Techniken zur schnelleren Berechnung von Gomory-Hu-Bäumen

Diplomarbeitsverteidigung

Sascha Grau
Matr-Nr. 33690

10. Oktober 2007

Inhalt

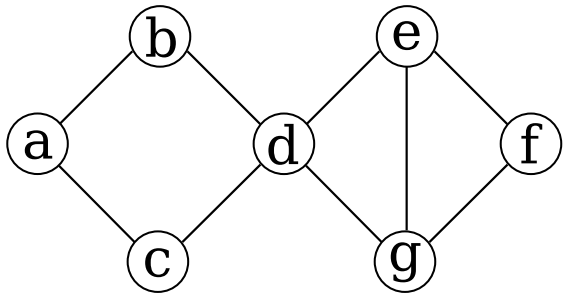
Inhalt

Problemstellung

Definition (ALL-PAIRS-MINIMUM-CUT)

Ist $G = (V, E)$ ein ungerichteter Graph mit Kantengewichtsfunktion $w_G : E \to \mathbb{R}^+$, berechne für jedes Knotenpaar $u, v \in V$ den *Wert* $\lambda_{u,v}$ und *Verlauf* eines minimalen u-v-Schnitts in G.

Es gibt $\binom{n}{2}$ solcher Knotenpaare.
Wie lässt sich ein solches Ergebnis optimal repräsentieren?

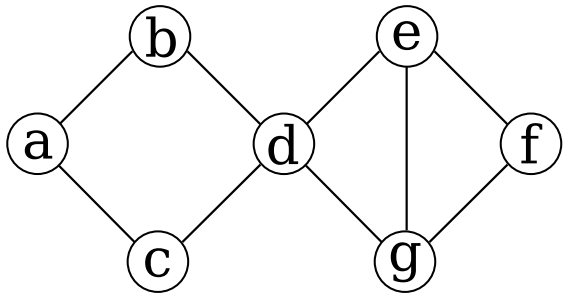


Problemstellung

Definition (ALL-PAIRS-MINIMUM-CUT)

Ist $G = (V, E)$ ein ungerichteter Graph mit Kantengewichtsfunktion $w_G : E \to \mathbb{R}^+$, berechne für jedes Knotenpaar $u, v \in V$ den *Wert* $\lambda_{u,v}$ und *Verlauf* eines minimalen u-v-Schnitts in G.

Es gibt $\binom{n}{2}$ solcher Knotenpaare.
Wie lässt sich ein solches Ergebnis optimal repräsentieren?

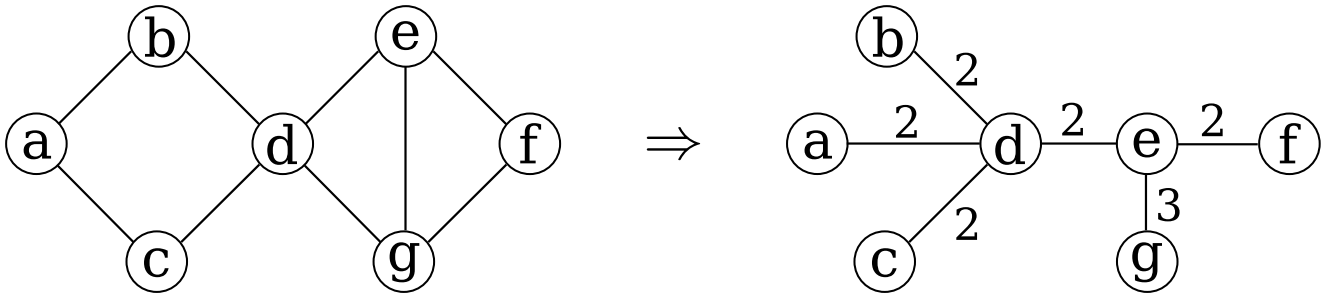


Problemstellung

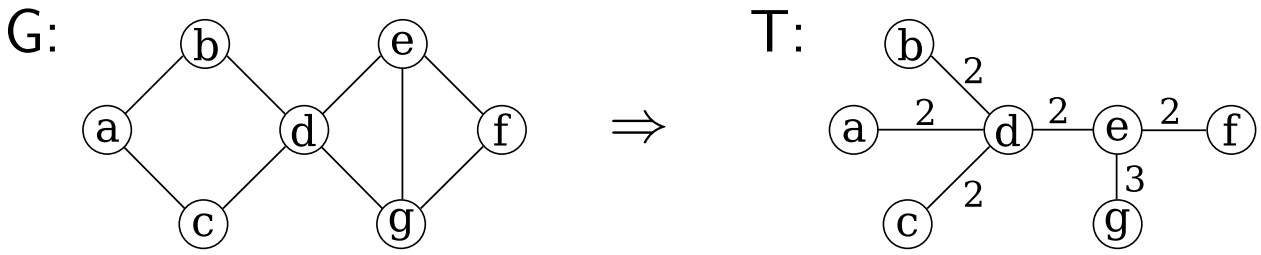
Definition (ALL-PAIRS-MINIMUM-CUT)

Ist $G = (V, E)$ ein ungerichteter Graph mit Kantengewichtsfunktion $w_G : E \to \mathbb{R}^+$, berechne für jedes Knotenpaar $u, v \in V$ den *Wert* $\lambda_{u,v}$ und *Verlauf* eines minimalen u-v-Schnitts in G.

Es gibt $\binom{n}{2}$ solcher Knotenpaare.
Wie lässt sich ein solches Ergebnis optimal repräsentieren?



Der Gomory-Hu-Baum



Definition (Gomory-Hu-Baum)

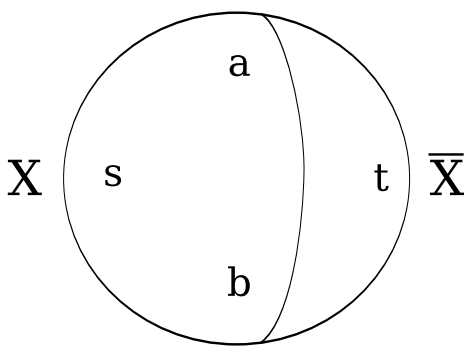
Der Baum $T = (\mathbf{V}, A)$ mit Kantengewichtsfunktion $w_T : A \to \mathbb{R}^+$ ist genau dann ein *Gomory-Hu-Baum* für $G = (\mathbf{V}, E)$, falls gilt:

1. Die minimal gewichtete Kante auf jedem Pfad zwischen zwei Knoten $u, v \in V$ in T, ist mit dem minimalen u-v-Schnittwert in G gewertet.
2. $\forall a \in A$: Sind T_1 und T_2 die Knotenmengen der durch Löschung von a aus T entstehenden Teilbäume, so gilt $w_T(a) := c_G(T_1)$.

Warum lassen sich die Schnitte in Baumform repräsentieren?

Lemma (Kreuzungsfreiheit - Gomory und Hu, 1961)

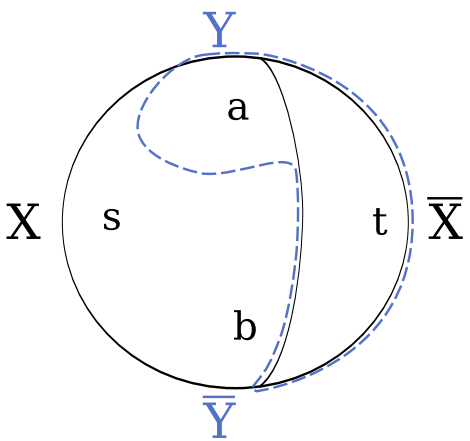
Seien X und $\overline{X}$ die beiden Schnittmengen eines minimalen s-t-Schnittes mit $s \in X$ und $t \in \overline{X}$. Existieren weiterhin zwei Knoten $a, b \in X$, so gibt es mindestens einen minimalen a-b-Schnitt $\{Y, \overline{Y}\}$, so dass $\overline{X} \subseteq Y$ oder $\overline{X} \subseteq \overline{Y}$ gilt.

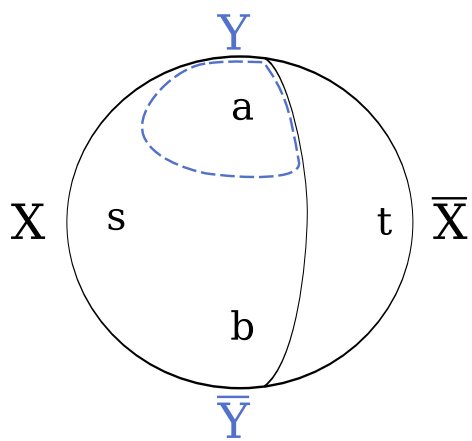


Warum lassen sich die Schnitte in Baumform repräsentieren?

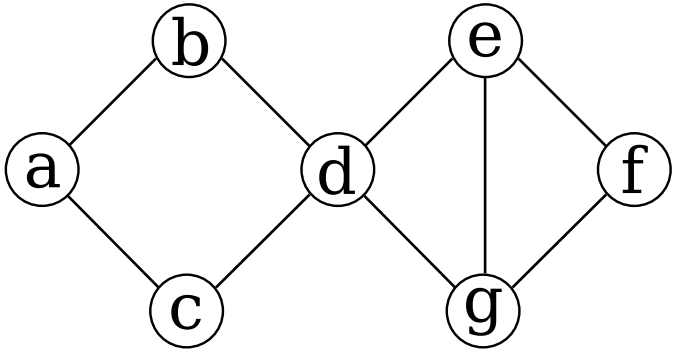
Lemma (Kreuzungsfreiheit - Gomory und Hu, 1961)

Seien X und $\overline{X}$ die beiden Schnittmengen eines minimalen s-t-Schnittes mit $s \in X$ und $t \in \overline{X}$. Existieren weiterhin zwei Knoten $a, b \in X$, so gibt es mindestens einen minimalen a-b-Schnitt $\{Y, \overline{Y}\}$, so dass $\overline{X} \subseteq Y$ oder $\overline{X} \subseteq \overline{Y}$ gilt.

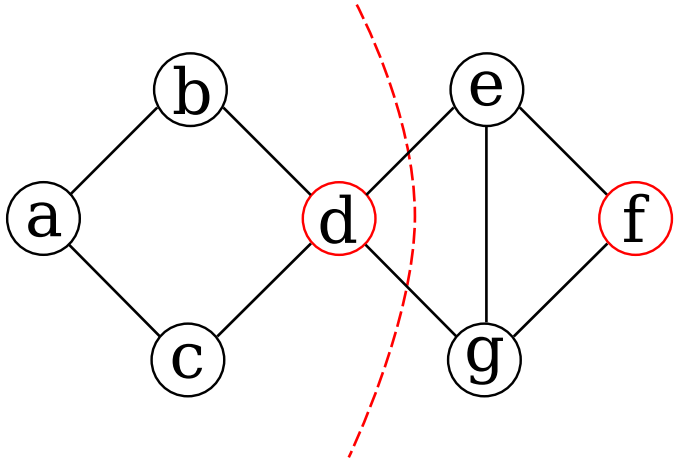




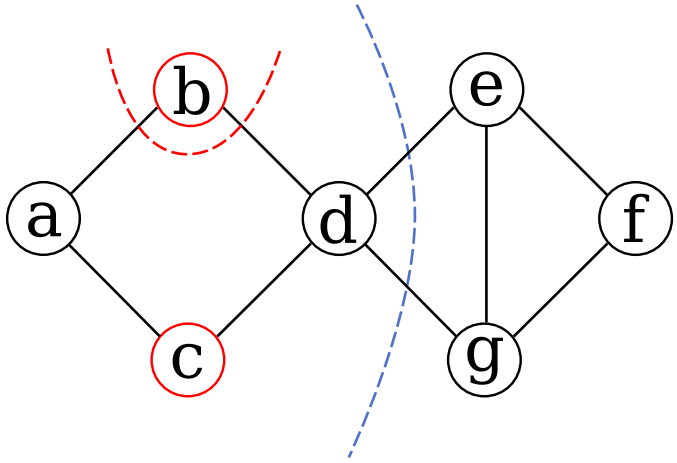
Kreuzungsfreiheit Minimaler Schnitte



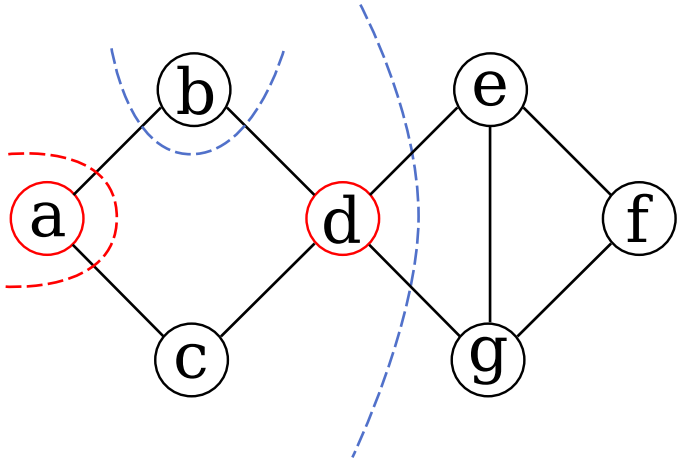
Kreuzungsfreiheit Minimaler Schnitte



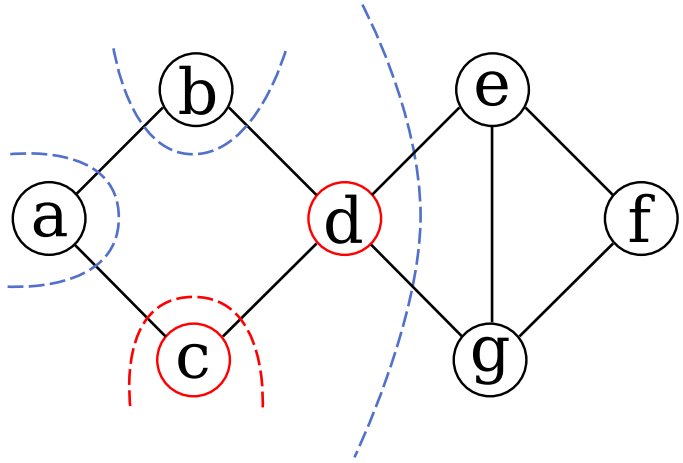
Kreuzungsfreiheit Minimaler Schnitte



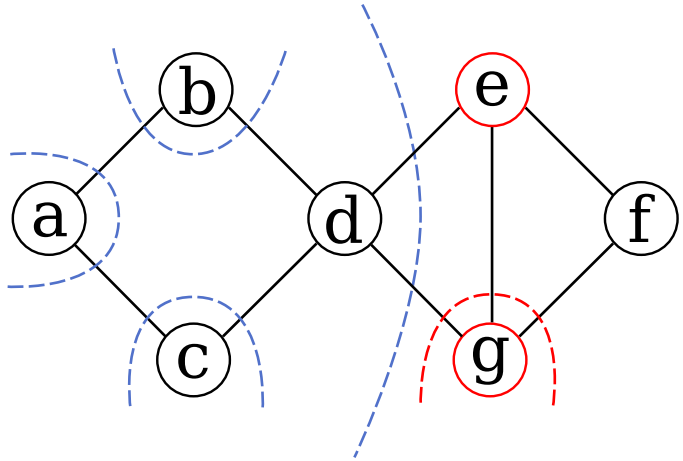
Kreuzungsfreiheit Minimaler Schnitte



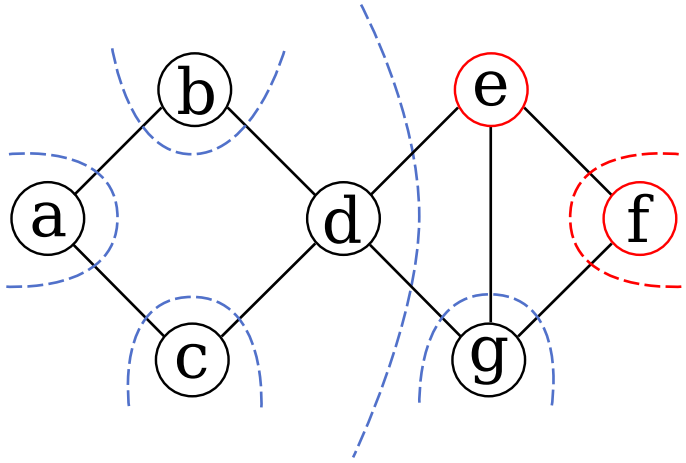
Kreuzungsfreiheit Minimaler Schnitte



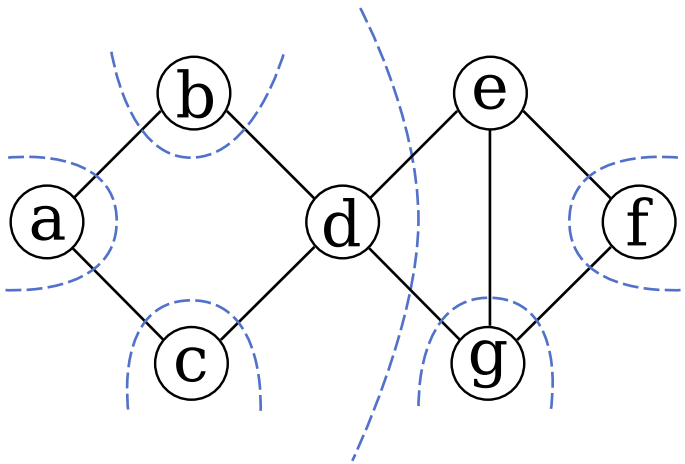
Kreuzungsfreiheit Minimaler Schnitte

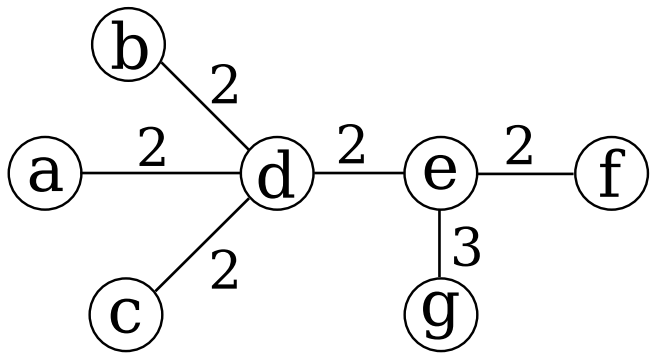


Kreuzungsfreiheit Minimaler Schnitte



Kreuzungsfreiheit Minimaler Schnitte





Gomory-Hu-Baum

Inhalt

Der Algorithmus von Gomory & Hu

Ziel

Finde kreuzungsfreie Auswahl minimaler Schnitte zwischen unterschiedlichen Knotenpaaren, so dass jedes Paar aus V getrennt wird.

