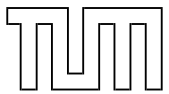
Fortgeschrittene Netzwerk- und Graph-Algorithmen

Prof. Dr. Hanjo Täubig

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

Wintersemester 2010/11



Knotenzusammenhang (Even & Tarjan)

Algorithmus 14 : 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
- 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.

$\Rightarrow$ 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.

$\Rightarrow$ Komplexität: $\mathcal{O}(\sqrt{n}m^2)$

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 $\ell \in L_S$ 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 15 : 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: $n - \delta - 1 + \kappa(2\delta - \kappa - 3)/2$

Kantenzusammenhangsalgorithmen

- Kantenzusammenhang λ kann in ungerichteten ungewichteten Graphen ebenfalls mit der MaxFlow-Prozedur berechnet werden.
- Ersetze dafür jede ungerichtete Kante durch zwei antiparallele gerichtete Kanten mit Kapazität 1
- Berechne dann lokalen Zusammenhang zwischen der entsprechenden Quelle s und der Senke t.

$\Rightarrow$ Resultierendes Netzwerk ist Unit Capacity Network vom Typ 1

$\Rightarrow$ Komplexität zur Berechnung des lokalen Zusammenhangs für ein Knotenpaar: $\mathcal{O}(m \cdot \min\{m^{1/2}, n^{2/3}\})$

Einfache Kantenzusammenhangsalgorithmen

- Trivialer Algorithmus: mit $n(n-1)/2$ MaxFlow-Aufrufen den lokalen Kantenzusammenhang aller Knotenpaare berechnen

- Besser: einen Knoten s festzuhalten und dann für alle anderen Knoten t die lokalen Zusammenhangszahlen $\lambda(s,t)$ berechnen

  Mindestens einer dieser Knoten muss auf der anderen Seite eines MinCuts liegen.

  Deshalb ist das Minimum aller dieser $(n-1)$ lokalen Zusammenhangszahlen $\lambda(s,t)$ gleich dem (globalen) Kantenzusammenhang λ des Graphen.

  Gesamtkomplexität: $\mathcal{O}(nm \cdot \min\{n^{2/3}, m^{1/2}\})$

λ-Covering

- Der Algorithmus funktioniert auch, wenn man statt der ganzen Knotenmenge nur eine Teilmenge verwendet, die zwei Knoten s, t enthält, deren lokaler Zusammenhang $\lambda(s, t)$ gleich dem globalen Zusammenhang λ ist.

- Eine solche Teilmenge heißt λ-Covering.

$\Rightarrow$ Versuche, die Kardinalität dieser Knotenmenge zu reduzieren.

Komponentengröße

Lemma

Sei S ein Minimum Edge-Cut eines Graphen $G = (V, E)$ und sei $L, R \subset V$ eine Partition der Knotenmenge, so dass L and R durch S separiert werden.

Wenn $\lambda(G) < \delta(G)$, dann besteht jede Komponente von $G - S$ aus mehr als $\delta(G)$ Knoten, d.h. es gilt $|L| > \delta(G)$ und $|R| > \delta(G)$.

Komponentengröße

Beweis.

- Seien $l_1, \ldots, l_k$ die Elemente von L und sei $E[L] = E(G[L])$ die Menge der durch L induzierten Kanten in G.
- Es gilt:

$$\begin{aligned} \delta_G \cdot k &\leq \sum_{i=1}^{k} d_G(l_i) \\ &\leq 2 \cdot |E[L]| + |S| \\ &\leq 2 \cdot \frac{k(k-1)}{2} + |S| \\ &< k(k-1) + \delta_G \end{aligned}$$

- Aus $\delta_G \cdot (k-1) < k(k-1)$ folgt $|L| = k > 1$ und $|L| = k > \delta_G$ (sowie $|R| > \delta(G)$).

Komponentengröße

Folgerung

Wenn gilt $\lambda_G < \delta_G$, dann enthält jede Komponente von $G - S$ einen Knoten, der mit keiner Kante in S inzident ist.

Kantenzusammenhang λ

Lemma

Sei $\lambda_G < \delta_G$ und sei T ein Spannbaum von G.
Dann enthalten alle Komponenten von $G - S$ mindestens einen Knoten, der kein Blatt in T ist, d.h. die inneren Knoten von T bilden ein λ-Cover.

Beweis.

- Nehmen wir an, dass es nicht so wäre (d.h. alle Knoten in L sind Blätter von T).
- Also für keine Kante von T sind beide Endknoten in L, d.h. $|L| = |S|$.
- Aus dem vorhergehenden Lemma ($\lambda_G < \delta_G \Rightarrow |L| > \delta_G$ und $|R| > \delta_G$) folgt, dass $\lambda_G = |S| = |L| > \delta_G$ (Widerspruch zur Annahme).

