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

All-Pairs MaxFlow / MinCut

- gegeben:
  Graph $G = (V, E)$ mit n Knoten,
  p ausgewählte Knoten

- gesucht:
  (lokale) MaxFlow/MinCut-Werte für alle Paare aus den p gegebenen Knoten

$\Rightarrow$ kann einfach durch Lösen von $\binom{p}{2} = \frac{1}{2}p(p-1)$ MaxFlow-Problemen berechnet werden.

- kann in ungerichteten Graphen aber auch mit nur $p-1$ MaxFlow-Berechnungen gelöst werden.

Equivalent Flow Trees

Sei G also ungerichtet und sei $v_{ij} = v_{ji}$ der Wert eines MaxFlows zwischen den Knoten i und j.

Lemma

1. *Für alle Knoten $i, j, k \in \{1, \ldots, n\}$ gilt:*

$$v_{ik} \geq \min\{v_{ij}, v_{jk}\}$$

2. *Es gibt einen Baum T auf den Knoten 1 bis n, so dass für alle Paare von Knoten i, j gilt:*

$$v_{ij} = \min\{v_{ij_1}, v_{j_1 j_2}, \ldots, v_{j_k j}\},$$

wobei $i - j_1 - \cdots - j_k - j$ der (eindeutige) Pfad von Knoten i zu Knoten j in T ist.

Equivalent Flow Trees

Beweis.

1. Sei $(X, \bar{X})$ ein i, k-MinCut, d.h. $v_{ik} = w(X, \bar{X})$.
   - Falls $j \in \bar{X}$, dann gilt $v_{ij} \leq w(X, \bar{X}) = v_{ik}$.
   - Falls $j \in X$, dann gilt $v_{jk} \leq w(X, \bar{X}) = v_{ik}$.
2. Betrachte K_n (vollständiger Graph mit n Knoten), wobei jede Kante (i, j) Gewicht v_{ij} hat.
   Sei T darin ein Spannbaum maximalen Gewichts.
   - Induktiv folgt aus dem vorangegangenen Punkt, dass für jedes Knotenpaar (i, j) gilt: $v_{ij} \geq \min\{v_{ij_1}, v_{j_1 j_2}, \ldots, v_{j_k j}\}$, wobei $i - j_1 - \cdots - j_k - j$ der (eindeutige) Pfad von Knoten i zu Knoten j in T ist.
   - Angenommen für ein Paar (i, j) gilt echte Ungleichheit.
     Dann gibt es eine Kante (i', j') auf dem Pfad zwischen i und j mit $v_{ij} > v_{i'j'}$.
     Das würde bedeuten, dass der Baum, der aus T durch Tausch von (i', j') gegen (i, j) entsteht, ein größeres Gewicht hätte. (Widerspruch)

□

Algorithmus von Gomory und Hu

- Die Existenz eines Equivalent Flow Trees hat natürlich gleichzeitig die Konsequenz, dass unter den $\binom{n}{2}$ MaxFlow- bzw. MinCut-Werten v_{ij} nur höchstens $n-1$ verschiedene Werte existieren können (nämlich die Kantengewichte von T).
- Außerdem kommt man so zu der Vermutung, dass $n-1$ MaxFlow-Berechnungen genügen, um einen solchen Equivalent Flow Tree T zu konstruieren und damit alle v_{ij}-Werte zu berechnen. Ein Algorithmus von Gomory und Hu erreicht dies tatsächlich.

Algorithmus von Gomory und Hu

- Gegeben zwei Knoten i, j und ein i, j-MinCut $(X, \bar{X})$.
- Definiere den kontrahierten Graph G^c als den Graph, den man aus G erhält, indem man die Knoten in $\bar{X}$ zu einem einzelnen (speziellen) Knoten $u_{\bar{X}}$ zusammenfasst und für jeden Knoten $v \in X$ alle über den Schnitt führenden Kanten durch eine einzelne Kante $\{v, u_{\bar{X}}\}$ mit der entsprechend summierten Gesamtkapazität ersetzt.

Lemma

Für jedes Paar von (normalen) Knoten i', j' in G^c (d.h. $i', j' \in X$) hat der MaxFlow bzw. MinCut zwischen i' und j' in G^c den gleichen Wert wie der MaxFlow/MinCut im Original-Graphen.
(Entsprechendes gilt natürlich für Knoten $i', j' \in \bar{X}$, wenn man X in G kontrahiert.)

