Seminar Approximationsalgorithmen Steinerwald und Steinernetzwerk

Sebastian Bauer

Institut für Theoretische Informatik (ITI) Wagner

12. Juli 2008

Teil I

Steinerwald

1. Einführung
2. LP
3. Approximationsalgorithmus
4. Analyse

Intro

Was ist ein Steinerwald?

- benannt nach Jakob Steiner (1796 - 1863)
- Vergleichbar mit **Steinerbaum**
- Wald in einem Graph
- Wald hat **minimale Kosten**
- bestimmte Knoten müssen **verbunden** sein

Definition

Gegeben: ungerichteter Graph $G = (V, E)$
Kostenfunktion $c : E \to \mathbb{Q}^+$
Sammlung **disjunkter** Teilmengen von V: $S_1, \ldots, S_k$

Gesucht: Teilgraph G' von G mit:
G' hat minimale Kosten **und** jeweils
alle Knotenpaare aus S_i sind verbunden

Idee

- **geeignete** Verbindungen finden
- zerteilen des Graphen in **Schnitte** S
- Schnitte **bewerten** & **klassifizieren**

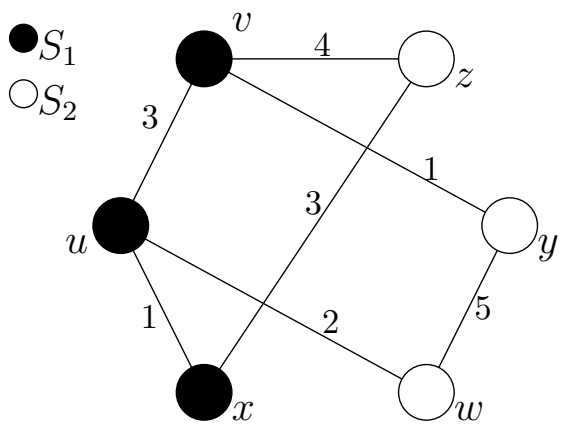
Vorüberlegungen I

Verbindungsbedarfsfunktion:

Definition

$$r : V \times V \rightarrow \{0, 1\}$$

$$u, v \mapsto \begin{cases} 1, \text{ wenn } u \text{ und } v \in S_i \\ 0, \text{ sonst} \end{cases}$$



Beispiel: $r(u, v) = 1,\ r(u, w) = 0$

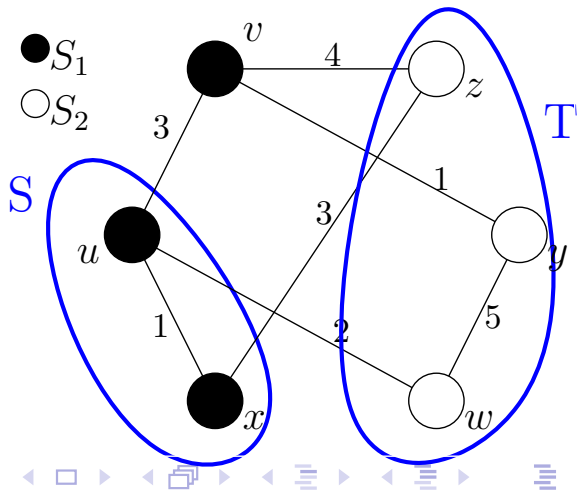
Vorüberlegungen II

Schnittbedarfsfunktion:

Definition

$$f : 2^V \rightarrow \{0,1\}$$

$$S \mapsto \begin{cases} 1, \text{ wenn } \exists u \in S_i \text{ und } v \in \overline{S_i} \text{ mit } r(u,v) = 1 \\ 0, \text{ sonst} \end{cases}$$



Beispiel: $f(S) = 1,\ f(T) = 0$

Primales LP

minimiere $\sum_{e \in E} c_e x_e$

Nebenbedingungen: $\sum_{e : e \in \delta(S)} x_e \geq f(S), \quad S \subseteq V$

$x_e \geq 0, \quad e \in E$

wobei gilt:

- $x_e = \begin{cases} 1, \text{ wenn } e \text{ verwendet wird} \\ 0, \text{ sonst} \end{cases}$
- $\delta(S) =$ Menge an Kanten, die den Schnitt $(S, \overline{S})$ kreuzen.

Duales LP

maximiere $\sum_{S \subseteq V} f(S) \cdot y_S$

Nebenbedingungen: $\sum_{S : e \in \delta(S)} y_S \leq c_e, \quad e \in E$ [1]

$y_S \geq 0, \quad S \subseteq V$

[1] alle Schnitte, die **die Kante** e kreuzt

Idee I

- geeignete Schnitte suchen zum **Anheben**
- initial alle primale x_e und duale $y_S = 0$ setzen
  $\Rightarrow$ keine Kanten gewählt, keine Schnitte bewertet
- alle y_S **synchron** anheben
  $\Rightarrow$ Herantasten an günstige Lösung
  $\Rightarrow$ simultan viele verschiedene Lösungen ausprobieren
- **Primales-Duales Schema mittels Synchronisation**
- Problem: Wann soll nicht mehr erhöht werden?