Datenstruktur: Der Baum T

Reell gewichteter Baum T von *Superknoten*.

- Jeder Superknoten steht für Teilmenge von V
- T modelliert Partitionierung von V durch bisherige Schnitte
- Kanten tragen Schnittgewichte aus G

Initialisierung: $T = (\{V\}, \emptyset)$

Der Algorithmus von Gomory & Hu

Ziel (verfeinert)

Solange bisher ungetrenntes Knotenpaar (s, t) existiert, finde minimalen s-t-Schnitt kreuzungsfrei zu allen bisherigen Schnitten.

Datenstruktur: Der Baum T

Reell gewichteter Baum T von *Superknoten*.

- Jeder Superknoten steht für Teilmenge von V
- T modelliert Partitionierung von V durch bisherige Schnitte
- Kanten tragen Schnittgewichte aus G

Initialisierung: $T = (\{V\}, \emptyset)$

Der Algorithmus von Gomory & Hu

Ziel (verfeinert)

Solange bisher ungetrenntes Knotenpaar (s, t) existiert, finde minimalen s-t-Schnitt kreuzungsfrei zu allen bisherigen Schnitten.

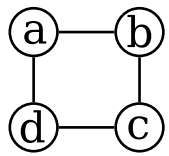
Datenstruktur: Der Baum T

Reell gewichteter Baum T von *Superknoten*.

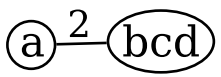
- Jeder Superknoten steht für Teilmenge von V
- T modelliert Partitionierung von V durch bisherige Schnitte
- Kanten tragen Schnittgewichte aus G

Initialisierung: $T = (\{V\}, \emptyset)$

G:



T:



Der Algorithmus von Gomory & Hu

Ziel (verfeinert)

Solange bisher ungetrenntes Knotenpaar (s, t) existiert, finde minimalen s-t-Schnitt kreuzungsfrei zu allen bisherigen Schnitten.

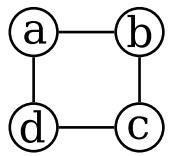
Datenstruktur: Der Baum T

Reell gewichteter Baum T von *Superknoten*.

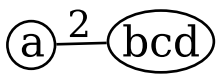
- Jeder Superknoten steht für Teilmenge von V
- T modelliert Partitionierung von V durch bisherige Schnitte
- Kanten tragen Schnittgewichte aus G

Initialisierung: $T = (\{V\}, \emptyset)$

G:



T:



Ist mit $\{X, \overline{X}\}$ ein minimaler s-t-Schnitt mit $a, b \in X$ bekannt. Wie kann ein zu $\overline{X}$ kreuzungsfreier minimaler a-b-Schnitt berechnet werden?

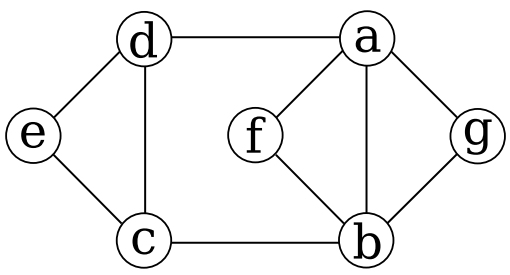
Ist mit $\{X, \overline{X}\}$ ein minimaler s-t-Schnitt mit $a, b \in X$ bekannt. Wie kann ein zu $\overline{X}$ kreuzungsfreier minimaler a-b-Schnitt berechnet werden?

Kontraktionsgraphen zur Verhinderung von Schnittkreuzungen

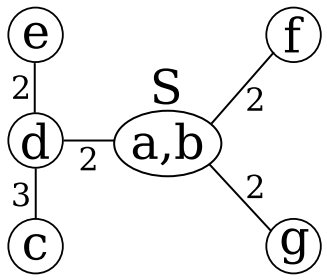
Erstellung einer kontrahierten Version des Graphen G für Superknoten S mit $a, b \in S$:

1. Definiere S als Baumwurzel von T.
2. Für jeden von S abgehenden Teilbaum B:
   1. Verschmelze die Knotenmengen aller Superknoten in B.
   2. Kontrahiere die entstandene Knotenmenge in G.
3. Berechne minimalen a-b-Schnitt im Kontraktionsgraphen

G:



T:



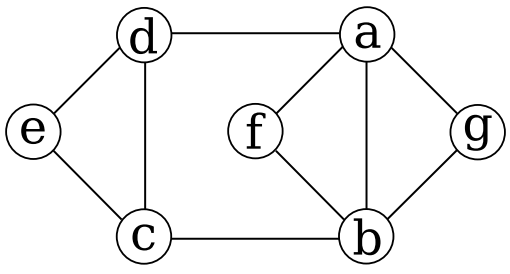
Ist mit $\{X, \overline{X}\}$ ein minimaler s-t-Schnitt mit $a, b \in X$ bekannt. Wie kann ein zu $\overline{X}$ kreuzungsfreier minimaler a-b-Schnitt berechnet werden?

Kontraktionsgraphen zur Verhinderung von Schnittkreuzungen

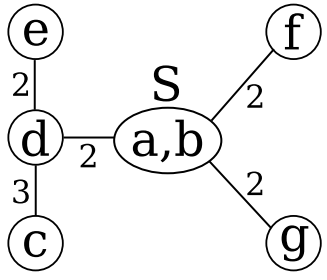
Erstellung einer kontrahierten Version des Graphen G für Superknoten S mit $a, b \in S$:

1. Definiere S als Baumwurzel von T.
2. Für jeden von S abgehenden Teilbaum B:
   1. Verschmelze die Knotenmengen aller Superknoten in B.
   2. Kontrahiere die entstandene Knotenmenge in G.
3. Berechne minimalen a-b-Schnitt im Kontraktionsgraphen

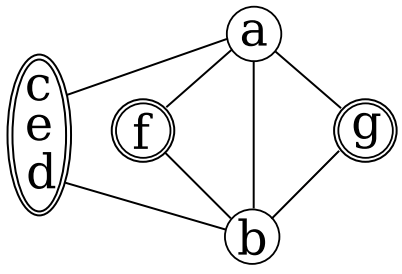
G:



T:



G':



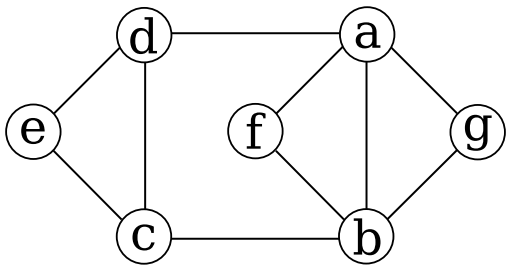
Ist mit $\{X, \overline{X}\}$ ein minimaler s-t-Schnitt mit $a, b \in X$ bekannt. Wie kann ein zu $\overline{X}$ kreuzungsfreier minimaler a-b-Schnitt berechnet werden?

Kontraktionsgraphen zur Verhinderung von Schnittkreuzungen

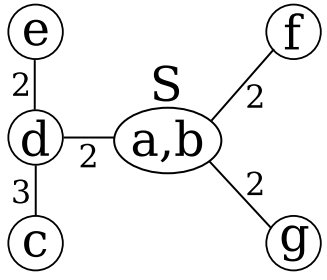
Erstellung einer kontrahierten Version des Graphen G für Superknoten S mit $a, b \in S$:

1. Definiere S als Baumwurzel von T.
2. Für jeden von S abgehenden Teilbaum B:
   1. Verschmelze die Knotenmengen aller Superknoten in B.
   2. Kontrahiere die entstandene Knotenmenge in G.
3. Berechne minimalen a-b-Schnitt im Kontraktionsgraphen

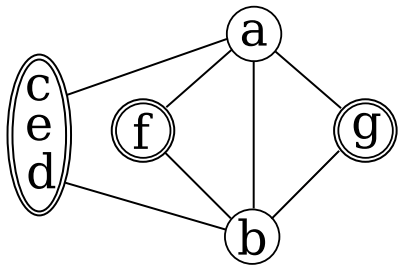
G:



T:



G':



Der Algorithmus von Gomory & Hu

$T \leftarrow (\{V\}, \emptyset)$;

while $\exists$ *Superknoten* S *in* T *mit* $|S| > 1$ **do**
- Erstelle Kontraktionsgraph G' für S;
- $\{X, \overline{X}\} \leftarrow$ Minimaler s-t-Schnitt in G' mit $s, t \in S$;
- $S_1 \leftarrow \{v \in S | v \in X\}$;
- $S_2 \leftarrow \{v \in S | v \in \overline{X}\}$;
- Nimm S_1, S_2 in T auf;
- Verbinde S_1, S_2 mit T-Kante des Werts $\lambda_{s,t}$;
- **foreach** *Teilbaum* T_i *ausgehend von* S **do**
  - **if** *Kontraktionsknoten zu* T_i *in* X **then**
    - Hänge T_i an S_1;
  - **else** Hänge T_i an S_2;
- Lösche S aus T;

`return` (T);

Der Algorithmus von Gomory & Hu

$T \leftarrow (\{V\}, \emptyset)$;

while $\exists$ *Superknoten* S *in* T *mit* $|S| > 1$ **do**
  Erstelle Kontraktionsgraph G' für S;
  $\{X, \overline{X}\} \leftarrow$ Minimaler s-t-Schnitt in G' mit $s, t \in S$;
  $S_1 \leftarrow \{v \in S | v \in X\}$;
  $S_2 \leftarrow \{v \in S | v \in \overline{X}\}$;
  Nimm S_1, S_2 in T auf;
  Verbinde S_1, S_2 mit T-Kante des Werts $\lambda_{s,t}$;
  foreach *Teilbaum* T_i *ausgehend von* S **do**
    if *Kontraktionsknoten zu* T_i *in* X **then**
      Hänge T_i an S_1;
    else Hänge T_i an S_2;
  Lösche S aus T;
`return` (T);

Der Algorithmus von Gomory & Hu

$T \leftarrow (\{V\}, \emptyset)$;

while $\exists$ *Superknoten* S *in* T *mit* $|S| > 1$ **do**
- Erstelle Kontraktionsgraph G' für S;
- $\{X, \overline{X}\} \leftarrow$ Minimaler s-t-Schnitt in G' mit $s, t \in S$;
- $S_1 \leftarrow \{v \in S | v \in X\}$;
- $S_2 \leftarrow \{v \in S | v \in \overline{X}\}$;
- Nimm S_1, S_2 in T auf;
- Verbinde S_1, S_2 mit T-Kante des Werts $\lambda_{s,t}$;
- **foreach** *Teilbaum* T_i *ausgehend von* S **do**
  - **if** *Kontraktionsknoten zu* T_i *in* X **then**
    - Hänge T_i an S_1;
  - **else** Hänge T_i an S_2;
- Lösche S aus T;

`return` (T);

Der Algorithmus von Gomory & Hu

$T \leftarrow (\{V\}, \emptyset)$;

while $\exists$ *Superknoten* S *in* T *mit* $|S| > 1$ **do**
- Erstelle Kontraktionsgraph G' für S;
- $\{X, \overline{X}\} \leftarrow$ Minimaler s-t-Schnitt in G' mit $s, t \in S$;
- $S_1 \leftarrow \{v \in S | v \in X\}$;
- $S_2 \leftarrow \{v \in S | v \in \overline{X}\}$;
- Nimm S_1, S_2 in T auf;
- Verbinde S_1, S_2 mit T-Kante des Werts $\lambda_{s,t}$;
- **foreach** *Teilbaum* T_i *ausgehend von* S **do**
  - **if** *Kontraktionsknoten zu* T_i *in* X **then**
    - Hänge T_i an S_1;
  - **else** Hänge T_i an S_2;
- Lösche S aus T;

`return` (T);

Der Algorithmus von Gomory & Hu

$T \leftarrow (\{V\}, \emptyset)$;

while $\exists$ *Superknoten* S *in* T *mit* $|S| > 1$ **do**
- Erstelle Kontraktionsgraph G' für S;
- $\{X, \overline{X}\} \leftarrow$ Minimaler s-t-Schnitt in G' mit $s, t \in S$;
- $S_1 \leftarrow \{v \in S | v \in X\}$;
- $S_2 \leftarrow \{v \in S | v \in \overline{X}\}$;
- Nimm S_1, S_2 in T auf;
- Verbinde S_1, S_2 mit T-Kante des Werts $\lambda_{s,t}$;
- **foreach** *Teilbaum* T_i *ausgehend von* S **do**
  - **if** *Kontraktionsknoten zu* T_i *in* X **then**
    - Hänge T_i an S_1;
  - **else** Hänge T_i an S_2;
- Lösche S aus T;

`return` (T);

Der Algorithmus von Gomory & Hu

$T \leftarrow (\{V\}, \emptyset)$;

while $\exists$ *Superknoten* S *in* T *mit* $|S| > 1$ **do**
- Erstelle Kontraktionsgraph G' für S;
- $\{X, \overline{X}\} \leftarrow$ Minimaler s-t-Schnitt in G' mit $s, t \in S$;
- $S_1 \leftarrow \{v \in S | v \in X\}$;
- $S_2 \leftarrow \{v \in S | v \in \overline{X}\}$;
- Nimm S_1, S_2 in T auf;
- Verbinde S_1, S_2 mit T-Kante des Werts $\lambda_{s,t}$;
- **foreach** *Teilbaum* T_i *ausgehend von* S **do**
  - **if** *Kontraktionsknoten zu* T_i *in* X **then**
    - Hänge T_i an S_1;
  - **else** Hänge T_i an S_2;
- Lösche S aus T;

return (T);

Der Algorithmus von Gomory & Hu

$T \leftarrow (\{V\}, \emptyset)$;

while $\exists$ *Superknoten* S *in* T *mit* $|S| > 1$ **do**
- Erstelle Kontraktionsgraph G' für S;
- $\{X, \overline{X}\} \leftarrow$ Minimaler s-t-Schnitt in G' mit $s, t \in S$;
- $S_1 \leftarrow \{v \in S | v \in X\}$;
- $S_2 \leftarrow \{v \in S | v \in \overline{X}\}$;
- Nimm S_1, S_2 in T auf;
- Verbinde S_1, S_2 mit T-Kante des Werts $\lambda_{s,t}$;
- **foreach** *Teilbaum* T_i *ausgehend von* S **do**
  - **if** *Kontraktionsknoten zu* T_i *in* X **then**
    - Hänge T_i an S_1;
  - **else** Hänge T_i an S_2;
- Lösche S aus T;

return (T);

