

# Das Problem des minimalen Steiner-Baumes

Ein polynomieller Approximationsalgorithmus

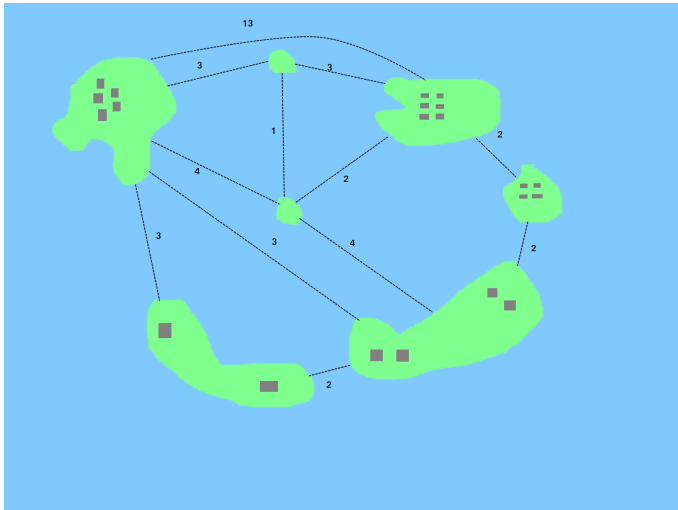
Benedikt Wagner | 14.05.2018

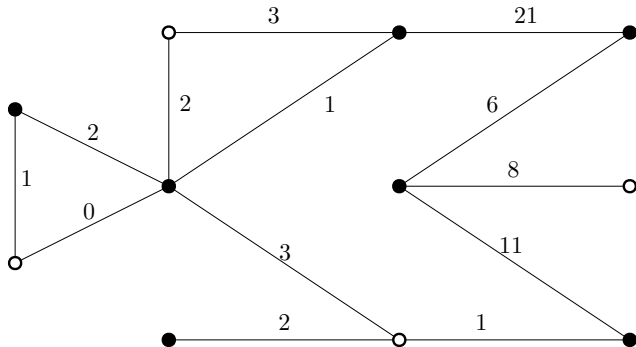
INSTITUT FÜR THEORETISCHE INFORMATIK, LEHRSTUHL ALGORITHMIK 1



- 1 Problemstellung
  - Motivation
  - Formale Definition
  - Beispiele
- 2 Andere Arbeiten
- 3 Approximationsalgorithmus
  - Definition des Algorithmus von Kou, Markowsky und Berman
- 4 Analyse
  - Laufzeit
  - Relative Gütegarantie
- 5 Zusammenfassung

# Motivation





## Gegeben:

- Ungerichteter gewichteter zusammenhängender Graph  
 $G = (V, E, d), E \subseteq \{\{u, v\} \in V \times V \mid u \neq v\},$   
 $d : E \rightarrow \mathbb{R}_0^+$
- Steinerpunkte  $S \subseteq V$

## Gesucht:

- Baum  $T = (V', E')$  sodass  $v \in S \Rightarrow v \in V'$  (Steinerbaum) wobei  
 $\sum_{e \in E'} d(e)$  minimal

Das Steiner-Baum-Problem ist  $\mathcal{NP}$ -schwer

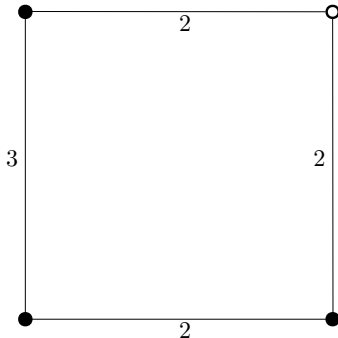
## Gegeben:

- Ungerichteter gewichteter zusammenhängender Graph  
 $G = (V, E, d), E \subseteq \{\{u, v\} \in V \times V \mid u \neq v\},$   
 $d : E \rightarrow \mathbb{R}_0^+$
- Steinerpunkte  $S \subseteq V$

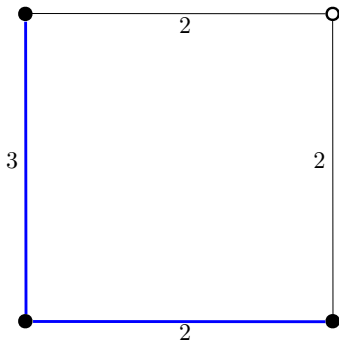
## Gesucht:

- Baum  $T = (V', E')$  sodass  $v \in S \Rightarrow v \in V'$  (Steinerbaum) wobei  
 $\sum_{e \in E'} d(e)$  minimal

