Effiziente Algorithmen (SS2015)

Kapitel 6
Approximation I

Walter Unger

Lehrstuhl für Informatik 1

14:07 Uhr, den 19. Dezember 2018

Inhalt I

Einleitung

- NP-schwere Probleme doch lösen
  - Exakt: "$\Longrightarrow$" exponentiale Laufzeit
  - Nicht Exakt: "$\Longrightarrow$" hoffentlich polynomiale Laufzeit
- Hier: Approximationsalgorithmen in polynomialer Laufzeit.
- D.h.: welche Approximationsfaktoren sind möglich?
- Kommt man beliebig nah an das Optimum?

Einfaches Beispiel (Knotenfärbung)

$\mathbb{N}_k = \{\mathbf{1}, \ldots, k\}$

Definition (Knotenfärbung)

Sei $G = (V, E)$ ungerichteter Graph. Dann wird definiert:

- k-Färbungsfunktion $c : V \mapsto \mathbb{N}_k : \forall\{v, w\} \in E : c(v) \neq c(w)$.
- Färbungszahl
  $\chi(G) = \min_{k \in \mathbb{N}}\{k \mid \exists c : V \mapsto \mathbb{N}_k : \forall\{v, w\} \in E : c(v) \neq c(w)\}$
- Färbungsproblem:

  COL: Gegeben: $G = (V, E), k$, Frage: Gilt $\chi(G) \leqslant k$
- k-Färbungsproblem:

  k-COL: Gegeben: $G = (V, E)$, Frage: Gilt $\chi(G) \leqslant k$
- praktisches Färbungsproblem:
  - Gegeben: $G = (V, E)$
  - Bestimme: $c : V \mapsto \mathbb{N}_{\chi(G)} : \forall\{v, w\} \in E : c(v) \neq c(w)$.

Einfaches Beispiel (Knotenfärbung)

$\mathbb{N}_k = \{\mathbf{1}, \ldots, k\}$

Definition (Planarer Graph)

$G = (V, E)$ heißt planar, wenn er kreuzungsfrei in die Ebene eingebettet werden kann.

Theorem (Färbung Planarer Graphen)

Für planare Graphen G gilt:

- $\chi(G) \leqslant 4$.
- $3\text{-}COL(G) \in \mathcal{NPC}$.

Folgerung für das Färben von planaren Graphen:

- kann mit additivem Fehler von 1 gelöst werden.
- kann mit multipikativem Fehler von 4/3 gelöst werden.

Einfaches Beispiel (Kantenfärbung)

$\mathbb{N}_k = \{1, \ldots, k\}$

Definition (Kantenfärbung)

Die Kantenfärbung auf G entspricht der Knotenfärbung auf $L(G)$:

$$\chi'(G) = \chi(L(G)).$$

Theorem (Kantenfärbung)

Für Graphen G gilt:

- $\delta(G) \leqslant \chi'(G) \leqslant \delta(G) + 1.$
- $(\delta(G)) - EDGECOL(G) \in \mathcal{NPC}.$

Folgerung für die Kantenfärbung:

- kann mit additivem Fehler von 1 gelöst werden.
- kann mit multipikativem Fehler von $(\delta(G) + 1)/\delta(G)$ gelöst werden.

Definition (Konstante Approximation)

$\mathbb{N}_k = \{1, \ldots, k\}$

Definition

Ein Algorithmus A hat einen additiven Approximationfehler, falls für alle Eingabeinstanzen I gilt:

- $opt(I) \leqslant A(I) + k$ (Maximierungsproblem) oder
- $opt(I) \geqslant A(I) - k$ (Minimierungsproblem)

Definition

Ein Algorithmus A hat einen multiplikativen Approximationfehler, falls gilt:

- $\forall I : \frac{A(I)}{opt(I)} \leqslant \alpha$ und $\alpha \geqslant 1$ bei einem Minimierungsproblem und
- $\forall I : \frac{A(I)}{opt(I)} \geqslant \alpha$ und $\alpha \leqslant 1$ bei einem Maximierungsproblem.

Oft wird vereinfacht: $\max\{\frac{A(I)}{opt(I)}, \frac{opt(I)}{A(I)}\} \leqslant \alpha$.

Definition (Approximation)

$\mathbb{N}_k = \{1, \ldots, k\}$

Definition

Sei $L : \mathbb{N} \mapsto \mathbb{R}$ eine Funktion und seien weiter I_n die Eingaben für Algorithmus A. Dann hat A einen Approximationsfaktor, falls gilt:

- $\forall n \in \mathbb{N} : \forall I \in I_n : \frac{A(I)}{opt(I)} \leqslant L(n)$ bei einem Minimierungsproblem und
- $\forall n \in \mathbb{N} : \forall I \in I_n : \frac{opt(I)}{A(I)} \leqslant L(n)$ bei einem Maximierungsproblem.

Kreuzprodukt von Graphen

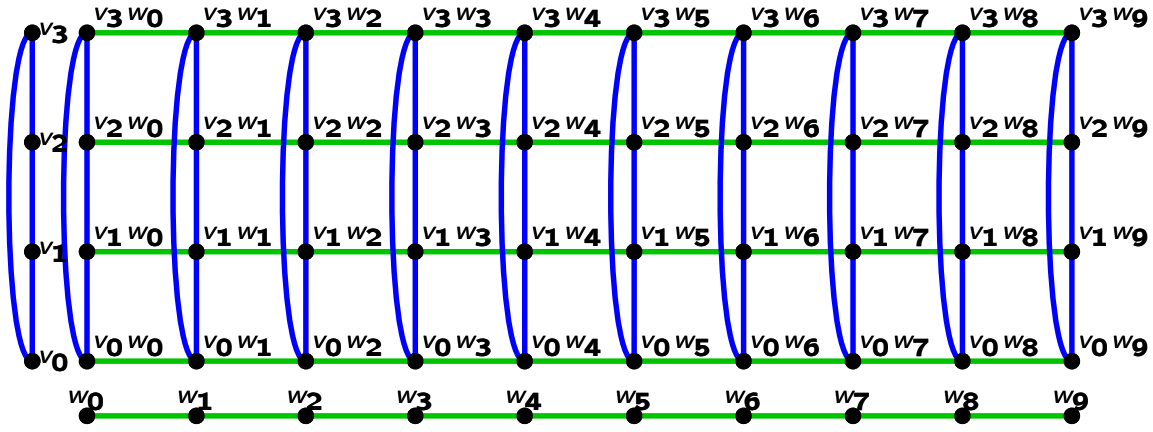
$\omega(G) = \max\{|C| \mid C \subset V \wedge \forall v, w \in C : \{v, w\} \in E\}$

Definition:

Seien $G = (V, E)$ und $G' = (V', E')$ zwei Graphen. Dann wird mit $G \times G'$ das Kreuzprodukt von G und G' bezeichnet:

- $G \times G' = (V \times V', E_1 \cup E_2)$.
- $E_1 = \{((a, a'), (b, b')) \mid a' = b' \wedge (a, b) \in E\}$.
- $E_2 = \{((a, a'), (b, b')) \mid a = b \wedge (a', b') \in E'\}$.

Beispiel $L(10) \times C(4)$:



Kreuzprodukt von Graphen

$\omega(G) = \max\{|C| \mid C \subset V \wedge \forall v, w \in C : \{v, w\} \in E\}$

Definition:

Seien $G = (V, E)$ und $G' = (V', E')$ zwei Graphen. Dann wird mit $G \times G'$ das Kreuzprodukt von G und G' bezeichnet:

- $G \times G' = (V \times V', E_1 \cup E_2)$.
- $E_1 = \{((a, a'), (b, b')) \mid a' = b' \wedge (a, b) \in E\}$.
- $E_2 = \{((a, a'), (b, b')) \mid a = b \wedge (a', b') \in E'\}$.

Cliquenproblem

$\omega(G) = \max\{|C| \mid C \subset V \wedge \forall v, w \in C : \{v, w\} \in E\}$

Theorem

Falls $\mathcal{P} \neq \mathcal{NP}$, gibt es für beliebiges $k \in \mathbb{N}$ keinen Polynomzeitalgorithmus mit Approximationsfehler k für das Cliquenproblem.

- Zeigen hier nur additiven Fehler.
- Sei G ein Graph und sei $G^k = G \times C_k$.
- Damit: $\omega(G) \cdot k = \omega(G^k)$.
- Angenommen: es gibt einen Algorithmus A mit additiven Approximationsfehler k für das Cliquenproblem.
- Eingabe für A wird G^{k+1}.
- Damit gilt: $opt(I) \leqslant A(I) + k$ und weiter:
- $\omega(G^{k+1}) - A(G^{k+1}) \leqslant k$.
- $(k+1) \cdot \omega(G) - A(G^{k+1}) \leqslant k$.

Cliquenproblem

$\omega(G) = \max\{|C| \mid C \subset V \wedge \forall v, w \in C : \{v, w\} \in E\}$

Theorem

Falls $\mathcal{P} \neq \mathcal{NP}$, gibt es für beliebiges $k \in \mathbb{N}$ keinen Polynomzeitalgorithmus mit Approximationsfehler k für das Cliquenproblem.

- $(k+1) \cdot \omega(G) - A(G^{k+1}) \leqslant k$.
- Sei G_i $(1 \leqslant i \leqslant k+1)$ die Kopien von G in G^{k+1}.
- Seien C_i die Lösungsanteile in G_i.
- $\forall i : 1 \leqslant i \leqslant k+1 : C_i \leqslant \omega(G)$
- $(k+1) \cdot \omega(G) - \sum_{i=1}^{k+1} |C_i| \leqslant k$.
- $\sum_{i=1}^{k+1} \omega(G) - |C_i| \leqslant k$.
- $\exists i : C_i = \omega(G)$.
- Damit ist das Cliquenproblem in $\mathcal{P}$.
- Kann auf multiplikativen Fehler erweitert werden.

Einleitung

Definition

Sei $G = (V, E)$. $C \subset V$ heißt Vertex Cover von G, falls

$$\forall e \in E : C \cap e \neq \emptyset.$$

- Das folgende Problem ist in $\mathcal{NPC}$:
  - Gegeben: G, k.
  - Frage: Gibt es ein Vertex Cover C in G mit $|C| \leqslant k$.
- Reduktionsidee: $V \setminus C$ ist eine stabile Menge.
- Versuche nun mittels Greedy das Vertex Cover Problem zu approximieren.
- Strategie: Wähle Knoten vom höchsten Grad.
- Betrachte dazu das folgende Beispiel:

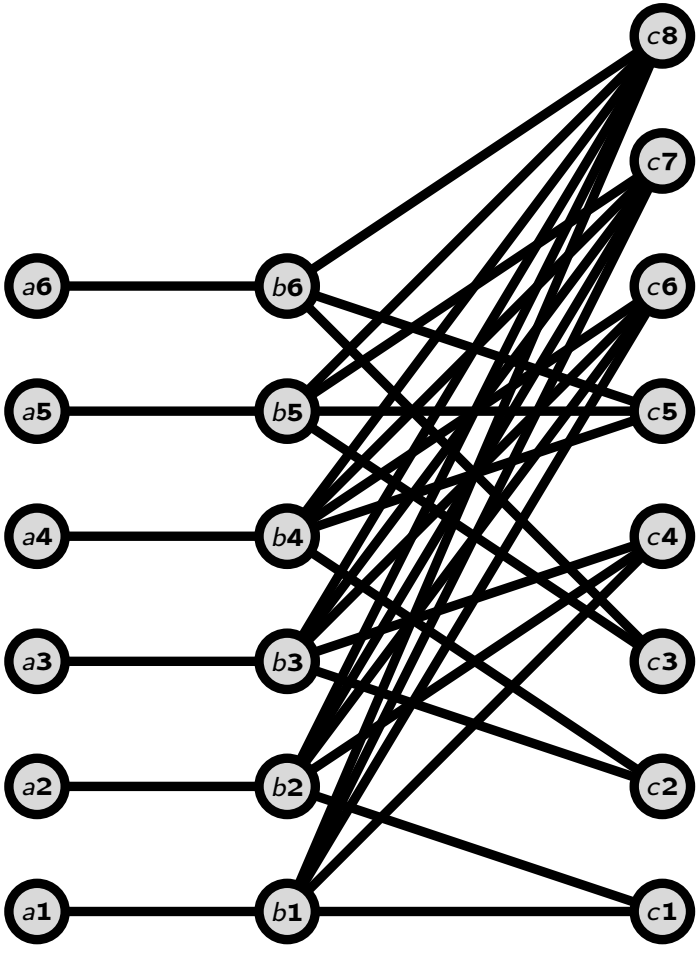
Ideen

Beim Vertex Cover erscheint ein Greedy-Verfahren, welches Knoten vom großen Knotengrad bevorzugt, da dann viele Kanten überdeckt werden, sinnvoll. Aber das muss nicht immer sinnvoll sein. Dazu konstruieren wir einen tripatiten Graphen, bei dem es nur Kanten zwischen der Knotenmenge A und B gibt, sowie zwichen B und C. Das optimale Vertex Cover wird aus der Menge B bestehen. Es werden möglichst viele Knoten vom grossen Grad in C sein. Der Greedy Algorithmus wird dann o.E.d.A. die Mengen C und A auswählen und einen grossen Fehler machen.