Idee II

Wir **klassifizieren** die Schnitte:

- S ist **unbefriedigt**, wenn gilt:
  - $f(S) = 1$
  - keine Kante kreuzt den Schnitt $(S, \overline{S})$
- S ist **aktiv**, wenn gilt:
  - S ist unbefriedigt
  - S ist minimal unter Inklusion

Algorithmus 1 : Steinerwald

input : Graph $G = (V, E)$

Initialisierung:

$F \leftarrow \emptyset$

foreach $S \subseteq V$ **do**

  $y_S \leftarrow 0$

Kantenzuwachs:

while $\exists$ *unbefriedigte Menge* **do**

  repeat

    foreach *aktiver Schnitt* y_S **do**

      erhöhe y_S

  until *Kante* e *berührt die Grenze*

  $F \leftarrow F \cup \{e\}$

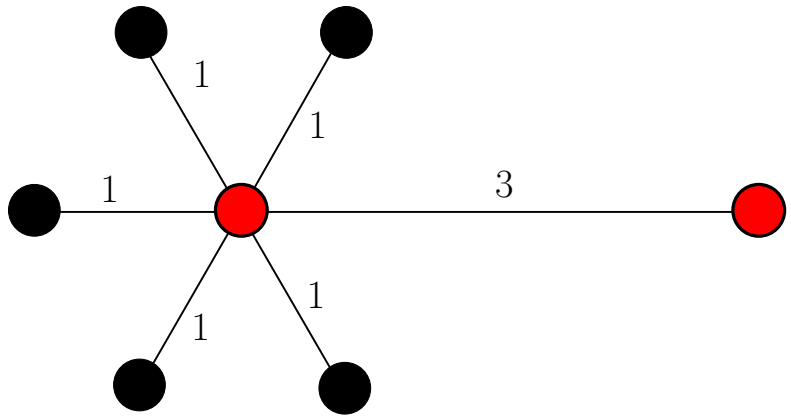
Algorithmus 2 : Steinerwald (Fort.)

output : Restkantenmenge F'

1 *Säuberung:*

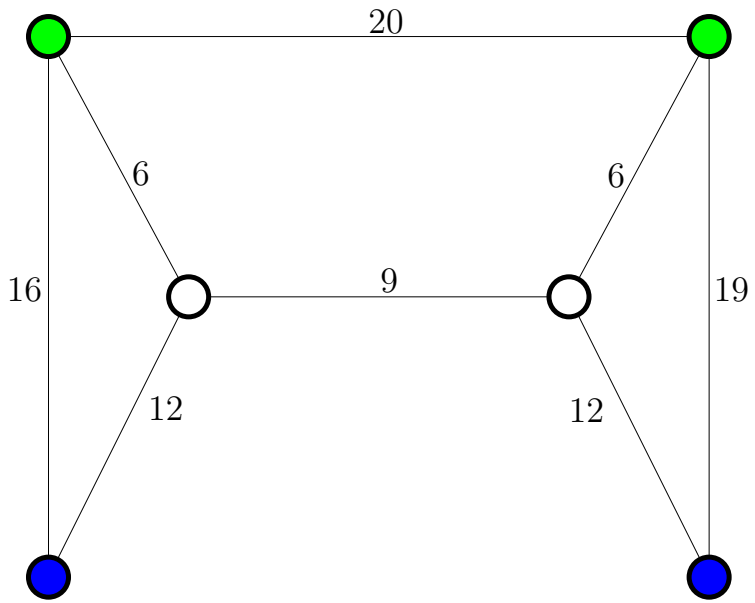
2 $F' = \{e \in F | F - \{e\}$ ist primal zulässig $\}$

- rote Knoten zu verbinden
- Algo. nimmt zuerst die "1"-Kanten
- $\Rightarrow$ *redundante* Kanten entfernen