Das Steiner-Baum-Problem ist  $\mathcal{NP}$ -schwer

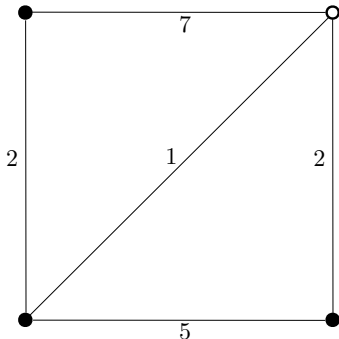


⇒ Minimaler Spannbaum entspricht nicht dem minimalen Steinerbaum

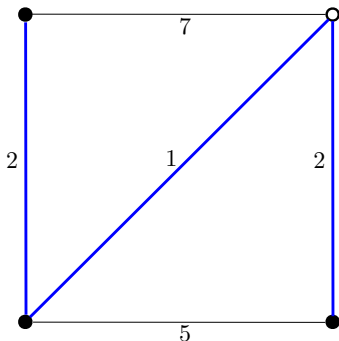


⇒ Minimaler Spannbaum entspricht nicht dem minimalen Steinerbaum



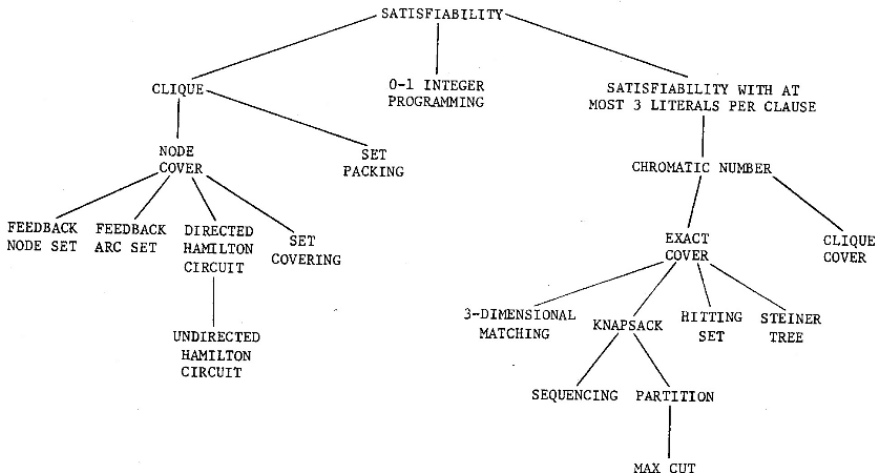


⇒ Knoten, die nicht markiert sind, können nicht einfach entfernt werden



⇒ Knoten, die nicht markiert sind, können nicht einfach entfernt werden

- eines von Karps 21  $\mathcal{NP}$ -**vollständigen** Problemen [Kar72]
- diverse Approximationsalgorithmen z.B. [Meh88] [RZ00]
- Gröpl, Hougardy, Nierhoff, Prömel zeigen:  
 $\mathcal{P} \neq \mathcal{NP} \Rightarrow$  es gibt kein PTAS für das Steinerbaumproblem [GHNP73]



aus [Kar72]



## Gegeben:

Endliche Menge  $M = \{u_1, \dots, u_n\}$ , Teilmengen  $X = \{S_1, \dots, S_m\}$

## Gefragt:

Gibt es eine Auswahl  $\{S_j, S_j, \dots, S_k\} \subseteq X$ , sodass

- $S_i, S_j, \dots, S_k$  disjunkt
- $\bigcup_{S \in X} S = M$

## Gegeben:

Endliche Menge  $M = \{u_1, \dots, u_n\}$ , Teilmengen  $X = \{S_1, \dots, S_m\}$

## Gefragt:

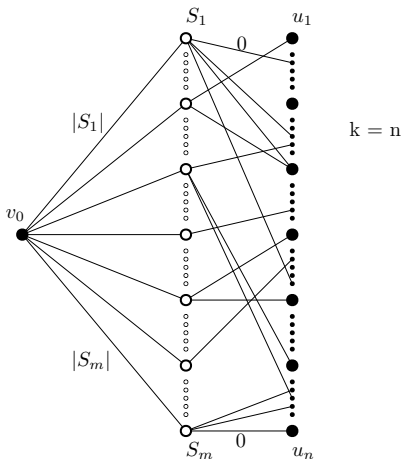
Gibt es eine Auswahl  $\{S_i, S_j, \dots, S_k\} \subseteq X$ , sodass

- $S_i, S_j, \dots, S_k$  disjunkt
- $\bigcup_{S \in X} S = M$

# EXACT-COVER $\leq_k$ STEINER TREE

Gegeben:  $M = \{u_1, \dots, u_n\}$ ,  $m$   
Teilmengen  $S_i \subseteq M$

EXACT-COVER Instanz ist Ja-  
Instanz  $\Leftrightarrow$  Graph hat Steinerbaum  
mit Gewicht kleiner gleich  $k$



# Der Approximationsalgorithmus von Kou, Markowsky und Berman





# Der Approximationsalgorithmus [KMB81]

Eingabe:  $G = (V, E, d), S \subseteq V$

1 Vollständiger Graph  $G_1 = (S, E_1, d_1)$   
 $d_1(\{u, v\}) = d(\text{ShortestPath}(G, u, v))$

2  $T_1 \leftarrow \text{MST}(G_1)$

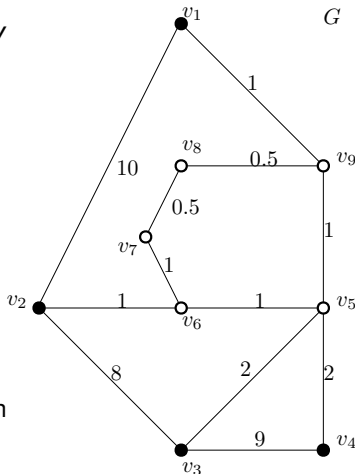
3 Packe kürzeste Wege in  $T_1$  aus  $\Rightarrow$   
erhalte  $G_S$

4  $T_S \leftarrow \text{MST}(G_S)$

5  $T_H \leftarrow T_S$  ohne unnötige Zweige

Ausgabe:  $T_H$

(wobei  $\text{MST}(G)$  einen minimalen Spannbaum in  
 $G$  berechnet)



# Der Approximationsalgorithmus [KMB81]

Eingabe:  $G = (V, E, d), S \subseteq V$

1 Vollständiger Graph  $G_1 = (S, E_1, d_1)$   
 $d_1(\{u, v\}) = d(\text{ShortestPath}(G, u, v))$

2  $T_1 \leftarrow \text{MST}(G_1)$

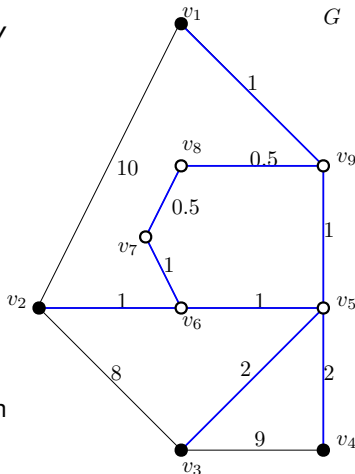
3 Packe kürzeste Wege in  $T_1$  aus  $\Rightarrow$   
erhalte  $G_S$