Der Algorithmus von Gomory & Hu

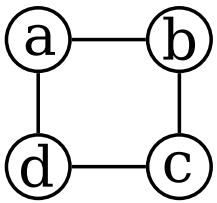
$T \leftarrow (\{V\}, \emptyset)$;

while $\exists$ *Superknoten* S *in* T *mit* $|S| > 1$ **do**
- Erstelle Kontraktionsgraph G' für S;
- $\{X, \overline{X}\} \leftarrow$ Minimaler s-t-Schnitt in G' mit $s, t \in S$;
- $S_1 \leftarrow \{v \in S | v \in X\}$;
- $S_2 \leftarrow \{v \in S | v \in \overline{X}\}$;
- Nimm S_1, S_2 in T auf;
- Verbinde S_1, S_2 mit T-Kante des Werts $\lambda_{s,t}$;
- **foreach** *Teilbaum* T_i *ausgehend von* S **do**
  - **if** *Kontraktionsknoten zu* T_i *in* X **then**
    - Hänge T_i an S_1;
  - **else** Hänge T_i an S_2;
- Lösche S aus T;

`return` (T);

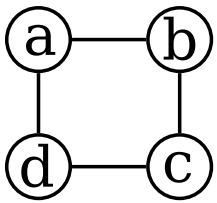
Beispiel

G



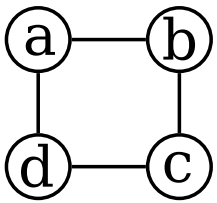
T

G'



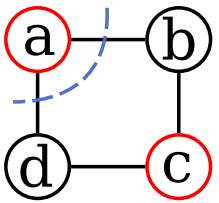
Beispiel

G



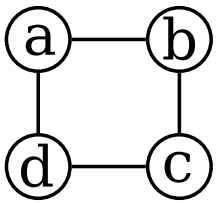
T

G'

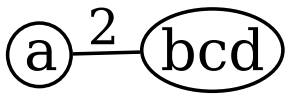


Beispiel

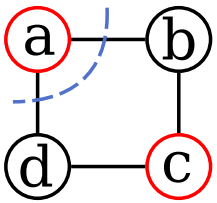
G

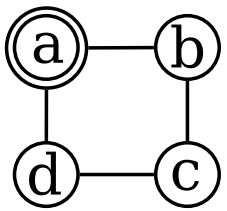


T



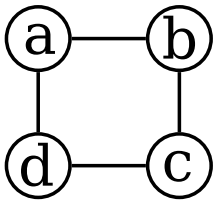
G'



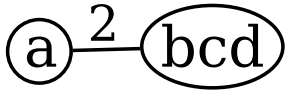


Beispiel

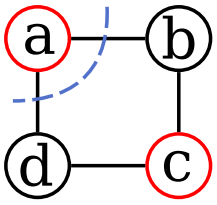
G

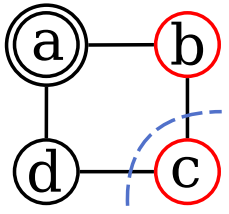


T

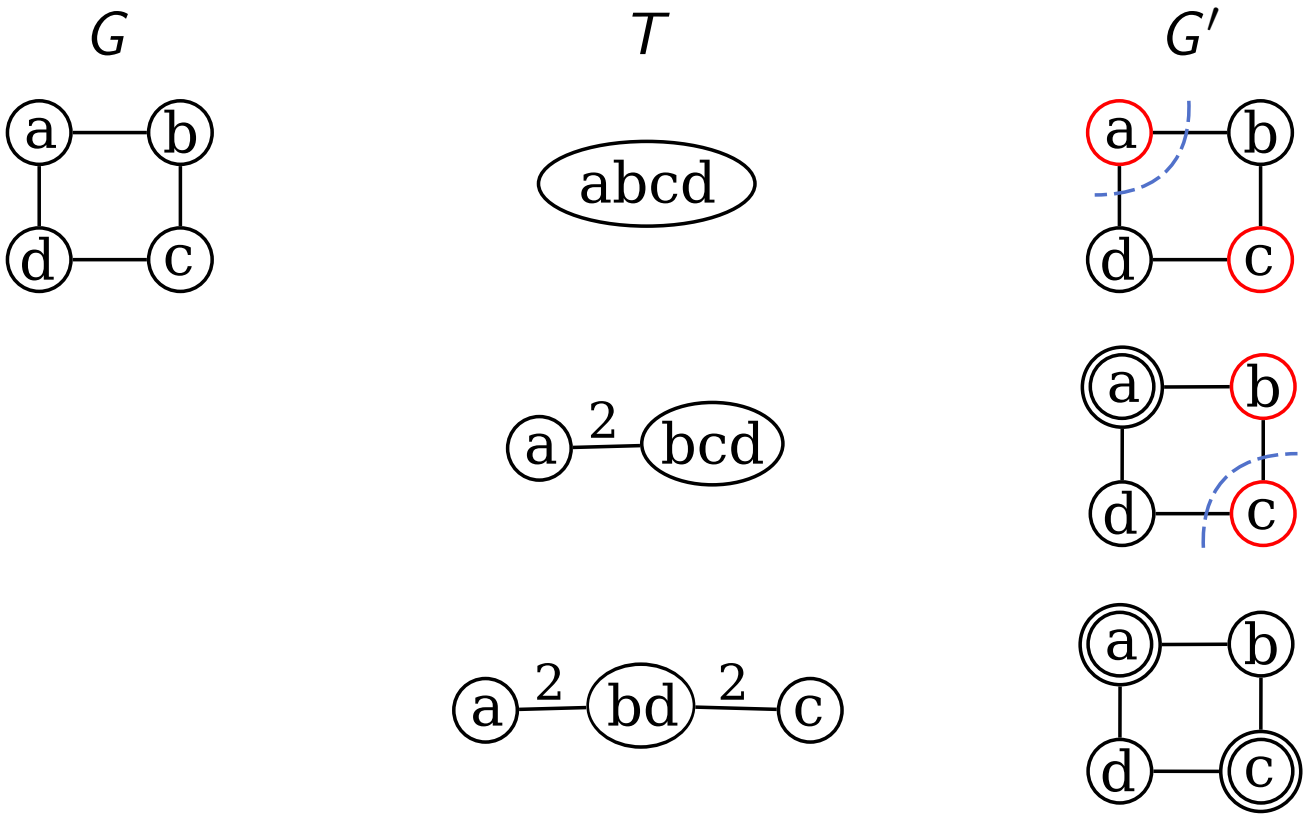


G'

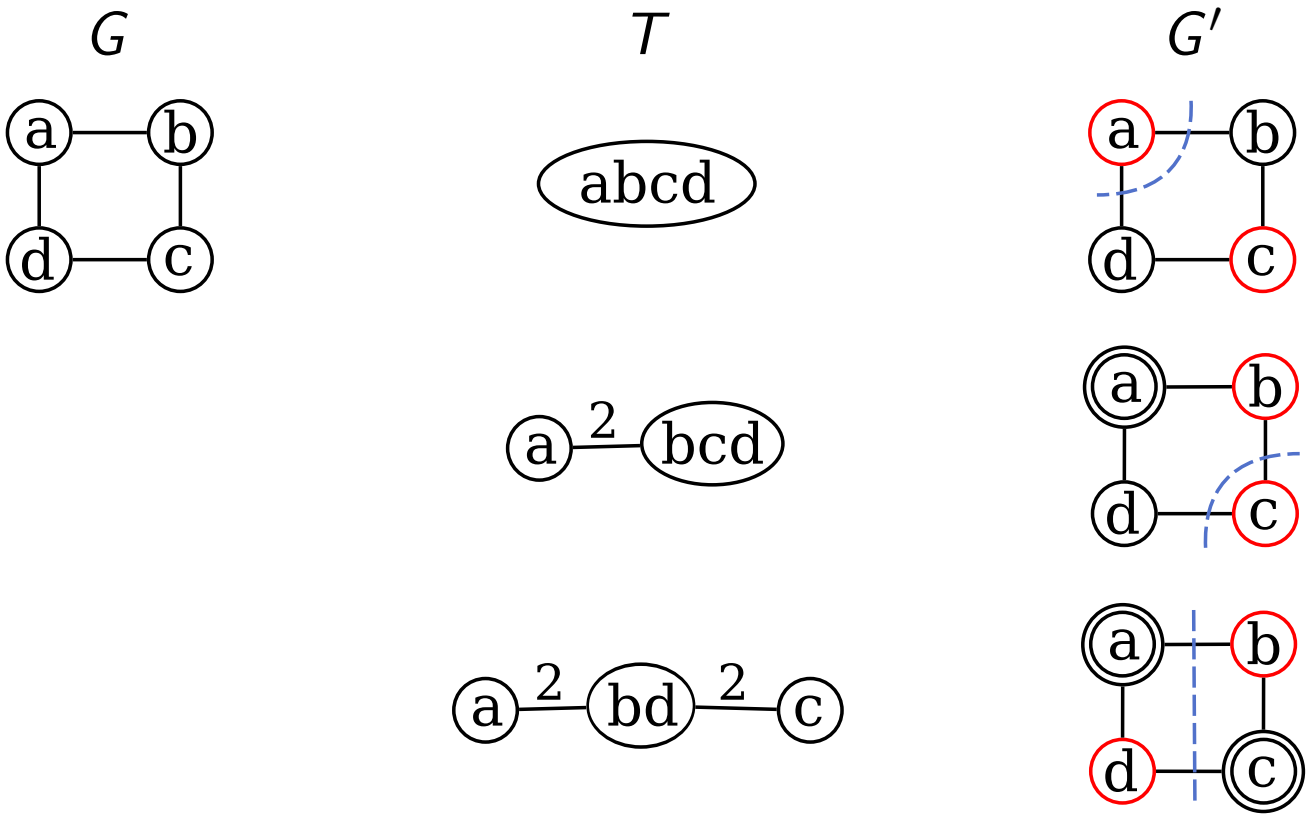




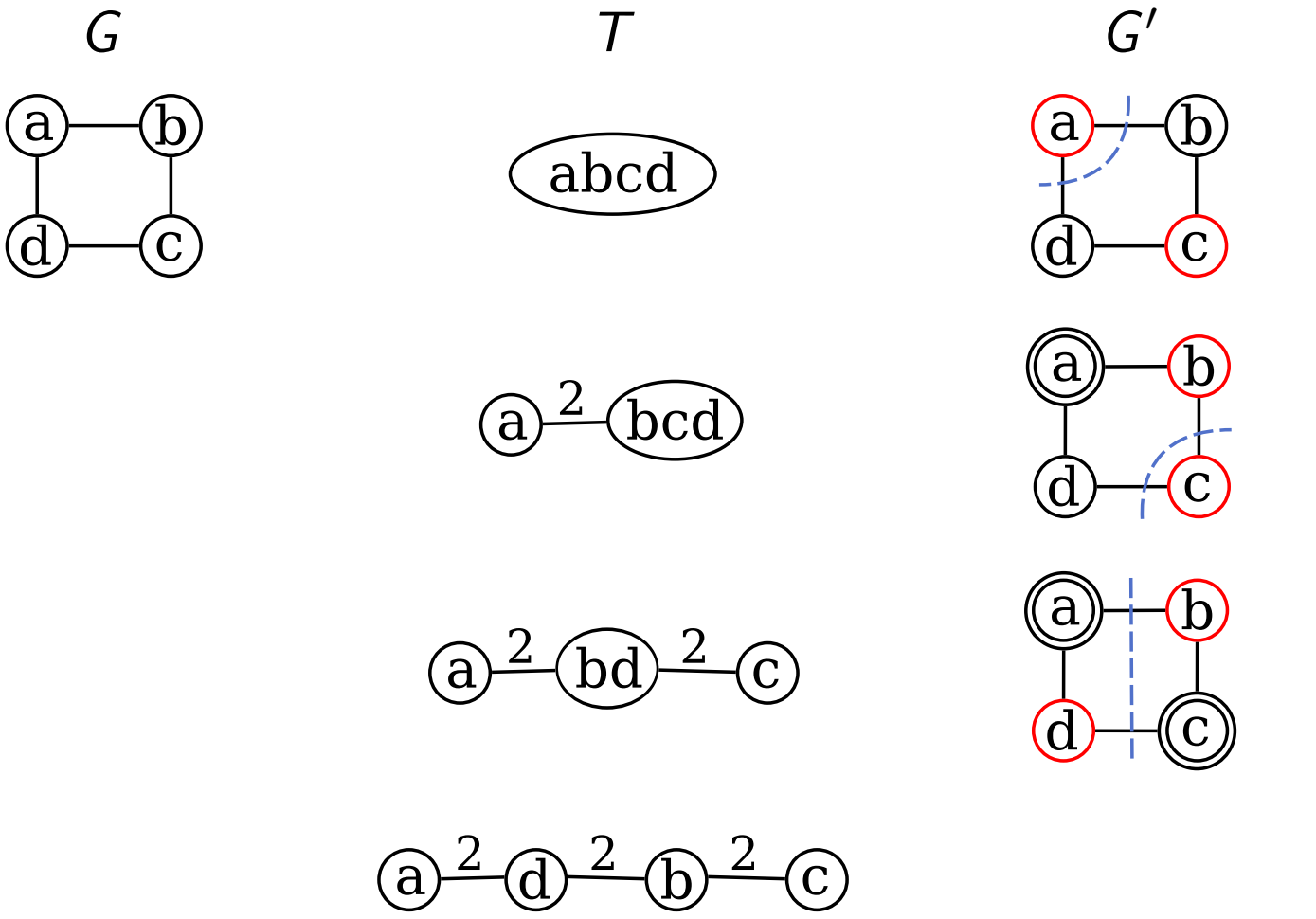
Beispiel



Beispiel



Beispiel



Bestimmende Laufzeitfaktoren

- Gezielte Minimale s-t-Schnitte werden über s-t-Maximalfluss berechnet
- Beste bekannte Laufzeitschranke: $\mathcal{O}(nm\log(n))$ (Goldberg-Rao) für $n = |V|$ und $m = |E|$
- Grundsätzlich $(n-1)$ Berechnungen notwendig
- Resultierende Laufzeitkomplexität: $\mathcal{O}(n(nm\log(n)))$

Bestimmende Laufzeitfaktoren

- Gezielte Minimale s-t-Schnitte werden über s-t-Maximalfluss berechnet
- Beste bekannte Laufzeitschranke: $\mathcal{O}(nm\log(n))$ (Goldberg-Rao) für $n = |V|$ und $m = |E|$
- Grundsätzlich $(n-1)$ Berechnungen notwendig
- Resultierende Laufzeitkomplexität: $\mathcal{O}(n(nm\log(n)))$

Inhalt

Neue Ansätze

Ziele

- Aufwändige Maximalflussberechnungen möglichst ersetzen.
- Zahl notwendiger Maximalflussberechnungen senken.
- Verbleibende Maximalflussberechnungen vereinfachen.

Neue Ansätze

Ziele

- Aufwändige Maximalflussberechnungen möglichst ersetzen.
- Zahl notwendiger Maximalflussberechnungen senken.
- Verbleibende Maximalflussberechnungen vereinfachen.

Neue Ansätze

Ziele

- Aufwändige Maximalflussberechnungen möglichst ersetzen.
- Zahl notwendiger Maximalflussberechnungen senken.
- Verbleibende Maximalflussberechnungen vereinfachen.

Neue Ansätze

Ziele

- Aufwändige Maximalflussberechnungen möglichst ersetzen.
  - Maximale Adjazenzordnungen
- Zahl notwendiger Maximalflussberechnungen senken.
  - Flusskomponentenzerlegung
- Verbleibende Maximalflussberechnungen vereinfachen.
  - Flusskomponentenzerlegung

Einsatz Maximaler Adjazenzordnungen

Idee

Bildung Maximaler Adjazenzordnung ($\mathcal{O}(n \log(n) + m)$) kann bestimmte Maximalflussberechnungen ($\mathcal{O}(nm \log(n))$) ersetzen.

Probleme

- Wenig Einfluss auf Endpunkte der berechneten maximalen Flüsse
- Häufige Wiederholung von Ergebnissen
- Leistung stark graphenabhängig
- Geschwindigkeitsvorteil in Implementierung gering

Einsatz Maximaler Adjazenzordnungen

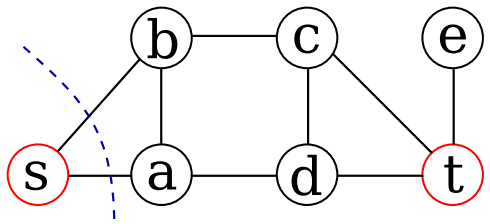
Idee

Bildung Maximaler Adjazenzordnung ($\mathcal{O}(n \log(n) + m)$) kann bestimmte Maximalflussberechnungen ($\mathcal{O}(nm \log(n))$) ersetzen.