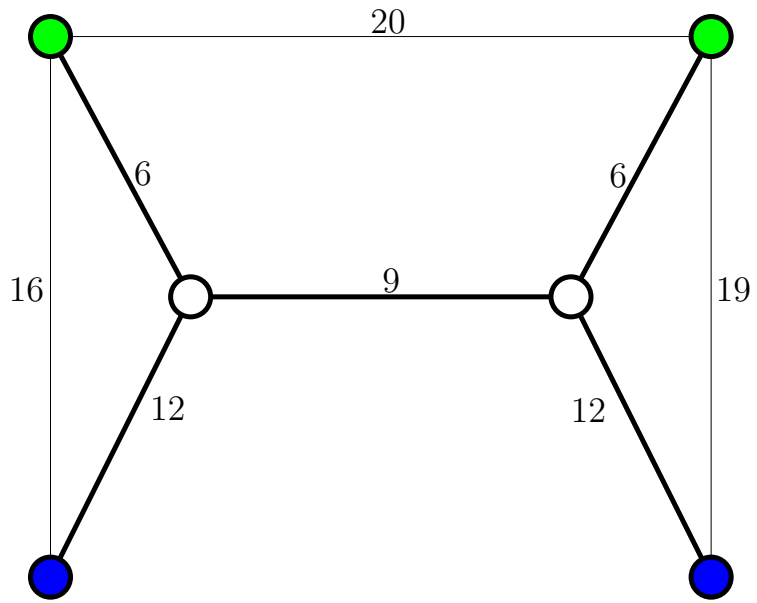
Beispiel

Ausgangsgraph:



