Algorithmentheorie

Daniel Neuen (Universität Bremen)

WiSe 2023/24

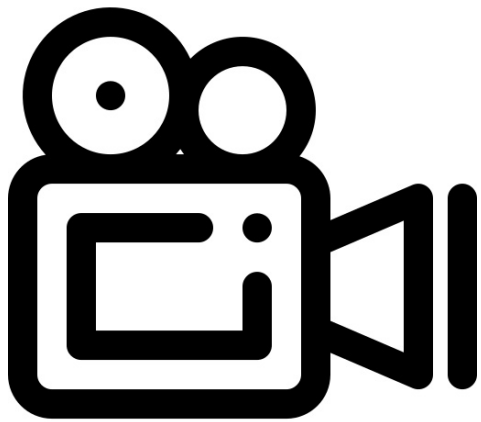
Ausblick

13. Vorlesung

Aufzeichnung der Vorlesung

Diese Vorlesung wird aufgezeichnet und live gestreamt.

- Aufzeichnungen nur der Lehrenden durch sich selbst.
- Bei Rückfragen aus dem Auditorium und Diskussion bitte deutlich anzeigen, falls das Mikro stumm geschaltet werden soll.



Organisatorisches

Klausurtermin: 19. Februar, 14:15-15:45 Uhr, HS 1010 und HS 2010

Wiederholungstermin: 08. März, 10:15-11:45 Uhr, MZH 1380/1400

Nächste Woche:

- Wiederholung
- Fragestunde (Vorschläge für Fragen/Themen in Discord)
- Aufgaben zur Klausurvorbereitung (Abstimmung im StudIP)

Effiziente Algorithmen versus NP-Vollständigkeit

Algorithmenwunschliste

1. **Optimal**
2. **Schnell** (Polynomialzeit)
3. **Universell** (für alle Instanzen)

Komplexitätstheorie sagt, wir können nicht (immer) alles haben.

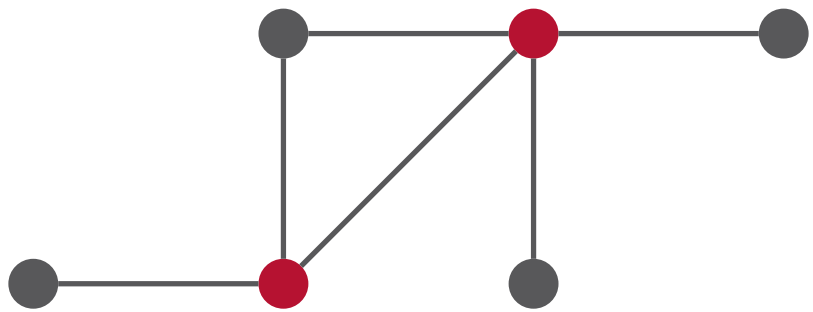
Alternativen

- Eingrenzen der Instanzen (Aufgeben von 3)
  - Ausnutzen spezieller Strukturen (Bsp.: Bäume, planare Graphen)
  - Parametrisierte Algorithmen
- Nicht-polynomielle Laufzeit erlauben (Aufgeben von 2)
  - Parametrisierte Algorithmen
- Suboptimale Algorithmen (Aufgeben von 1)
  - Heuristiken (ohne jede Gütegarantie)
  - Approximationsalgorithmen

Knotenüberdeckung

Definition

Sei $G = (V, E)$ ein Graph. Eine Menge $S \subseteq V$ heißt Knotenüberdeckung (vertex cover) falls für jede Kante $(u, v) \in E$ gilt, dass $u \in S$ oder $v \in S$.



Das Problem Knotenüberdeckung

Knotenüberdeckung (Vertex Cover)

Gegeben: ein Graph $G = (V, E)$.
Gesucht: eine kardinalitätsminimale Knotenüberdeckung S in G.