Probleme

- Wenig Einfluss auf Endpunkte der berechneten maximalen Flüsse
- Häufige Wiederholung von Ergebnissen
- Leistung stark graphenabhängig
- Geschwindigkeitsvorteil in Implementierung gering

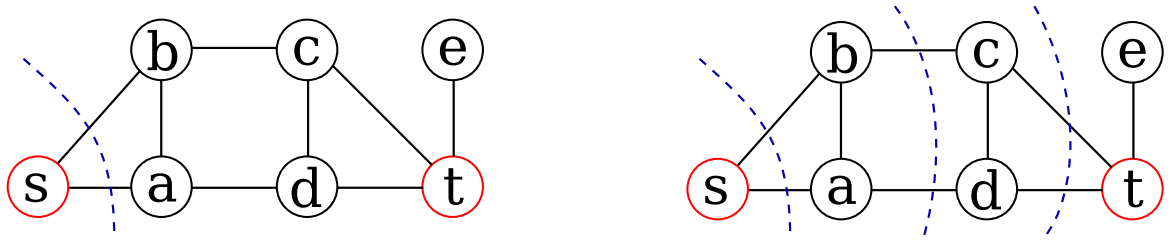
Flusskomponentenzerlegung



Beobachtungen

- Gomory-Hu-Algorithmus verwendet Maximalflussberechnungen um minimale s-t-Schnitte zu bestimmen
- Dabei erfolgt je Flussberechnung eine Bipartition der Knotenmengen V
- Aus einem maximalen s-t-Fluss lassen sich jedoch sämtliche minimalen s-t-Schnitte aufzählen

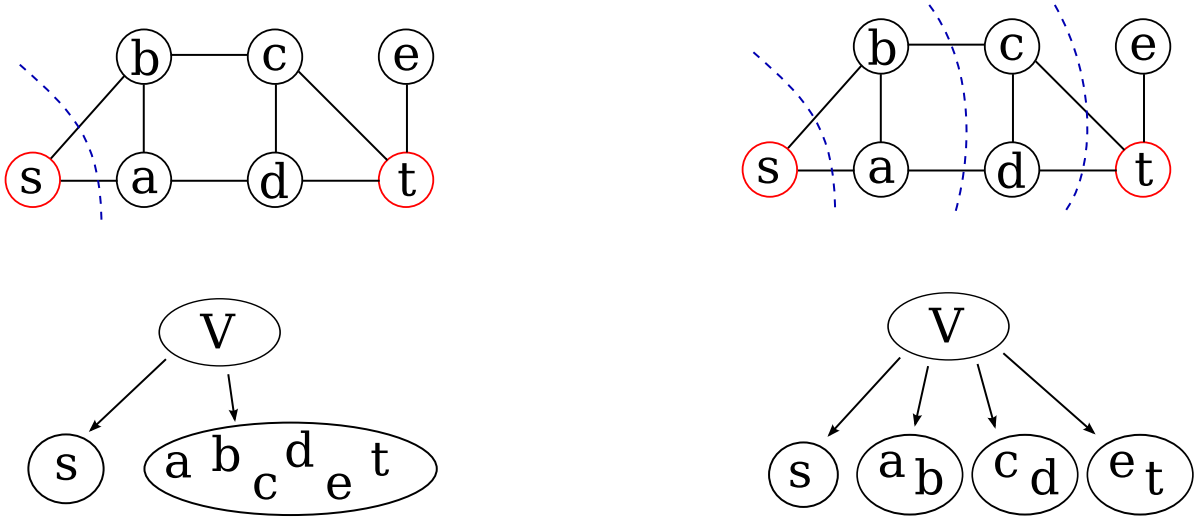
Flusskomponentenzerlegung



Beobachtungen

- Gomory-Hu-Algorithmus verwendet Maximalflussberechnungen um minimale s-t-Schnitte zu bestimmen
- Dabei erfolgt je Flussberechnung eine Bipartition der Knotenmengen V
- Aus einem maximalen s-t-Fluss lassen sich jedoch sämtliche minimalen s-t-Schnitte aufzählen

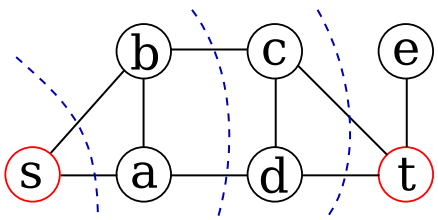
Flusskomponentenzerlegung (2)

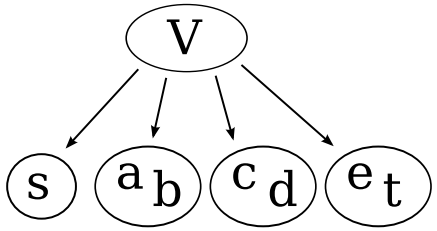


Idee

- Für jeden maximalen s-t-Fluss wird eine maximale Auswahl kreuzungsfreier minimaler s-t-Schnitte getroffen.
- Deren Differenzmengen, die *Flusskomponenten*, lassen sich ähnlich den Schnittmengen des Gomory-Hu-Algorithmus verwenden.

Flusskomponenten



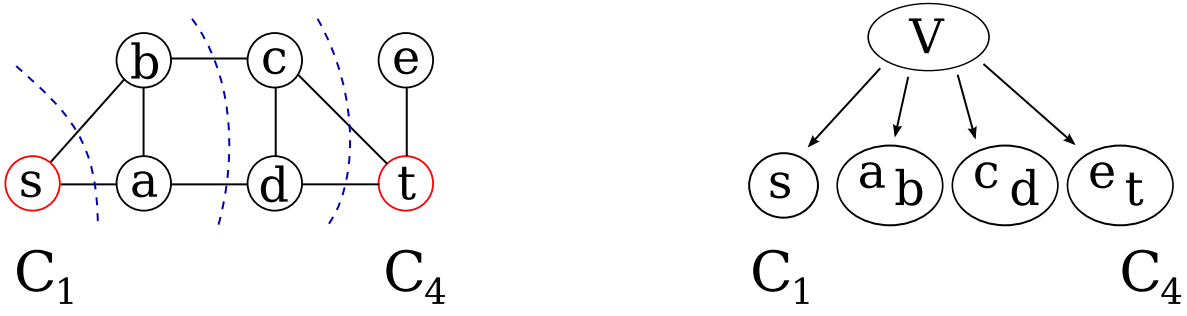


Definition (Flusskomponente C_i)

Knotenmenge $C_i \subset V$, die für einen maximalen s-t-Fluss in G

- einer s- oder t-Schnittmenge entspricht, oder
- eine nichtleere Differenzmenge zwischen den s-Schnittmengen zweier minimaler s-t-Schnitte in G ist.

Flusskomponenten

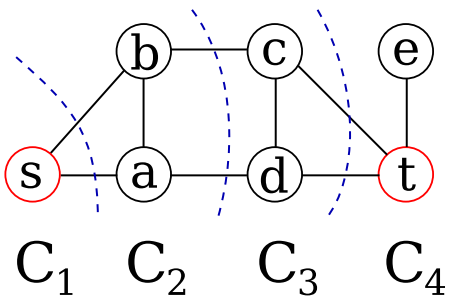


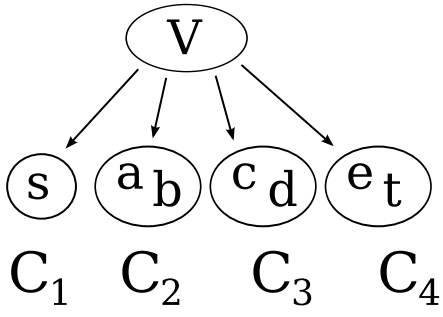
Definition (Flusskomponente C_i)

Knotenmenge $C_i \subset V$, die für einen maximalen s-t-Fluss in G

- einer s- oder t-Schnittmenge entspricht, oder
- eine nichtleere Differenzmenge zwischen den s-Schnittmengen zweier minimaler s-t-Schnitte in G ist.

Flusskomponenten



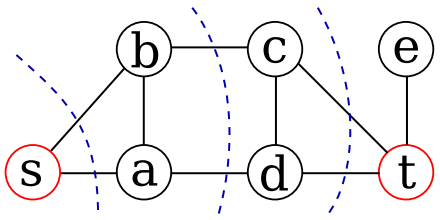


Definition (Flusskomponente C_i)

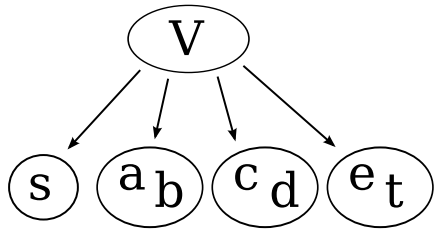
Knotenmenge $C_i \subset V$, die für einen maximalen s-t-Fluss in G

- einer s- oder t-Schnittmenge entspricht, oder
- eine nichtleere Differenzmenge zwischen den s-Schnittmengen zweier minimaler s-t-Schnitte in G ist.

Flusskomponentenfolge



(C_1, C_2, C_3, C_4)



(C_1, C_2, C_3, C_4)

Definition (Flusskomponentenfolge $(C_1, \cdots, C_k)$)

Folge von Flusskomponenten eines maximalen s-t-Flusses, mit

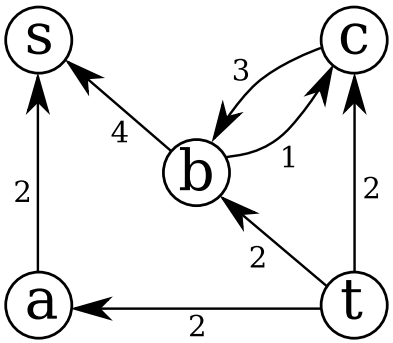
- C_1 ist minimaler s-t-Schnitt mit $s \in C_1$
- C_k ist minimaler s-t-Schnitt mit $t \in C_k$
- $X_i = \sum_{j=1..i} C_j$ mit $i < k$ ist minimaler s-t-Schnitt

Berechnung der Flusskomponenten

Berechnung einer Flusskomponentenfolge $(C_1, \cdots, C_k)$

Input: $G = (V, E)$ mit s-t-Maximalfluss, Residualgraph G_f

$\{C_1, \cdots, C_k\} \leftarrow$ Starke Zusammenhangskomponenten von G_f;
// G_f-Kanten definieren Halbordnung über $\{C_1, \cdots, C_k\}$
Vereinige $C_1 \ni s$ mit seinen transitiven Nachfolgern;
Vereinige $C_k \ni t$ mit seinen transitiven Vorgängern;
$(C_1, \cdots, C_k) \leftarrow$ Topologische Sortierung von $\{C_1, \cdots, C_k\}$;
return $(C_1, \cdots, C_k)$;

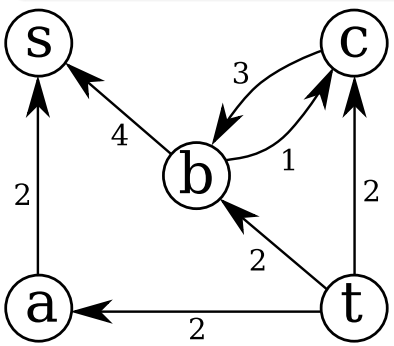


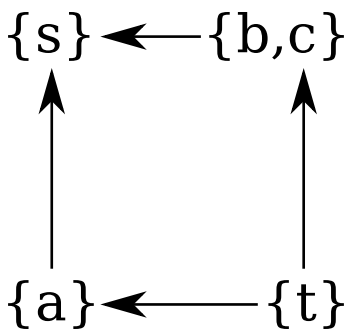
Berechnung der Flusskomponenten

Berechnung einer Flusskomponentenfolge $(C_1, \cdots, C_k)$

Input: $G = (V, E)$ mit s-t-Maximalfluss, Residualgraph G_f

$\{C_1, \cdots, C_k\} \leftarrow$ Starke Zusammenhangskomponenten von G_f;
// G_f-Kanten definieren Halbordnung über $\{C_1, \cdots, C_k\}$
Vereinige $C_1 \ni s$ mit seinen transitiven Nachfolgern;
Vereinige $C_k \ni t$ mit seinen transitiven Vorgängern;
$(C_1, \cdots, C_k) \leftarrow$ Topologische Sortierung von $\{C_1, \cdots, C_k\}$;
return $(C_1, \cdots, C_k)$;



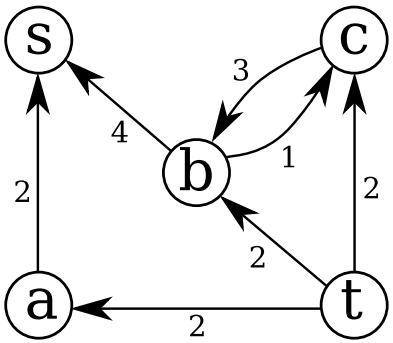


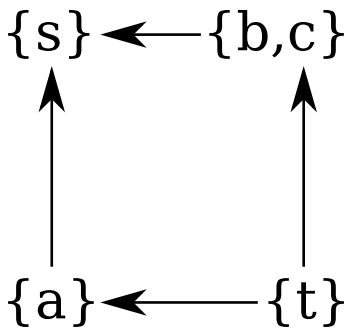
Berechnung der Flusskomponenten

Berechnung einer Flusskomponentenfolge $(C_1, \cdots, C_k)$

Input: $G = (V, E)$ mit s-t-Maximalfluss, Residualgraph G_f

$\{C_1, \cdots, C_k\} \leftarrow$ Starke Zusammenhangskomponenten von G_f;
// G_f-Kanten definieren Halbordnung über $\{C_1, \cdots, C_k\}$
Vereinige $C_1 \ni s$ mit seinen transitiven Nachfolgern;
Vereinige $C_k \ni t$ mit seinen transitiven Vorgängern;
$(C_1, \cdots, C_k) \leftarrow$ Topologische Sortierung von $\{C_1, \cdots, C_k\}$;
return $(C_1, \cdots, C_k)$;





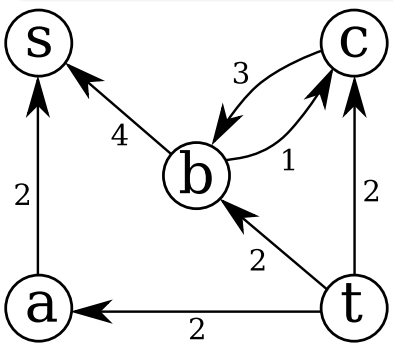
{s}←{a}←{b,c}←{t}

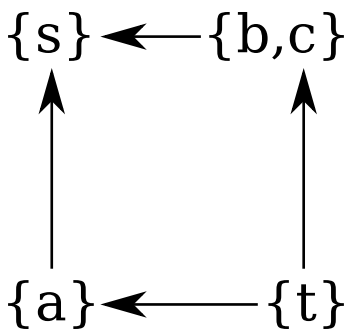
Berechnung der Flusskomponenten

Berechnung einer Flusskomponentenfolge $(C_1, \cdots, C_k)$

Input: $G = (V, E)$ mit s-t-Maximalfluss, Residualgraph G_f

$\{C_1, \cdots, C_k\} \leftarrow$ Starke Zusammenhangskomponenten von G_f;
// G_f-Kanten definieren Halbordnung über $\{C_1, \cdots, C_k\}$
Vereinige $C_1 \ni s$ mit seinen transitiven Nachfolgern;
Vereinige $C_k \ni t$ mit seinen transitiven Vorgängern;
$(C_1, \cdots, C_k) \leftarrow$ Topologische Sortierung von $\{C_1, \cdots, C_k\}$;
return $(C_1, \cdots, C_k)$;



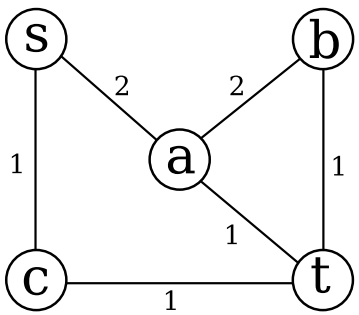


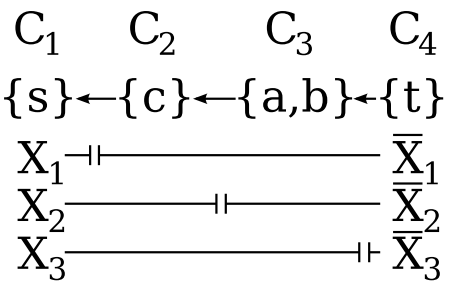
{s}←{a}←{b,c}←{t}

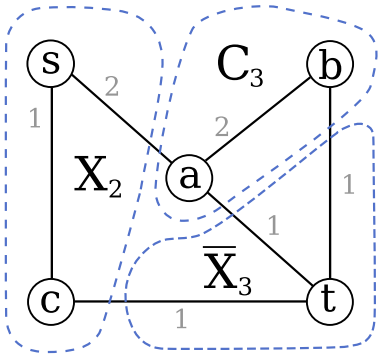
Eigenschaften

Eigenschaften von $(C_1, \cdots, C_k)$

- Jede Menge $X_i = \sum_{j=1..i} C_j$ ist ein minimaler s-t-Schnitt
- $X_1, \cdots, X_{k-1}$ sind kreuzungsfrei
- Für $a, b \in C_i$ existiert ein minimaler a-b-Schnitt, der X_{i-1} und $\overline{X}_i$ nicht kreuzt.
- Für $a \in C_i$, $b \in C_j$ mit $i < j$ existiert ein minimaler a-b-Schnitt A mit $A \subseteq C_i$ oder $A \subseteq C_j$ oder $A = X_i$.