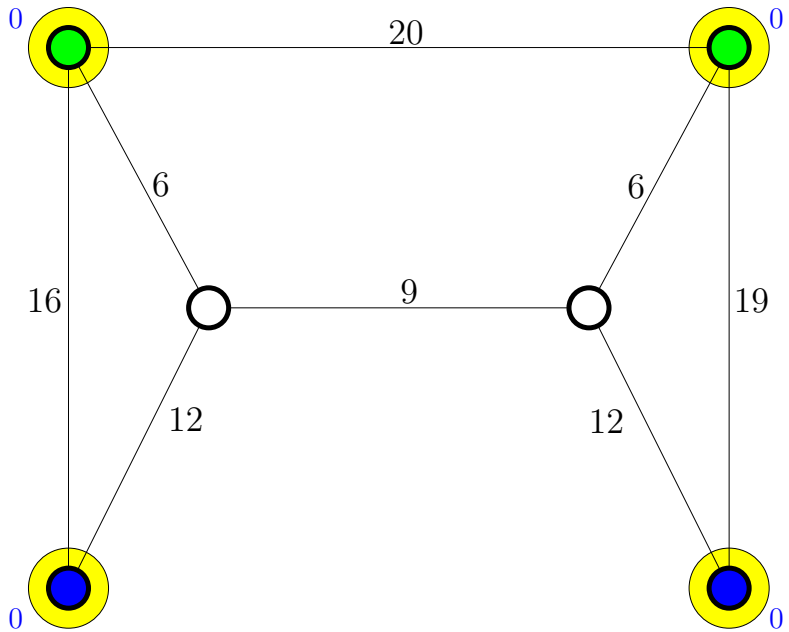
Beispiel

optimale Lösung: Kosten von 45



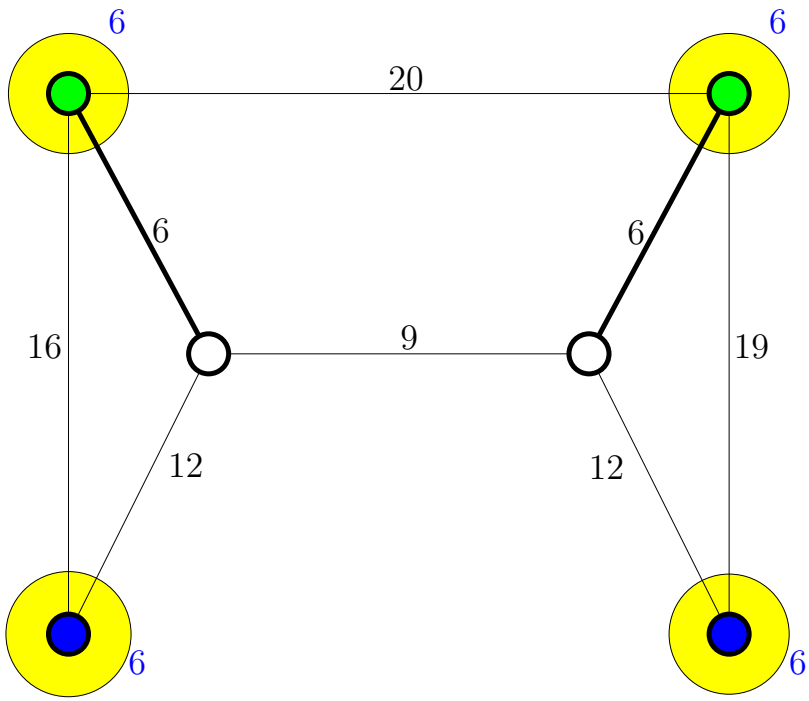
Beispiel

Start: aktive Mengen sind nur die zu verbindenden Knoten



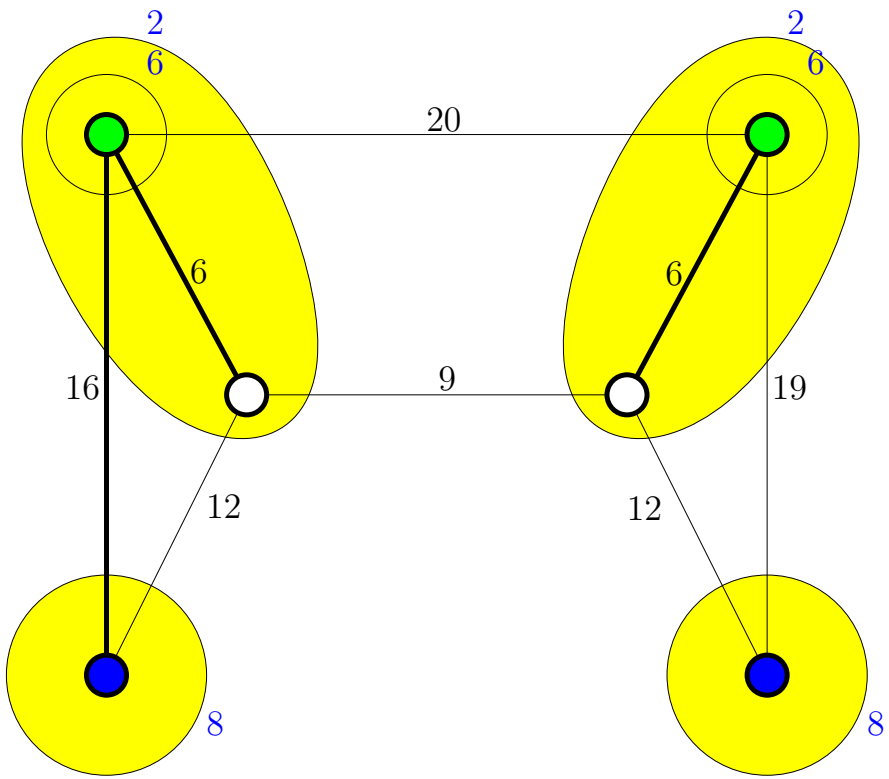
Beispiel

Zwischenstatus 1: erste Kanten berühren die Grenze



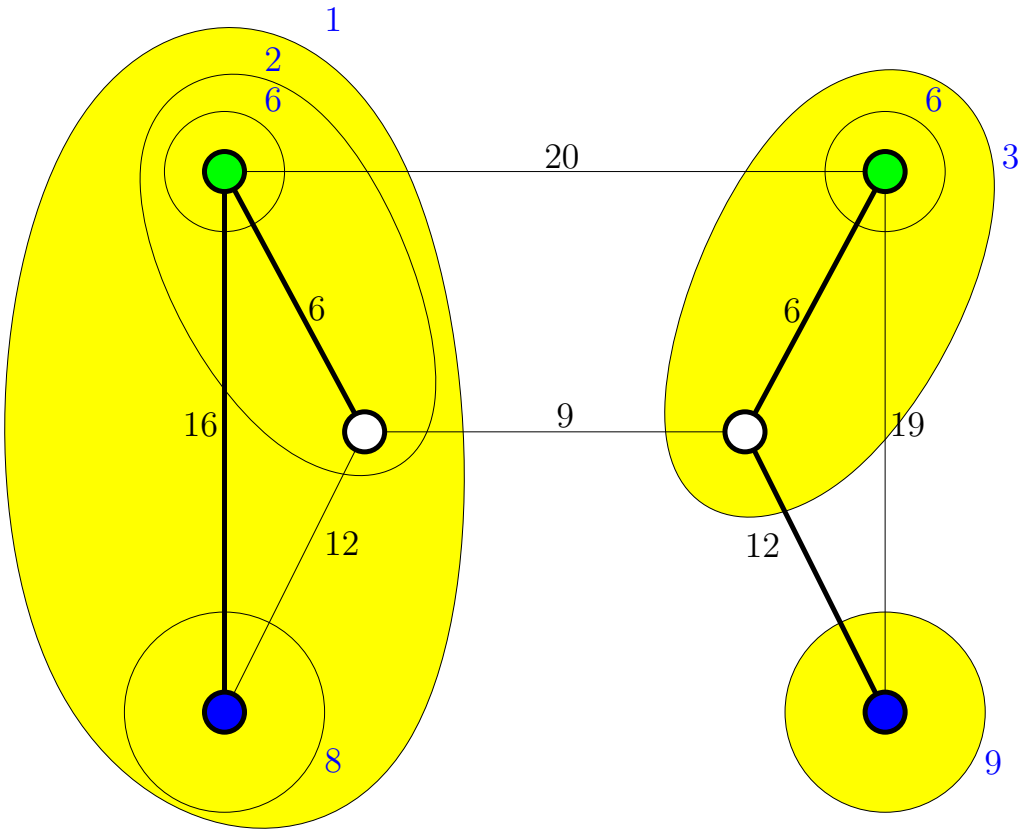
Beispiel