4  $T_S \leftarrow \text{MST}(G_S)$

5  $T_H \leftarrow T_S$  ohne unnötige Zweige

Ausgabe:  $T_H$

(wobei  $\text{MST}(G)$  einen minimalen Spannbaum in  
 $G$  berechnet)



# Der Approximationsalgorithmus [KMB81]

Eingabe:  $G = (V, E, d), S \subseteq V$

1 Vollständiger Graph  $G_1 = (S, E_1, d_1)$   
 $d_1(\{u, v\}) = d(\text{ShortestPath}(G, u, v))$

2  $T_1 \leftarrow \text{MST}(G_1)$

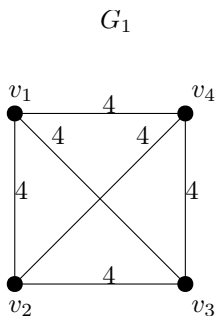
3 Packe kürzeste Wege in  $T_1$  aus  $\Rightarrow$   
erhalte  $G_S$

4  $T_S \leftarrow \text{MST}(G_S)$

5  $T_H \leftarrow T_S$  ohne unnötige Zweige

Ausgabe:  $T_H$

(wobei  $\text{MST}(G)$  einen minimalen Spannbaum in  
 $G$  berechnet)



# Der Approximationsalgorithmus [KMB81]

Eingabe:  $G = (V, E, d), S \subseteq V$

1 Vollständiger Graph  $G_1 = (S, E_1, d_1)$   
 $d_1(\{u, v\}) = d(\text{ShortestPath}(G, u, v))$

2  $T_1 \leftarrow \text{MST}(G_1)$

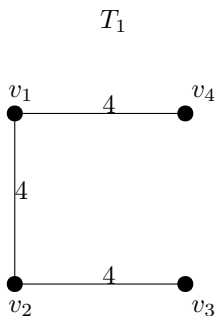
3 Packe kürzeste Wege in  $T_1$  aus  $\Rightarrow$   
erhalte  $G_S$

4  $T_S \leftarrow \text{MST}(G_S)$

5  $T_H \leftarrow T_S$  ohne unnötige Zweige

Ausgabe:  $T_H$

(wobei  $\text{MST}(G)$  einen minimalen Spannbaum in  
 $G$  berechnet)



# Der Approximationsalgorithmus [KMB81]

Eingabe:  $G = (V, E, d), S \subseteq V$

1 Vollständiger Graph  $G_1 = (S, E_1, d_1)$   
 $d_1(\{u, v\}) = d(\text{ShortestPath}(G, u, v))$

2  $T_1 \leftarrow \text{MST}(G_1)$

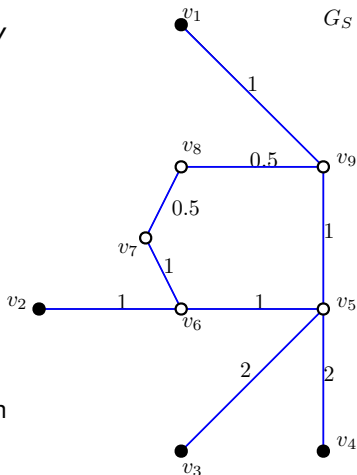
3 Packe kürzeste Wege in  $T_1$  aus  $\Rightarrow$   
erhalte  $G_S$

4  $T_S \leftarrow \text{MST}(G_S)$

5  $T_H \leftarrow T_S$  ohne unnötige Zweige

Ausgabe:  $T_H$

(wobei  $\text{MST}(G)$  einen minimalen Spannbaum in  
 $G$  berechnet)



# Der Approximationsalgorithmus [KMB81]

Eingabe:  $G = (V, E, d), S \subseteq V$

1 Vollständiger Graph  $G_1 = (S, E_1, d_1)$   
 $d_1(\{u, v\}) = d(\text{ShortestPath}(G, u, v))$

2  $T_1 \leftarrow \text{MST}(G_1)$

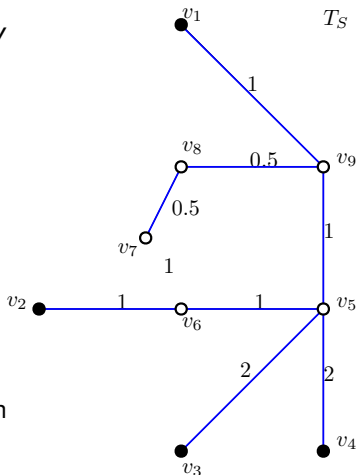
3 Packe kürzeste Wege in  $T_1$  aus  $\Rightarrow$   
erhalte  $G_S$

4  $T_S \leftarrow \text{MST}(G_S)$

5  $T_H \leftarrow T_S$  ohne unnötige Zweige

Ausgabe:  $T_H$

(wobei  $\text{MST}(G)$  einen minimalen Spannbaum in  
 $G$  berechnet)



# Der Approximationsalgorithmus [KMB81]

Eingabe:  $G = (V, E, d), S \subseteq V$

1 Vollständiger Graph  $G_1 = (S, E_1, d_1)$   
 $d_1(\{u, v\}) = d(\text{ShortestPath}(G, u, v))$

2  $T_1 \leftarrow \text{MST}(G_1)$

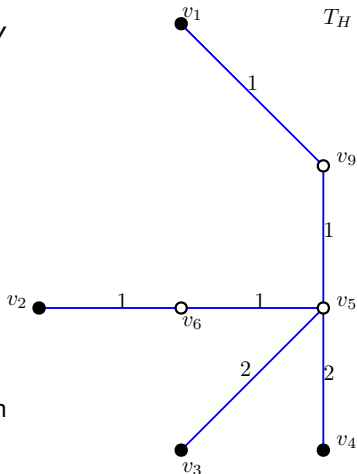
3 Packe kürzeste Wege in  $T_1$  aus  $\Rightarrow$   
erhalte  $G_S$