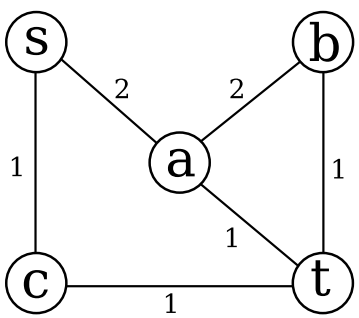


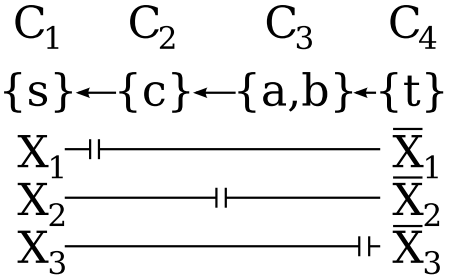


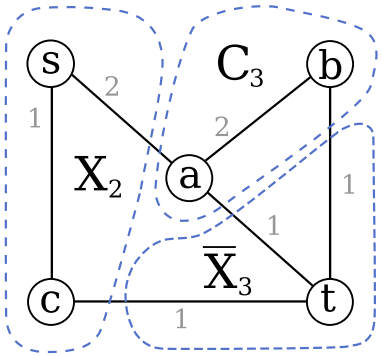
Eigenschaften

Eigenschaften von $(C_1, \cdots, C_k)$

- Jede Menge $X_i = \sum_{j=1..i} C_j$ ist ein minimaler s-t-Schnitt
- $X_1, \cdots, X_{k-1}$ sind kreuzungsfrei
- Für $a, b \in C_i$ existiert ein minimaler a-b-Schnitt, der X_{i-1} und $\overline{X}_i$ nicht kreuzt.
- Für $a \in C_i$, $b \in C_j$ mit $i < j$ existiert ein minimaler a-b-Schnitt A mit $A \subseteq C_i$ oder $A \subseteq C_j$ oder $A = X_i$.



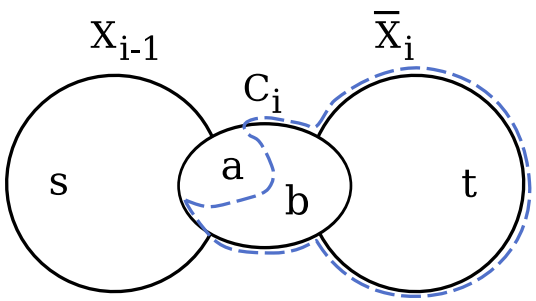


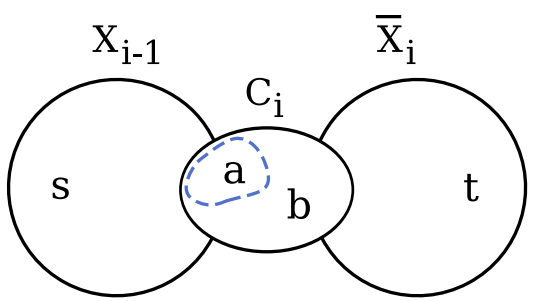


Eigenschaften

Eigenschaften von $(C_1, \cdots, C_k)$

- Jede Menge $X_i = \sum_{j=1..i} C_j$ ist ein minimaler s-t-Schnitt
- $X_1, \cdots, X_{k-1}$ sind kreuzungsfrei
- Für $a, b \in C_i$ existiert ein minimaler a-b-Schnitt, der X_{i-1} und $\overline{X}_i$ nicht kreuzt.
- Für $a \in C_i$, $b \in C_j$ mit $i < j$ existiert ein minimaler a-b-Schnitt A mit $A \subseteq C_i$ oder $A \subseteq C_j$ oder $A = X_i$.

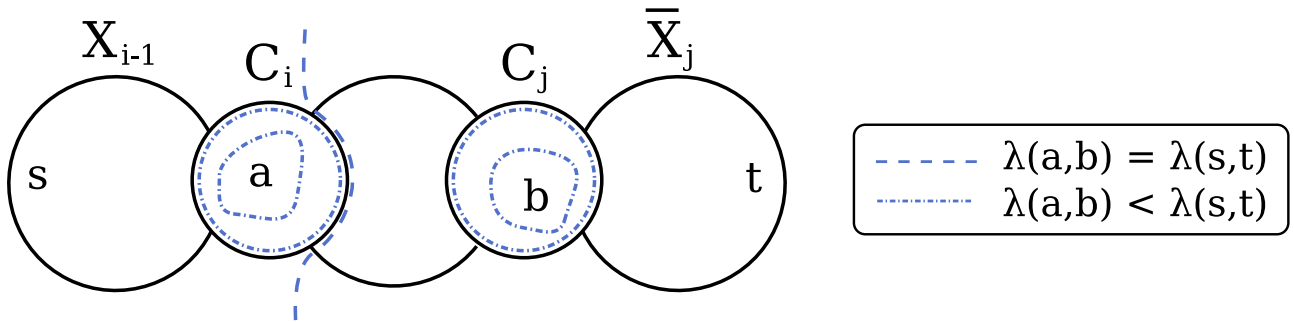




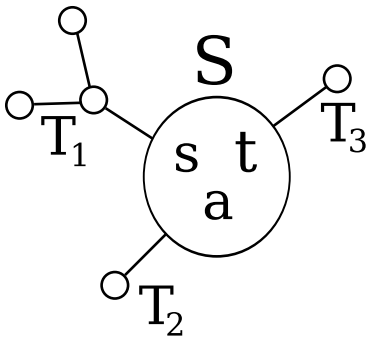
Eigenschaften

Eigenschaften von $(C_1, \cdots, C_k)$

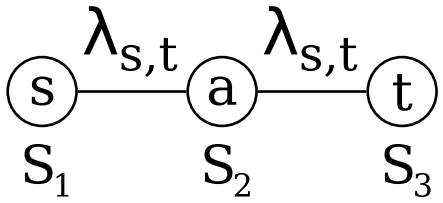
- Jede Menge $X_i = \sum_{j=1..i} C_j$ ist ein minimaler s-t-Schnitt
- $X_1, \cdots, X_{k-1}$ sind kreuzungsfrei
- Für $a, b \in C_i$ existiert ein minimaler a-b-Schnitt, der X_{i-1} und $\overline{X}_i$ nicht kreuzt.
- Für $a \in C_i$, $b \in C_j$ mit $i < j$ existiert ein minimaler a-b-Schnitt A mit $A \subseteq C_i$ oder $A \subseteq C_j$ oder $A = X_i$.



Zerlegung von Superknoten in T



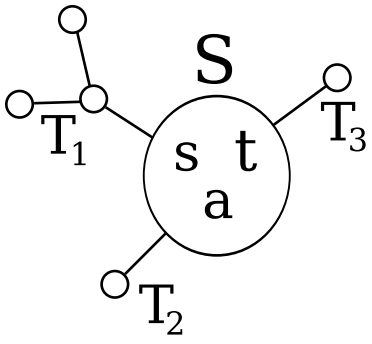
$$C_1 = \{s, [T_1]\}$$
$$C_2 = \{a\}$$
$$C_3 = \{t, [T_2], [T_3]\}$$



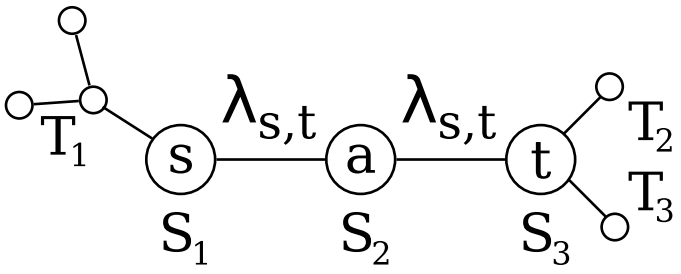
Flusskomponentenzerlegung

- Verwendung der Flusskomponenten analog zu den Schnittmengen des Gomory-Hu-Algorithmus
- Aufspaltung des Superknoten S in Superknotenkette $S_1, \cdots, S_k$ verbunden mit $\lambda_{s,t}$-wertigen Kanten
- Anhängen jedes von S abgehenden T-Teilbaums T_l an $S_i \in \{S_1, \cdots, S_k\}$ falls Kontraktionsknoten $[T_l] \in C_i$

Zerlegung von Superknoten in T



$$C_1 = \{s, [T_1]\}$$
$$C_2 = \{a\}$$
$$C_3 = \{t, [T_2], [T_3]\}$$



Flusskomponentenzerlegung

- Verwendung der Flusskomponenten analog zu den Schnittmengen des Gomory-Hu-Algorithmus
- Aufspaltung des Superknoten S in Superknotenkette $S_1, \cdots, S_k$ verbunden mit $\lambda_{s,t}$-wertigen Kanten
- Anhängen jedes von S abgehenden T-Teilbaums T_l an $S_i \in \{S_1, \cdots, S_k\}$ falls Kontraktionsknoten $[T_l] \in C_i$

Wirkung: Einfachere Maximalflussberechnungen

- Schnellerer Verkleinerung der mit den Superknoten von T assoziierten Teilmengen von V

$\Rightarrow$ Potentiell kleinere Kontraktionsgraphen als im Gomory-Hu-Algorithmus

$\Rightarrow$ Beschleunigung der durchzuführenden Maximalflussberechnungen ($\mathcal{O}(nm\log(n))$)

Notwendigkeit der Speicherung bekannter Schnitte

Problem

Flusskomponentenzerlegung kann für jeden Superknoten S in T einen Zustand $|S| = 1$ (Abbruchbedingung Gomory-Hu) erreichen, bevor alle benötigten Schnitte festgestellt wurden.

Notwendigkeit der Speicherung bekannter Schnitte

Problem

Flusskomponentenzerlegung kann für jeden Superknoten S in T einen Zustand $|S| = 1$ (Abbruchbedingung Gomory-Hu) erreichen, bevor alle benötigten Schnitte festgestellt wurden.

Notwendige Schritte

- Identifikation noch ausstehender Schnittberechnungen
  - $\Rightarrow$ Buchführung über verwendete minimale Schnitte und Schnittpartner
- Einfaches Festhalten neuer Information (ohne Möglichkeit zur Superknotenteilung)
  - $\Rightarrow$ Relaxierung des Konzepts des Gomory-Hu-Baums

Notwendigkeit der Speicherung bekannter Schnitte

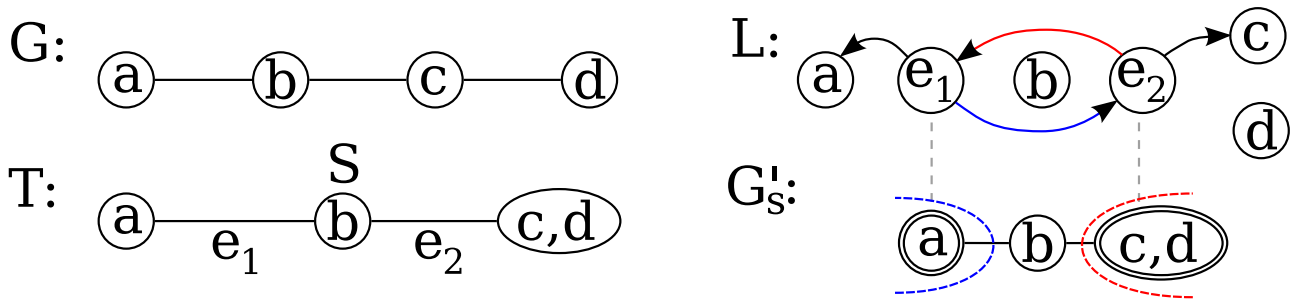
Problem

Flusskomponentenzerlegung kann für jeden Superknoten S in T einen Zustand $|S| = 1$ (Abbruchbedingung Gomory-Hu) erreichen, bevor alle benötigten Schnitte festgestellt wurden.

Notwendige Schritte

- Identifikation noch ausstehender Schnittberechnungen
  - $\Rightarrow$ Buchführung über verwendete minimale Schnitte und Schnittpartner
- Einfaches Festhalten neuer Information (ohne Möglichkeit zur Superknotenteilung)
  - $\Rightarrow$ Relaxierung des Konzepts des Gomory-Hu-Baums

Identifikation ausstehender Schnittberechnungen



Blattschnittbeziehungsgraph L

- Speicherung der für jeden aus T bildbaren Kontraktionsgraphen bereits bekannten minimalen Schnitte
- Ermöglicht:
  - Bestimmung von Knotenpaaren mit noch nicht untersuchtem minimalen Schnitt
  - Neuformulierung der Abbruchbedingung: *Spannbaum in spezifischem Teilgraphen*

Nebenwirkung Blattschnittbeziehungsgraph

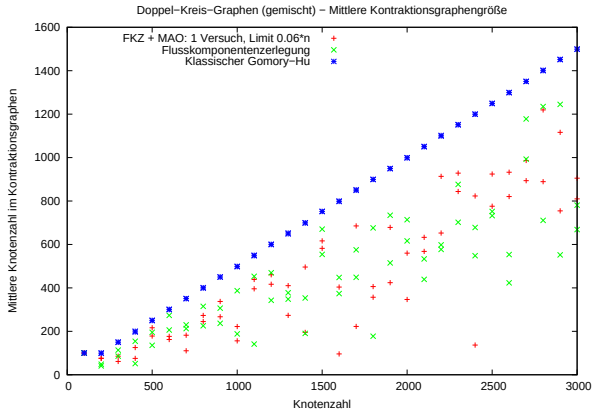
Einsparung von Schnittberechnungen

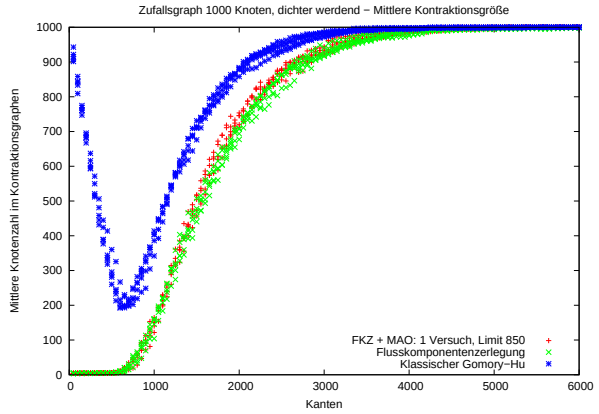
- Auswertung der Information aus L
- Günstige Kombinationen zweier Schnitte erspart dritte Schnittberechnung
- Voraussetzung: Zerlegungen mit ≥ 3 Komponenten
- Im Idealfall 50% Einsparungen

Inhalt

Mittlere Kontraktionsgraphengröße

Doppel-Kreise





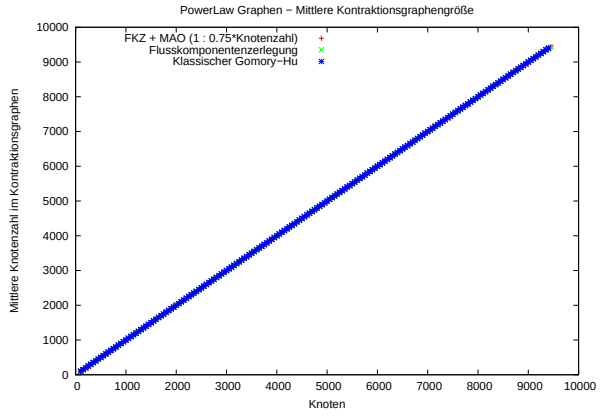
Zufallsgraph, sich verdichtend

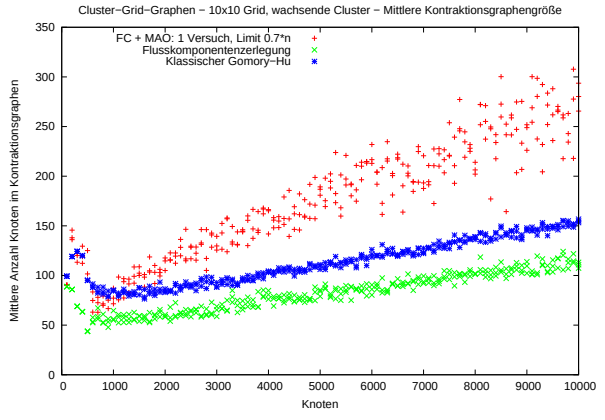
Entwicklung durchschnittlicher Kontraktionsgraphengröße