Dieses Beispiel wird auch zeigen, wie man eine bessere Approximation erreicht. Die beiden Mengen A und B können perfekt gematcht werden. Da ein Matching eine unabhängige Menge von Kanten ist, muss für jede Kante ein Knoten im Vertex Cover sein. Damit erhält man eine einfache 2-Approximation für das Vertex Cover Problem.

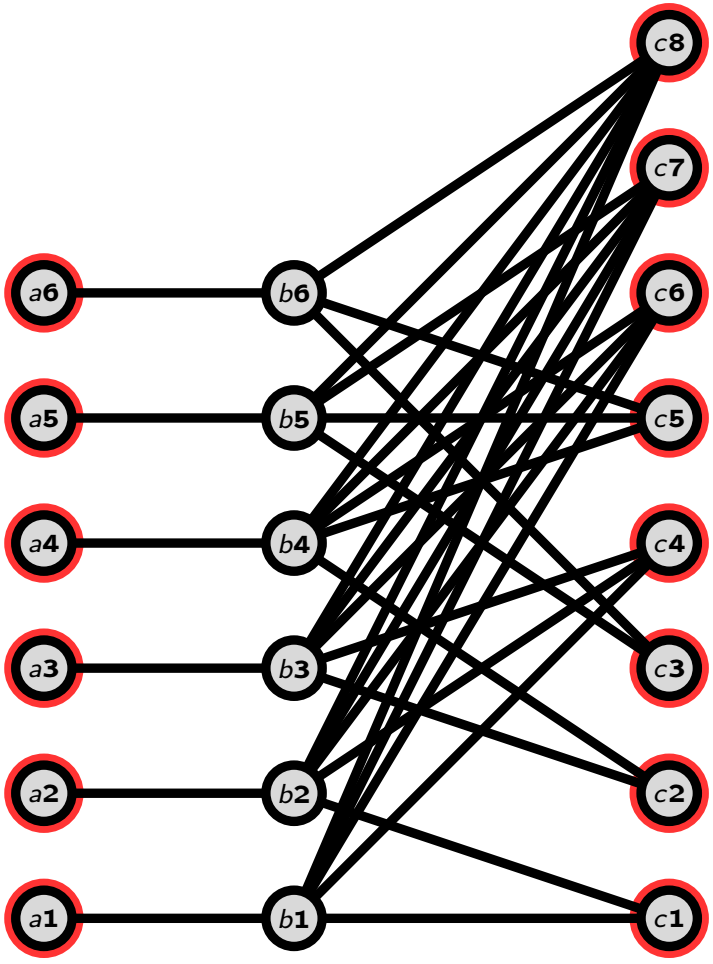
Gleichzeitig erkennt man: Zum Beweis der Korrektheit benötigt man eine untere Schranke (hier das maximale Matching). Hier ist diese auch effizient zu berechnen und dient als Grundlage für die algorithmische Approximation.

Vertex Cover (Greedy)



- $G_k = (A_k \cup B_k \cup C_{k'}, \cup_{i=1}^{k} E_i)$
- $A_k = \{a_1, a_2, \ldots, a_k\}$
- $B_k = \{b_1, b_2, \ldots, b_k\}$
- $C_k = \{c_1, c_2, \ldots, c_{k'}\}$ mit $k' = \sum_{j=2}^{k} \lfloor k/j \rfloor$
- $E_1 = \{\{a_i, b_i\} \mid 1 \leqslant i \leqslant k\}$
- $k_i = \sum_{j=2}^{i} \lfloor k/j \rfloor$ für $i = 2, 3, \ldots, k$
- $k_1 = 0$
- $E_i = \{\{c_{k_{i-1}+j}, b_{i*(j-1)+x}\} \mid 1 \leqslant j \leqslant \lfloor k/i \rfloor \wedge 1 \leqslant x \leqslant i\}$ für $i = 2, 3, \ldots, k$
- In E_i sind $\lfloor k/i \rfloor$ Knoten aus C_k beteiligt.

Vertex Cover (Greedy)



- Wähle Knoten in folgender Reihenfolge:
  - c_8 (Grad 6)
  - c_7 (Grad 5)
  - c_6 (Grad 4)
  - c_4, c_5 (Grad 3)
  - c_1, c_2, c_3 (Grad 2)
  - $a_1, a_2, \ldots, a_6$ (Grad 1)

Vertex Cover (Greedy)

- Wir bestimmen nun den Approximationsfaktor $F(n)$.

$$\begin{aligned}
F(n) &\geqslant \frac{|C_{greedy}(G_k)|}{|C_{opt}(G_k)|} \\
&= \frac{1}{k} \cdot \left(k + \sum_{i=2}^{k} \left\lfloor \frac{k}{i} \right\rfloor\right) \\
&= \frac{1}{k} \cdot \sum_{i=1}^{k} \left\lfloor \frac{k}{i} \right\rfloor \\
&\geqslant \frac{1}{k} \cdot \left(\sum_{i=1}^{k} \frac{k}{i} - (k-2)\right) \\
&\geqslant \sum_{i=1}^{k} \frac{1}{i} - 1 \\
&\geqslant \int_{i=1}^{k+1} \frac{dx}{x} - 1 \\
&\geqslant \ln k - 1 \\
&\geqslant \ln \Delta(G_k) - 1
\end{aligned}$$

- Damit kann dieser Greedyalgorithmus keinen konstanten Approximationsfaktor haben.

Aufbau der Idee

- Jede Kante muss überdeckt werden.
- Für die Abschätzung des Faktors brauchen wir eine untere Schranke.
- Für zwei unabhängige Kanten braucht man zwei Knoten zur Überdeckung.
- Für k paarweise unabhängige Kanten braucht man k Knoten zur Überdeckung.
- Man weiß aber nicht, welcher Knoten von diesen k Kanten im Cover ist.
- Idee: Wähle beide für einen Approximationsfaktor von zwei.

Vertex Cover

Theorem

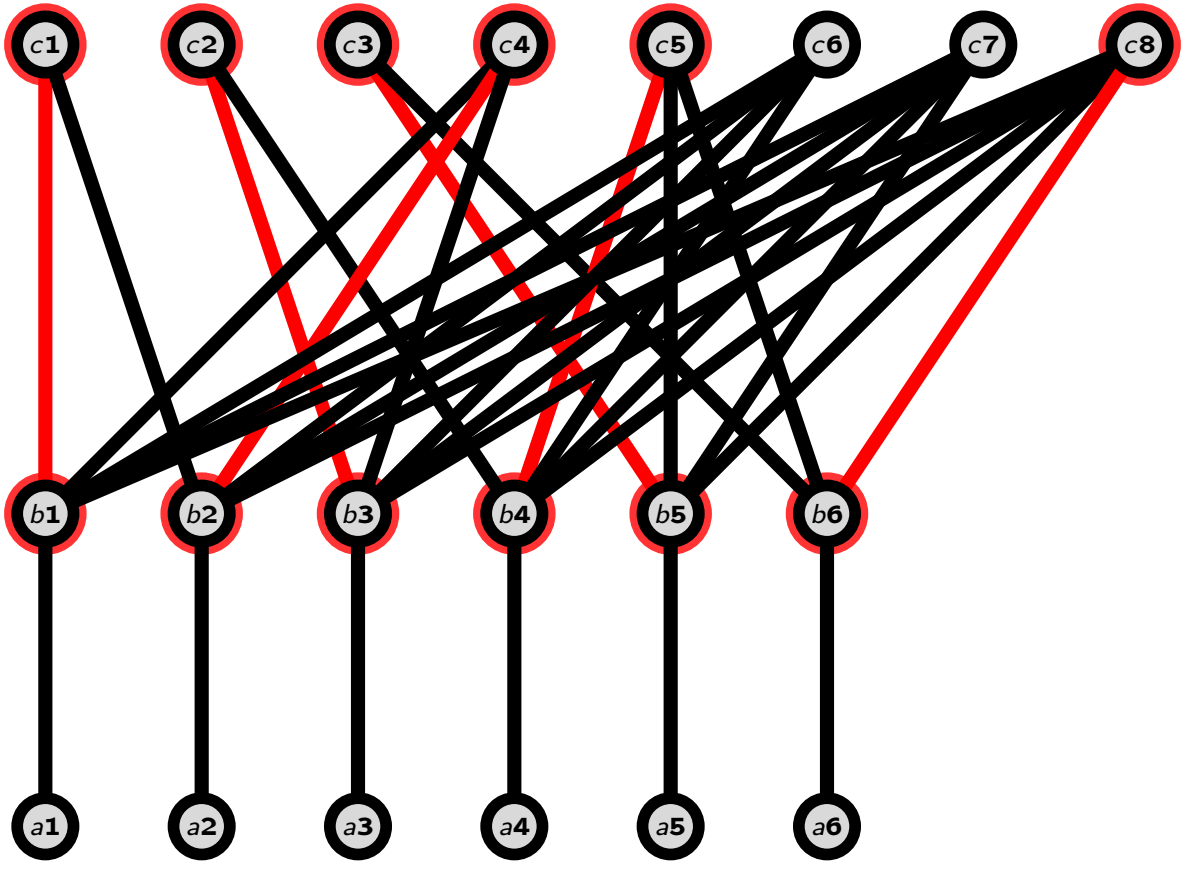
Das Vertex Cover Problem kann mit einem Faktor von 2 approximiert werden.

1. Bestimme inklusions-maximales Matching $M \subset E$ auf Eingabegraph $G = (V, E)$.
2. Wähle Vertex Cover $C = \cup_{e \in M} e$ und gib C aus.

- Zeige Korrektheit:
  - Angenommen $C = \cup_{e \in M} e$ ist kein Vertex Cover, d.h. $\exists e : e \cap C = \emptyset$.
  - Damit ist M kein inklusions-maximales Matching, Widerspruch.
- Zeige Approximationsfaktor: Setze $\tau(G) = |V| - \alpha(G)$ und
  - $M_{\max}(G)$ Maximales Matching von G.
  - Schätze Approximationsfaktor ab:

$$\begin{aligned} \frac{2 \cdot |M|}{\tau(G)} &\leqslant \frac{2 \cdot |M_{\max}(G)|}{\tau(G)} && \text{beachte: } |M_{\max}(G)| \geqslant |M| \\ &\leqslant \frac{2 \cdot |M_{\max}(G)|}{|M_{\max}(G)|} && \text{beachte: } |M_{\max}(G)| \leqslant \tau(G) \\ &= 2 \end{aligned}$$

Beispiel



Vertex Cover

Theorem (Monien)

Das Vertex Cover Problem kann mit einem Faktor von $2 - \frac{\log \log n}{2 \cdot \log n}$ *approximiert werden.*

- Zusammenfassung:
- Vertex Cover ist mit Faktor 2 approximierbar.
- Clique ist mit keinem konstanten Faktor approximierbar.
- Independent Set ist mit keinem konstanten Faktor approximierbar.
- $\mathcal{NP}$-vollständige Probleme unterscheiden sich in ihrer Approximierbarkeit.

