$\begin{aligned}
p(i+1) &\geq p(i) - \frac{p(i)^2}{4} \geq \frac{1}{i+1} - \frac{0.25}{(i+1)^2} \\
&= \frac{(i+1)(i+2) - 0.25(i+2)}{(i+2)(i+1)^2} = \frac{i^2+3i+2-0.25i-0.5}{(i+2)(i+1)^2} \\
&= \frac{i^2+2i+1+0.75i+0.5}{(i+2)(i+1)^2} = \frac{(i+1)^2}{(i+2)(i+1)^2} + \frac{0.75i+0.5}{(i+2)(i+1)^2} \\
&\geq \frac{1}{i+2}
\end{aligned}$$

□

Mit Probability Amplification

Theorem 10.13

Unabhängige Wiederholungen von Algorithmus 10.10 (Fast Cut) finden für Graph $G = (V, E)$ mit $|V| = n$ und jede Konstante $c > 0$ in Zeit $O(n^2 \log^3 n)$ einen minimalen Schnitt mit W'keit mindestens $1 - 1/n^c$.

Beweis: für Fast Cut gilt $\exists k_1 : T \leq k_1 \cdot n^2 \log n$ (Lemma 10.11)
$\exists k_2$: Prob (Erfolg) $\geq k_2 / \log n$ (Lemma 10.12)

Wähle Anzahl unabh. Wiederholungen $w := \lceil (c/k_2) \ln^2 n \rceil$

Beobachtung Gesamtrechenzeit $O(n^2 \log^3 n)$ ✓

Wir nutzen $1 - x \leq e^{-x}$ für alle x

Prob(Misserfolg) $< \left(1 - \frac{k_2}{\log n}\right)^w \leq e^{-c \ln n} = \frac{1}{n^c}$ ✓

□

Design-Techniken im Verlauf der Vorlesung

1. Die Greedy-Methode: Rucksackproblem (Kap. 7)
2. Dynamische Programmierung: Rucksackproblem (FPTAS) (Kap. 7)
3. Inkrementelle Algorithmen für Partitionsprobleme: Bin Packing Problem (Kap. 7)
4. Spezielle, problemabhängige Verfahren: Traveling Salesman Problem (Kap. 8)
5. Zufalls-basierte Verfahren (Kap. 9, Kap. 10)
6. LP-basierte Verfahren: MaxSAT (Kap. 9)
7. Lokale Suchverfahren: Max Cut (Kap. 10)