Algorithmus von Gomory und Hu

Corollary

Für jedes Paar von Knoten $i', j' \in X$ (oder $\bar{X}$) gibt es einen Minimum i', j'-Cut, so dass sich alle Knoten von $\bar{X}$ (bzw. X) auf der gleichen Seite des Schnitts befinden.

- Für zwei Knoten i, j und einen i, j-MinCut $(X, \bar{X})$ repräsentiert man die aktuelle Situation durch einen Baum mit zwei Knoten (die für X und $\bar{X}$ stehen) und einer Kante mit dem Gewicht v_{ij}. Die Kante repräsentiert dabei den Schnitt $(X, \bar{X})$.
- Sei A die Menge der p zu betrachtenden Knoten, für die die MaxFlow/MinCut-Werte berechnet werden sollen.
- Die ersten beiden Knoten i und j werden aus A gewählt.

Algorithmus von Gomory und Hu

Definition

Ein Baum T wird als Semi-Cut Tree bezeichnet, falls er folgende Eigenschaften besitzt:

1. Jeder Knoten U von T entspricht einer Teilmenge der Knoten von G und enthält mindestens einen Knoten der Menge A.
2. Jede Kante (U, V) ist mit einem Label v versehen, so dass es Knoten $i, j \in A$ mit $i \in U$ und $j \in V$ gibt und der MaxFlow/MinCut zwischen i und j den Wert v hat.
3. Jede Kante (U, V) repräsentiert einen i, j-MinCut mit $i, j \in A$ und i ist in einem Knoten des Teilbaums auf der Seite von U enthalten und j ist in einem Knoten des Teilbaums auf der Seite von V enthalten (und die zwei Teile des Cut bestehen aus der jeweiligen Knotenmenge).

Wenn jeder Knoten eines Semi-Cut Trees T genau einen Knoten der Menge A enthält, dann bezeichnet man T als Cut Tree für A.

Algorithmus von Gomory und Hu

Satz

Sei T ein Cut Tree für A.
Dann gilt für jedes Paar $i, j \in A$: $v_{ij} = \min\{v_1, \ldots, v_{k+1}\}$, wobei v_1 bis v_{k+1} die Kantengewichte in T auf dem Pfad vom Knoten, der i enthält, zum Knoten, der j enthält, sind.

Beweis.

- Sei $i - j_1 - \ldots - j_k - j$ die Folge von A-Knoten, die dem Pfad in T vom Knoten mit i zum Knoten mit j entsprechen.
- Aufgrund der Eigenschaften des Cut Trees sind die Kantenlabels einfach $v_{ij_1}, \ldots, v_{j_k j}$.
- Aufgrund des Lemmas gilt: $v_{ij} \geq \min\{v_{ij_1}, \ldots, v_{j_k j}\}$.
- Andererseits entspricht jede Kante einem i, j-Schnitt, wobei die Kapazität dem Kantenlabel entspricht. Es gilt also: $v_{ij} \leq \min\{v_{ij_1}, \ldots, v_{j_k j}\}$ und damit $v_{ij} = \min\{v_{ij_1}, \ldots, v_{j_k j}\}$.

Algorithmus von Gomory und Hu

Satz

Ein Cut Tree für A existiert und kann mit Hilfe von nur $p-1$ MaxFlow-Berechnungen konstruiert werden.

Beweis.

- Angenommen, wir haben einen Semi-Cut Tree T für A und T hat noch einen Knoten U, der zwei Knoten $i, j \in A$ enthält.
- Aufgrund des Lemmas hat der MaxFlow zwischen i und j in G genau den gleichen Wert wie der MaxFlow zwischen i und j in dem (kontrahierten) Graph G^c, der entsteht, wenn man die Knotenmengen in jedem zu U verbundenen Teilbaum zu einem einzelnen (speziellen) Knoten kontrahiert und für jeden (Original-)Knoten $v \in U$ die Kanten, die in die jeweiligen Teilbäume führen, durch einzelne Kanten zu den entsprechenden neuen (speziellen) Knoten mit aufsummiertem Kantengewicht ersetzt.