TSP

Definition (TSP)

Gegeben $G = (V, E)$, $L \in \mathbb{Q}$ und $c : E \mapsto \mathbb{Q}$.
Bestimme spannenden einfachen Kreis C mit $c(C) \leqslant L$.
D.h. ein Kreis, der jeden Knoten genau einmal besucht.

- Problem ist in $\mathcal{NPC}$ (Reduktion von Hamilton Kreis).
  - $G = (V, E)$ Eingabe für Hamilton Kreis.
  - Eingabe für TSP ist Clique $C = (V, E')$ mit:
  - $c(e) = 1$ falls $e \in E$ und
  - $c(e) = 2$ falls $e \notin E$.
  - Falls G einen Hamiltonkreis hat, dann G' Tour mit Kosten n.
  - Anderenfalls hat G' eine Tour mit Kosten $\geqslant n + 1$.

TSP

Theorem

Sei $\alpha(n)$ eine polynomialzeit-berechenbare Funktion, $G = (V, E)$ und $n = |V|$. Falls $\mathcal{P} \neq \mathcal{NP}$, gilt: TSP kann nicht in Polynomialzeit mit Faktor $\alpha(n)$ approximiert werden.

- Problem ist in $\mathcal{NPC}$ (Reduktion von Hamilton Kreis).
- Gleiche Reduktion zeigt obigen Satz:
  - $G = (V, E)$ Eingabe für Hamilton Kreis.
  - Eingabe für TSP ist Clique $C = (V, E')$ mit:
  - $c(e) = 1$ falls $e \in E$ und
  - $c(e) = \alpha(n) \cdot n + 1$ falls $e \notin E$.
  - Falls G einen Hamiltonkreis hat, dann G' Tour mit Kosten n.
  - Anderenfalls hat G' eine Tour mit Kosten $\geqslant n - 1 + \alpha(n) \cdot n$.

Δ-TSP

Definition (Δ-TSP)

Gegeben $G = (V, E)$, $L \in \mathbb{Q}$ und $c : E \mapsto \mathbb{Q}$

- mit: $\forall v, w \in V : e = \{v, w\} \in E, c(e) \geqslant 0$ und
- mit: $\forall v, w, z \in V : c(v, z) \leqslant c(v, w) + c(w, z)$.

Bestimme spannenden einfachen Kreis C mit $c(C) \leqslant L$.

- Obige Eigenschaft auf c wird Metrik genannt.
- Motivation: Die Abstände von Punkten in der Ebene bilden eine Metrik.

Theorem

Δ-TSP mit Gewichtsfunktion $c \mapsto \{1, 2\}$ ist in $\mathcal{NPC}$.

Beweis: Reduktion von Hamilton Kreis auf planare Graphen.

Idee zu Δ-TSP

Analog wie beim Vertex Cover brauchen wir für das TSP eine effizient zu berechnende untere Schranke, die zur Approximation genutzt wird. Man beachte: im Folgenden ist wegen der Dreiecksungleichung der Graph eine Clique.

Da eine TSP Tour auch ein zusammenhängender spannender Graph (Teilgraph, der alle Knoten beinhaltet) ist, können wir als untere Schrankte einen minimalen Spannbaum nutzen. Um aus diesem Spannbaum eine TSP Tour zu bestimmen, werden die Kanten verdoppelt. Dadurch entsteht ein Eulergraph, der wegen der Dreiecksungleichung zu einer TSP Tour umgeformt werden kann. Bei der Umformung werden einfach die Mehrfachbesuche eines Knotens übersprungen. Wegen der Verdopplung der Kanten erhalten wir einen Approximationsfaktor von 2.

Dieser Faktor kann noch verbessert werden, denn es reicht aus, nur jedem Knoten von ungeradem Grad in dem Spannbaum eine neu Kante hinzuzufügen. Daher wird ein kostenminimales Matching zwischen den Knoten von ungeradem Grad des Spannbaums gesucht. Danach ist das Vorgehen wie oben. Nun erhalten wir eine Approximationsfaktor von 1.5.

Bei den Beweisen wird von der optimalen Lösung ausgehend gezeigt, das der minimale Spannbaum weniger Kosten hat als eine optimale TSP Tour und das kostenminimale Matching nur die halben Kosten wie eine optimale TSP Tour.

Aufbau der Idee

- Suche untere Schranke:
  - TSP-Tour ist ein spannender Graph (enthält alle Knoten)
  - Jeder kostenminimale zusammenhängende Graph bildet eine untere Schranke.
  - Idee: Wähle minimalen Spannbaum!
- Nutze Dreiecksungleichung aus:
  - D.h. jede Abkürzung verringert die Kosten.
  - Damit kann ein beliebiger Kreis, der jeden Knoten mindestens einmal besucht, zu einer kostengünstigeren TSP-Tour umgeformt werden.
  - Idee: Konstruiere daher einen Eulerkreis (besucht alle Kanten).
- $G = (V, E)$ ist Eulergraph (enthält Eulerkreis):
  - G ist zusammenhängend.
  - Alle Knoten haben geraden Grad.
- Kombination: Mit Hilfe des Spannbaums wird Eulerkreis konstruiert.

Approximation

Theorem

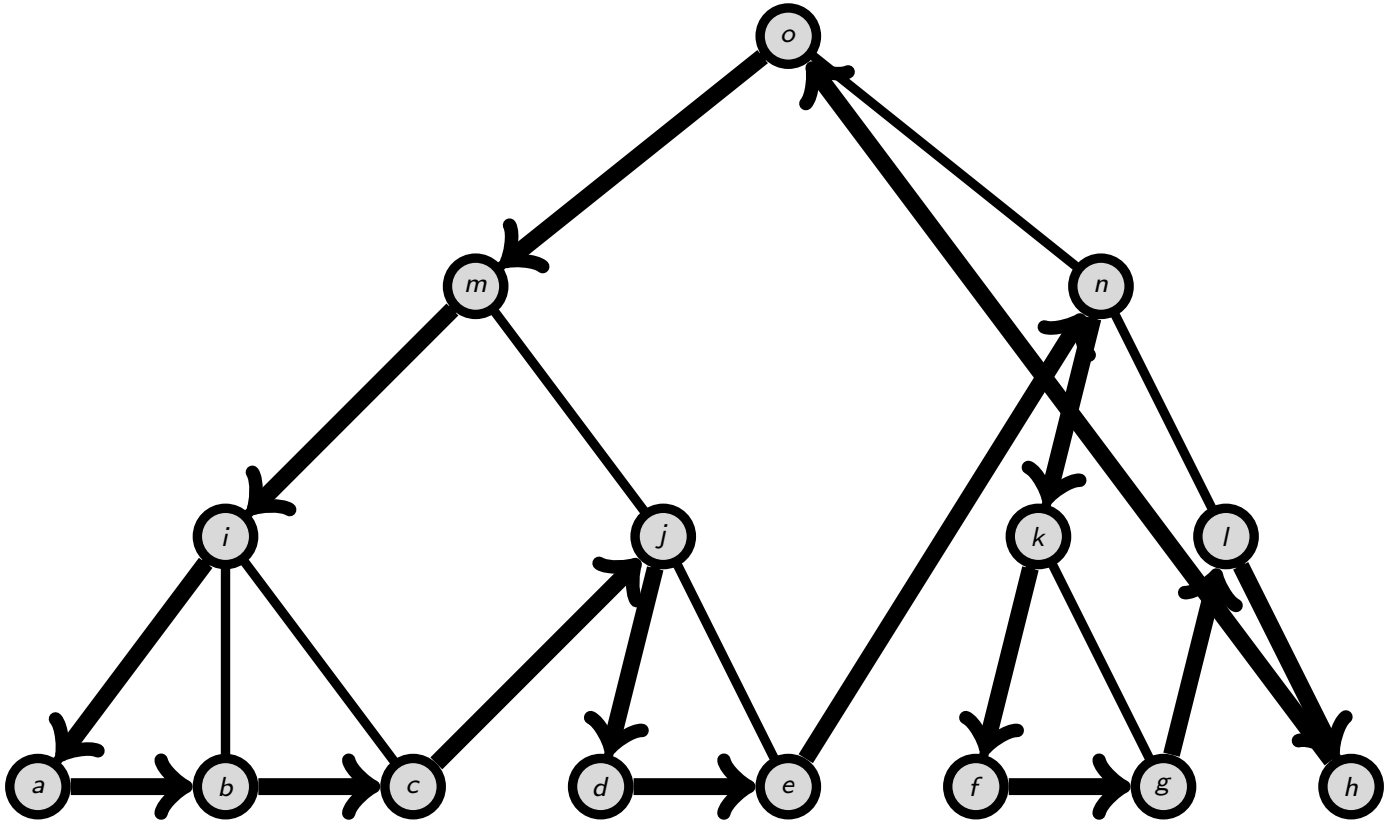
Δ-TSP ist mit einem Faktor von 2 in Zeit $O(n^2 \log n)$ approximierbar.

1. Bestimme minimalen Spannbaum T von G.
2. Verdoppele die Kanten von T und erzeuge Graphen T'.
3. Damit sind alle Knotengrade in T' gerade (da verdoppelt).
4. Bestimme in T' einen Euler-Kreis C'.
5. Verkürze C' durch Überspringen doppelter Knoten.
6. Dadurch entsteht Kreis C, gebe C aus.

Beispiel

Bestimme Eulertour: Gehe einmal inorder um den Baum herum.

Eulertour: $o, m, i, a, i, b, i, c, i, m, j, d, j, e, j, m, o, n, k, f, k, g, k, n, l, h, l, n, o$.



Approximation (Beweis)

$G - e$ ist G ohne eine beliebige Kante

- Seien T minimaler Spannbaum von G,
  T' mit verdoppelte Kanten,
  C' Eulerkreis in T' und
  C gefundene Lösung.
- Sei C^* eine minimale TSP Tour.
- $c(T) \leqslant c(C^* - e) \leqslant c(C^*)$.
  - Beachte: $C^* - e$ ist ein Spannbaum.
- $c(C) \leqslant c(C') = 2 \cdot c(T)$.
  - Da die Kanten verdoppelt worden sind.
- $\frac{c(C)}{c(C^*)} \leqslant \frac{2 \cdot c(T)}{c(C^*)} \leqslant \frac{2 \cdot c(C^*)}{c(C^*)} = 2$
- Laufzeit $O(n^2 \log n)$.

Aufbau der Idee

- Durch die Verdopplung aller Kanten des Spannbaums bekommen wir den Faktor von 2.
- D.h.: Falls v bereits gerade ist, wird trotzdem der Knotengrad verdoppelt.
- Daher: "verdoppele" den Knotengrad nur von den ungeraden Knoten.
- Oder noch besser: Erhöhe den Knotengrad der ungeraden Knoten um eins.
- D.h. jeder ungerade Knoten bekommt eine zusätzliche Kante.
- Das entspricht einem Matching zwischen den Knoten vom ungeraden Grad.
- Geht nur, wenn die Anzahl der ungeraden Knoten gerade ist:
  - Zählen wir die Kombinationen X:
    Knoten v inzident zu Kante $\{v, w\}$:
  - Aus der Sicht der Kanten: $|X| = 2 \cdot |E|$.
  - Aus der Sicht der Knoten: $|X| = \sum_{v \in V} \delta(v)$.
  - Sei U die Menge der Knoten mit ungeradem Grad.
  - $2 \cdot |E| = \sum_{v \in U} \delta(v) + \sum_{v \notin U} \delta(v)$.

