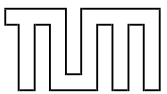
Fortgeschrittene Netzwerk- und Graph-Algorithmen

Dr. Hanjo Täubig

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

Wintersemester 2009/10



Übersicht

Unit Capacity Networks

Definition

- Ein Graph wird als Unit Capacity Network (oder 0-1 *Network*) bezeichnet, falls die Kapazität aller Kanten gleich 1 ist.
- Ein Unit Capacity Network ist vom Typ 1, falls es keine parallelen Kanten hat.
- Es ist vom Typ 2, falls für jeden Knoten v ($v \neq s$, $v \neq t$) entweder der Eingangsgrad $d^-(v)$ oder der Ausgangsgrad $d^+(v)$ gleich 1 ist.

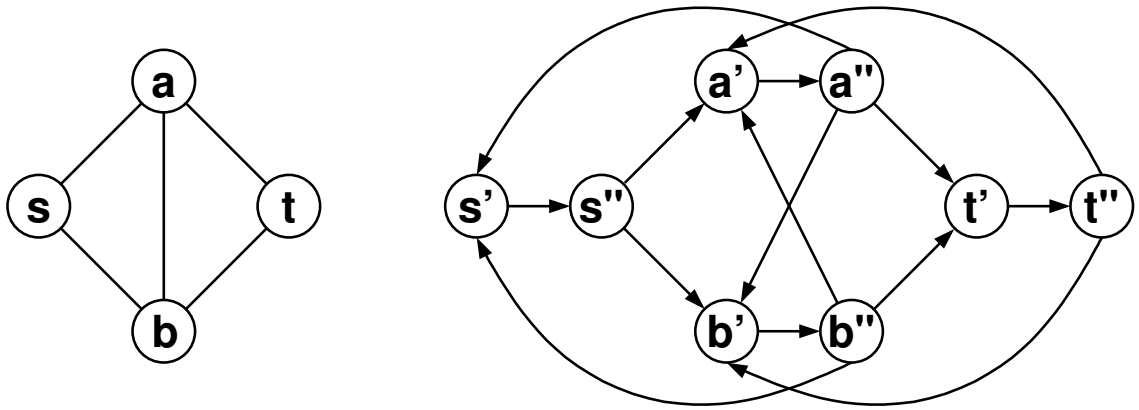
Unit Capacity Networks

Lemma

- *Ein MaxFlow/MinCut kann für ein Unit Capacity Network (mit Dinitz' Algorithmus) in Zeit $\mathcal{O}(m^{3/2})$ berechnet werden.*
- *Für Unit Capacity Networks vom Typ 1 ist die Zeitkomplexität von Dinitz' Algorithmus $\mathcal{O}(n^{2/3}m)$.*
- *Für Unit Capacity Networks vom Typ 2 ist die Zeitkomplexität von Dinitz' Algorithmus $\mathcal{O}(n^{1/2}m)$.*

(Beweis: siehe Shimon Even, Graph Algorithms, 1979)

Ungerichtete ungewichtete Graphen



- Gegeben: ungerichteter (ungewichteter) Graph $G = (V, E)$ mit n Knoten und m Kanten
- Konstruiere gerichteten Graph $\bar{G} = (\bar{V}, \bar{E})$ mit $|\bar{V}| = 2n$ und $|\bar{E}| = 2m + n$ wie folgt:
  - Ersetze jeden Knoten $v \in V$ durch zwei Knoten $v', v'' \in \bar{V}$, verbunden durch eine (interne) Kante $e_v = (v', v'') \in \bar{E}$.
  - Ersetze jede Kante $e = (u, v) \in E$ durch zwei (externe) Kanten $e' = (u'', v')$ und $e'' = (v'', u') \in \bar{E}$.

Ungerichtete ungewichtete Graphen

- $\kappa(s,t)$ wird nun berechnet als MaxFlow in $\bar{G}$ von Quelle s'' zu Senke t' mit Unit Capacity-Kanten
- Hinweis: $c(e_v) = 1$, $c(e') = c(e'') = \infty$ führt übrigens zum gleichen Ergebnis.
- Für jedes Paar $v', v'' \in \bar{V}$, das einen internen Knoten $v \in V$ repräsentiert, ist die interne Kante (v', v'') die einzige von v' ausgehende Kante und die einzige eingehende Kante von v''. Der Graph $\bar{G}$ ist also ein UCN vom Typ 2.
- Nach dem Lemma kann die Berechnung des MaxFlow bzw. des lokalen Knotenzusammenhangs in Zeit $\mathcal{O}(\sqrt{n}m)$ erfolgen.

Ungerichtete ungewichtete Graphen

Trivialer Algorithmus zur Bestimmung von $\kappa(G)$:

- Bestimme Minimum aller lokalen Knotenzusammenhangszahlen.
- Für die Endknoten jeder Kante (s, t) in G gilt:

$$\kappa_G(s, t) = n - 1$$

- Anzahl notwendiger MaxFlow-Berechnungen:

$$\frac{n(n-1)}{2} - m$$

Ungerichtete ungewichtete Graphen

Besserer Algorithmus zur Bestimmung von $\kappa(G)$:

