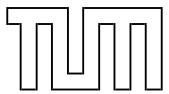
Fortgeschrittene Netzwerk- und Graph-Algorithmen

Prof. Dr. Hanjo Täubig

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

Wintersemester 2010/11



Korrektheit des Stoer/Wagner-Algorithmus

Satz

Ein 'Cut of the Phase' mit minimaler Kantenkapazität ist ein (globaler) MinCut des gegebenen Graphen.

Korrektheit des Stoer/Wagner-Algorithmus

Beweis.

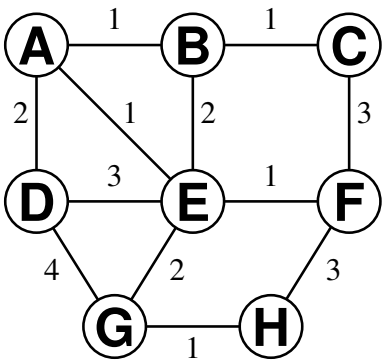
- Für $|V| = 2$ trivial: bei zwei Knoten existiert nur *ein* Cut.
- Für $|V| > 2$ Fallunterscheidung:
  Betrachte eine Phase mit letzten Knoten s und t
  1. Entweder der Graph hat einen MinCut, der gleichzeitig ein (lokaler) Minimum s-t-Cut ist.
     $\Rightarrow$ Cut of the Phase ist nach Lemma globaler MinCut des Graphen
  2. Andernfalls hat der Graph einen globalen MinCut, bei dem s und t nicht separiert werden (sich also auf der gleichen Seite befinden). Dann wird der globale MinCut durch das Verschmelzen von s und t nicht verändert.

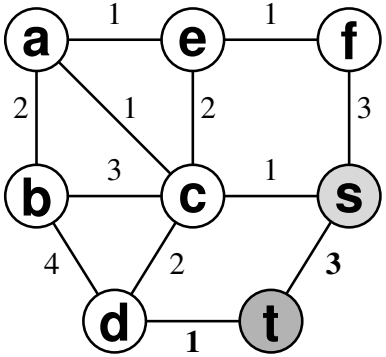
$\Rightarrow$ (Induktion über die Anzahl der Knoten)
'Cut of the Phase' mit minimaler Gewichtssumme ist globaler MinCut.