Zwischenstatus 2: neue aktive Mengen, nächste Kante grenzt an



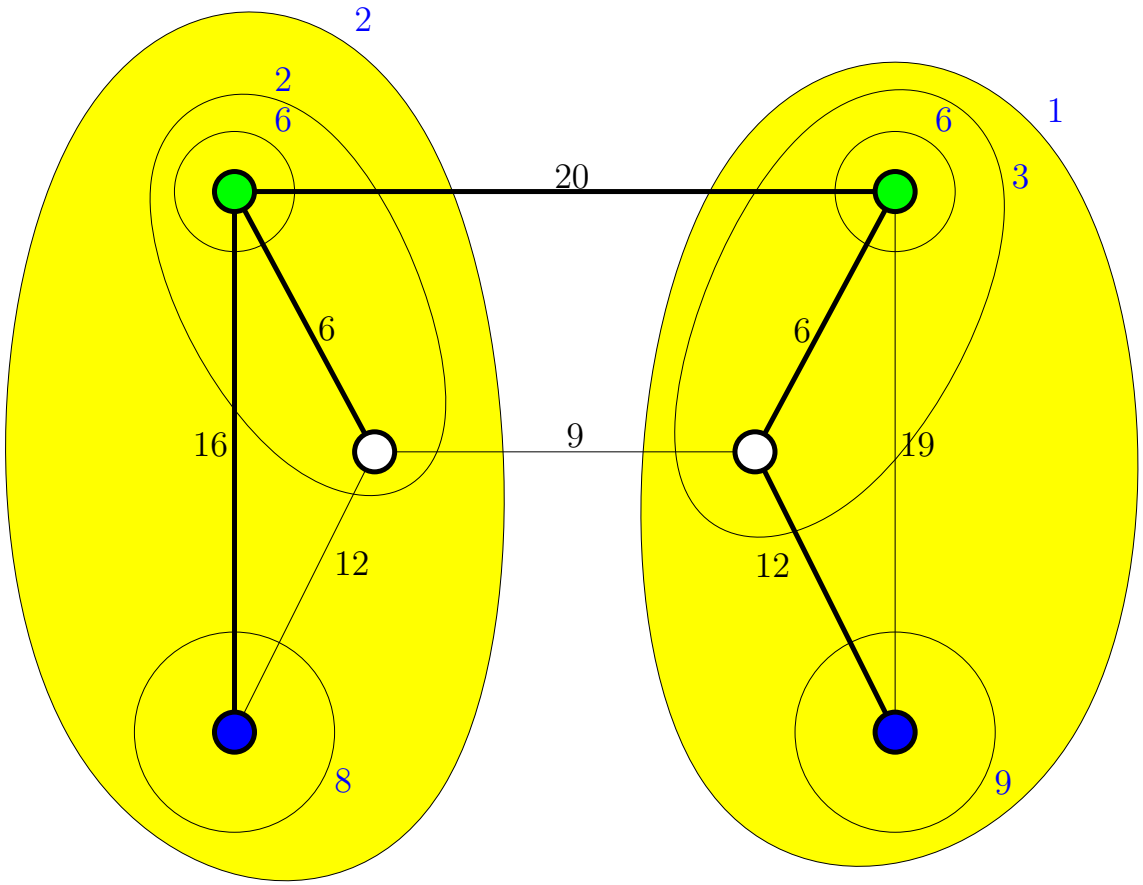
Beispiel

Zwischenstatus 3: Schnitte kommen dazu



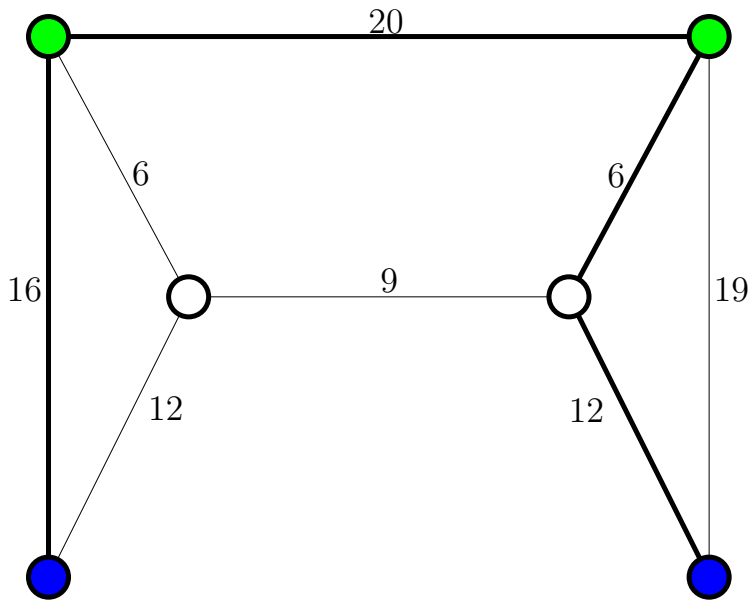
Beispiel

Zwischenstatus 4: letzte benötigte Kante wird grenzt an



Beispiel

Ergebnis: Kosten von 54



zulässige Lösung

Lemma

Am Ende des Algorithmus sind F' und y primal bzw. dual **zulässige Lösungen**.