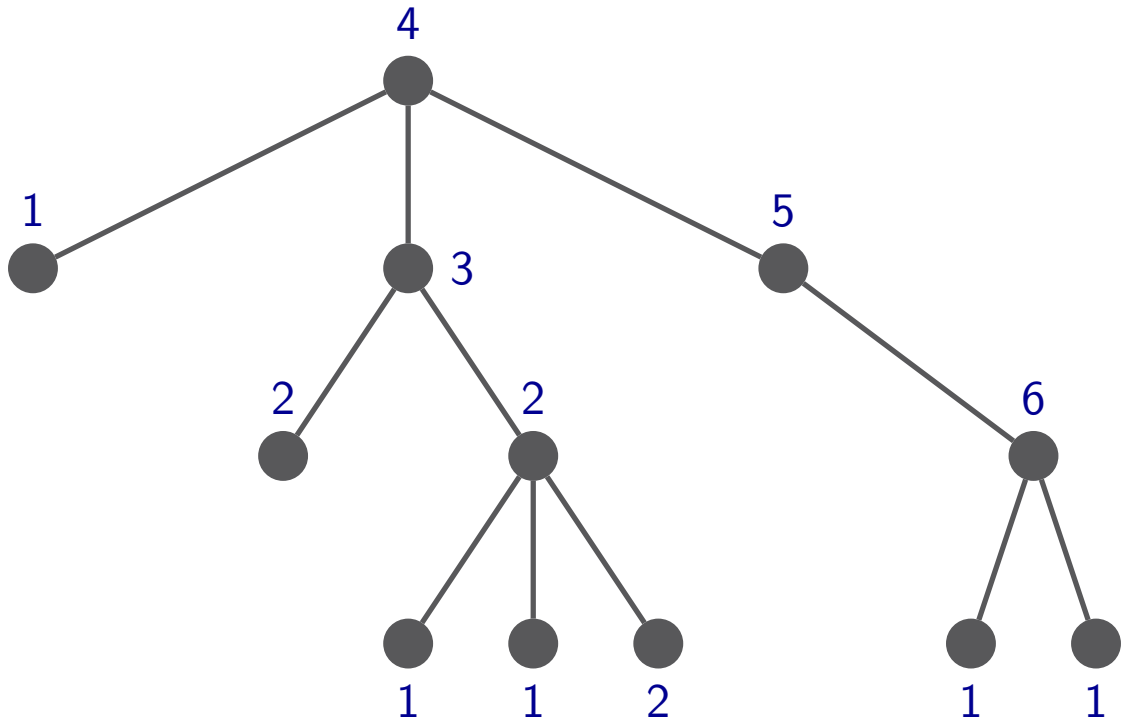
Gewichtete Knotenüberdeckung (Weighted Vertex Cover)

Gegeben: ein Graph $G = (V, E, w)$ mit Gewichten $w(v)$ für alle $v \in V$.
Gesucht: eine gewichtsminimale Knotenüberdeckung S in G.

- Beide Probleme sind NP-schwer, d.h., wir erwarten keinen Polynomialzeit-Algorithmus (der für alle Graphen funktioniert).
- Eingrenzen der Instanzen: Wir konstruieren einen Linearzeit-Algorithmus, der für alle Bäume funktioniert.

Knotenüberdeckung für Bäume

Sei $T = (V, E, w)$ ein gewichteter Baum. Wir wählen einen beliebigen Knoten r als Wurzel. Für jeden Knoten $u \in V$ sei T_u der Unterbaum von Knoten u.



Wir nutzen dynamische Programmierung.

DP-Tabelle

DP-Tabelle (Teilprobleme):
Für $u \in V$ sei $M_+[u]$ der Wert einer gewichtsminimalen Knotenüberdeckung S von T_u mit $u \in S$.

Für $u \in V$ sei $M_-[u]$ der Wert einer gewichtsminimalen Knotenüberdeckung S von T_u mit $u \notin S$.

Basisfälle: Für jedes Blatt gilt $M_+[u] = w(u)$ und $M_-[u] = 0$.

Wie können wir $M_+[u]$ und $M_-[u]$ rekursiv berechnen?

Seien $u_1, \ldots, u_\ell$ die Kinder von u. Dann ist

$$M_-[u] = \sum_{i=1}^{\ell} M_+[u_i]$$

$$M_+[u] = w(u) + \sum_{i=1}^{\ell} \min\Big(M_+[u_i], M_-[u_i]\Big)$$

Der Wert einer gew.-min. Knotenüberd. von T ist $\min(M_+[r], M_-[r])$.

Analyse

Satz

Das Problem Gewichtete Knotenüberdeckung kann auf Bäumen in Zeit $\mathcal{O}(n)$ gelöst werden.

- Berechne DP-Tabelle mit Tabularization.
- Nutze Backtracking um gew.-min. Knotenüberd. zu finden.
- Laufzeit: Für jeden Knoten iterieren über sein Nachbarn. Also $\mathcal{O}(n+m) = \mathcal{O}(n)$, da Bäume genau $n-1$ Kanten haben.