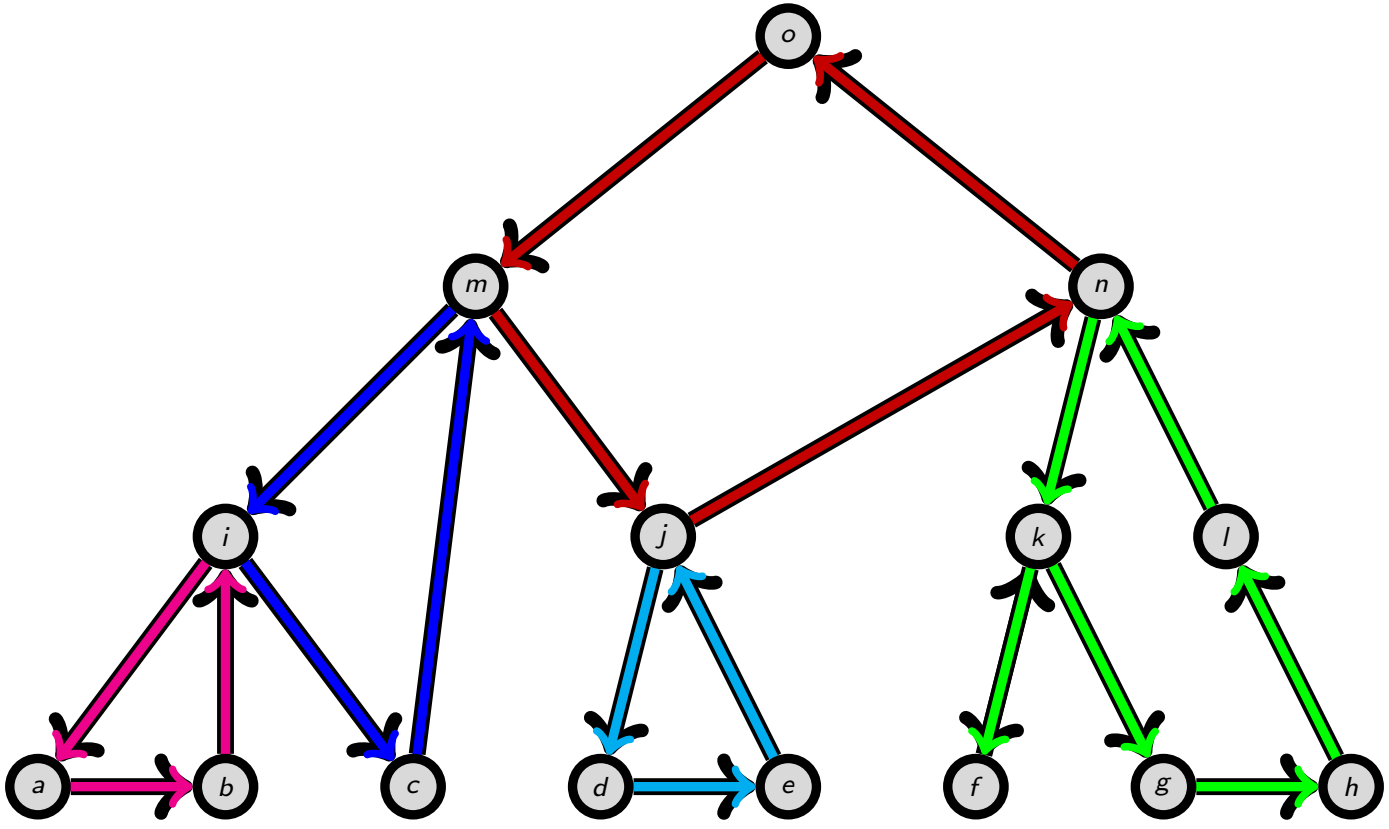
1.5-Approximation

Theorem

Δ-TSP ist mit einem Faktor von 1.5 in Zeit $O(n^3)$ approximierbar.

- Bestimme minimalen Spannbaum T von G.
- Seien U die Knoten von ungeraden Grad in T.
- Beachte $|U|$ ist gerade: $\sum_{v \in U} \delta(v) + \sum_{v \notin U} \delta(v) = 2|E|$.
- Bestimme kostenminimales Matching M zwischen den Knoten aus U.
- Bestimme in $T \cup M$ einen Eulerkreis C'.
- Verkürze C' durch Überspringen doppelter Knoten.
- Dadurch entsteht Kreis C.
- Gebe C aus.

Beispiel (Bestimmen der Eulertour)



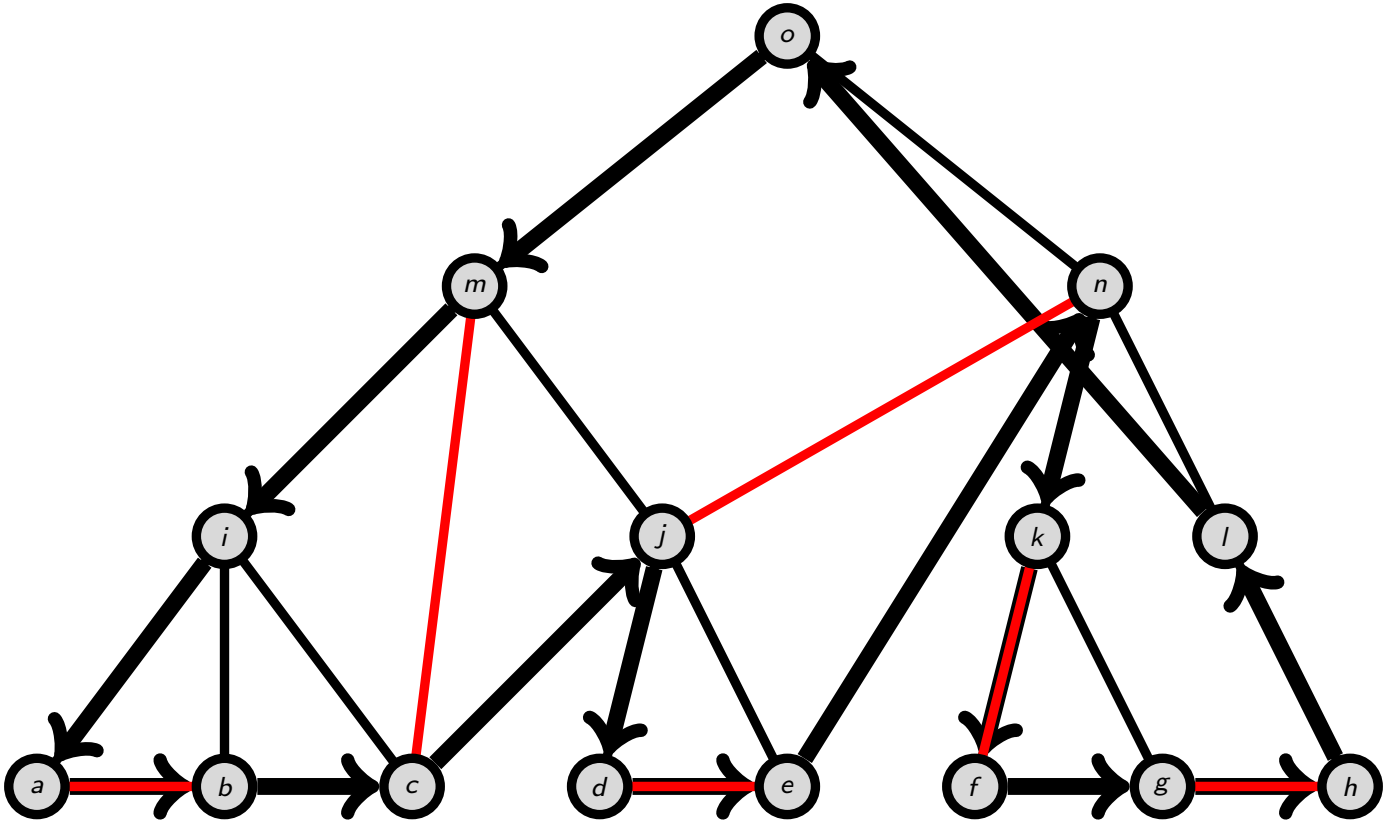
Bestimmung des Eulerkreises

1. Gegeben $G = (V, E)$ Eulergraph.
2. Setze $i = 0$
3. Wähle Startknoten v_0.
4. Bestimme Kreis C_0, der bei v_0 startet.
5. Solange es noch ungenutzte Kanten gibt:
   1. Bestimme von v_i aus auf C_i den ersten Knoten v_{i+1} mit ungenutzter Kante.
   2. Setze $i = i + 1$.
   3. Bestimme Kreis C_i, der bei v_i startet.
6. Nun können die Kreise einfach kombiniert werden:
   1. Gehe auf C_0 von v_0 bis v_1.
   2. Durchlaufe rekursiv alle Kreise $C_1, C_2, \ldots$.
   3. Gehe auf C_0 von v_1 bis v_0.

Laufzeit: $O(|E|)$.

Beispiel (1.5 Approximation)

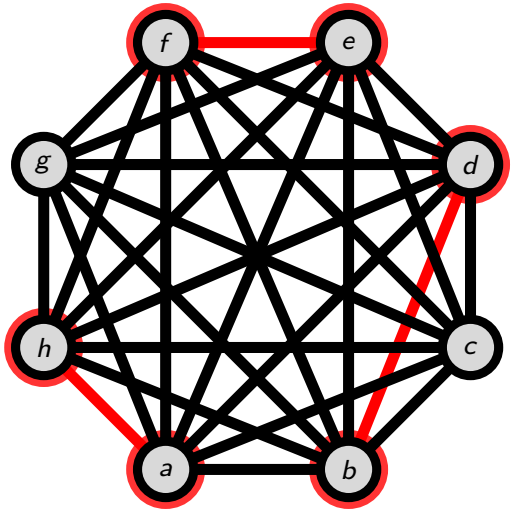
Eulertour: $o, m, i, a, b, i, c, m, j, d, e, j, n, k, f, k, g, h, l, n, o$



1.5-Approximation (Beweis)

- Seien T minimaler Spannbaum von G,
  M das Matching,
  C' Eulerkreis in T' und
  C gefundene Lösung.
- Sei C^* eine minimale TSP Tour.
- $c(T) \leqslant c(C^* - e) \leqslant c(C^*)$.
  - Beachte: $C^* - e$ ist ein Spannbaum
- $c(M) \leqslant c(C^*)/2$.
  - Siehe dazu nächste Folie.
- $\frac{c(C)}{c(C^*)} \leqslant \frac{c(T)+c(M)}{c(C^*)} \leqslant \frac{c(C^*)+c(C^*)/2}{c(C^*)} \leqslant \frac{1.5 \cdot c(C^*)}{c(C^*)} = 1.5$

1.5-Approximation (Zeige $c(M) \leqslant c(C^*)/2$)



- Sei C^* eine optimale Lösung.
- Seien U die ungeraden Knoten auf T.
- Sei C'' der Kreis, der sich aus C^* ergibt durch Überspringen der Knoten $V \setminus U$.
- C'' hat gerade Länge.
- C'' hat zwei disjunkte perfekte Matchings M_1, M_2
- $c(M) \leqslant c(M_1)$ und $c(M) \leqslant c(M_2)$.
- $2 \cdot c(M) \leqslant c(M_1) + c(M_2) = c(C'') \leqslant c(C^*)$
- $c(M) \leqslant c(C^*)/2$