- Verhalten graphenabhängig.
- Flusskomponentenzerlegung senkt mittlere Kontraktionsgraphengröße in meisten Fällen
- Wirkunglos auf Graphklassen die zu Gomory-Hu-Bäumen mit kleinem Durchmesser (Stern) tendieren
- Einsatz Maximaler Adjazenzordnungen kann Größe steigern

Mittlere Kontraktionsgraphengröße

PowerLaw-Graphen





ClusterGrid-Graphen

Entwicklung durchschnittlicher Kontraktionsgraphengröße

- Verhalten graphenabhängig.
- Flusskomponentenzerlegung senkt mittlere Kontraktionsgraphengröße in meisten Fällen
- Wirkunglos auf Graphklassen die zu Gomory-Hu-Bäumen mit kleinem Durchmesser (Stern) tendieren
- Einsatz Maximaler Adjazenzordnungen kann Größe steigern

Weitere Ergebnisse

Schnitteinsparungen durch Flusskomponentenzerlegung

- In Experimenten gering $(0 - 5\%)$
- Setzen Zerlegungen mit ≥ 3 Komponenten voraus
- Durch Wechselwirkung mit Maximalen Adjazenzordnungen gesteigert

Beobachtungen

- Anzahl an Komponenten meist gering (selten über 3)
- Positiv wirken: (nahezu) uniforme Gewichtung, Tendenz zu Gomory-Hu-Bäumen mit großem Durchmesser (Kreise, Doppel-Kreise, Tree-Seeded-Graphen)

Weitere Ergebnisse

Schnitteinsparungen durch Flusskomponentenzerlegung

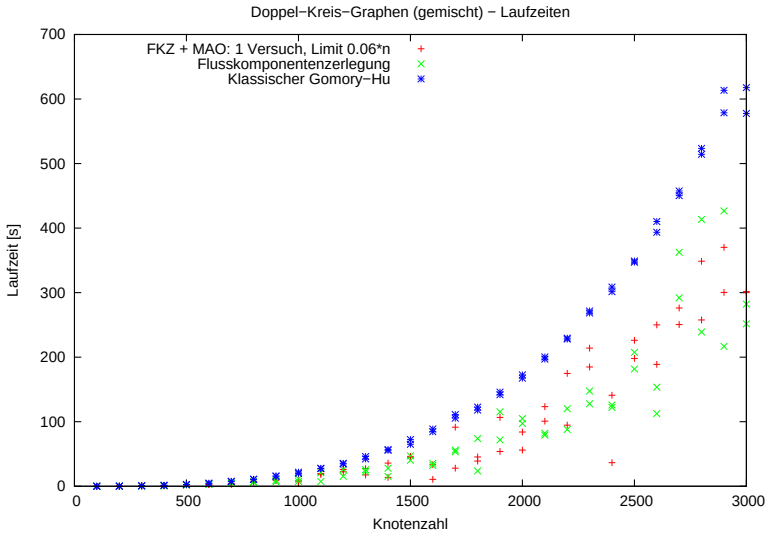
- In Experimenten gering $(0 - 5\%)$
- Setzen Zerlegungen mit ≥ 3 Komponenten voraus
- Durch Wechselwirkung mit Maximalen Adjazenzordnungen gesteigert

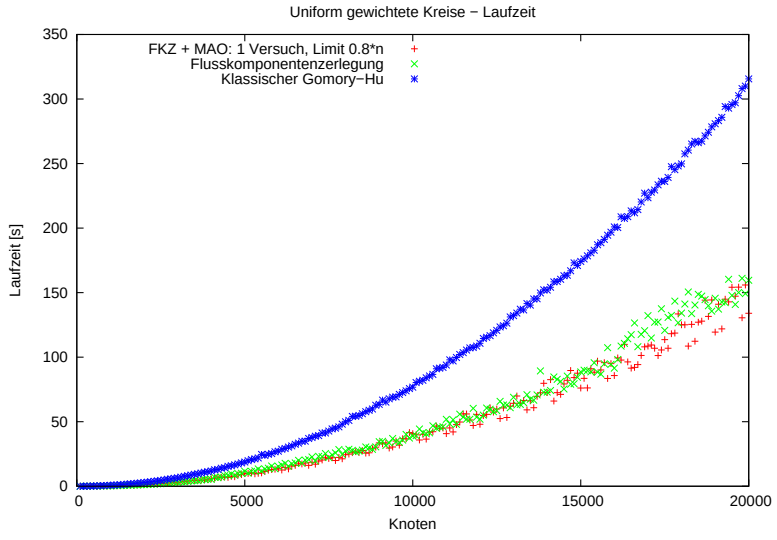
Beobachtungen

- Anzahl an Komponenten meist gering (selten über 3)
- Positiv wirken: (nahezu) uniforme Gewichtung, Tendenz zu Gomory-Hu-Bäumen mit großem Durchmesser (Kreise, Doppel-Kreise, Tree-Seeded-Graphen)

Laufzeitentwicklung

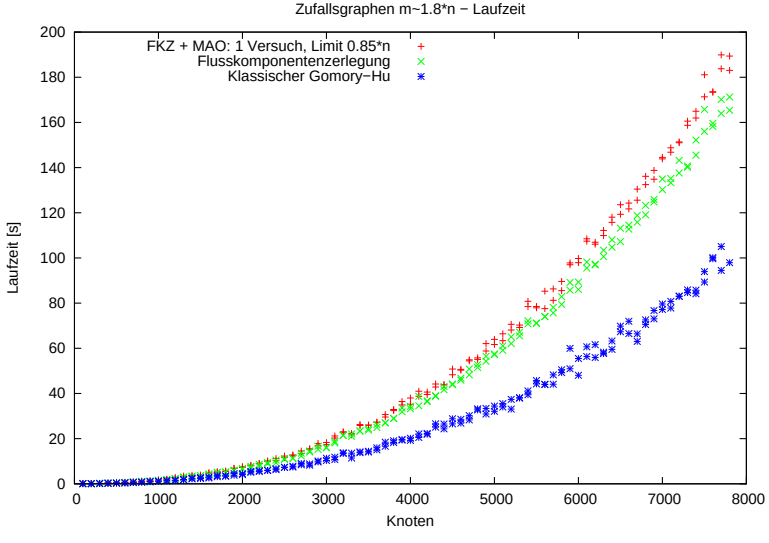
Doppel-Kreise

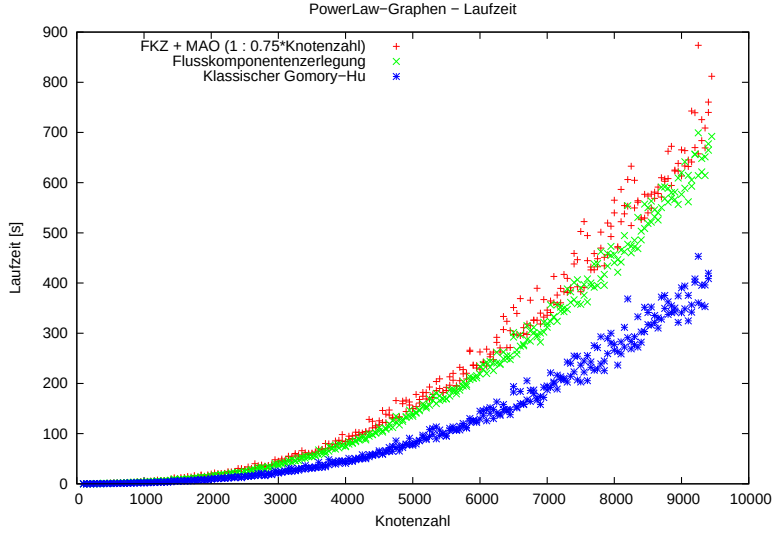




Uniforme Kreise

Zufallsgraphen





PowerLaw-Graphen

Begründung Laufzeitentwicklung

Feststellung

- Laufzeitvorteile ergeben sich nur auf wenigen Graphklassen
- Erfolgreiche Beeinflussung der Zielparameter schlägt sich nicht nieder

Gründe

- Empirische Laufzeitentwicklung des Maximalflussalgorithmus oft pseudo-linear

$\Rightarrow$ Zusatzaufwand für Flusskomponentenzerlegung ist proportional zu Maximalflussaufwand

- Feststellung: viele Graphklassen streben zu Gomory-Hu-Bäumen mit kleinem Durchmesser (keine Verkleinerung der Kontraktionsgraphen)

Begründung Laufzeitentwicklung

Feststellung

- Laufzeitvorteile ergeben sich nur auf wenigen Graphklassen
- Erfolgreiche Beeinflussung der Zielparameter schlägt sich nicht nieder

Gründe

- Empirische Laufzeitentwicklung des Maximalflussalgorithmus oft pseudo-linear

$\Rightarrow$ Zusatzaufwand für Flusskomponentenzerlegung ist proportional zu Maximalflussaufwand

- Feststellung: viele Graphklassen streben zu Gomory-Hu-Bäumen mit kleinem Durchmesser (keine Verkleinerung der Kontraktionsgraphen)

Zusammenfassung

Einsatz Maximaler Adjazenzordnungen

- Anteil nutzbarer Ergebnisse sehr klein
- Geringer realer Geschwindigkeitsvorteil

Flusskomponentenzerlegung

- Kann durchschnittliche Kontraktionsgraphen verkleinern
- Kann Teil der Schnittberechnungen einsparen
- worst-case Laufzeitkomplexität wie Gomory-Hu-Algorithmus: $\mathcal{O}(n(nm\log(n)))$
- In Experimenten Zusatzaufwand oft größer als Einsparung

Vielen Dank für ihre Aufmerksamkeit.

Maximale Adjazenzordnungen

Adjazenz

Die *Adjazenz eines Knotens v zu einer Knotenmenge $X \subseteq V$* ist definiert als:

$$d(X, v) = \sum_{x \in X, \{v,x\} \in E} w(\{v, x\})$$

Definition (Maximale Adjazenzordnung)

Eine *Maximale Adjazenzordnung* für $G = (V, E)$ mit $V = \{v_1, \cdots, v_n\}$ ist eine **Knotenordnung** $V_n = (v_1, \cdots, v_n)$ **mit gewähltem Startknoten** $v_1 \in V$, in der für jeden Knoten v_i mit $i > 1$ und die Teilordnungen $V_j = (v_1, \cdots, v_j)$ mit $j < n$ gilt:

$$v_i = \underset{x \in V \setminus V_{i-1}}{\arg\max}\ d(V_{i-1}, x)$$

Bildung Maximaler Adjazenzordnungen

Algorithmus

$X := \{v_1\}$; /* Alternativ: $X := \emptyset$ */
while $X \neq V$ **do**
  $v := \arg\max_{x \in V \setminus X} d(X, x)$;
  $X := X \cup \{v\}$;
  order $[|X|] := v$;
return(order);

Implementierung

- Auswahl maximal adjazenter Knoten über Heap-Struktur
- Komplexität mit Fibonacci-Heap: $\mathcal{O}(n \log(n) + m)$

Ersatz von Maximalflussberechnungen

Lemma (Stoer und Wagner, 1997)

Ist $V_n = (v_1, \cdots, v_{n-1}, v_n)$ eine Maximale Adjazenzordnung für den Graphen G, so existiert in G ein minimaler v_{n-1}-v_n-Schnitt mit den Schnittmengen $\{v_n\}$ und $V \setminus \{v_n\}$.

Problematik

- Einflussmöglichkeit auf Knotenpaar v_{n-1},v_n alleinig durch gewählten Startknoten
- Auch bei Verwendung sämtlicher Knoten des Graphen G als Startknoten, sind nicht mehr als 2 gefundene minimale Schnitte garantiert.
- Mögliche Neuberechnung bereits gefundener minimaler Schnitte

Ersatz von Maximalflussberechnungen (2)

Verwendung im Algorithmus

- Einsatz als Orakel für minimalen s-t-Schnitt
- Bei nicht-verwertbarem Ergebnis:
  - Wiederholung mit anderem Startknoten, oder
  - Klassisches Verfahren mit Maximalflussalgorithmus
- Anwendung besonders zu Beginn des Algorithmus
  - Wenige bereits bekannte Schnitte, damit wenig Kollisionsmöglichkeiten.
  - Kontraktionsgraphen haben noch kaum an Größe verloren, damit potentiell größerer Einfluss der besseren worst-case Laufzeit

Ergebnisse Maximaler Adjazenzordnungen

Einsatzerfolg Maximaler Adjazenzordnungen

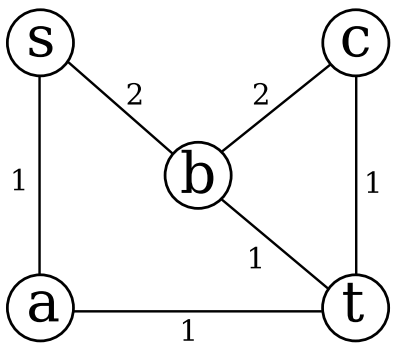
- Ergebnisse äußerst graphenabhängig
- Anzahl überhaupt auffindbarer Minimalschnitte abhängig von Anteil der Knoten mit Minimalgrad
- Schwierige Dosierung
- Schnelle Wiederholung von Ergebnissen
- Meist geringer Laufzeitunterschied zu Flussberechnung für selbes Knotenpaar (oft isolierte Knoten)
- Wechselwirkungen mit Flusskomponentenzerlegung

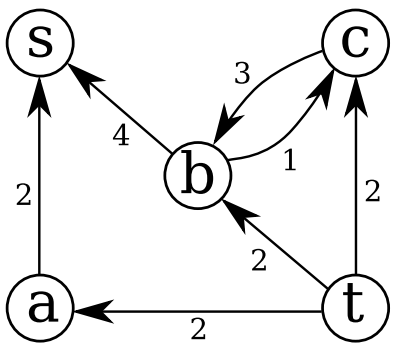
Lemma (Picard-Queyranne, 1980)

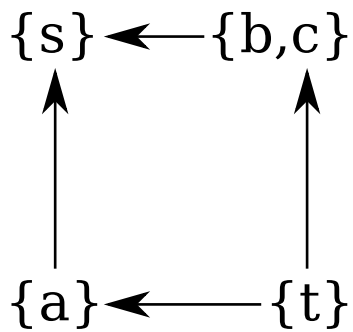
Ist $G_f = (V, A)$ Residualgraph zu maximalem s-t-Fluss in G und sei R Relation mit

$$iRj \Leftrightarrow (i,j) \in A$$

Dann ist ein Schnitt $(X, \overline{X})$ genau dann minimal, wenn X eine transitive Hülle von R ist und $s \in X$, $t \notin X$ gilt.