Parametrisierte Algorithmen

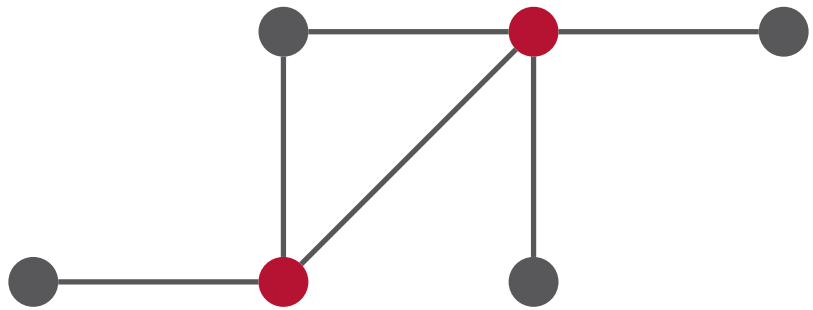
Parametrisierte Algorithmen

- Wir erlauben Nicht-polynomielle Laufzeit, wollen aber möglichst gute Laufzeit erzielen
- Parametrisierte Komplexität = multivariate Komplexität
- Wir betrachten mehr als nur die Eingabelänge
  - Parameter Lösungsgröße
  - Strukturelle Parameter
  - ...
- Effizient lösbar bezüglich Parameter k:
  - Slicewise polynomial XP: $n^{f(k)}$
  - Fixed-parameter tractable FPT: $f(k) \cdot n^c$
- Abhängigkeit zwischen Effizienz und Allgemeinheit durch Parameter beschrieben:
  - Instanzen mit kleinem Parameter: effizient lösbar.

Knotenüberdeckung

Knotenüberdeckung (Entscheidungsvariante)

Gegeben: ein Graph $G = (V, E)$ und eine Zahl $k \geq 1$
Frage: Gibt es eine Knotenüberdeckung S in G mit $|S| \leq k$?

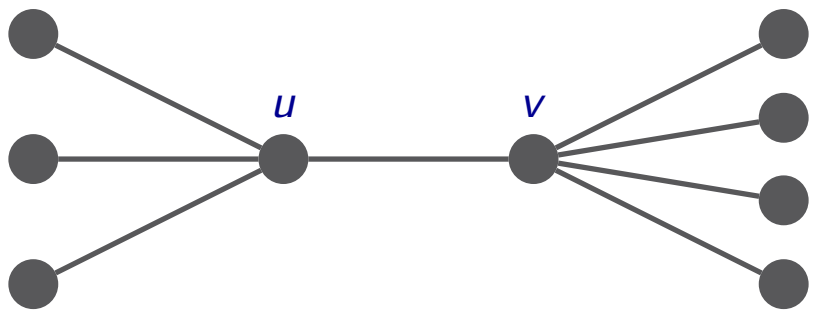


- Trivialer Algorithmus: teste alle $\binom{n}{k}$ möglichen Lösungen.
  - Für $n = 1000$, $k = 10$: $\binom{1000}{10} \approx 2.63 \cdot 10^{23}$.
  - Praktisch unbrauchbar...
- Mit einfachen Einsichten können wir einen besseren Algorithmus bilden.

Besserer Algorithmus

Idee:

- Betrachte beliebige Kante $e = \{u, v\}$.
- Jede Knotenüberdeckung enthält u oder v.
- Teste beide Möglichkeiten.
- Wenn wir einen Knoten w auswählen, dann sind alle inzidenten Kanten überdeckt und können gelöscht werden.



Besserer Algorithmus

VertexCoverRec

Input : Graph $G = (V, E)$ und k
Output : True, falls eine Knotenüberdeckung der Größe k existiert

```
1 if k < 0 then
2     return false
3 if E = ∅ then
4     return true
5 Wähle beliebige Kante e = {u, v} in G
6 Erhalte G_u aus G durch Löschen aller Kanten inzident zu u
7 Erhalte G_v aus G durch Löschen aller Kanten inzident zu v
8 return VertexCoverRec(G_u, k − 1) ∨ VertexCoverRec(G_v, k − 1)
```

Analyse

Satz

Der Algorithmus gibt True zurück genau dann wenn es eine Knotenüberdeckung der Größe k gibt. Die Laufzeit ist $\mathcal{O}(2^k(n+m))$.