4  $T_S \leftarrow \text{MST}(G_S)$

5  $T_H \leftarrow T_S$  ohne unnötige Zweige

Ausgabe:  $T_H$

(wobei  $\text{MST}(G)$  einen minimalen Spannbaum in  
 $G$  berechnet)



# Der Approximationsalgorithmus [KMB81]

Eingabe:  $G = (V, E, d), S \subseteq V$

1 Vollständiger Graph  $G_1 = (S, E_1, d_1)$   
 $d_1(\{u, v\}) = d(\text{ShortestPath}(G, u, v))$

2  $T_1 \leftarrow \text{MST}(G_1)$

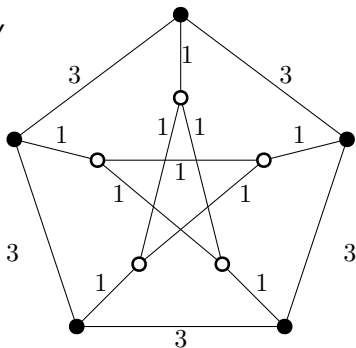
3 Packe kürzeste Wege in  $T_1$  aus  $\Rightarrow$   
erhalte  $G_S$

4  $T_S \leftarrow \text{MST}(G_S)$

5  $T_H \leftarrow T_S$  ohne unnötige Zweige

Ausgabe:  $T_H$

(wobei  $\text{MST}(G)$  einen minimalen Spannbaum in  
 $G$  berechnet)





# Der Approximationsalgorithmus [KMB81]

Eingabe:  $G = (V, E, d), S \subseteq V$

1 Vollständiger Graph  $G_1 = (S, E_1, d_1)$   
 $d_1(\{u, v\}) = d(\text{ShortestPath}(G, u, v))$

2  $T_1 \leftarrow \text{MST}(G_1)$

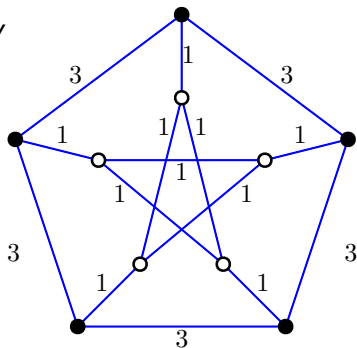
3 Packe kürzeste Wege in  $T_1$  aus  $\Rightarrow$   
erhalte  $G_S$

4  $T_S \leftarrow \text{MST}(G_S)$

5  $T_H \leftarrow T_S$  ohne unnötige Zweige

Ausgabe:  $T_H$

(wobei  $\text{MST}(G)$  einen minimalen Spannbaum in  
 $G$  berechnet)



# Der Approximationsalgorithmus [KMB81]

Eingabe:  $G = (V, E, d), S \subseteq V$

1 Vollständiger Graph  $G_1 = (S, E_1, d_1)$   
 $d_1(\{u, v\}) = d(\text{ShortestPath}(G, u, v))$

2  $T_1 \leftarrow \text{MST}(G_1)$

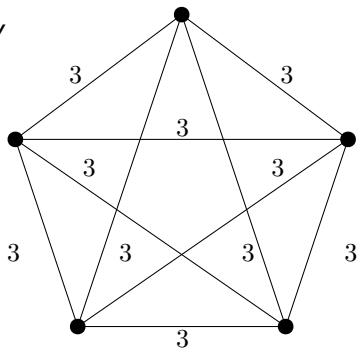
3 Packe kürzeste Wege in  $T_1$  aus  $\Rightarrow$   
erhalte  $G_S$

4  $T_S \leftarrow \text{MST}(G_S)$

5  $T_H \leftarrow T_S$  ohne unnötige Zweige

Ausgabe:  $T_H$

(wobei  $\text{MST}(G)$  einen minimalen Spannbaum in  
 $G$  berechnet)



# Der Approximationsalgorithmus [KMB81]

Eingabe:  $G = (V, E, d), S \subseteq V$

1 Vollständiger Graph  $G_1 = (S, E_1, d_1)$   
 $d_1(\{u, v\}) = d(\text{ShortestPath}(G, u, v))$

2  $T_1 \leftarrow \text{MST}(G_1)$

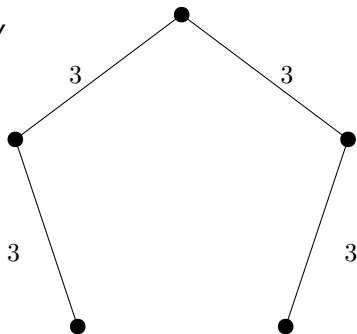
3 Packe kürzeste Wege in  $T_1$  aus  $\Rightarrow$   
erhalte  $G_S$

4  $T_S \leftarrow \text{MST}(G_S)$

5  $T_H \leftarrow T_S$  ohne unnötige Zweige

Ausgabe:  $T_H$

(wobei  $\text{MST}(G)$  einen minimalen Spannbaum in  
 $G$  berechnet)



# Der Approximationsalgorithmus [KMB81]

Eingabe:  $G = (V, E, d), S \subseteq V$

1 Vollständiger Graph  $G_1 = (S, E_1, d_1)$   
 $d_1(\{u, v\}) = d(\text{ShortestPath}(G, u, v))$

2  $T_1 \leftarrow \text{MST}(G_1)$

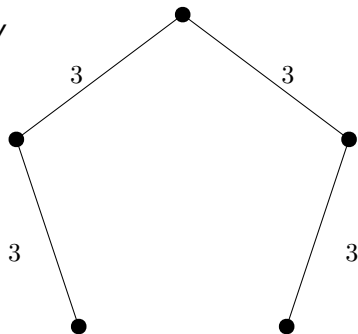
3 Packe kürzeste Wege in  $T_1$  aus  $\Rightarrow$   
erhalte  $G_S$