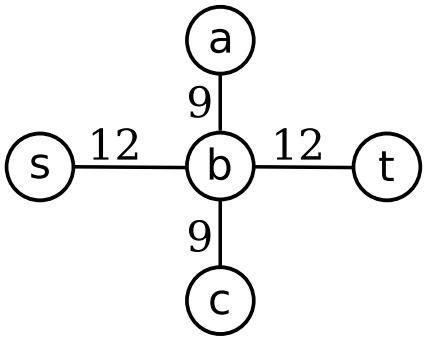




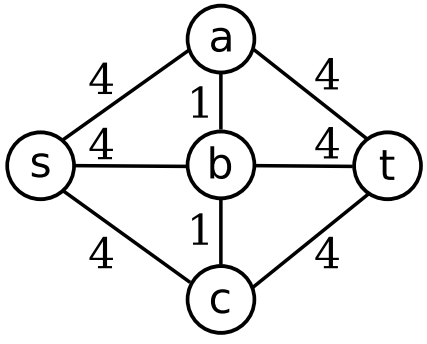


Relaxierung Gomory-Hu-Baum

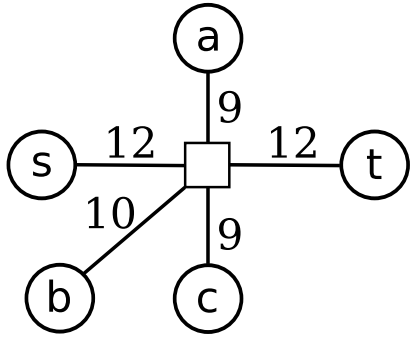
T vorher



Graph G

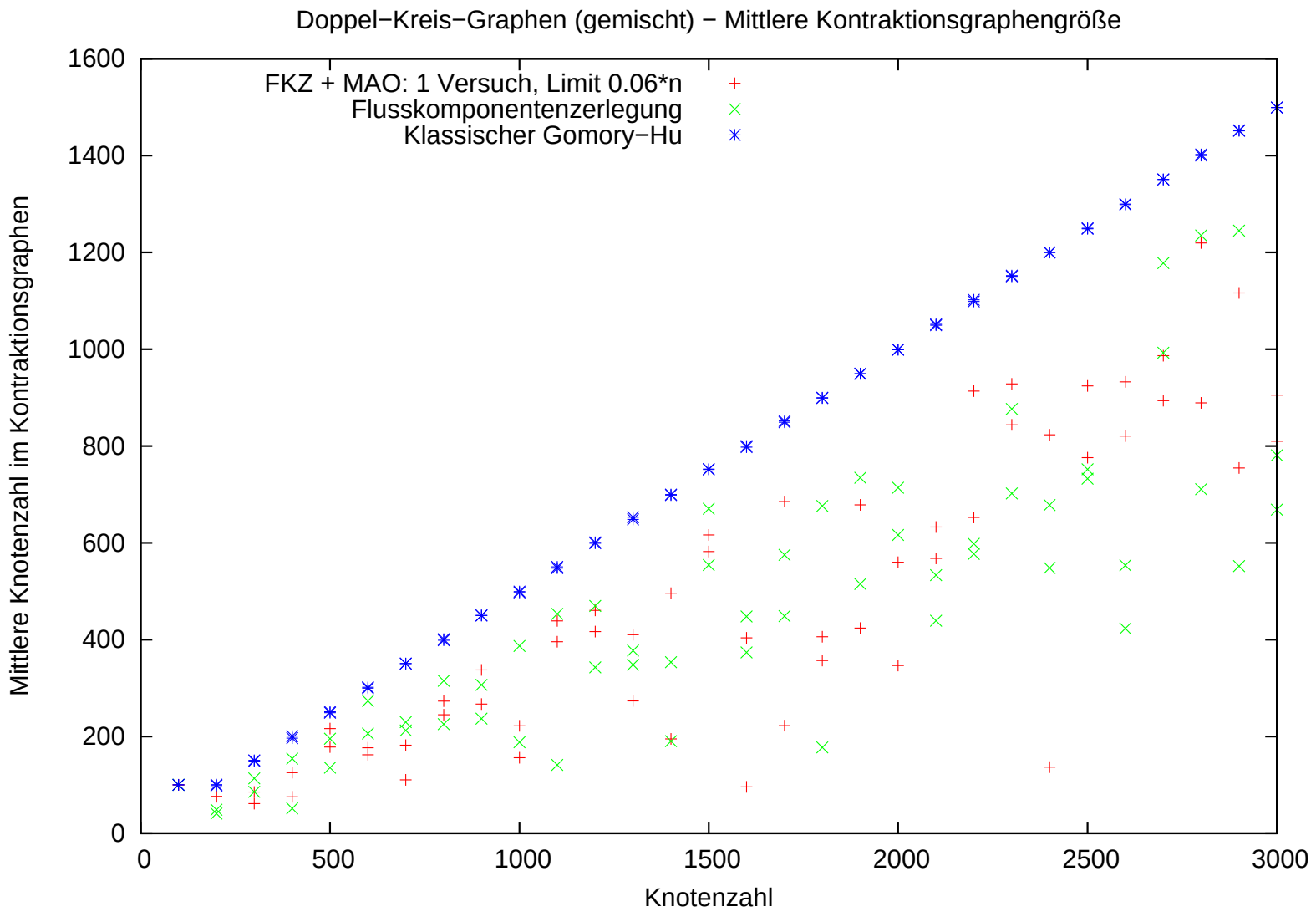


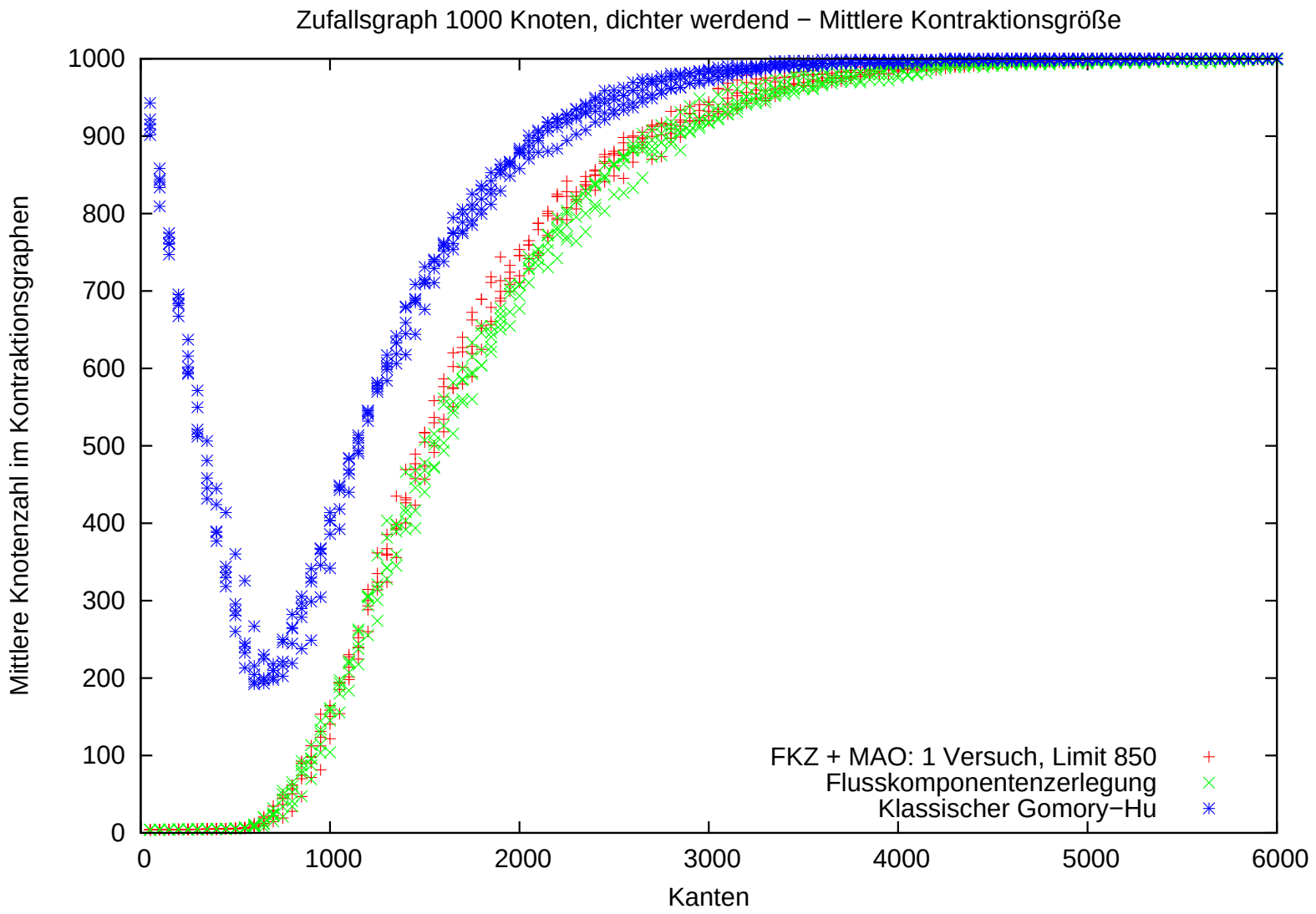
T mit b-t-MinCut

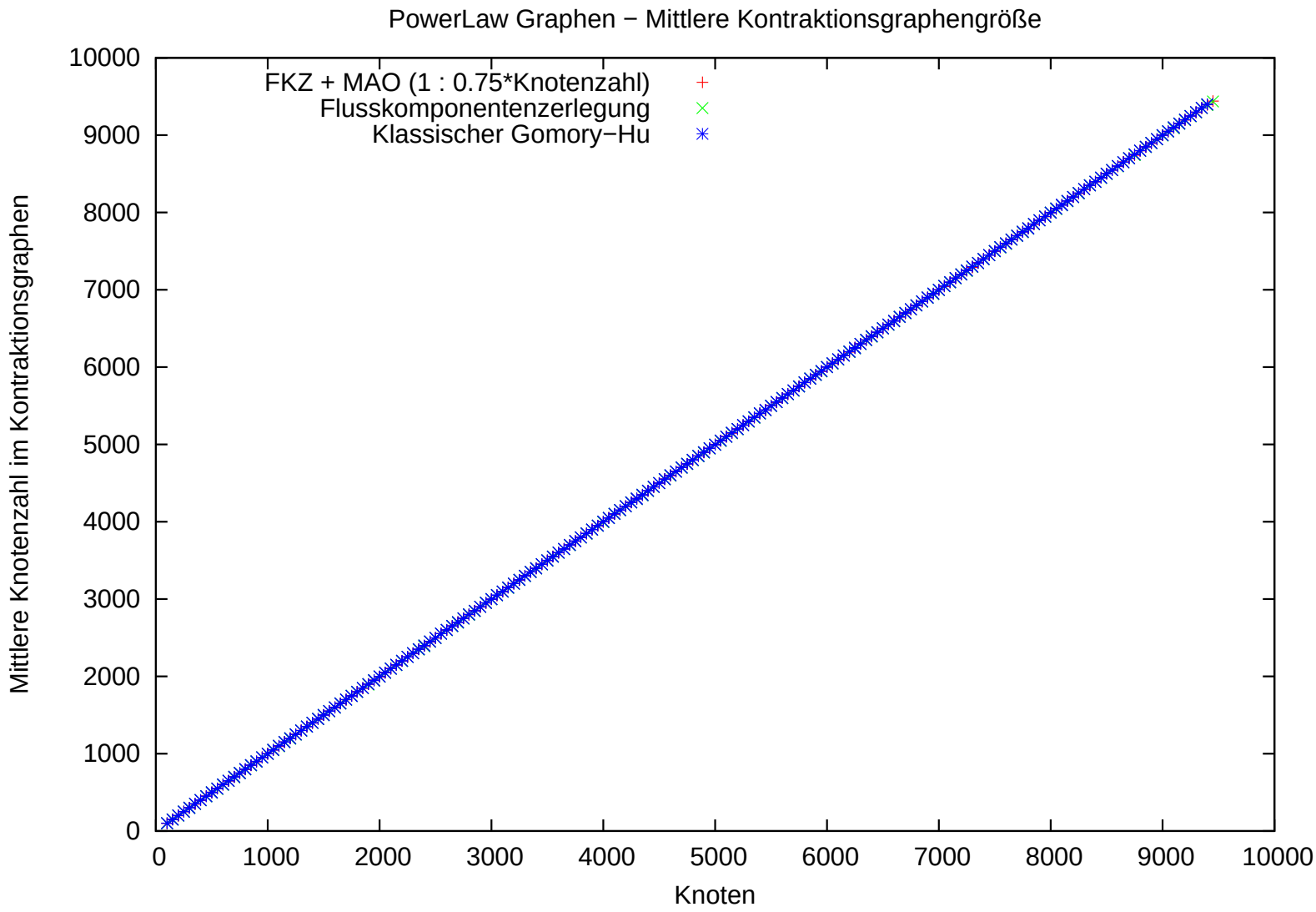


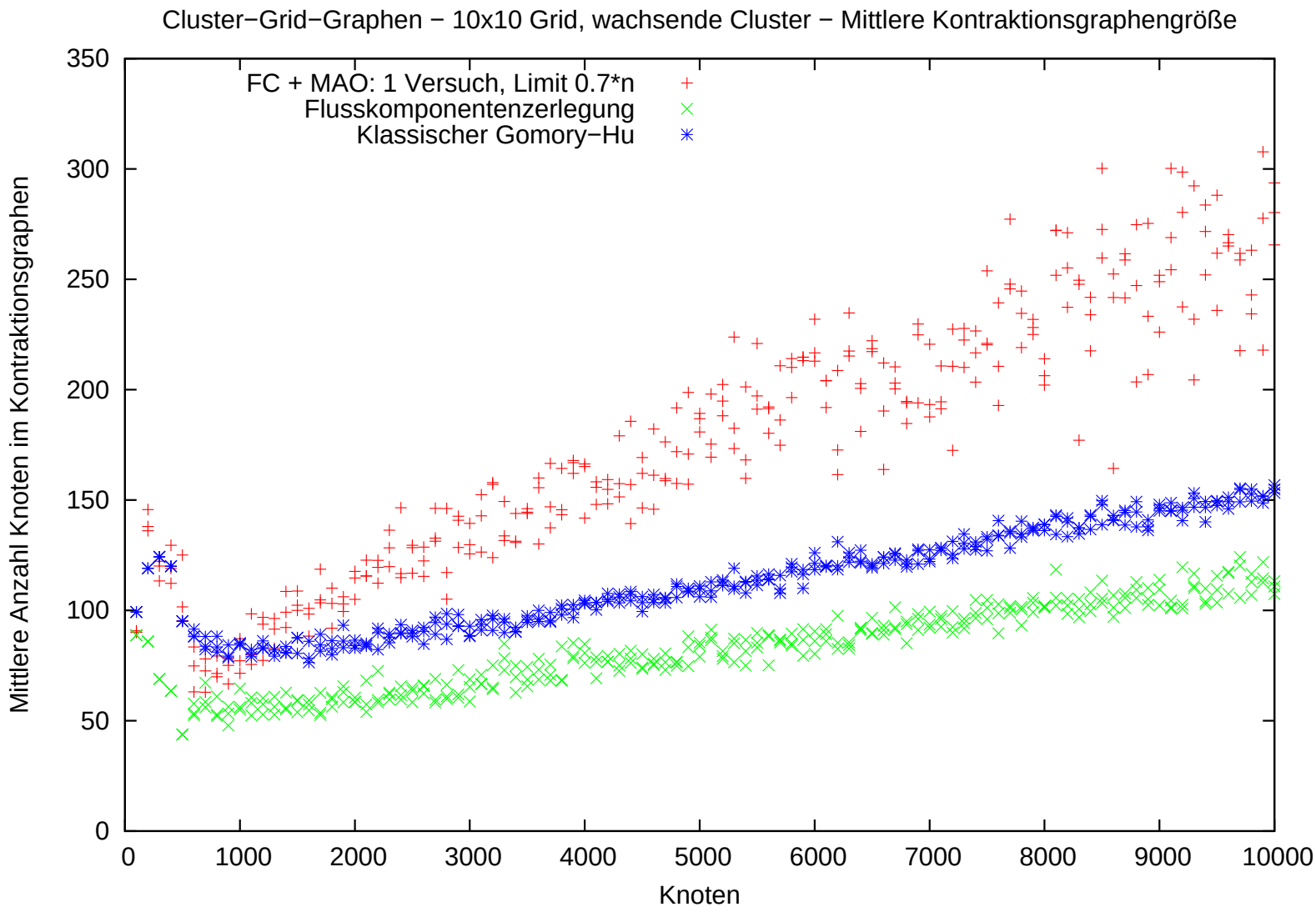
Relaxierung Gomory-Hu-Baum

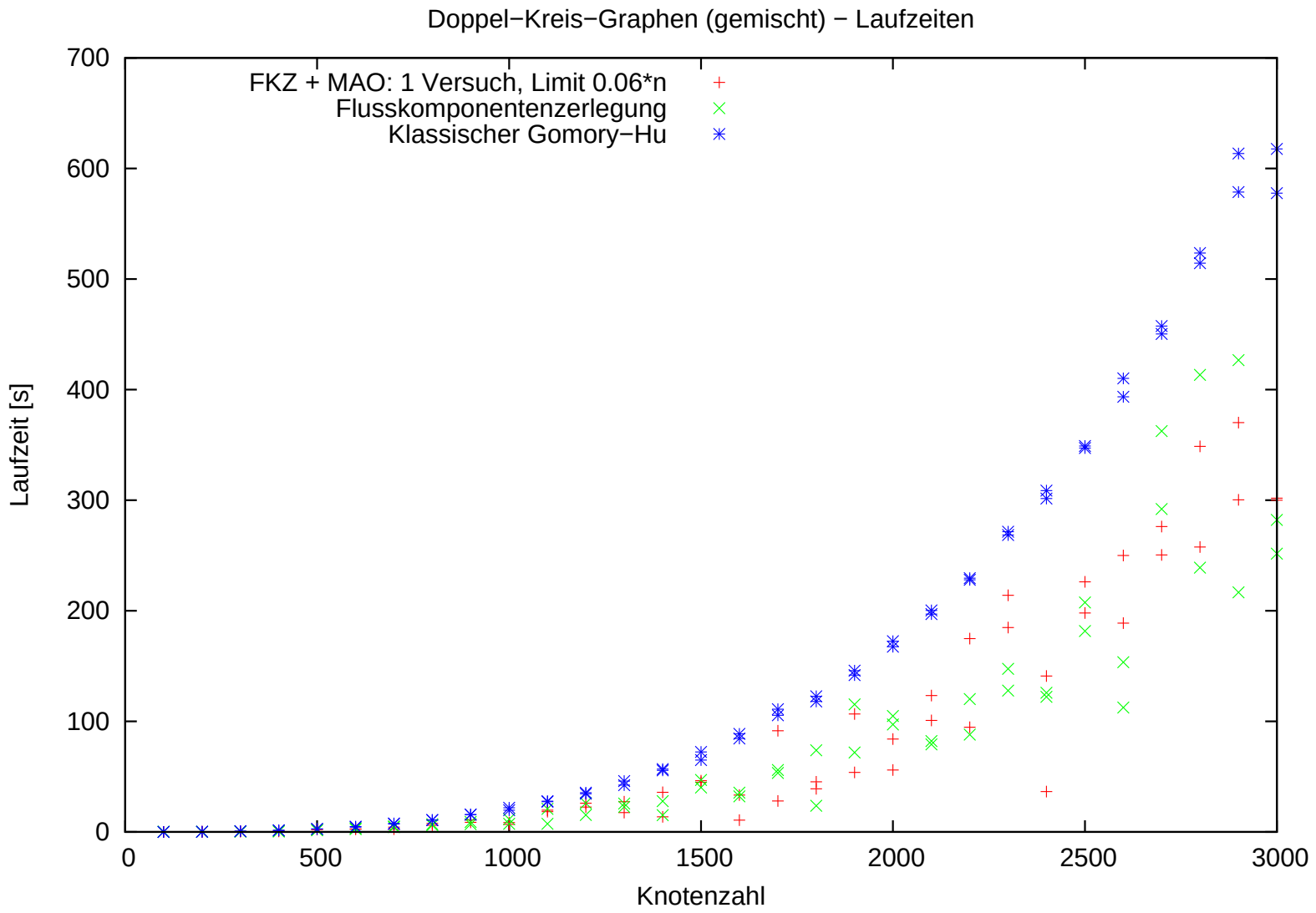
- Erweiterung der Knotenmenge um *Leere Knoten* mit leerer assoziierter Knotenmenge
- Ermöglicht lokale Umstrukturierung von T um:
  - Neue Schnittinformation aufzunehmen
  - ohne bestehende Schnittinformation zu verfälschen
- Umformung in Gomory-Hu-Baum in $\mathcal{O}(n^2)$ möglich

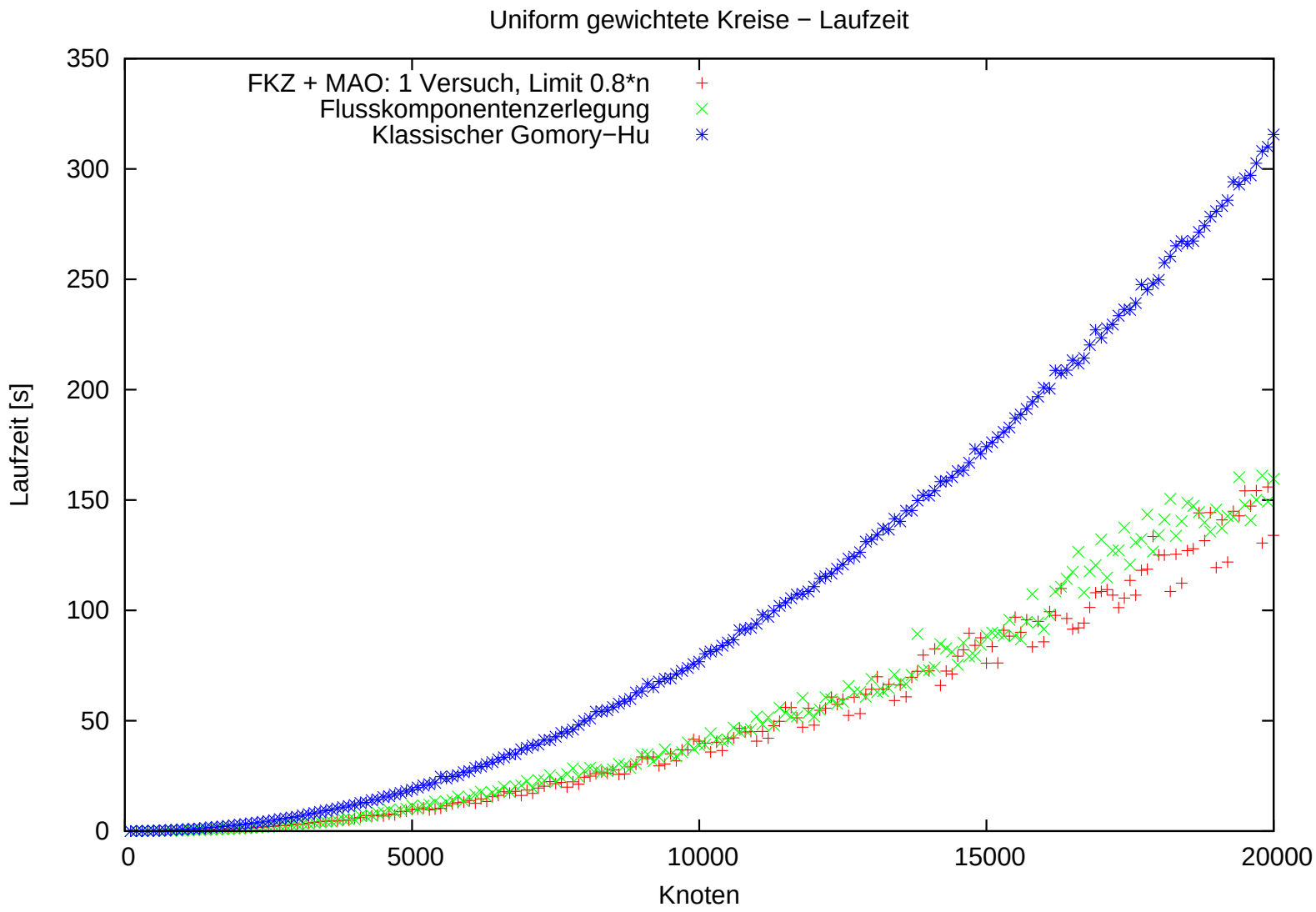

Doppel−Kreis−Graphen (gemischt) − Mittlere Kontraktionsgraphengröße
FKZ + MAO: 1 Versuch, Limit 0.06*n
Flusskomponentenzerlegung
Klassischer Gomory−Hu
Mittlere Knotenzahl im Kontraktionsgraphen
Knotenzahl
0
200
400
600
800
1000
1200
1400
1600
0
500
1000
1500
2000
2500
3000

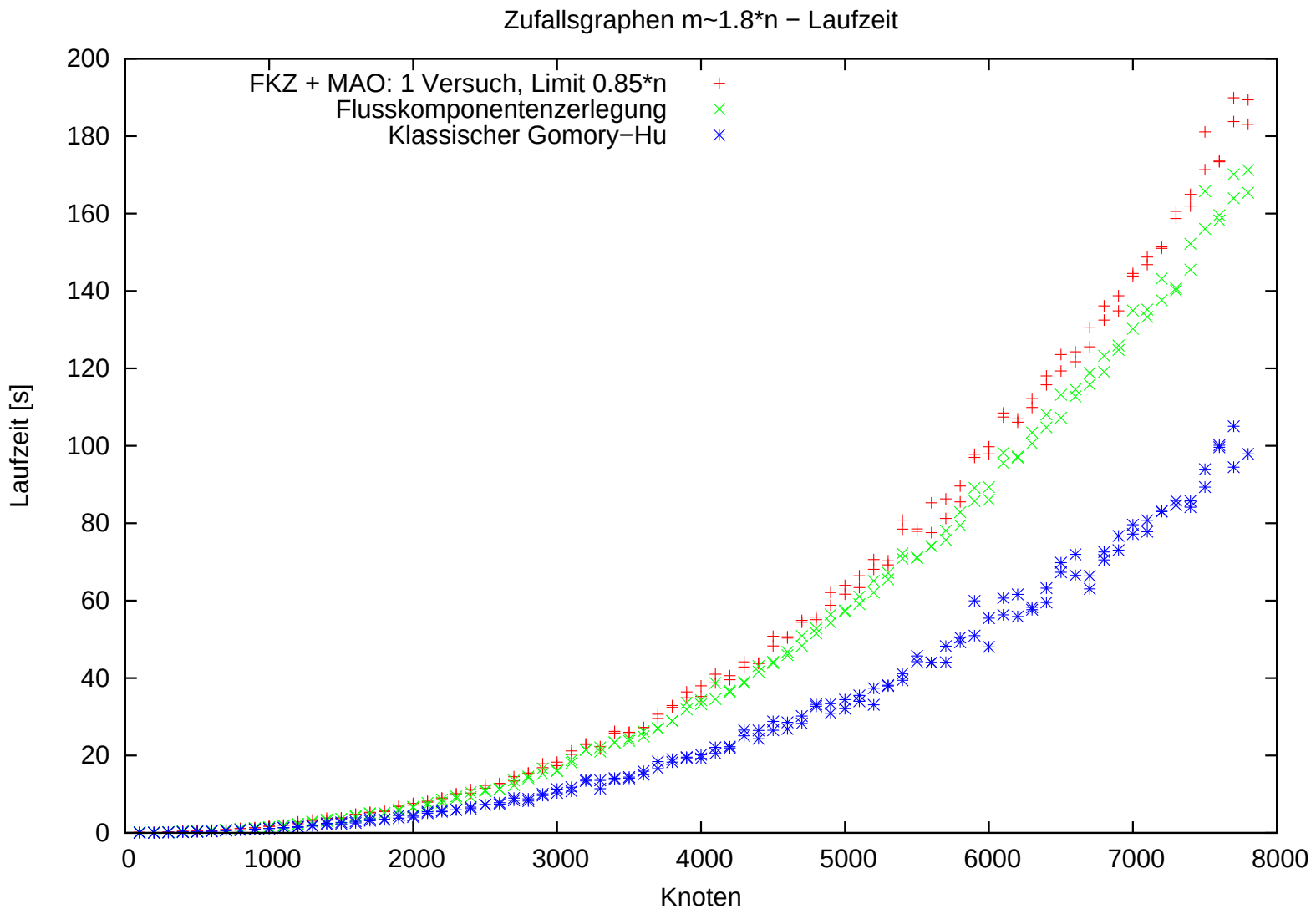

Zufallsgraph 1000 Knoten, dichter werdend − Mittlere Kontraktionsgröße
Mittlere Knotenzahl im Kontraktionsgraphen
Kanten
0
100
200
300
400
500
600
700
800
900
1000
0
1000
2000
3000
4000
5000
6000
FKZ + MAO: 1 Versuch, Limit 850
Flusskomponentenzerlegung
Klassischer Gomory−Hu

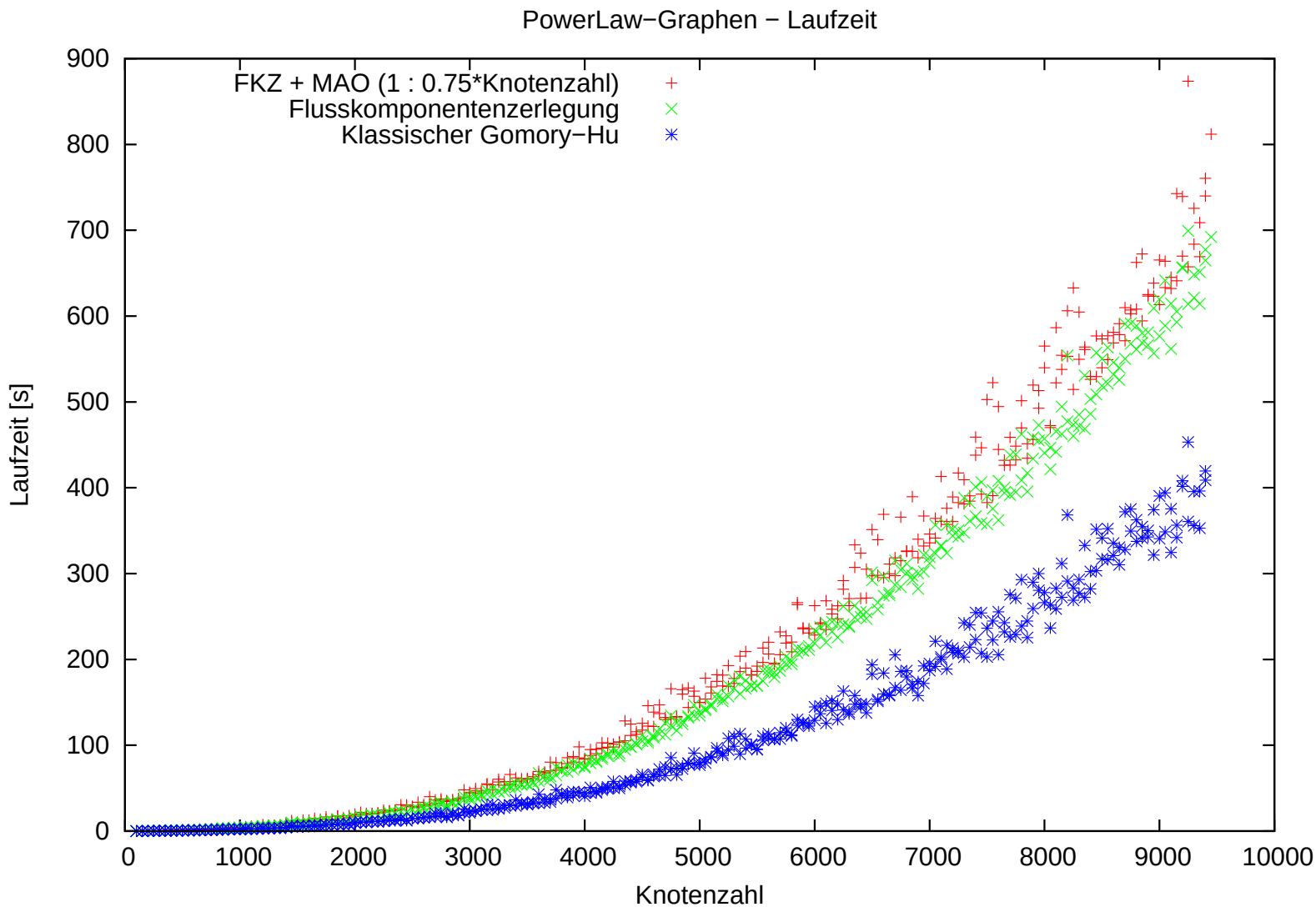