Beweisskizze:

- Lösung ist zu jeder Zeit ein **Wald**
- nichtredundante Kanten werden nicht gelöscht $\Rightarrow$ **primal zulässig**
- **Neueinteilung** der aktiven Schnitte **verhindert Überladen** von Kanten $\Rightarrow$ **dual zulässig**

Approximationsfaktor

Lemma

$$\sum_{e \in F'} c_e \leq 2 \sum_{S \subseteq V} y_S$$

Beweis: nicht hier

Lemma

Der Approximationsalgorithmus erreicht eine Approximationsgarantie vom **Faktor 2** für das Steiner-Wald-Problem.

Approximationsfaktor

Lemma

$$\sum_{e \in F'} c_e \leq 2 \sum_{S \subseteq V} y_S$$

Beweis: nicht hier

Lemma

Der Approximationsalgorithmus erreicht eine Approximationsgarantie vom **Faktor 2** für das Steiner-Wald-Problem.

Teil II

Steinernetzwerk

5 Steinernetzwerk

6 LP

7 Approximationsalgorithmus

Intro

Gemeinsamkeiten

- Knoten aus der **selben Menge** müssen **verknüpft** werden
- Ergebnisgraph kann ein Wald sein
- Approximationsalgorithmus löst das Problem **iterativ**

Unterschiede

- eine Menge besteht nur noch aus 2 Knoten
- Kanten haben zusätzlich eine **Kapazität**
- Kanten werden beim Auswählen **kopiert**
- jede Kante darf **nur einmal** verwendet werden

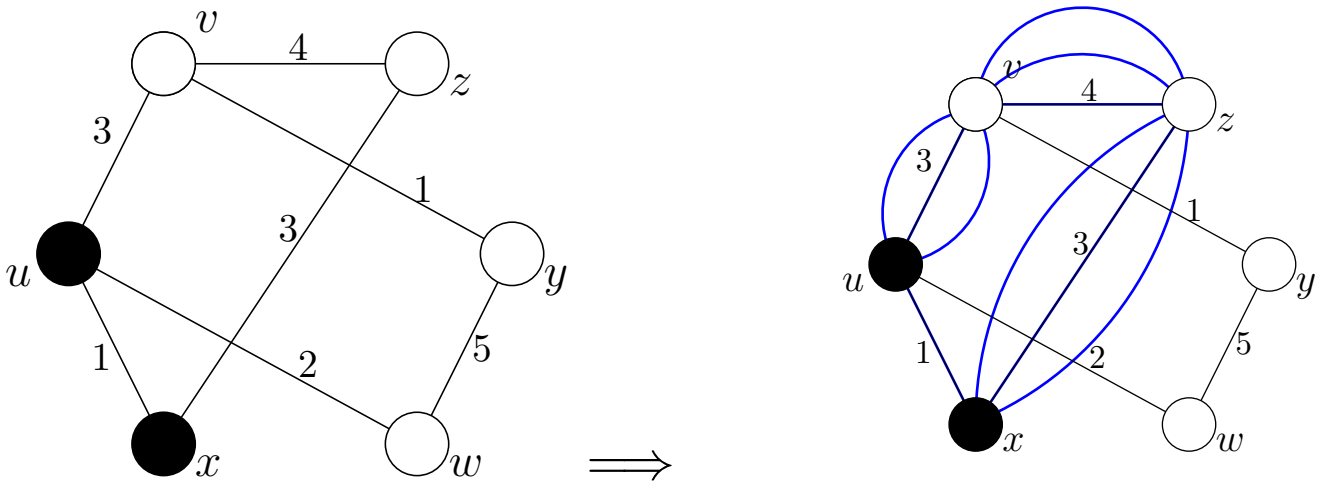
Definition

Gegeben: ungerichteter Graph $G = (V, E)$
Kostenfunktion $c : E \to \mathbb{Q}^+$
Verbindungsbedarfsfunktion $r : V \times V \to \mathbb{Z}^+$
Kapazitätsfunktion $u : E \to \mathbb{Z}^+ \cup \{\infty\}$
($\Rightarrow$ Anzahl an Kopien / Kante)

Gesucht: Multigraph über V mit:
$r(u, v)$ **disjunkter** Pfade $\forall u, v \in V$
minimalen Kosten
jede Kopie einer Kante kostet $c(e)$