4  $T_S \leftarrow \text{MST}(G_S)$

5  $T_H \leftarrow T_S$  ohne unnötige Zweige

Ausgabe:  $T_H$

(wobei  $\text{MST}(G)$  einen minimalen Spannbaum in  $G$  berechnet)



# Der Approximationsalgorithmus [KMB81]

Eingabe:  $G = (V, E, d), S \subseteq V$

1 Vollständiger Graph  $G_1 = (S, E_1, d_1)$   
 $d_1(\{u, v\}) = d(\text{ShortestPath}(G, u, v))$

2  $T_1 \leftarrow \text{MST}(G_1)$

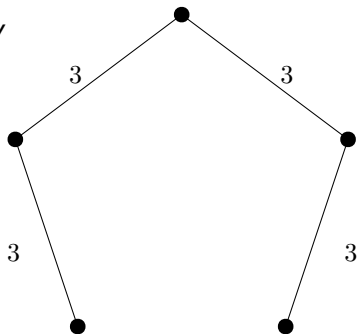
3 Packe kürzeste Wege in  $T_1$  aus  $\Rightarrow$   
erhalte  $G_S$

4  $T_S \leftarrow \text{MST}(G_S)$

5  $T_H \leftarrow T_S$  ohne unnötige Zweige

Ausgabe:  $T_H$

(wobei  $\text{MST}(G)$  einen minimalen Spannbaum in  
 $G$  berechnet)



# Der Approximationsalgorithmus [KMB81]

Eingabe:  $G = (V, E, d), S \subseteq V$

1 Vollständiger Graph  $G_1 = (S, E_1, d_1)$   
 $d_1(\{u, v\}) = d(\text{ShortestPath}(G, u, v))$

2  $T_1 \leftarrow \text{MST}(G_1)$

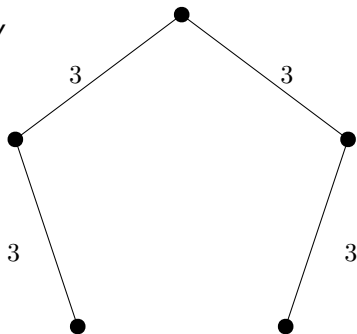
3 Packe kürzeste Wege in  $T_1$  aus  $\Rightarrow$   
erhalte  $G_S$

4  $T_S \leftarrow \text{MST}(G_S)$

5  $T_H \leftarrow T_S$  ohne unnötige Zweige

Ausgabe:  $T_H$

(wobei  $\text{MST}(G)$  einen minimalen Spannbaum in  $G$  berechnet)



# Der Approximationsalgorithmus [KMB81]

Eingabe:  $G = (V, E, d), S \subseteq V$

1 Vollständiger Graph  $G_1 = (S, E_1, d_1)$   
 $d_1(\{u, v\}) = d(\text{ShortestPath}(G, u, v))$

2  $T_1 \leftarrow \text{MST}(G_1)$

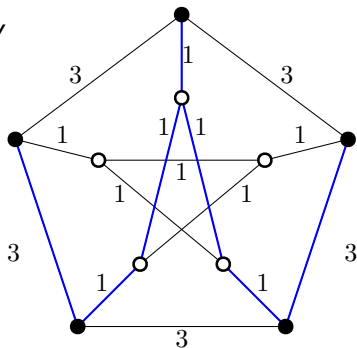
3 Packe kürzeste Wege in  $T_1$  aus  $\Rightarrow$   
erhalte  $G_S$

4  $T_S \leftarrow \text{MST}(G_S)$

5  $T_H \leftarrow T_S$  ohne unnötige Zweige

Ausgabe:  $T_H$

(wobei  $\text{MST}(G)$  einen minimalen Spannbaum in  
 $G$  berechnet)



- 1 Konstruktion von  $G_1$  aus kürzesten Wegen:  $\mathcal{O}(|S||V|^2)$
- 2 Minimaler Spannbaum in  $G_1$ :  $\mathcal{O}(|S|^2)$
- 3 Konstruktion von  $G_S$ :  $\mathcal{O}(|V|^2)$
- 4 Minimaler Spannbaum in  $G_S$ :  $\mathcal{O}(|V|^2)$
- 5 Unnötige Blätter entfernen:  $\mathcal{O}(|V|)$

---

Gesamt:  $\mathcal{O}(|S||V|^2)$



# Laufzeit von Schritt 3

Schritt 3: Konstruiere aus  $T_S$  den Graphen  $G_S$

→ Kanten durch kürzeste Wege ersetzen

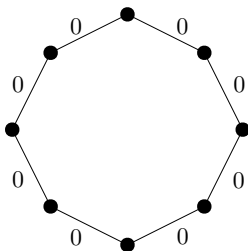
*Step 3 could be done in  $\mathcal{O}(|V|)$  time [KMB81]*

# Laufzeit von Schritt 3

Schritt 3: Konstruiere aus  $T_S$  den Graphen  $G_S$

→ Kanten durch kürzeste Wege ersetzen

*Step 3 could be done in  $\mathcal{O}(|V|)$  time [KMB81]*



- Kantengewichte dürfen 0 sein
- Wahl der kürzesten Wege ist beliebig

- $T_H$ : Steinerbaum, erzeugt vom Approximationsalgorithmus [KMB81]
- $D_H$ : Summe der Kantengewichte in  $T_H$
- $T_{MIN}$ : minimaler Steinerbaum
- $D_{MIN}$ : Summe der Kantengewichte in  $T_{MIN}$
- $b$ : Anzahl der Blätter in  $T_{MIN}$
- $d(P)$ : Länge eines einfachen Pfades/Zyklus  $P$

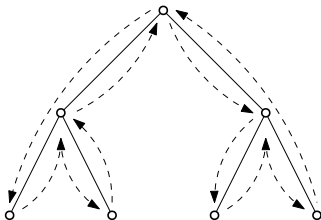
## Theorem

Sei mit  $G = (V, E, d)$  und  $S$  eine beliebige Eingabe für den Algorithmus [KMB81] gegeben. Dann gilt:  $D_H / D_{MIN} \leq 2(1 - \frac{1}{b})$

## Lemma

Sei  $T$  ein Baum mit  $m \geq 1$  Kanten. Dann gibt es einen Zyklus  $C = (u_0, \dots, u_{2m})$  in  $T$  wobei gilt:

- i jede Kante aus  $T$  kommt genau zwei mal in  $C$  vor
- ii jedes Blatt aus  $T$  kommt genau einmal in  $C$  vor
- iii wenn  $u_i$  und  $u_j$  Blätter in  $C$  sind und kein anderes Blatt dazwischen ist, ist  $(u_i, u_{i+1}, \dots, u_j)$  ein einfacher Pfad.