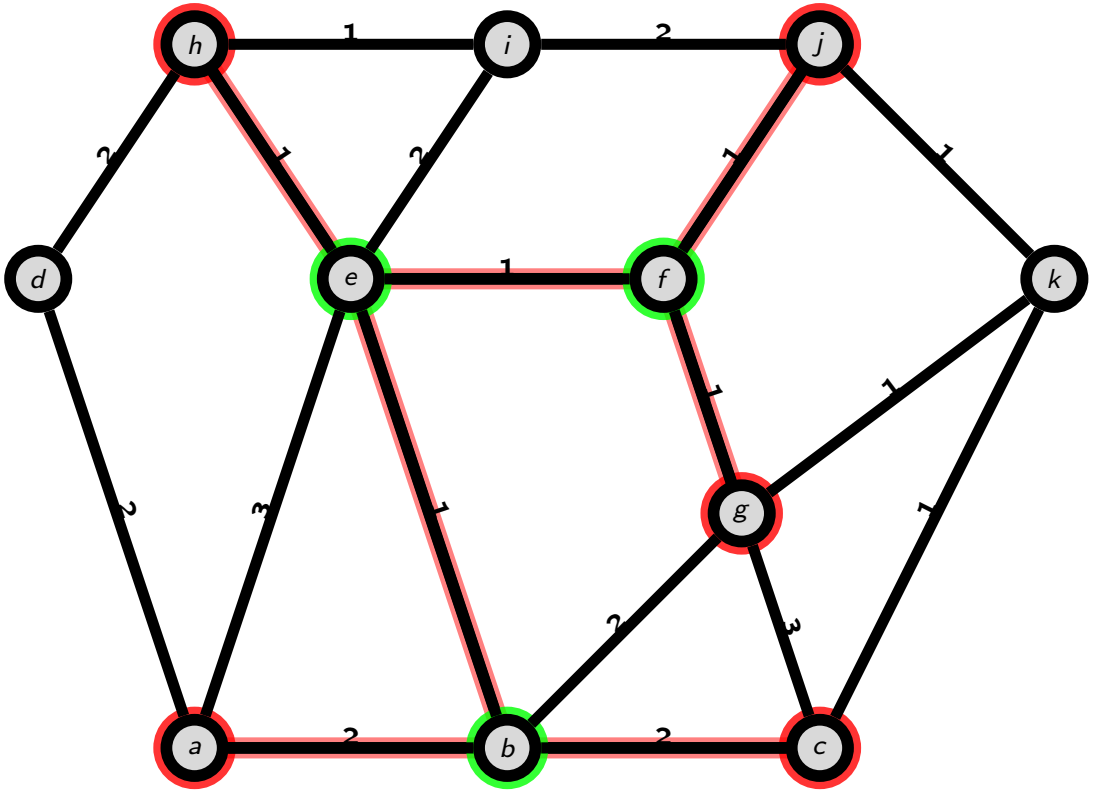
Steinerbaum

Definition (Steinerbaum Problem)

Gegeben $G = (V, E)$, $T \subset V$ und $c : E \mapsto \mathbb{Q}^+$.
Bestimme Spannbaum $S = (T \cup P, F)$,
der die Knoten von T minimal $\sum_{e \in F} c(e)$ verbindet.

- Die Blätter von S sind alle aus T.
- Die Menge P heißt Steinerpunkte.
- Falls $T = V$ gilt, ist ein minimaler Spannbaum zu finden.
- Falls $|T| = 2$ gilt, ist ein minimaler Weg zu finden.
- Anwendung: VLSI-Chip-Design
- Entscheidungsvariante: Gibt es Menge $P \subset V \setminus T$ der Größe k, so daß $G[S \cup P]$ zusammenhängend ist.

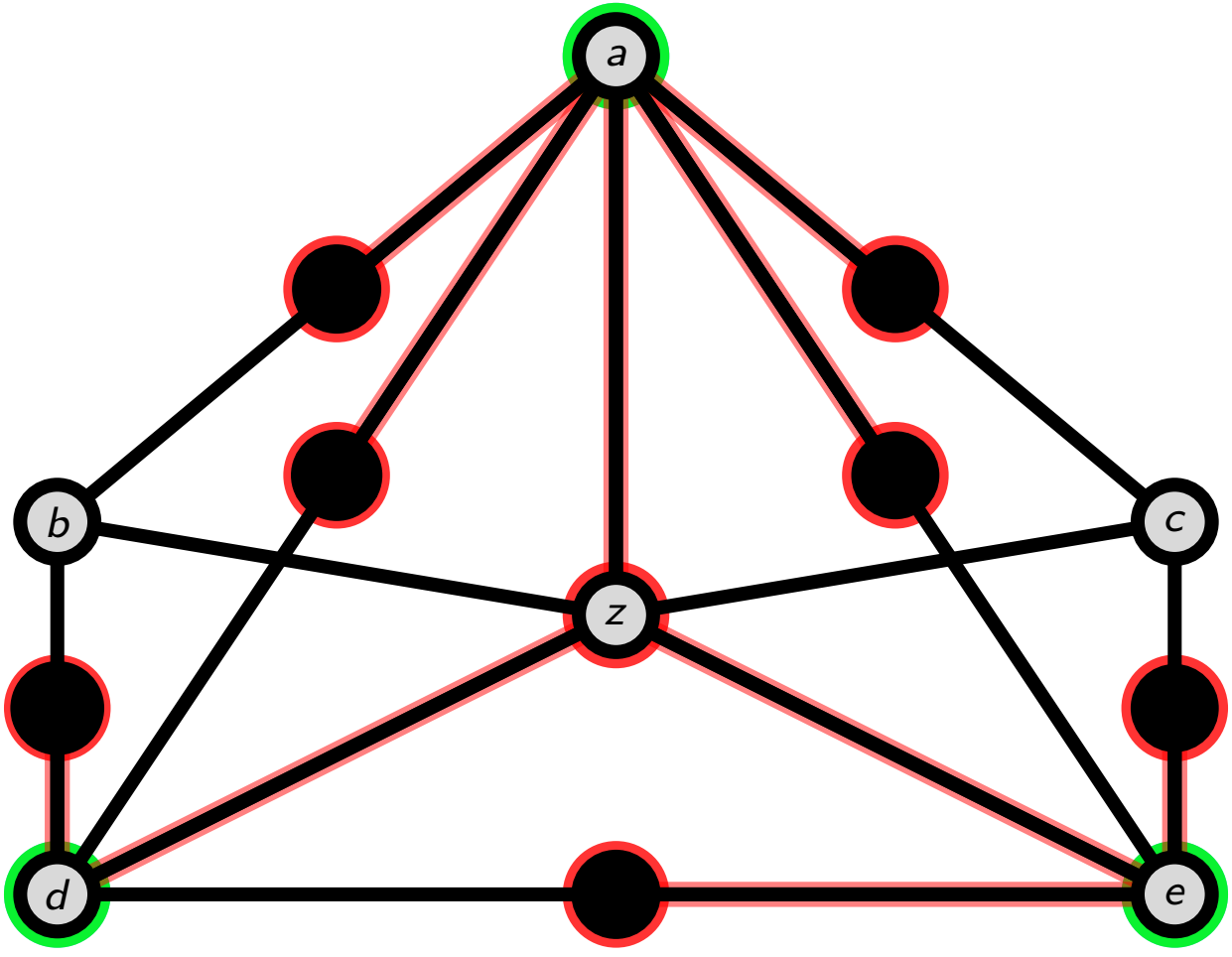
Beispiel



- Die Terminals T sind rot.
- Die Steinerpunkte sind grün.
- Die Kosten sind: 9.

Steinerbaum (Idee der Reduktion)

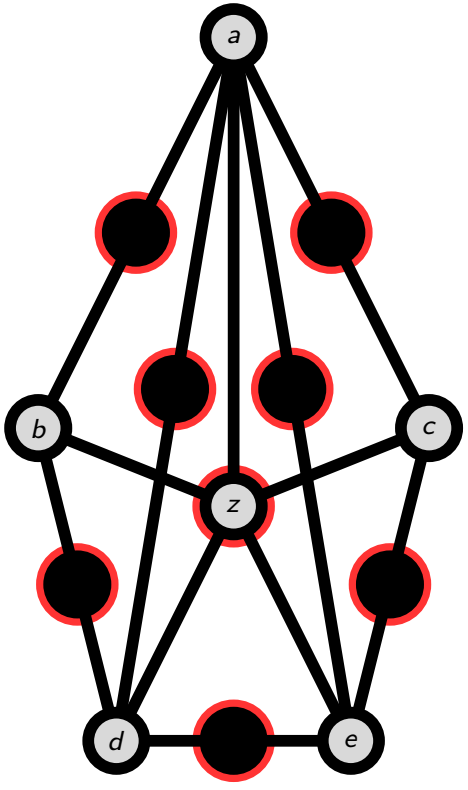
Zeige: Das Steinerbaumproblem ist in $\mathcal{NPC}$ (Reduktion von Vertex Cover).



Steinerbaum

Theorem

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

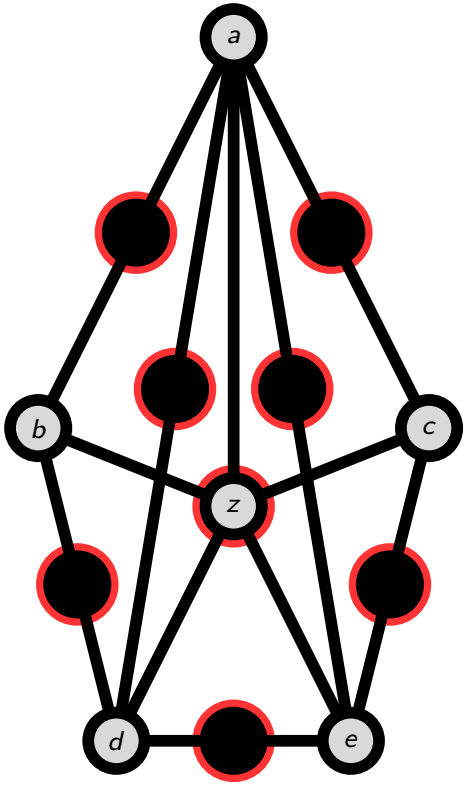


- Sei $G = (V, E)$, k Eingabe des Vertex Cover Problems
- Konstruiere Graphen G' aus G wie folgt:
  - $G' = (V \dot\cup V_e \dot\cup \{z\}, F \cup E_z)$
  - $V_e = \{v_e \mid e \in E\}$
  - $F = \{\{a, v_e\}, \{v_e, b\} \mid e = \{a, b\} \in E\}$
  - $E_z = \{\{v, z\} \mid v \in V\}$
  - Setze $T = V_e \cup \{z\}$.

Steinerbaum

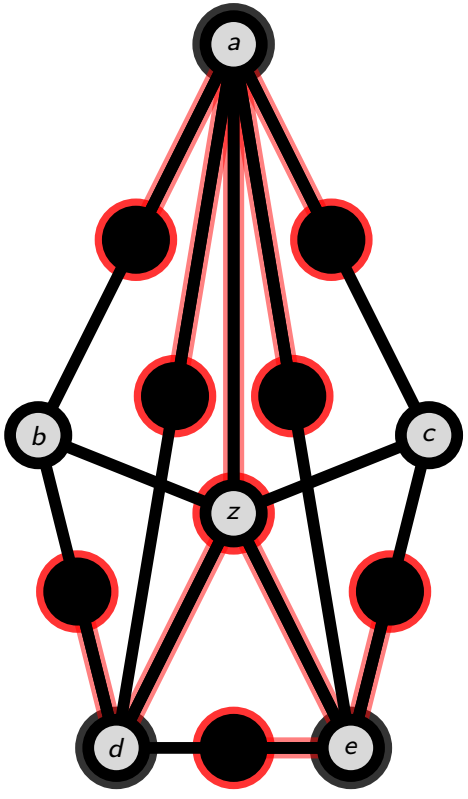
Theorem

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



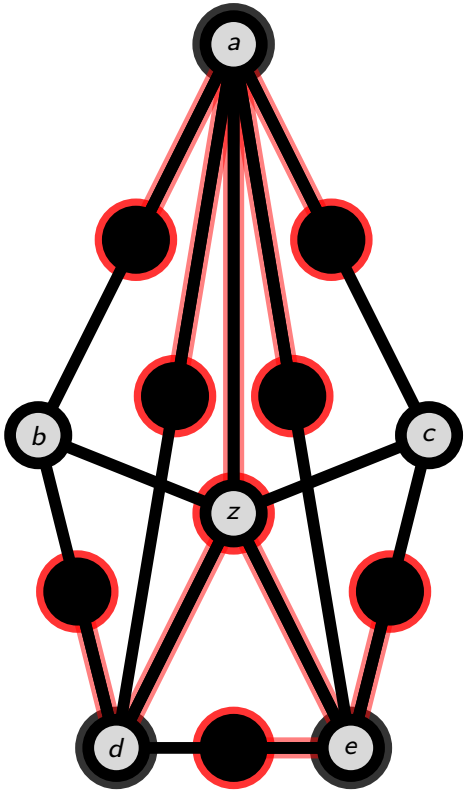
- Erinnerung: G hat Vertex Cover der Größe $\tau(G)$
- G' hat $n + m + 1$ Knoten.
- G' ist bipartit.
- Beachte: $T = V_e \cup \{z\}$.
- G' wird $\tau(G)$ Steinerpunkte haben.
- Der Steinerbaum wird $\gamma(G') = \tau(G) + m$ Kanten haben.
- Zeige im Weiteren:
  - $\gamma(G') \leqslant \tau(G) + m$
  - $\gamma(G') \geqslant \tau(G) + m$

Steinerbaum (Beweis)



- Zeige: $\gamma(G') \leqslant \tau(G) + m$
- Sei C Vertex Cover der Größe $\tau(G)$.
- Wähle C als Steinerpunkte von G'.
- Mit $\tau(G)$ Kanten werden die Knoten aus C mit z verbunden.
- Da C ein Vertex Cover ist,
  ist jeder Knoten aus V_e mit einen Knoten aus C verbunden.
- Damit sind alle Knoten aus V_e
  mit jeweils einem Knoten aus C verbunden.
- Das sind dann nochmals m Kanten

Steinerbaum (Beweis)



- Zeige: $\gamma(G') \geqslant \tau(G) + m$
- Sei S Steinerbaum mit $\gamma(G')$ Kanten.
- Die Knoten $C = V(S) \setminus T$ sind dann die Steinerknoten.
- Jeder Knoten aus V_e ist mit einem Knoten aus C über S verbunden.
- Damit ist C ein Vertex Cover für G.
- Weiter hat S $\gamma(G') + 1$ Knoten.
- $\tau(G) \leqslant |C| = \gamma(G') + 1 - (m+1) = \gamma(G') - m$.
- $\tau(G) + m \leqslant \gamma(G')$.