Beispiel $r(u, v)$

Start- und Zielgraph für $r(u, x) = 4$:



Vorüberlegung

Schnittbedarfsfunktion (vgl. Steiner Wald):

Definition

$$f : 2^V \rightarrow \mathbb{Z}^+$$

S := größter Verbindungsbedarf, der durch den Schnitt $(S, \overline{S})$ getrennt wird

z.B. $f(S) = \max\{r(u,v) | u \in S \wedge v \in \overline{S}\}$

LP

minimiere $\sum_{e \in E} c_e x_e$

Nebenbedingungen: $\sum_{e : e \in \delta(S)} x_e \geq f(S), \quad S \subseteq V$

$x_e \geq 0, \quad e \in E$ und $u_e = \infty$

$u_e \geq x_e \geq 0, \quad e \in E$ und $u_e \neq \infty$

Lösungsansatz?

- LP-Relaxation wie z.B. bei VERTEX COVER
- Aufrunden von halben Zahlen auf 1
- $\Rightarrow$ Faktor-2 Algorithmus

Aber:

- funktioniert hier nicht!
- Gegenbeispiel: Petersens Graph

Lösungsansatz?

- LP-Relaxation wie z.B. bei VERTEX COVER
- Aufrunden von halben Zahlen auf 1
- $\Rightarrow$ Faktor-2 Algorithmus

Aber:

- funktioniert **hier nicht**
- Gegenbeispiel: Petersens Graph

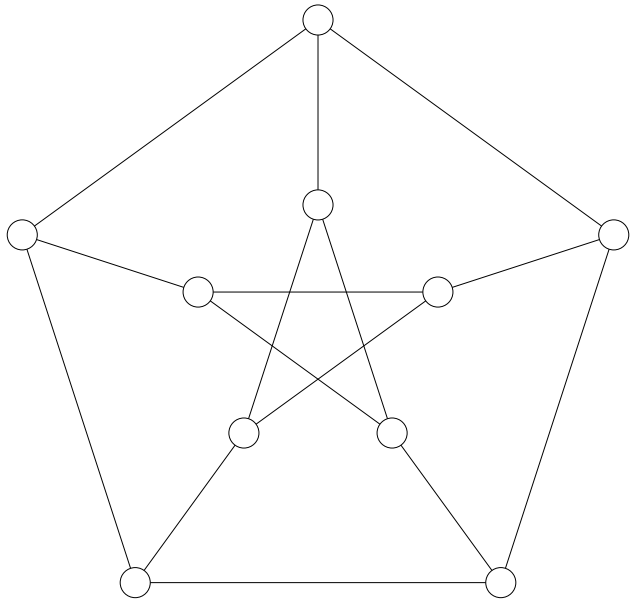
Petersens Graph I

Werte für das Problem:

- $\forall u, v \in V : r(u, v) = 1$
- $\forall e, f \in E : c(e) = c(f)$

optimale Lösung:

- $\forall x_e : x_e = \frac{1}{3} \Rightarrow$ Kosten von 5
- Problem: alle auf- oder abrunden?



Petersens Graph II

Annahme: $\exists$ **halbzahlige optimale Lösung**:

- optimale Lösung hat Kosten von **mindestens 5**
- mindestens **10 Kanten** werden gewählt (zu $\frac{1}{2}$)
- 10 Knoten sind zu verbinden
- $\Rightarrow$ Lösung ist ein Hamlitonkreis (HC)
- Petersens Graph hat **keinen** HC 

$\Rightarrow \nexists$ Algorithmus für Steinernetzwerk mittels Runden einer halbzahligen Lösung

Lösungsansatz!

- man betrachtet eine “Extrempunktlösung”
  - fasse alle x_e zu einem Vektor $\boldsymbol{x}$
  - $\boldsymbol{x}$ ist eine Basislösung
  - nicht notwendigerweise halbzahlig
  - kann nicht als Linearkombination zweier günstiger Lösungen dargestellt werden
- Approximation bedient sich der Extrempunktlösung:
  - Ecken $\geq \frac{1}{2}$ werden aufgerundet
  - danach eine weitere Iteration durchgeführt
  - Algorithmus erreicht Faktor 2

Vorbereitung I

Restbedarfsfunktion:

Definition

$$\begin{array}{lcl} f' : 2^V & \rightarrow & \mathbb{Z}^+ \\ S & \mapsto & f(S) - |\delta_H(S)| \end{array}$$

wobei $\delta_H(S)$ die Menge an Kanten ist, die den Schnitt $(S, \overline{S})$ im Graphen H kreuzt.

Vorbereitung II

Theorem (Basistheorem)

Für jede schwach supermodulare Funktion f *und* **jede Extermpunktlösung** x *muss das LP* **mindestens eine Kante** e *von mindestens der Größe* $\frac{1}{2}$ *auswählen (z.B.* $x_e \geq \frac{1}{2}$*).*

Beweis: hier nicht.