# Baumtraversierung - Beweis des Lemmas

*Beweis.* (Induktion über  $m$ )

*I.A.*  $m = 1$  und  $T$  ist ein Baum  $\Rightarrow T$  besteht aus zwei Knoten  $V = \{u, v\} \Rightarrow C = (u, v, u)$  leistet *i, ii, iii* ( $u$  Wurzel)



# Baumtraversierung - Beweis des Lemmas

*I.V.* Es gelte die Behauptung für  $m = k \geq 1$

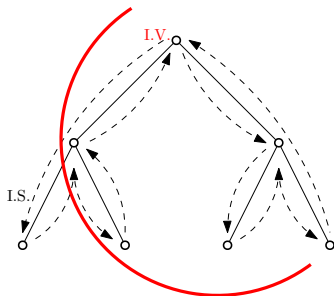
*I.S.* Sei  $m = k + 1$ .

Sei  $v_p$  ein Blatt. Wende auf *I.V.* auf

$G - v_p := (V \setminus \{v_p\}, \{e \in E \mid v_p \notin e\})$  an

$\Rightarrow \exists$  Zyklus  $C = (u'_0, \dots, u'_{2k})$ ,

der *i, ii, iii* in  $G - v_p$  erfüllt. Sei  $v_q$  der Elternnoten von  $v_p$ . Dann ersetze in  $C$  das erste  $v_q$  durch  $v_q, v_p, v_q$ . □



# Baumtraversierung - Beweis des Lemmas

*I.V.* Es gelte die Behauptung für  $m = k \geq 1$

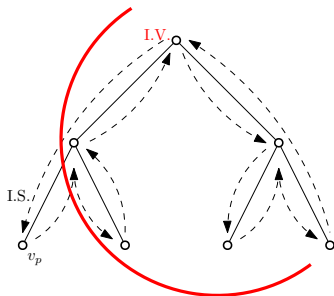
*I.S.* Sei  $m = k + 1$ .

Sei  $v_p$  ein Blatt. Wende auf *I.V.* auf

$G - v_p := (V \setminus \{v_p\}, \{e \in E \mid v_p \notin e\})$  an

$\Rightarrow \exists$  Zyklus  $C = (u'_0, \dots, u'_{2k})$ ,

der *i, ii, iii* in  $G - v_p$  erfüllt. Sei  $v_q$  der Elternnoten von  $v_p$ . Dann ersetze in  $C$  das erste  $v_q$  durch  $v_q, v_p, v_q$ . □





# Baumtraversierung - Beweis des Lemmas

*I.V.* Es gelte die Behauptung für  $m = k \geq 1$

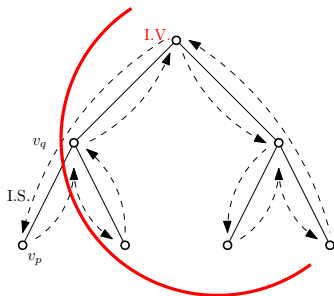
*I.S.* Sei  $m = k + 1$ .

Sei  $v_p$  ein Blatt. Wende auf *I.V.* auf

$G - v_p := (V \setminus \{v_p\}, \{e \in E \mid v_p \notin e\})$  an

$\Rightarrow \exists$  Zyklus  $C = (u'_0, \dots, u'_{2k})$ ,

der *i*, *ii*, *iii* in  $G - v_p$  erfüllt. Sei  $v_q$  der Elternknoten von  $v_p$ . Dann ersetze in  $C$  das erste  $v_q$  durch  $v_q, v_p, v_q$ . □



## Theorem

Sei mit  $G = (V, E, d)$  und  $S$  eine beliebige Eingabe für den Algorithmus [KMB81] gegeben. Dann gilt:  $D_H/D_{MIN} \leq 2(1 - \frac{1}{b})$

*Beweis.* Wende Lemma auf  $T_{MIN}$  an  $\Rightarrow$  liefert Zyklus  $L$  in  $T_{MIN}$  mit

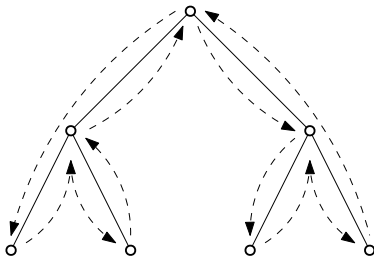
- i jede Kante von  $T_{MIN}$  genau zweimal in  $L \Rightarrow d(L) = 2 \cdot D_{MIN}$
- ii jedes Blatt aus  $T_{MIN}$  genau einmal in  $L$
- iii wenn  $u_i$  und  $u_j$  Blätter in  $L$  sind und kein anderes Blatt dazwischen ist, ist  $(u_i, u_{i+1}, \dots, u_j)$  ein einfacher Pfad.