Spannbaum-Berechnung (Esfahanian & Hakimi)

Algorithmus von Esfahanian & Hakimi:

- Berechne Spannbaum des gegebenen Graphen.
- Wähle beliebigen inneren Knoten v des Baums.
- Berechne für jeden anderen Knoten w, der kein Blatt ist, den lokalen Kantenzusammenhang $\lambda(v, w)$.
- Das Minimum dieser Wertemenge, zusammen mit δ_G, ergibt genau den Kantenzusammenhang λ_G.
- Ein Baum mit möglichst vielen Blättern wäre vorteilhaft, aber die Konstruktion eines optimalen Baums ist $\mathcal{NP}$-hart.

Spannbaum-Berechnung (Esfahanian & Hakimi)

Esfahanian & Hakimi:

- Algorithmus zur Berechnung eines Spannbaums T von G, so dass beide Mengen, L und R eines minimalen Kantenseparators mindestens ein Blatt von T enthalten, und nach dem letzten Lemma mindestens einen inneren Knoten.

Spannbaum-Berechnung (Esfahanian & Hakimi)

Algorithmus 16 : Spannbaum-Berechnung (Esfahanian & Hakimi)

Input : Ungerichteter Graph $G = (V, E)$
Output : Spannbaum T mit einem Blatt und einem inneren Knoten in L bzw. R

Wähle $v \in V$;
$T \leftarrow$ alle Kanten inzident mit v;
while $|E(T)| < n - 1$ **do**
  Wähle ein Blatt w in T, so dass für alle Blätter r in T gilt:
  $|N(w) \cap (V - V(T))| \geq |N(r) \cap (V - V(T))|$;
  $T \leftarrow T \cup \{(w, x) \in E : x \in (V - V(T))\}$
return T;

Kantenzusammenhang λ (Esfahanian & Hakimi)

Algorithmus 17 : Kantenzusammenhang λ (Esfahanian & Hakimi)

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

Konstruiere einen Spannbaum T (voriger Algorithmus);
Sei P die kleinere der beiden Mengen
(entweder die Blätter oder die inneren Knoten von T);
Wähle einen Knoten $u \in P$;
$c \leftarrow \min\{\lambda_G(u, v) : v \in P \setminus \{u\}\}$;
$\lambda \leftarrow \min(\delta(G), c)$;
return λ;

Kantenzusammenhang λ (Esfahanian & Hakimi)

- Da P als kleinere der beiden Mengen (Blätter / innere Knoten) gewählt wird, benötigt der Algorithmus höchstens $n/2$ Berechnungen für lokale Zusammenhangszahlen.

$\Rightarrow$ Gesamtzeitkomplexität: $\mathcal{O}(\lambda mn)$

Kantenzusammenhang λ (Matula)

Definition

Ein Dominating Set für einen Graphen $G = (V, E)$ ist eine Knotenteilmenge V' von V, so dass jeder Knoten, der nicht in V' ist, mit mindestens einem Knoten in V' über eine Kante verbunden ist.

Lemma

Im Fall $\lambda(G) < \delta(G)$ ist jedes Dominating Set von G auch ein λ-covering von G.

Kantenzusammenhang λ (Matula)

Verbesserter Algorithmus von Matula:

- Analog zur Spannbaum-Methode, wird λ hier berechnet, indem man
  - ein Dominating Set D von G berechnet,
  - einen beliebigen Knoten $u \in D$ auswählt und
  - den lokalen Kantenzusammenhang $\lambda(u, v)$ für alle anderen Knoten $v \in D \setminus \{u\}$ berechnet.
- Das Minimum der Wertemenge (wieder zusammen mit δ_G) ist der Kantenzusammenhang.
- Obwohl die Berechnung eines Dominating Sets minimaler Kardinalität $\mathcal{NP}$-hart ist, kann man zeigen, dass der Algorithmus in Zeit $\mathcal{O}(mn)$ läuft, wenn man das Dominating Set wie im folgenden Algorithmus konstruiert.

Kantenzusammenhang λ (Matula)

Algorithmus 18 : Berechnung eines Dominating Sets (Matula)

Input : Ungerichteter Graph $G = (V, E)$
Output : Dominating Set D

Wähle $v \in V$;
$D \leftarrow \{v\}$;
while $V \setminus (D \cup N(D)) \neq \emptyset$ **do**
  Wähle einen Knoten $w \in V \setminus (D \cup N(D))$;
  $D \leftarrow D \cup \{w\}$;
return D;