□

Beispiel für den Stoer/Wagner-Algorithmus

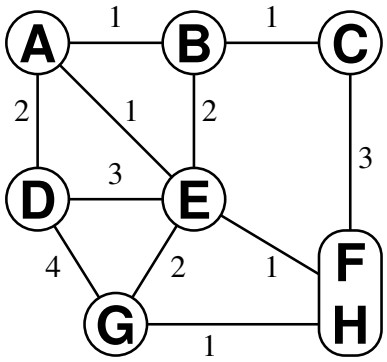
Phase 1

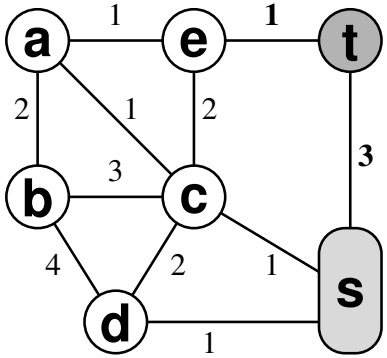




4

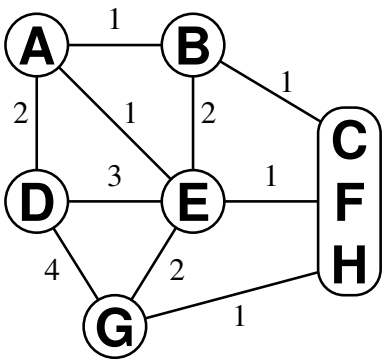
Phase 2

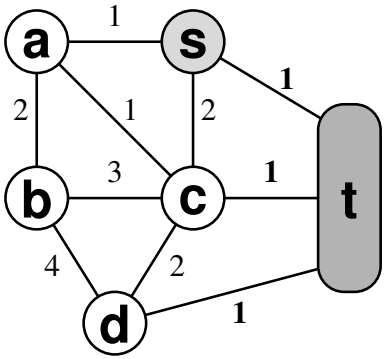




4

Phase 3

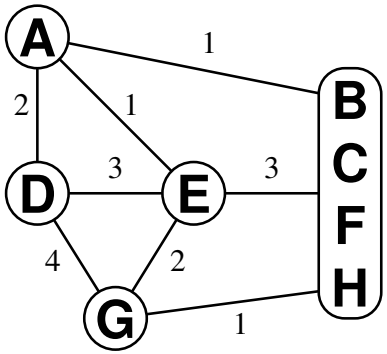


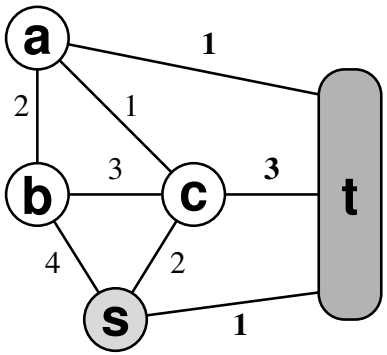


3

Beispiel für den Stoer/Wagner-Algorithmus

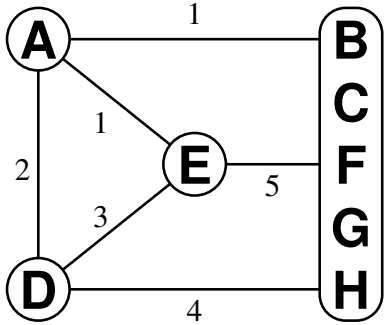
Phase 4

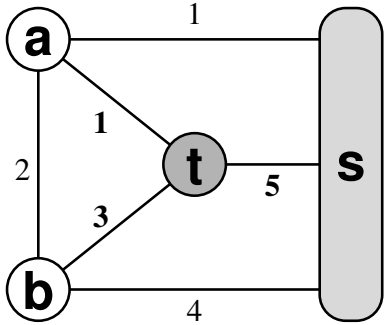




5

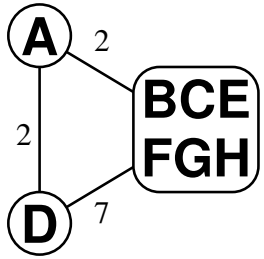
Phase 5

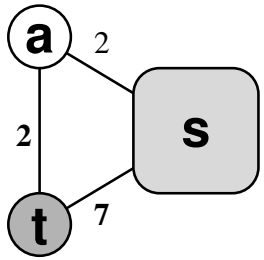




9

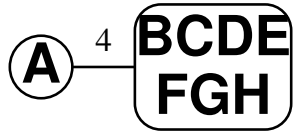
Phase 6

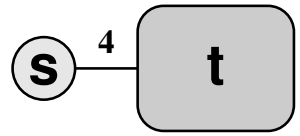




9

Phase 7





4

Komplexität des Stoer/Wagner-Algorithmus

- Adjazenzwerte für verbleibende Knoten ($V \setminus A$) eines MAOs können in einer Priority Queue (Maximum-Variante) gespeichert werden.
- Der nächste Knoten, der zu A hinzugefügt werden soll, wird per `deleteMax`-Operation bestimmt.
- Beim Hinzufügen eines Knotens zu A wird `increaseKey` für alle in $V \setminus A$ verbleibenden Nachbarn des Knotens aufgerufen.
- Bei Verwendung von Fibonacci-Heaps ergibt sich pro Phase (MAO) Komplexität $\mathcal{O}(m + n \log n)$
  (wie bei den Algorithmen von Prim bzw. Dijkstra)
- Gesamtkomplexität für $n - 1$ Phasen: $\mathcal{O}(mn + n^2 \log n)$

Randomisierter MinCut-Algorithmus von Karger

- Gesucht: globaler Minimum Cut in einem Multi-Graphen
- D. Karger (1992): Vorschlag eines sehr einfachen Algorithmus ohne augmentierende Pfade und Flussberechnungen
- Ansatz: randomisierter (Monte-Carlo-)Algorithmus, der
  - mit gewisser (Erfolgs-)Wahrscheinlichkeit den Minimum Cut liefert und
  - mit gewisser (Fehler-)Wahrscheinlichkeit einen beliebigen anderen Cut

Randomisierter MinCut-Algorithmus von Karger

- Algorithmus wählt beliebige Kante $e = \{u, v\}$ von G (gleichverteilt, d.h. jede Kante wird mit gleicher Wahrscheinlichkeit gezogen)
- gewählte Kante wird kontrahiert, d.h. Multi-Graph G' wird erzeugt, bei dem Knoten u und v zu einem neuen Knoten w verschmolzen sind
- Kanten zwischen u und v verschwinden.
- Andere Kanten bleiben erhalten. Wenn genau ein Endknoten entweder u oder v ist, dann endet dafür eine neue Kante am (Super-)Knoten w.
- Hinweis: auch wenn G keine Multikanten enthält, kann G' welche enthalten.
- Das Verfahren wird rekursiv auf G' angewendet.