## Theorem

Sei mit  $G = (V, E, d)$  und  $S$  eine beliebige Eingabe für den Algorithmus [KMB81] gegeben. Dann gilt:  $D_H/D_{MIN} \leq 2(1 - \frac{1}{b})$

*Beweis.* Wende Lemma auf  $T_{MIN}$  an  $\Rightarrow$  liefert Zyklus  $L$  in  $T_{MIN}$  mit

- i jede Kante von  $T_{MIN}$  genau zweimal in  $L \Rightarrow d(L) = 2 \cdot D_{MIN}$
- ii jedes Blatt aus  $T_{MIN}$  genau einmal in  $L$
- iii wenn  $u_i$  und  $u_j$  Blätter in  $L$  sind und kein anderes Blatt dazwischen ist, ist  $(u_i, u_{i+1}, \dots, u_j)$  ein einfacher Pfad.



Seien  $P_1, \dots, P_b$  die Pfade, die die Blätter verbinden.

Sei  $P_{MAX} := \operatorname{argmax}_{i=1}^b d(P_i) \Rightarrow d(P_{MAX}) \geq \frac{1}{b}d(L)$

Sei  $P := L - P_{MAX}$  der Zyklus  $L$  ohne Kanten aus  $P_{MAX}$

$$\Rightarrow d(P) \leq \left(1 - \frac{1}{b}\right)d(L) = \left(1 - \frac{1}{b}\right) \cdot 2 \cdot D_{MIN}$$

$$d(P) \leq 2 \cdot \left(1 - \frac{1}{b}\right) \cdot D_{MIN} \quad (1)$$

# Relative Gütegarantie - Beweis

$$d(P) \leq 2 \cdot \left(1 - \frac{1}{b}\right) \cdot D_{MIN} \quad (1)$$

$$D_H \leq d(MST(G_1))$$

(Schritte 3-5 verlängern die Distanz nicht)

Seien  $w_1, w_2, \dots, w_k$  die Steinerpunkte,  
in dieser Reihenfolge in  $P$  vorkommend

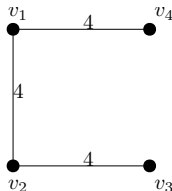
Sei  $\mathcal{T}$  ein Spannbaum

in  $G_1$ , der genau  $\{w_1, w_2\}, \dots, \{w_{k-1}, w_k\}$  enthält.

$$d(MST(G_1)) \leq d(\mathcal{T})$$

Jede Kante aus  $T_{MIN}$  mindestens einmal in  $P \Rightarrow$

$$d(\mathcal{T}) \leq d(P)$$



Es folgt:  $D_H \leq d(P)$  (2) (1),(2)  $\Rightarrow$  Beh.  $\square$

# Relative Gütegarantie - Beweis

$$d(P) \leq 2 \cdot \left(1 - \frac{1}{b}\right) \cdot D_{MIN} \quad (1)$$

$$D_H \leq d(MST(G_1))$$

(Schritte 3-5 verlängern die Distanz nicht)

Seien  $w_1, w_2, \dots, w_k$  die Steinerpunkte,  
in dieser Reihenfolge in  $P$  vorkommend

Sei  $\mathcal{T}$  ein Spannbaum

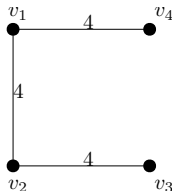
in  $G_1$ , der genau  $\{w_1, w_2\}, \dots, \{w_{k-1}, w_k\}$  enthält.

$$d(MST(G_1)) \leq d(\mathcal{T})$$

Jede Kante aus  $T_{MIN}$  mindestens einmal in  $P \Rightarrow$

$$d(\mathcal{T}) \leq d(P)$$

Es folgt:  $D_H \leq d(P)$  (2) (1),(2)  $\Rightarrow$  Beh.  $\square$



# Relative Gütegarantie - Beweis

$$d(P) \leq 2 \cdot \left(1 - \frac{1}{b}\right) \cdot D_{MIN} \quad (1)$$

$$D_H \leq d(MST(G_1))$$

(Schritte 3-5 verlängern die Distanz nicht)

Seien  $w_1, w_2, \dots, w_k$  die Steinerpunkte,  
in dieser Reihenfolge in  $P$  vorkommend

Sei  $\mathcal{T}$  ein Spannbaum

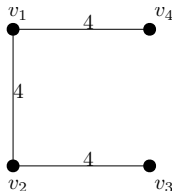
in  $G_1$ , der genau  $\{w_1, w_2\}, \dots, \{w_{k-1}, w_k\}$  enthält.

$$d(MST(G_1)) \leq d(\mathcal{T})$$

Jede Kante aus  $T_{MIN}$  mindestens einmal in  $P \Rightarrow$

$$d(\mathcal{T}) \leq d(P)$$

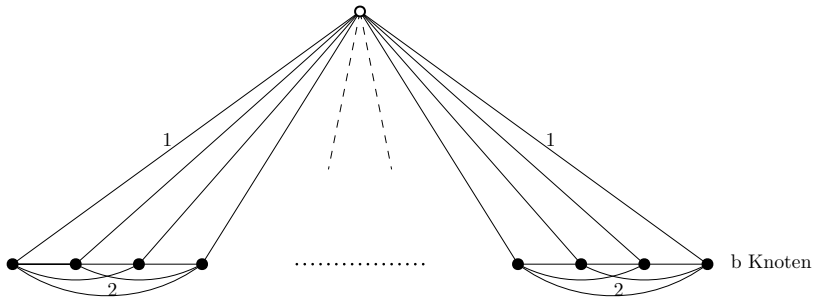
Es folgt:  $D_H \leq d(P)$  (2) (1),(2)  $\Rightarrow$  Beh.  $\square$



## Theorem

Sei  $b \geq 2$ . Dann gibt es einen Graphen  $G = (V, E, d)$  und Steinerpunkte  $S \subseteq V$  sodass  $T_{MIN}$   $b$  Blätter hat und im Worst-Case  $D_H/D_{MIN} = 2(1 - \frac{1}{b})$