Algorithmus 3 : Steinernetzwerk

input : Graph $G = (V, E)$

output : Graph H

$H \leftarrow \emptyset$, $f' \leftarrow f$.

while $f' \not\equiv 0$ **do**

  finde Extrempunktlösung $\boldsymbol{x}$ für das LP mit
    Schnittbedarfsfunktion f'

  foreach *Ecke* e *mit* $x_e \geq \frac{1}{2}$ **do**

    $H = H \cup \lceil x_e \rceil \cdot \{e\}$

    $u_e = u_e - \lceil x_e \rceil$

  Aktualisiere f': **for** $S \subseteq V$ **do**

    $f'(S) \leftarrow f(S) - |\delta_H(S)|$

Iteriertes Runden I

1. Iteration:

- **Max-Flow**-Routine als Basis für ein **Separations-Orakel**
- Prüfen auf Korrektheit:
  - Neuer Graph über V mit Kantenkapazitäten x_e
  - **Prüfung**, ob $\forall u, v \in V : \exists$ ein Fluss von u nach v mit Kapazität von min. $r(u, v)$
  - Falls nicht erfüllt $\Rightarrow$ **verletzter Schnitt**

Iteriertes Runden II

Folgeiteration:

- f' neue Schnittvoraussetzung
- Lösung x entsteht auf Basis der alten Lösung x':
  - setzte für $x'_e \geq \frac{1}{2}$: $x_e = \lceil x'_e \rceil$
  - x_e zur Ganzzahl machen
  - Weiterberechnen des LP mit den übrigen Variablen
- $\Rightarrow$ **Iteriertes Runden**

Approximationsgarantie I

Theorem

Der Algorithmus für Steinernetzwerk erreicht eine Approximationsgarantie von Faktor 2.

Approximationsgarantie II

Beweis mittels Induktion über die Iterationsschritte:

- Voraussetzung
  - nur Kanten mit $x_e \geq \frac{1}{2}$ werden gerundet $\Rightarrow$ Behauptung
- Induktionsanfang
  - $\boldsymbol{x}$ ist Lösung nach Schritt 1
  - $\boldsymbol{x'}$ erhält man durch Ausnullen von Einträgen $< \frac{1}{2}$, $x' \neq 0$
  - Aufrunden nur von Einträgen $\geq \frac{1}{2} \Rightarrow \text{cost}(H) \leq 2\ \text{cost}(x')$

Approximationsgarantie III

- Induktionsschritt
  - H' und f' seien Graph und Funktion einer Folgeiteration
  - Beobachtung: $\boldsymbol{x} - \boldsymbol{x}'$ ist zulässige Lösung für f'
  - $\overset{IV}{\Rightarrow}$ $\text{cost}(H') \leq 2\ \text{cost}(\boldsymbol{x} - \boldsymbol{x}')$
  - $\text{cost}(H + H') \leq 2\ \text{cost}(H) + 2\ \text{cost}(H')$
  - $\text{cost}(H + H') \leq 2\ \text{cost}(\boldsymbol{x}') + 2\ \text{cost}(\boldsymbol{x} - \boldsymbol{x}') \leq 2\ \text{cost}(\boldsymbol{x})$

Korollar

Der *itegrality gap* des LP ist beschränkt durch 2.

Approximationsgarantie III

- Induktionsschritt
  - H' und f' seien Graph und Funktion einer Folgeiteration
  - Beobachtung: $\boldsymbol{x} - \boldsymbol{x}'$ ist zulässige Lösung für f'
  - $\overset{IV}{\Rightarrow}$ $\text{cost}(H') \leq 2\ \text{cost}(\boldsymbol{x} - \boldsymbol{x}')$
  - $\text{cost}(H + H') \leq 2\ \text{cost}(H) + 2\ \text{cost}(H')$
  - $\text{cost}(H + H') \leq 2\ \text{cost}(\boldsymbol{x}') + 2\ \text{cost}(\boldsymbol{x} - \boldsymbol{x}') \leq 2\ \text{cost}(\boldsymbol{x})$

Korollar

Der *itegrality gap* des LP ist beschränkt durch 2.

Zusammenfassung

- Steinerwald
  - Graph wird in Schnitte aufgeteilt
  - Schnitte werden iterativ erhöht
  - Ausprobieren vieler Lösungen gleichzeitig
- Steinernetzwerk
  - auf LP-Relaxation wird zurückgriffen
  - Zwischenlösungen wählen Kanten durch Runden aus
  - Herantasten an die optimale Lösung

noch Fragen?

Quellen

Vijay V. Vazirani. Approximation Algorithms. Springer-Verlag, 2001
Kapitel 12, 22, 23