Idee zur Approximation eines Steinerbaums

Wieder brauchen wir eine gute und effizent zu berechnende untere Schranke. Eine einfache Schranke ist der kürzeste Weg zwischen zwei beliebigen Terminals. Nur ein kürzester Weg zwischen zwei Terminals ist aber als gute Schranke unzureichend, denn es müssen ja alle Terminals miteinander verbunden werden.

Daher bestimmen wir zwischen jedem Terminalpaar a, b einen kürzesten Weg $P_{a,b}$. Danach wählen wir eine kostenoptimale Menge aus diese Wegen, so das alle Terminals miteinander verbunden sind. Dies ist dann eine 2 Approximation des Steinerbaums.

Beim Beweis dieses Faktors, starten wir wieder mit einer optimalen Lösung. Von dieser Lösung verdoppeln wir die Kanten. Danach ist es möglich, eine Zuordnung von den Kanten auf die gewählten Wege zu machen. Nach der Verdopplung kann jede Kante genau einem gewählten Weg zugeordnet werden. Damit ist dann der Faktor von 2 bewiesen.

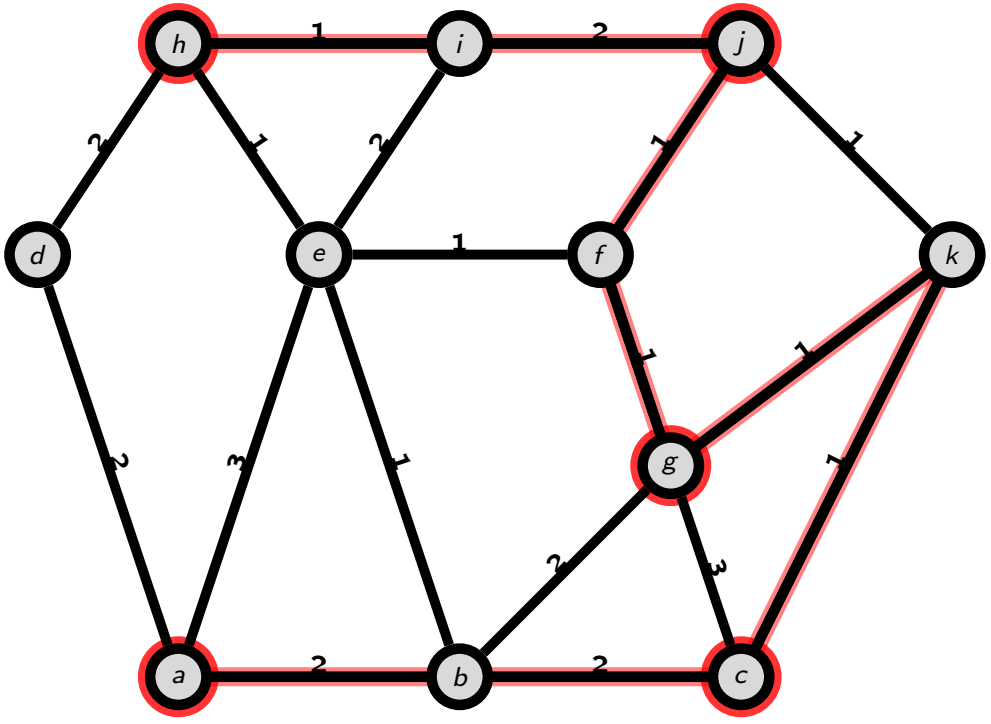
Steinerbaum (Approximation)

- Aufbau der Idee
- Alle Knoten aus T sind zu verbinden.
- Der kürzeste Weg zwischen zwei Terminals ist eine einfache untere Schranke.
- Eine kostenminimale Menge von diesen kürzesten Wegen, die alle Knoten aus T verbinden, sind kein untere Schranke.
- Aber wir werden sehen, das liefert eine 2-Approximation.
- Damit ergibt sich das folgende Vorgehen:
  - Bestimme paarweise kürzeste Wege.
  - Bestimme kostenminimale Menge kürzester Wege, die T verbinden.
  - Bestimme daraus den Steinerbaum.

Steinerbaum (Approximation)

- Verfahren von Kou, Markowsky und Berman:
  1. Gegeben $G = (V, E)$, $T \subset V$ und $c : E \mapsto \mathbb{Q}$.
  2. Bestimme Clique $G_D = (T, F)$ und
     - $c : F \mapsto \mathbb{Q}$ mit: $c_D(\{a, b\}) = dist_G(a, b)$.
  3. Bestimme minimalen Spannbaum $S_D = (T, F')$ auf G_D.
  4. Für jede Kante $e \in F'$ sei $W_e = (V_e, F_e)$ ein Weg der Länge $c_D(e)$ in G.
  5. Bestimme $H = (\cup_{e \in F'} V_e, \cup_{e \in F'} F_e)$ als Teilgraph von G.
  6. Bestimme minimalen Spannbaum S_H auf H.
  7. Lösche sukzessive Blätter aus S_H, die nicht in T sind.
  8. Ergebnis davon ist dann der Baum S_{KMB}.
- Damit gilt: $c(S_{KMB}) \leqslant c(H) \leqslant c(S_D)$.

Beispiel



- Erzeuge G_D und c_D:
- Bestimme S_D auf G_D
- Bestimme passende Wege in G.
- Kosten: 11.

x	y	$c(x,y)$	$\in S_D$
c	g	2	1
g	j	2	2
c	j	2	
h	j	3	3
g	h	3	
a	c	4	4
a	g	4	
a	h	4	
c	h	4	
a	j	5	

Steinerbaum (Approximation)

Theorem

Das Verfahren von Kou, Markowsky und Berman berechnet eine 2-Approximation für das Steinerbaumproblem.

- Beachte: $c(S_{KMB}) \leqslant c(H) \leqslant c(S_D)$.
- Sei S^* ein minimaler Steinerbaum.
- Schätze nun $c(S^*)$ und $c(S_D)$ zueinander ab. Ziel: $c(S_D) \leqslant 2 \cdot c(S^*)$.
- Verdopple die Kanten von S^*.
- Damit ergibt sich Zyklus Z, der alle Knoten aus T trifft.
- Z definiert spannenden Baum auf G_D:
  - Start bei $s_1 \in T$ auf Z.
  - Sei $s_2, s_3, \ldots, s_{|T|}$ die Folge der Erstbesuche der Knoten aus T.
  - Die Kanten $\{s_i, s_{i+1}\}$ bilden einen Spannbaum T_Z auf G_D.
  - Also: $c(T_Z) \leqslant c(Z)$ und damit $c(T_Z) \leqslant c(Z) = 2 \cdot c(S^*)$.
  - Insgesamt: $c(S_{KMB}) \leqslant c(H) \leqslant c(S_D) \leqslant c(T_Z) \leqslant c(Z) = 2 \cdot c(S^*)$.

Steinerbaum (Zusammenfassung)

Theorem

Das Steinerbaum Problem kann mit einem Faktor von 2 approximiert werden.

Theorem

Das Steinerbaum Problem kann mit einem Faktor von $\frac{11}{6}$ approximiert werden.

Wird hier nicht bewiesen.

Zentrumsproblem

- Gegeben $G = (V, E)$ und $c : E \mapsto \mathbb{Q}^+$.
- Sei $Z \subset V$
- $dist(v, Z) := \min_{z \in Z} dist(z, v)$
- $\text{rad}(Z) := \max_{v \in V} dist(v, Z)$

Definition (Zentrumsproblem (Entscheidungsvariante))

Gegeben $G = (V, E)$, $k \in \mathbb{N}$, $L \in \mathbb{Q}^+$ und $c : E \mapsto \mathbb{Q}^+$.
Bestimme $\exists Z : |Z| = k$ und $\text{rad}(Z) \leqslant L$.

Definition (Zentrumsproblem)

Gegeben $G = (V, E)$, $k \in \mathbb{N}$ und $c : E \mapsto \mathbb{Q}^+$.
Bestimme Z mit $|Z| = k$ und $\text{rad}(Z)$ minimal.

Komplexität

Theorem

Falls $\mathcal{P} \neq \mathcal{NP}$ gilt, gibt es für beliebiges $r < 2$ keinen Polynomzeitalgorithmus mit Approximationsfaktor r für das Zentrumsproblem.

Beweis:

- Reduktion vom Dominating Set Problem.

Definition (Dominating Set (Entscheidungsvariante))

Gegeben $G = (V, E)$, $k \in \mathbb{N}$.
Bestimme $\exists D : |D| = k$ und $V = D \cup \{v \mid \exists w \in D : \{v, w\} \in E\}$.

- Alternativ:
  - $V = D \cup \{v \mid \exists w \in D : \{v, w\} \in E\}$ genau dann, wenn
  - $\text{rad}(D) = 1$.

Komplexität (Beweis)

- Reduktion vom Dominating Set Problem.
- Sei (G, k) Eingabe für das Dominating Set Problem.
- Setze $c(e) = 1$ für $e \in E(G)$.
- Dann hat G Dominating Set der Größe k, falls G Zentrum mit k Knoten und Radius 1 hat.
- Falls es einen Algorithmus A mit Approximationsfaktor $r < 2$ gibt,
- So liefert A bei Eingabe $(G, 1, k, c)$ ein Dominating Set Problem der Größe k.

Idee zur Approximation

Es könnten zwei Parameter Approximiert werden: Die Größe des Zentrums oder der Radius. Hier werden wir den Radius mit einem Faktor von zwei approximieren. Dazu wird einfach ein Greedy-Algorithmus mit allen sinnvoll möglichen Radien gestartet. Das Greedy-Verfahren wählt einen "beliebigen" Knoten, der noch nicht überdeckt ist. Im Algorithmus wird aber der Faktor von zwei beim Radius genutzt und dann das Zentrum der kleinsten Kardinalität gewählt. Daher muss der Faktor, er ist ja schon Teil des Algorithmuses, nicht mehr bewiesen werden. Es muss aber gezeigt werden, das die berechnete Lösung nicht größer ist als eine optimal Lösung mit einfachem Radius.

Sei A die berechnete Lösung mit doppelten Radius und O eine optimale Lösung mit einfachem Radius R. Nun wird jedem Knoten v aus A genau ein Knoten aus O zugeordnet. Dem Knoten v wird der Knoten $r(v)$ aus O zugeordnet, der in der optimalen Lösung den Knoten v überdeckt (der Abstand ist höchstens R). Alle Knoten im Radius R um $r(v)$ haben auch einen Abstand von höchstens $2R$ von v. Anschaulich: Die Scheibe mit Radius R um $r(v)$ ist vollständig in der Scheibe mit Radius $2R$ um v enthalten. Daher wird der Greedy-Algorithmus keinen weiteren Knoten v' wählen, so das die Scheibe mit Radius R um $r(v)$ ist vollständig in der Scheibe mit Radius $2R$ um v' enthalten ist. Die Zuordnung ist also eindeutig.