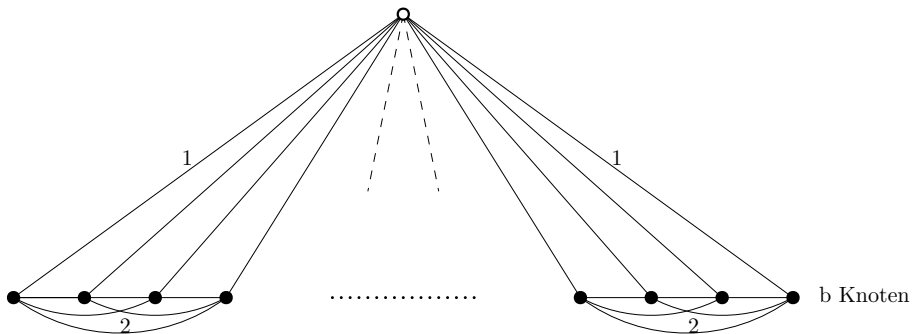
Beweis. Für  $b \geq 2$  betrachte:





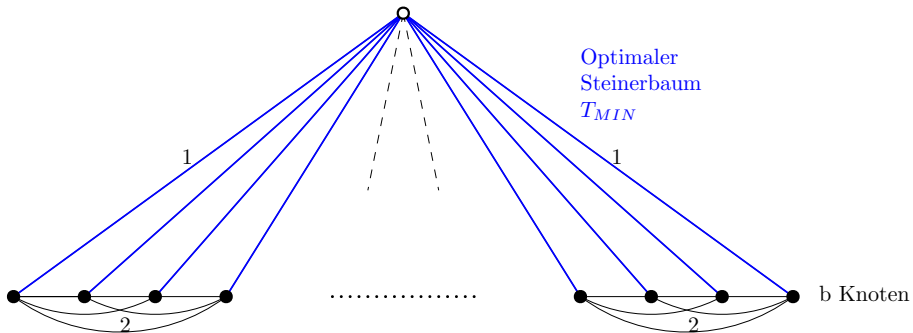
vollständiger Graph  $G := (V, E, d)$   $V := \{v_1, v_2, \dots, v_{b+1}\}$

$$d(\{v_i, v_j\}) = \begin{cases} 1, & v_i = v_{b+1} \vee v_j = v_{b+1} \\ 2, & \text{sonst} \end{cases}, S := V \setminus \{v_{b+1}\}$$



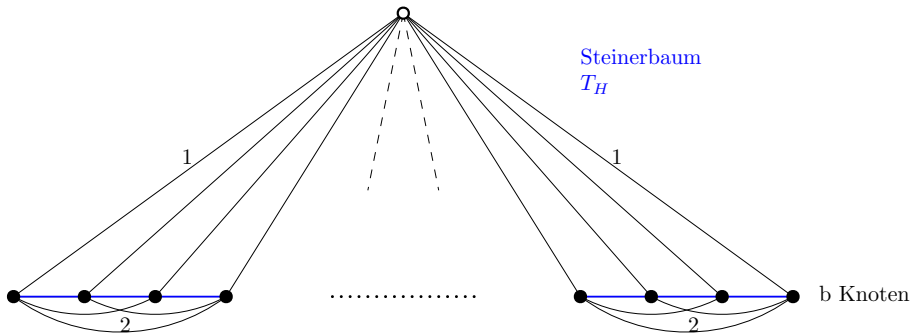
vollständiger Graph  $G := (V, E, d)$   $V := \{v_1, v_2, \dots, v_{b+1}\}$

$$d(\{v_i, v_j\}) = \begin{cases} 1, & v_i = v_{b+1} \vee v_j = v_{b+1} \\ 2, & \text{sonst} \end{cases}, S := V \setminus \{v_{b+1}\}$$

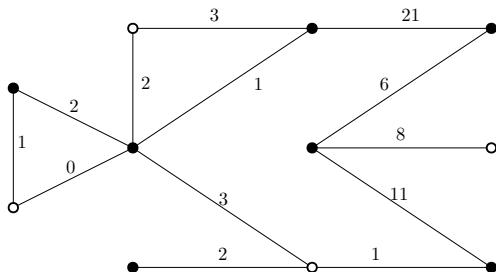


vollständiger Graph  $G := (V, E, d)$   $V := \{v_1, v_2, \dots, v_{b+1}\}$




$$d(\{v_i, v_j\}) = \begin{cases} 1, & v_i = v_{b+1} \vee v_j = v_{b+1} \\ 2, & \text{sonst} \end{cases}, S := V \setminus \{v_{b+1}\}$$





$$D_{MIN} = b \text{ und } D_H = 2(b - 1) \Rightarrow D_H/D_{MIN} = 2(b - 1)/b = 2(1 - \frac{1}{b}) \quad \square$$



- Steiner-Baum-Problem ist  $\mathcal{NP}$ -schwer
- Algorithmus, der Shortest Path und Minimum Spanning Tree nutzt
- Laufzeit:  $\mathcal{O}(|S||V|^2)$
- Garantie:  $D_H/D_{MIN} \leq 2(1 - \frac{1}{b})$

-  Clemens Gröpl, Stefan Hougardy, Till Nierhoff, and Hans Jürgen Prömel.  
Lower bounds for approximation algorithms for the steiner tree problem.  
*Lecture Notes in Computer Science*, page 217, 1973.
-  Richard M. Karp.  
Reducibility Among Combinatorial Problems.  
*R. E. Miller und J. W. Thatcher (Hrsg.): Complexity of Computer Computations*, pages 85–103, 1972.
-  L. Kou, G. Markowsky, and L. Berman.  
A Fast Algorithm for Steiner Trees.  
*Acta Informatica*, 15(2):141–145, Jun 1981.

-  K. Mehlhorn.  
A faster approximation algorithm for the steiner problem in graphs.  
*Information Processing Letters*, pages 125–128, 1988.
-  G. Robins and A. Zelikovsky.  
Improved steiner tree approximation in graphs.  
*Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms.*, pages 770–779, 2000.