Randomisierter MinCut-Algorithmus von Karger

- Der Algorithmus endet, wenn nur noch zwei (Super-)Knoten vorhanden sind.
- Jeder der beiden Knoten enthält eine gewisse Menge der Knoten des ursprünglichen Graphen.
- Diese Partition definiert einen Cut, der vom Algorithmus ausgegeben wird.

Randomisierter MinCut-Algorithmus von Karger

Satz

Der Monte-Carlo-Algorithmus von Karger gibt mit Wahrscheinlichkeit $\geq 1/\binom{n}{2}$ *den globalen Minimum Cut des Multigraphen zurück.*

- Nachdem es exponentiell viele Cuts in G gibt (und höchstens quadratisch viele MinCuts), würde man vermuten, dass die Wahrscheinlichkeit, den globalen MinCut zu finden, exponentiell klein ist.

$\Rightarrow$ Was also favorisiert die kleinen Cuts?

Randomisierter MinCut-Algorithmus: Analyse

Beweis.

- Betrachte globalen MinCut (A, B) in G.
- Sei die Schnitt-Kardinalität k, d.h. es gibt genau k Kanten ($F \subseteq E$), die einen Endpunkt in A und einen in B haben.
- gesucht: untere Schranke für die Wahrscheinlichkeit, dass der Algorithmus (A, B) zurück gibt.
- Was kann dabei schiefgehen?
- Eine Kante aus F könnte kontrahiert werden.
- Dann würden zwei Knoten kontrahiert, die auf unterschiedlichen Seiten des Cuts (A, B) liegen.
- Der Algorithmus könnte nicht mehr (A, B) ausgeben.
- Wenn eine beliebige Kante aus $E \setminus F$ kontrahiert wird, besteht noch die Chance auf die Ausgabe (A, B).

Randomisierter MinCut-Algorithmus: Analyse

Beweis.

- also gesucht: obere Schranke für die Wahrscheinlichkeit, dass eine Kante aus F kontrahiert wird.
- Wir brauchen eine untere Schranke für die Kardinalität von E.
- Wenn der globale MinCut den Wert $\lambda(G) = k$ hat, gilt für den Minimalgrad des Graphen $\delta(G) \geq \lambda(G) = k$.
- Es gilt also $|E| = m \geq kn/2$.

$\Rightarrow$ Die Wahrscheinlichkeit, dass eine Kante aus F kontrahiert wird, ist

$$\frac{k}{m} \leq \frac{k}{kn/2} = \frac{2}{n}$$

- Betrachte jetzt die Situation nach j Iterationen.
- Es gibt $n - j$ (Super-)Knoten im aktuellen G'.

Randomisierter MinCut-Algorithmus: Analyse

Beweis.

- Annahme: es wurde noch keine Kante aus F kontrahiert.
- Da jeder Cut in G' auch ein Cut in G ist, sind zu jedem Superknoten in G' mindestens k Kanten inzident.
- D.h. G' hat mindestens $k(n-j)/2$ Kanten.

$\Rightarrow$ Die Wahrscheinlichkeit, dass eine Kante aus F in der nächsten Iteration $j+1$ kontrahiert wird, ist

$$\frac{k}{k(n-j)/2} = \frac{2}{n-j}$$

- Cut (A, B) wird ausgegeben, wenn in Iteration $1, \ldots, n-2$ keine Kante aus F kontrahiert wird.

Randomisierter MinCut-Algorithmus: Analyse

Beweis.