Beweis:

- Die Korrektheit folgt per Induktion über die Anzahl der Kanten, da u oder v gewählt werden muss, und alle abgedeckten Kanten entfernt werden.
- Laufzeit:
  - Der Rekursionsbaum hat Verzweigungsgrad 2 und Tiefe höchstens $k+1$.
  - Somit gibt es maximal $2^{k+2} = \mathcal{O}(2^k)$ rekursive Aufrufe.
  - Die Berechnung von G_u und G_v dauert $\mathcal{O}(n+m)$. □

Für $n = 1000$, $k = 10$: $2^k \cdot n^2 \approx 1.03 \cdot 10^9$.

Approximations-Algorithmen

Approximationsalgorithmen

Definition

Ein α-Approximationsalgorithmus für ein Optimierungsproblem ist ein Algorithmus, der

- polynomielle Laufzeit hat,
- für jede Probleminstanz eine zulässige Lösung berechnet, und
- dessen Zielfunktionswert nicht mehr als einen Faktor $\alpha \geq 1$ vom Optimalwert abweicht.

ALG(I): Zielfunktionswert einer Lösung des Algorithmus auf Instanz I
OPT(I): Zielfunktionswert einer optimalen Lösung auf Instanz I

Für Minimierungsprobleme:

$$\text{ALG}(I) \leq \alpha \cdot \text{OPT}(I), \quad \forall I$$

Für Maximierungsprobleme:

$$\text{ALG}(I) \geq \frac{1}{\alpha} \cdot \text{OPT}(I), \quad \forall I$$

Wir bezeichnen $\alpha \geq 1$ als Approximationsfaktor oder Gütegarantie.

Der Schlüssel sind untere Schranken

Dilemma: Wie zeigt man $\mathrm{ALG}(I) \leq \alpha \cdot \mathrm{OPT}(I)$ für alle Instanzen I?

- möglicherweise unendlich viele Instanzen
- wir kennen $\mathrm{OPT}(I)$ nicht

Allgemeines Vorgehen (für Minimierungsprobleme)

1. Finde untere Schranke $\mathrm{LB}(I) \leq \mathrm{OPT}(I)$ für alle Instanzen I.
2. Zeige $\mathrm{ALG}(I) \leq \alpha \cdot \mathrm{LB}(I)$ für alle I.

Dann ist der Algorithmus ein α-Approximationsalgorithmus.

Wie finden wir (gute) untere Schranken an die Optimallösung?

- Kombinatorische untere Schranken... problemspezifisch
- Lineare Programmierung

Matching Schranke

Knotenüberdeckung (Vertex Cover)

Gegeben: ein Graph $G = (V, E)$.
Gesucht: eine kardinalitätsminimale Knotenüberdeckung S in G.

Lemma

Sei M ein inklusionsmaximales Matching in G und sei S eine Knotenüberdeckung. Dann gilt $|S| \geq |M|$.

Beweis:

- Für jede Kante $e \in M$ gilt, dass $e \cap S \neq \emptyset$.
- Jeder Knoten in S ist inzident zu höchstens einer Kante aus M.
- Zusammen folgt $|S| \geq |M|$. $\square$

Approximationsalgorithmus

Algorithmus

1. Finde inklusionsmaximales Matching M
2. Gebe $S = \bigcup_{e \in M} e$ zurück

Satz

Der Algorithmus ist eine 2-Approximation für das Knotenüberdeckungsproblem mit Laufzeit $\mathcal{O}(m + n)$.

Beweis: Sei S die berechnete Lösung. Angenommen S ist keine Knotenüberdeckung. Dann gibt es eine Kante e sodass $e \cap S = \emptyset$. Aber dann ist $M \cup \{e\}$ ebenfalls ein Matching im Widerspruch zur Maximalität von M. Außerdem ist $|S| \leq 2 \cdot |M| \leq 2 \cdot |\,\mathrm{OPT}\,|$ wobei OPT eine kardinalitätsminimale Knotenüberdeckung ist. Also erhalten wir eine 2-Approximation.
Ein inklusionsmaximales Matching kann mittels Greedy-Verfahren in Zeit $\mathcal{O}(m + n)$ gefunden werden. $\square$

LP Schranke

Gewichtete Knotenüberdeckung (Weighted Vertex Cover)

Gegeben: ein Graph $G = (V, E, w)$ mit Gewichten $w(v)$ für alle $v \in V$.
Gesucht: eine gewichtsminimale Knotenüberdeckung S in G.