- Betrachte minimalen Knoten-Separator $S \subset V$, der eine 'linke' Knotenteilmenge $L \subset V$ von einer 'rechten' Teilmenge $R \subset V$ separiert.
- Man könnte $\kappa(G)$ berechnen, indem man einen Knoten s in einer Teilmenge (L oder R) fixiert und die lokalen Zusammenhangszahlen $\kappa_G(s,t)$ für alle Knoten $t \in V \setminus \{s\}$ berechnet, wobei einer dieser Knoten auf der anderen Seite des Schnitts liegen muss.
- Problem: wie wählt man einen Knoten s, so dass s nicht zu jedem Minimum Vertex Separator gehört?
- Da $\kappa(G) \leq \delta(G)$, könnte man $\delta(G)+1$ Knoten für s versuchen. Einer davon kann nicht Teil aller Minimum Knoten-Separatoren sein.
- Der Algorithmus hat Komplexität $\mathcal{O}((\delta+1)\cdot(n-1)\cdot\sqrt{nm})) = \mathcal{O}(\delta n^{3/2} m)$

Knotenzusammenhang (Even & Tarjan)

Algorithmus 10 : Knotenzusammenhang κ (Even & Tarjan)

Input : Ungerichteter Graph $G = (V, E)$
Output : $\kappa(G)$

$\kappa_{\min} \leftarrow n - 1$;
$i \leftarrow 1$;
while $i \leq \kappa_{\min}$ **do**
  for $j \leftarrow i + 1$ **to** n **do**
    if $i > \kappa_{\min}$ **then**
      break;
    else if $\{v_i, v_j\} \notin E$ **then**
      Berechne $\kappa_G(v_i, v_j)$ mit MaxFlow-Prozedur;
      $\kappa_{\min} \leftarrow \min\{\kappa_{\min}, \kappa_G(v_i, v_j)\}$;
  $i \leftarrow i + 1$;
return $\kappa_{\min}$;

Knotenzusammenhang (Even & Tarjan)

Even/Tarjan-Algorithmus zur Berechnung des (globalen) Knotenzusammenhangs κ

- stoppt die Berechnung der lokalen Knotenzusammenhangszahlen $\kappa_G(v_i, v_j)$, falls das Minimum unter die Anzahl der momentan betrachteten Knoten i fällt
- Algorithmus betrachtet höchstens $\kappa + 1$ Knoten in der Schleife für Variable i
- Jeder Knoten hat mindestens $\delta(G)$ Nachbarn, also höchstens $n - \delta - 1$ Nicht-Nachbarn.

$\Rightarrow$ Es gibt maximal $\mathcal{O}((n - \delta - 1)(\kappa + 1))$ Aufrufe für die Berechnung des lokalen Zusammenhangs (MaxFlow für zwei gegebene Knoten).

$\Rightarrow$ Da $\kappa \leq \delta \leq \bar{d} = 2m/n$ wird der richtige Wert spätestens in Aufruf $2m/n + 1$ gefunden. Die Komplexität ist also $\mathcal{O}(\sqrt{n}m^2)$.

Knotenzusammenhang (Esfahanian & Hakimi)

Verbesserung von Esfahanian & Hakimi:

Lemma

Wenn ein Knoten v zu allen Knoten-Separatoren minimaler Kardinalität gehört, dann gibt es für jeden Minimum Vertex-Cut S zwei Knoten $l \in L_S$ und $r \in R_S$, die zu v adjazent sind.

Knotenzusammenhang (Esfahanian & Hakimi)

Beweis.

- Annahme: v ist an allen Minimum Vertex-Cutsets beteiligt.
- Betrachte die beiden (getrennten) Teile L und R des Restgraphen, der nach dem Löschen verbleibt.
- Jede der beiden Seiten muss einen Nachbarn von v enthalten, sonst wäre v nicht nötig, um die Teile zu trennen (und die Knotenmenge wäre damit kein minimaler Separator)
- Jede Seite, die mehr als einen Knoten enthält, muss sogar zwei Nachbarn von v enthalten, da man sonst durch Ersetzen von v durch den einzigen Nachbarn einen MinCut ohne v konstruieren könnte (Widerspruch zur Annahme).

□

Knotenzusammenhang (Esfahanian & Hakimi)

Algorithmus 11 : Knotenzusammenhang κ (Esfahanian & Hakimi)

Input : Ungerichteter Graph $G = (V, E)$
Output : $\kappa(G)$

$\kappa_{\min} \leftarrow n - 1$;
Wähle $v \in V$ mit minimalem Grad, also $d(v) = \delta(G)$;
Seien die Nachbarn $N(v) = \{v_1, v_2, \ldots, v_\delta\}$;

foreach *Nicht-Nachbar* $w \in V \setminus (N(v) \cup \{v\})$ **do**
    Berechne $\kappa_G(v, w)$ mit MaxFlow-Prozedur;
    $\kappa_{\min} \leftarrow \min\{\kappa_{\min}, \kappa_G(v, w)\}$;

$i \leftarrow 1$;
while $i \leq \kappa_{\min}$ **do**
    for $j \leftarrow i + 1$ **to** $\delta - 1$ **do**
        if $i \geq \delta - 2$ *or* $i > \kappa_{\min}$ **then**
            return $\kappa_{\min}$;
        else if $\{v_i, v_j\} \notin E$ **then**
            Berechne $\kappa_G(v_i, v_j)$ mit MaxFlow-Prozedur;
            $\kappa_{\min} \leftarrow \min\{\kappa_{\min}, \kappa_G(v_i, v_j)\}$;
    $i \leftarrow i + 1$;
return $\kappa_{\min}$;

Knotenzusammenhang (Esfahanian & Hakimi)

- erste Schleife:
  - Anzahl der Nicht-Nachbarn kann wieder höchstens $n - \delta - 1$ sein

  $\Rightarrow$ höchstens $n - \delta - 1$ MaxFlow-Aufrufe
- zweite Schleife: $\kappa(2\delta - \kappa - 3)/2$
- Gesamtkomplexität: $n - \delta - 1 + \kappa(2\delta - \kappa - 3)/2$