Zentrumsproblem (mit Knotenkosten)

- Gegeben $G = (V, E)$, $w : V \mapsto \mathbb{Q}^+$ und $c : E \mapsto \mathbb{Q}^+$.
- Sei $Z \subset V$
- $dist(v, Z) := \min_{z \in Z} dist(z, v)$
- $\text{rad}(Z) := \max_{v \in V} w(v) \cdot dist(v, Z)$ von nun an.

Definition (Zentrumsproblem)

Gegeben $G = (V, E)$, $k \in \mathbb{N}$, $w : V \mapsto \mathbb{Q}^+$ und $c : E \mapsto \mathbb{Q}^+$.
Bestimme Z mit $|Z| = k$ und $\text{rad}(Z)$ minimal.

Aufbau des Algorithmus

Bei dem Zentrumsproblem gibt es zwei Werte, die approximiert werden können. Zum einen der Radius und zum anderen die Anzahl der Knoten, die das Zentrum bilden.

Wir werden hier eine Lösung entwickeln, die nur beim Radius einen Fehler von zwei macht. Die Anzahl der Knoten, die dabei verwendet werden, wird nicht größer sein als bei der optimalen Lösung.

Wir werden den Algorithmus in zwei Schritten entwickeln. Zuerst wird ein Greedy Verfahren entwickelt. In diesem Verfahren werden wir den doppelten Radius nutzen. Wenn dieses Greedyverfahren mit dem optimalen Radius aufgerufen wird, dann wird auch einen Zentrum bestimmt, welches nicht mehr Knoten hat als das Optimum.

Im zweiten Schritt wird dieses Greedy-Verfahren für jeden sinnvollen Radius aufgerufen. Da unser Ziel ist, ein Zentrum mit k Knoten zu finden, wählen wir als Resultat das Ergebnis, welches bei Verwendung von k Knoten den kleinsten Radius hat. Dieser Radius ist dann eine Approximation von zwei.

Idee

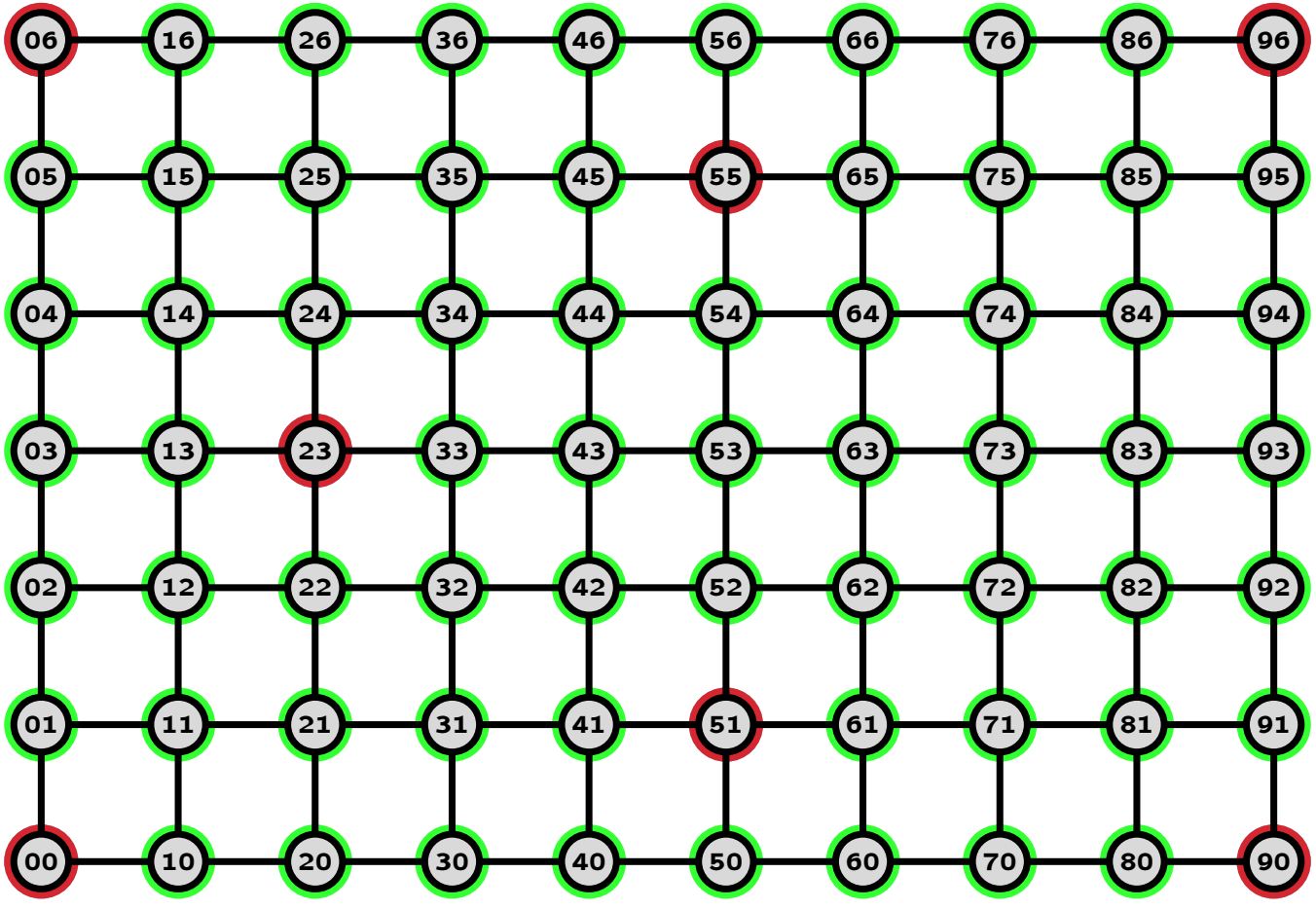
- Wir kennen die Knoten aus Z nicht.
- Wir kennen das Optimum $R = \text{rad}(Z)$ nicht.
- Idee: suche Z mit Greedy Verfahren für ein festes R.
  - Solange noch Knoten nicht durch Z überdeckt sind.
  - Wähle mit Greedy einen weiteren Knoten z zu Z hinzu.
  - Alle Knoten im Abstand $\underline{2 \cdot R}$ werden nun als überdeckt betrachtet.
- Achtung: der Algorithmus beinhaltet den Approximationsfaktor!
- Teste nun verschiedene Werte R mit Hilfe dieses Greedyverfahrens.

Approximation

- Algorithmus *GreedyZentrum*(G, c, w, R):
  1. $Z := \emptyset$ und $U := V$
  2. Solange $U \neq \emptyset$ mache
     1. $z := \text{argmax}\{w(u) \mid u \in U\}$
     2. $Z := Z \cup \{z\}$
     3. $U := U \setminus \{v \in U \mid w(v) \cdot dist(v, z) \leqslant 2 \cdot R\}$
  3. Ausgabe: Z
- Sei R^* optimale Lösung mit Z^*. Falls $R \geqslant R^*$, dann liefert *GreedyZentrum* ein Z mit $|Z| \leqslant |Z^*|$ (Siehe Folie: 59).
- Im Hauptalgorithmus wird *GreedyZentrum* daher für alle interessanten Werte für R aufgerufen (Siehe Folie: 60).
- Definiere: $f(R) := |GreedyZentrum(G, c, w, R)|$
  - Beachte $f(R)$ ist aber nicht monoton (Siehe Folie: 57).
- Definiere: $R' = \max_{v \in V} w(v) \cdot \sum_{e \in E} c(e)$
  - Mit so einem großen Radius kann ein Knoten alles überdecken.
  - $f(R') = 1$ und $f(0) = n$

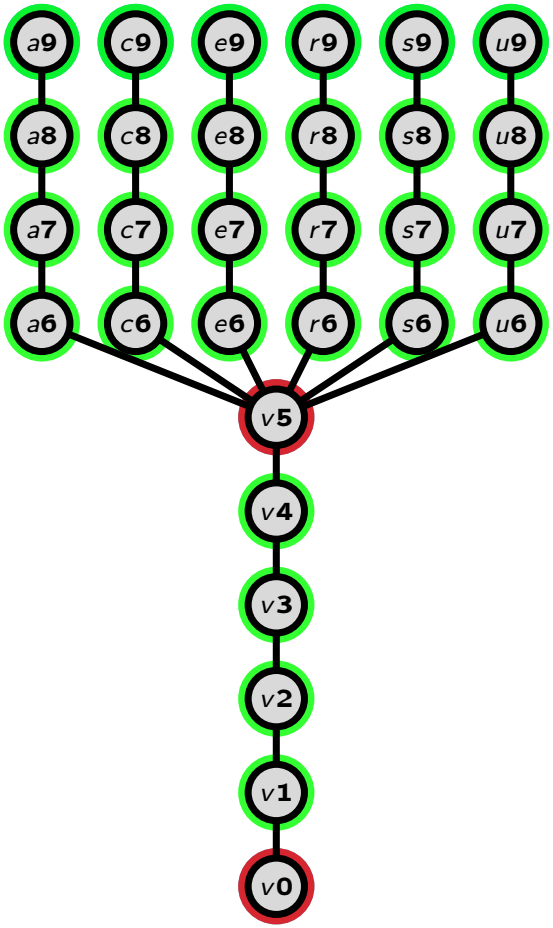
Bildhaftes Beispiel ($R = 2$)

$w(00) = w(06) = w(23) = w(90) = w(96) = w(51) = w(55) = 2$

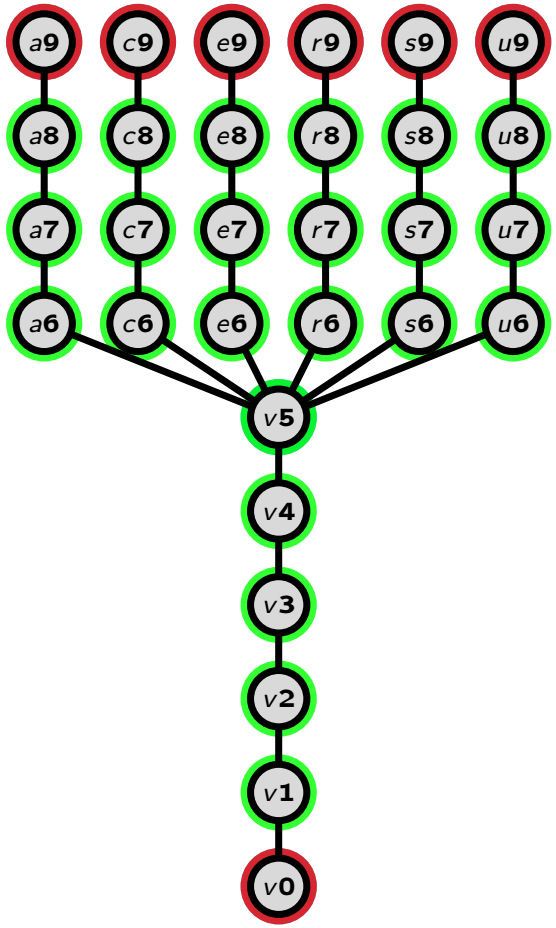


Beispiel für $f(R)$ ist nicht monoton

Bestimme $f(2) = 2$



Bestimme $f(3) = 7$

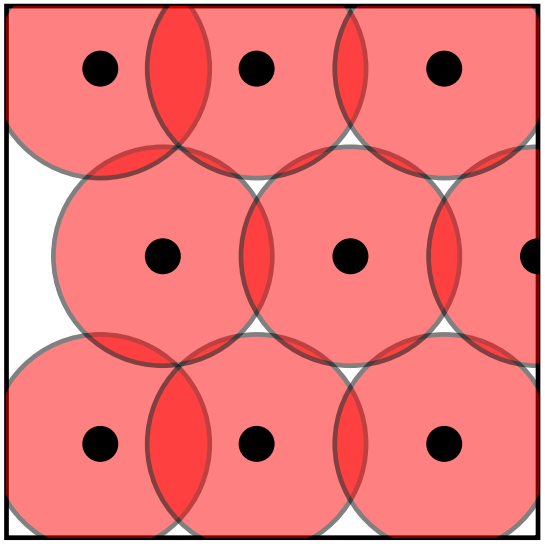


Approximation

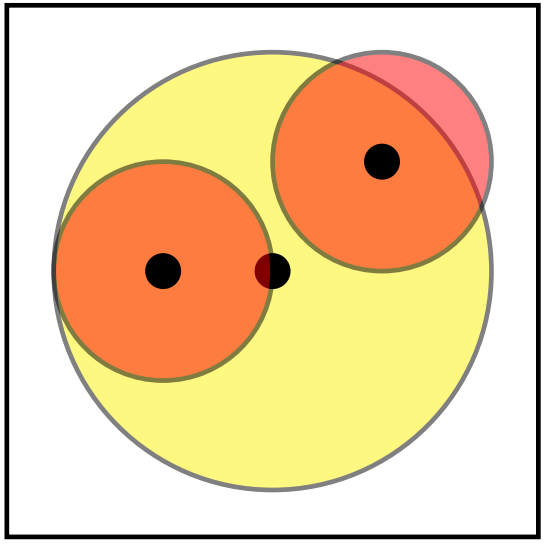
Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalen Radius und $R^* = \text{rad}(Z^*)$.
Dann gilt: $f(R) \leqslant k$ für alle $R \geqslant R^*$.