Algorithmus von Gomory und Hu

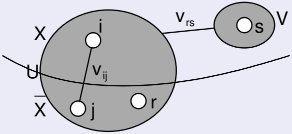
Beweis.

- Sei $(X, \bar{X})$ ein i, j-MinCut in G^c (notwendigerweise mit Kapazität v_{ij}).
- Konstruiere einen Baum T' wie folgt:
  - Spalte U in zwei Knoten X und $\bar{X}$.
  - Verbinde die Knoten X und $\bar{X}$ durch eine Kante mit Label v_{ij} und verbinde jeden Nachbarn V von U entweder zu X oder zu $\bar{X}$, in Abhängigkeit von der Seite des Cuts, auf der sich der zu V's Teilbaum gehörende spezielle Knoten in G^c befunden hat (mit Original-Kantenlabel).
- Die Behauptung ist nun, dass T' auch wieder ein Semi-Cut Tree für A ist.

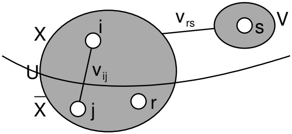
Algorithmus von Gomory und Hu

Beweis.

- Die einzige nicht-triviale zu überprüfende Eigenschaft ist der zweite Teil der Definition.
- Sei also V irgendein Nachbar von U in T und entspreche das Label der Kante (U, V) einem MaxFlow/MinCut-Wert zwischen $r \in U$ und $s \in V$ $(r, s \in A)$.
- Sei o.B.d.A. $j, r \in \bar{X}$, dann können folgende zwei Fälle auftreten:
  1. V wird mit $\bar{X}$ verbunden.
     Die Eigenschaften des Semi-Cut Trees bleiben erhalten.
  2. V wird mit X verbunden.



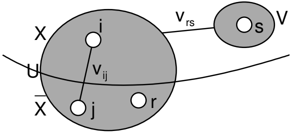
Algorithmus von Gomory und Hu



Beweis.

- Das Label v_{ij} für die Kante $\{X, \bar{X})$ entspricht der Definition.
- Betrachte nun das Label (v_{rs}) der Kante (X, V).
  Es wird gezeigt, dass $v_{is} = v_{rs}$:
  - $v_{is} \geq \min\{v_{ij}, v_{jr}, v_{rs}\}$ (Lemma).
  - Da i und s auf der gleichen Seite des Cuts sind, können die MaxFlow/MinCut-Werte zwischen Knoten in $\bar{X}$ nicht den Wert von v_{is} beeinflussen (v_{jr} ist also egal) und es gilt $v_{is} \geq \min\{v_{ij}, v_{rs}\}$.

Algorithmus von Gomory und Hu



Beweis.

- Andererseits muss v_{is} durch den zur Kante (U, V) in T gehörigen MinCut beschränkt sein, d.h. $v_{is} \leq v_{rs}$.
- $(X, \bar{X})$ hat den Wert v_{ij} und ist auch ein i-r-Cut, also $v_{ij} \geq v_{ir} \geq \min\{v_{is}, v_{rs}\}$
- $(X, \bar{X})$ ist auch ein r-s-Cut, also $v_{ij} \geq v_{rs}$
- Mit $v_{rs} \geq v_{is} \geq \min\{v_{ij}, v_{rs}\} = v_{rs}$ folgt daraus $v_{is} = v_{rs}$ und die Kante (X, V) entspricht dem MaxFlow zwischen i und s (mit Wert und Schnitt-Mengen wie im Original-Graph).

- Eine wiederholte Aufspaltung liefert den Cut Tree mit Hilfe von $p - 1$ MaxFlow-Berechnungen.

□

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.

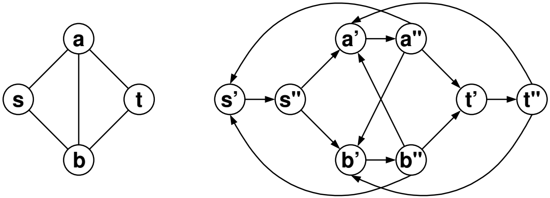
Unit Capacity Networks

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 $\mathcal{O}((\delta+1)\cdot(n-1)\cdot\sqrt{n}m)) = \mathcal{O}(\delta n^{3/2} m)$