PowerLaw Graphen − Mittlere Kontraktionsgraphengröße
FKZ + MAO (1 : 0.75*Knotenzahl)
Flusskomponentenzerlegung
Klassischer Gomory−Hu
Mittlere Knotenzahl im Kontraktionsgraphen
Knoten
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000


Cluster−Grid−Graphen − 10x10 Grid, wachsende Cluster − Mittlere Kontraktionsgraphengröße
FC + MAO: 1 Versuch, Limit 0.7*n
Flusskomponentenzerlegung
Klassischer Gomory−Hu
Mittlere Anzahl Knoten im Kontraktionsgraphen
Knoten
0
50
100
150
200
250
300
350
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000


Doppel−Kreis−Graphen (gemischt) − Laufzeiten
FKZ + MAO: 1 Versuch, Limit 0.06*n
Flusskomponentenzerlegung
Klassischer Gomory−Hu
Laufzeit [s]
Knotenzahl
0
100
200
300
400
500
600
700
0
500
1000
1500
2000
2500
3000


Uniform gewichtete Kreise − Laufzeit
FKZ + MAO: 1 Versuch, Limit 0.8*n
Flusskomponentenzerlegung
Klassischer Gomory−Hu
Laufzeit [s]
Knoten
0
50
100
150
200
250
300
350
0
5000
10000
15000
20000


Zufallsgraphen m~1.8*n − Laufzeit
FKZ + MAO: 1 Versuch, Limit 0.85*n
Flusskomponentenzerlegung
Klassischer Gomory−Hu
Laufzeit [s]
Knoten
0
20
40
60
80
100
120
140
160
180
200
0
1000
2000
3000
4000
5000
6000
7000
8000


PowerLaw−Graphen − Laufzeit
FKZ + MAO (1 : 0.75*Knotenzahl)
Flusskomponentenzerlegung
Klassischer Gomory−Hu
Laufzeit [s]
Knotenzahl
0
100
200
300
400
500
600
700
800
900
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000