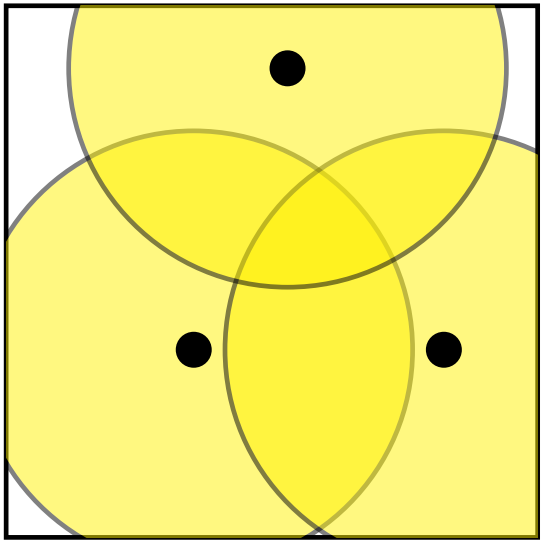
Optimale Lösung (R^*):



Vergleich:



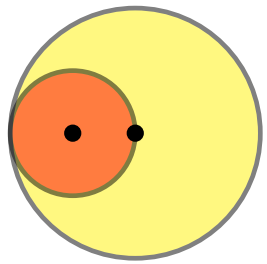
Approximation:



Approximation

Lemma

Sei $G = (V, E)$ Z^* ein Zentrum aus k Knoten mit minimalem Radius und $R^* = \text{rad}(Z^*)$.
Dann gilt: $f(R) \leqslant k$ für alle $R \geqslant R^*$.



- Sei $Z^* = \{z_1, z_2, \ldots, z_k\}$.
- Sei $V_i := \{v \in V \mid w(v) \cdot dist(v, z_i) \leqslant R^*\}$.
- Damit $\cup_{i=1}^{k} V_i = V$.
- Sei $z \in V_i$ der in der Schleife gewählte Knoten für ein i. Damit gilt:

$$\begin{aligned} w(v) \cdot dist(v, z) &\leqslant w(v) \cdot (dist(v, v_i) + dist(v_i, z)) \\ &\leqslant w(v) \cdot dist(v, v_i) + w(z) \cdot dist(v_i, z) \\ &\leqslant 2 \cdot R^* \\ &\leqslant 2 \cdot R \end{aligned}$$

- Alle Knoten aus V_i sind am Ende des Schleifendurchlaufes aus U gelöscht.
- Schleife terminiert nach höchstens $k = |Z^*|$ Runden. Und es gilt: $|Z| \leqslant k$.

Approximation

- Es werden nun die alle möglichen Distanzwerte untersucht:
  - $\{w(u) \cdot dist(u,v) \mid u,v \in V \wedge u \neq v\}$
  - $anz := |\{w(u) \cdot dist(u,v) \mid u,v \in V \wedge u \neq v\}| \leqslant n \cdot (n-1)$
- Algorithmus $ApproxZentrum(G, c, w, k)$
  1. Bestimme Distanzmatrix $(dist(u,v))_{u,v \in V}$ für (G, c).
  2. Sortiere Werte $\{w(u) \cdot dist(u,v) \mid u,v \in V \wedge u \neq v\}$.
  3. Seien $R_1 < R_2 < \ldots < R_{anz}$ diese Werte.
  4. $i := 0$, $Z = V$.
  5. Solange $|Z| > k$ führe aus:
     1. $i = i + 1$.
     2. $Z := GreedyZentrum(G, c, w, R_i)$.
  6. Ausgabe: Z.

Approximation

Theorem

Das Zentrumsproblem ist mit einem Faktor von 2 in Zeit $O(n^4)$ approximierbar.

- Wegen dem obigen Lemma terminiert die Schleife.
- Sei i_0 die Anzahl der Schleifendurchläufe.
- D.h. der Durchlauf mit Ausgabe Z mit $\text{rad}(Z) \leqslant 2 \cdot R_{i_0}$
- Durch Konstrukton von *GreedyZentrum* gilt: $\text{rad}(Z) \leqslant 2 \cdot R_{i_0}$
- Damit ist auch Z eine 2 Approximation.
- Laufzeit:
  - Distanzmatrix: $O(n^3)$.
  - Sortierung: $O(n^2 \log n)$.
  - *GreedyZentrum*: $O(n^2)$.
- Gesamtlaufzeit $O(n^4)$.

Erinnerung

Definition (Knotenfärbung (Erinnerung))

Sei $G = (V, E)$ ungerichteter Graph, Dann wird definiert:

- k-Färbungsfunktion $c : V \mapsto \mathbb{N}_k : \forall \{v, w\} \in E : c(v) \neq c(w)$.
- Färbungszahl
  $\chi(G) = \min_{k \in \mathbb{N}} \{k \mid \exists c : V \mapsto \mathbb{N}_k : \forall \{v, w\} \in E : c(v) \neq c(w)\}$
- Färbungsproblem:

  COL: Gegeben: $G = (V, E), k$, Frage: Gilt $\chi(G) \leqslant k$
- k-Färbungsproblem:

  k-COL: Gegeben: $G = (V, E)$, Frage: Gilt $\chi(G) \leqslant k$
- praktisches Färbungsproblem:
  - Gegeben: $G = (V, E)$
  - Bestimme: $c : V \mapsto \mathbb{N}_{\chi(G)} : \forall \{v, w\} \in E : c(v) \neq c(w)$.

Approximationsfehler von Greedy

1. Eingabe: $G = (V, E)$.
2. $\forall v \in V : c(v) = 0$.
3. Solange es $v \in V$ gibt, mit $c(v) = 0$:
   1. Wähle $v \in V$ gibt, mit $c(v) = 0$.
   2. Setze $c(v) = \max\{c(w) + 1 \mid \{v, w\} \in E\}$

- Greedyverfahren liefert Färbung mit $A_{Greedy} \leqslant \Delta(G) + 1$ Farben.
- Zweifärbung ist in $\mathcal{P}$.
- Approximationsfehler: $\frac{A_{Greedy}}{\chi(G)} \leqslant \frac{\Delta(G)+1}{3}$.

Lemma

Es gibt eine Folge der Knoten $(v_1, v_2, \ldots, v_n)$, so dass "Greedy" den Graphen mit $\chi(G)$ färbt.

Beweis: wähle Folge mit: $\chi(v_i) \leqslant \chi(v_{i+1})$.

Aussagen

Lemma

Sei $0 < c \leqslant 1$ Konstante. Es gibt einen linearen Algorithmus, der das Färbungsproblem mit Faktor $\max(1, c \cdot n)$ approximiert.

- Falls $|V| \leqslant 2/c$, dann färbe G:
  - Färbe per Greedy durch alle Permutationen der Knoten den Graphen optimal.
  - Laufzeit: $O((2/c)! \cdot \binom{(2/c)!}{2})$.
  - Laufzeit: $O(1)$ und Fehlerfaktor 1.
- Falls $|V| > 2/c$, dann färbe G:
  - Partitioniere $V(G)$ in $\lfloor c \cdot n \rfloor$ Teile der Größe $\lfloor n/\lfloor c \cdot n \rfloor \rfloor$ oder $\lceil n/\lfloor c \cdot n \rfloor \rceil$.
  - Jeder Teil hat Größe $\leqslant \frac{n}{cn-1} + 1 \leqslant \frac{2}{c} = O(1)$.
  - Jeder Teil kann in konstanter Zeit optimal gefärbt werden.
  - Gesamtfarbenanzahl: $\frac{\lfloor c \cdot n \rfloor \cdot \chi(G)}{\chi(G)} \leqslant cn$.

Aussagen

Lemma

Falls $\mathcal{P} \neq \mathcal{NP}$, dann gibt es keinen Polynomzeitalgorithmus mit Approximationsfehler $4/3$ für das Färbungsproblem.

Theorem (Garry, Johnson 1976)

Falls $\mathcal{P} \neq \mathcal{NP}$, dann gibt es keinen Polynomzeitalgorithmus mit Approximationsfehler 2 für das Färbungsproblem.

Theorem (Johnson 1974)

Das Färbungsproblem kann mit einem Faktor von $O(n/\log n)$ in Zeit $O(nm)$ approximiert werden.

Aussagen

ZPP = Zero-error Probabilistic Polynomial Time

Theorem (Lund, Yannakakis 1993)

Falls $\mathcal{P} \neq \mathcal{NP}$, dann gibt es für beliebiges $\varepsilon > 0$ keinen Polynomzeitalgorithmus mit Approximationsfehler n^{ε} für das Färbungsproblem.

Theorem (Feige, Kilian 1996)

Falls $\mathcal{P} \neq \mathcal{ZPP}$, dann gibt es für beliebiges $\varepsilon > 0$ keinen Polynomzeitalgorithmus mit Approximationsfehler $n^{1-\varepsilon}$ für das Färbungsproblem.

Literatur

- Vazirani: Approximation Algorithms, Springer Verlag, 2001.
- Jansen, Margraf: Approximative Algorithmen und Nichtapproximierbarkeit, de Gruyter, 2008.
- Wanka: Approximationsalgorithmen – Eine Einführung, Teubner Verlag 2006.
- Hochbaum: Approximation Algorithms for NP-Hard Problems, Thomson Publishing, 1996.
- Ausiello, Crescenzi, Gambosi, Kann, Marchetti-Spaccamela, Protasi: Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties, Springer Verlag, 1999.
- Garey, Johnson: Computers and Intractibility, Freeman and Company, 1979.

Fragen

- Wie kann das Problem Cliquenproblem approximiert werden? Welche untere Schranken sind dazu bekannt?
- Wie kann das Problem Färbungsproblem approximiert werden? Welche untere Schranken sind dazu bekannt?
- Wie kann das Problem Vertex Cover Problem approximiert werden? Welche untere Schranken sind dazu bekannt?
- Wie kann das Problem Independent Set Problem approximiert werden? Welche untere Schranken sind dazu bekannt?
- Wie kann das Problem Steinerbaum Problem approximiert werden? Welche untere Schranken sind dazu bekannt?
- Wie kann das Problem TSP approximiert werden? Welche untere Schranken sind dazu bekannt?
- Wie kann das Problem Δ-TSP approximiert werden? Welche untere Schranken sind dazu bekannt?
- Wie kann das Problem Zentrumsproblem approximiert werden? Welche untere Schranken sind dazu bekannt?

Legende

n : Nicht relevant

g : Grundlagen, die implizit genutzt werden

i : Idee des Beweises oder des Vorgehens

s : Struktur des Beweises oder des Vorgehens

w : Vollständiges Wissen