- Sei $\mathcal{E}_j$ das Ereignis, dass in Iteration j keine Kante aus F kontrahiert wird. Dann gilt also: $\Pr[\mathcal{E}_1] \geq 1 - 2/n$ und

$$\Pr[\mathcal{E}_{j+1} | \mathcal{E}_1 \cap \mathcal{E}_2 \cap \ldots \cap \mathcal{E}_j] \geq 1 - 2/(n-j)$$

- gesucht: untere Schranke für $\Pr[\mathcal{E}_1 \cap \mathcal{E}_2 \cap \ldots \cap \mathcal{E}_{n-2}]$ bzw. $\Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2|\mathcal{E}_1] \cdot \ldots \cdot \Pr[\mathcal{E}_{n-2}|\mathcal{E}_1 \cap \mathcal{E}_2 \cap \ldots \cap \mathcal{E}_{n-3}]$

$$\begin{aligned} &\geq \left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right) \ldots \left(1 - \frac{2}{n-j}\right) \ldots \left(1 - \frac{2}{3}\right) \\ &= \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdot \ldots \cdot \frac{2}{4} \cdot \frac{1}{3} \\ &= \frac{2 \cdot 1}{n(n-1)} = \binom{n}{2}^{-1} \end{aligned}$$

Randomisierter MinCut-Algorithmus: Analyse

- Wahrscheinlichkeit, dass der randomisierte MinCut-Algorithmus nicht den Cut (A, B) ausgibt, ist höchstens $1 - 1/\binom{n}{2}$
- Nach $\binom{n}{2}$ Läufen mit unabhängigen Zufallsentscheidungen ist die Wahrscheinlichkeit, dass nie der MinCut gefunden wurde, höchstens

$$\left(1 - 1/\binom{n}{2}\right)^{\binom{n}{2}} \leq \frac{1}{e}$$

- Wenn $c\binom{n}{2} \log n$ Läufe gestartet werden, sinkt die Fehlerwahrscheinlichkeit auf $\leq e^{-c \log n} = 1/n^c$.
- Laufzeit ist in dieser einfachen Form noch relativ hoch, verglichen mit dem besten deterministischen Algorithmus

Randomisierter MinCut-Algorithmus: Analyse

- Ansatz ist unbalanciert: Fehlerwahrscheinlichkeit erhöht sich zum Ende eines Laufs
- Wenn man den Algorithmus nur t Schritte laufen lässt, ist die Wahrscheinlichkeit, dass keine Kante aus F kontrahiert wird
  $\Pr[\mathcal{E}_1 \cap \mathcal{E}_2 \cap \ldots \cap \mathcal{E}_t]$ bzw.
  $\Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2|\mathcal{E}_1] \cdot \ldots \cdot \Pr[\mathcal{E}_t|\mathcal{E}_1 \cap \mathcal{E}_2 \cap \ldots \cap \mathcal{E}_{t-1}]$

$$
\begin{aligned}
&\geq \left(1-\frac{2}{n}\right)\left(1-\frac{2}{n-1}\right)\ldots\left(1-\frac{2}{n-(t-1)}\right)\\
&= \frac{n-2}{n}\cdot\frac{n-3}{n-1}\cdot\frac{n-4}{n-2}\cdot\ldots\cdot\frac{n-t}{n-t+2}\cdot\frac{n-t-1}{n-t+1}\\
&= \frac{(n-t)\cdot(n-t-1)}{n(n-1)} \in \Omega\left(\frac{(n-t)^2}{n^2}\right)
\end{aligned}
$$

Randomisierter MinCut-Algorithmus: Analyse

Verbesserte Variante:

- Lasse die einfache Variante mehrmals laufen
  z.B. 2-mal bis $\lceil n/\sqrt{2}+1 \rceil$ Knoten
  oder 4-mal bis $n/2$ Knoten
- Setze jeden der Läufe rekursiv mit der modifizierten Variante fort.
- Laufzeit: $\mathcal{O}(n^2 \log n)$ mit Erfolgswahrscheinlichkeit $\Omega(1/\log n)$
- $\mathcal{O}(\log n)$-malige Wiederholung des modifizierten Algorithmus sorgt für konstante Erfolgswahrscheinlichkeit.
- Um eine kleine Fehlerwahrscheinlichkeit ϵ zu erreichen, kann man $\mathcal{O}(\log n \log \epsilon^{-1})$ Wiederholungen durchführen, was zu einer Gesamtlaufzeit von $\mathcal{O}(n^2 \log^2 n \log \epsilon^{-1})$ führt.