Wir formulieren das Problem als Ganzzahliges Lineares Programm:

- Lösungsvariable $x_v \in \{0, 1\}$ für jeden Knoten $v \in V$, die entscheidet v zur Überdeckung S gehört oder nicht.

$$\begin{aligned} \min \sum_{v \in V} w(v) \cdot x_v \ &=: z_{\text{ILP}} \\ \text{s.d.} \quad x_u + x_v &\geq 1, && \text{für alle } (u, v) \in E \\ x_v &\in \{0, 1\}, && \text{für alle } v \in V. \end{aligned}$$

Um eine Lösung effizient finden zu können, erlauben wir zusätzliche rationale Lösungen.

LP Schranke

LP Relaxierung:

- Lösungsvariable $x_v \in [0, 1]$ für jeden Knoten $v \in V$.

$$\begin{aligned} \min \sum_{v \in V} w_v \cdot x_v \ &=: z_{\text{LP}} \\ \text{s.t.} \quad x_u + x_v &\geq 1, \qquad \text{für alle } (u, v) \in E \\ x_v &\geq 0, \quad \text{für alle } v \in V. \end{aligned}$$

Wir wissen: $z_{\text{LP}} \leq z_{\text{ILP}}$ (LP Schranke)

Satz

Lineare Programme können in Polynomialzeit optimal gelöst werden.

Einfaches LP Runden

Algorithmus (einfaches LP Runden)

1. Löse LP Relaxierung. Sei x^{LP} sie optimale LP Lösung.
2. Runde x^{LP} zu einer ganzzahligen Lösung x wie folgt:

$$x_v = \begin{cases} 1 & \text{wenn } x_v^{\text{LP}} \geq 1/2 \\ 0 & \text{sonst.} \end{cases}$$

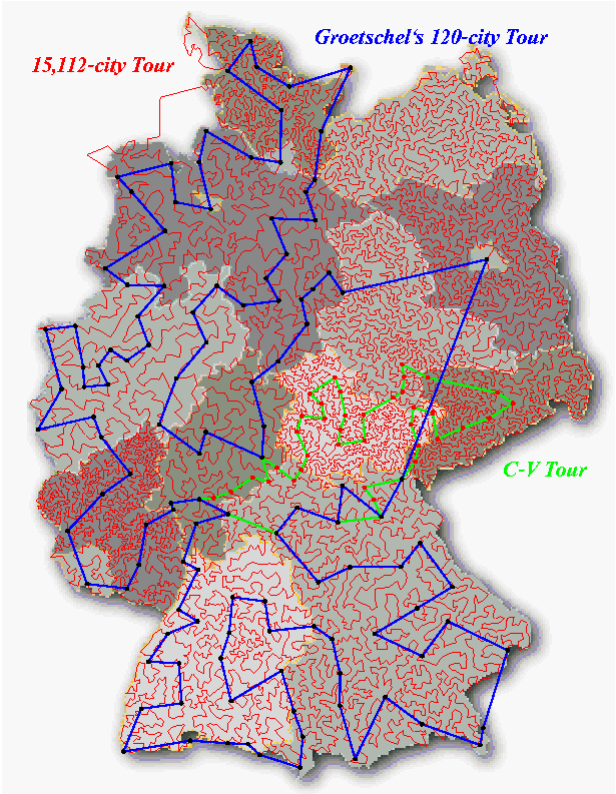
Satz

Der Algorithmus (einfaches LP Runden) ist eine 2-Approximation für das gewichtete Knotenüberdeckungsproblem.

Kombinatorische untere Schranken

Das Problem des Handlungsreisenden

Das Problem des Handlungsreisenden



http://www.math.uwaterloo.ca/tsp/

Engl. Traveling Salesperson Problem (TSP)

- Gegeben eine Menge an Städten und paarweise Distanzen dazwischen, finde eine kürzeste Tour durch alle Städte.
- eines der wichtigsten und meist untersuchten kombinatorischen Optimierungsprobleme
- TSP ist NP-schwer.
- Benchmark Problem für viele exakte und approximative Techniken

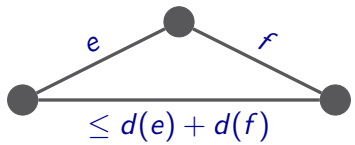
Auf dem Bild: 45 Städtetour (Commis-Voyageur, 1832), 120 Städte (Grötschel, 1980), 15.112 Städte (Bixby, Chvátal, Cook 2001)

Sehr sehenswert: Bill Cooks Webseite mit Geschichte, Methoden, Zahlen.

Metrisches TSP I

Input: vollständiger Graph $G = (V, E)$
Distanzen $d : E \to \mathbb{Q}_+$ mit $d(u, w) \leq d(u, v) + d(v, w)$ für alle $u, v, w \in V$

Finde: Hamiltonkreis C in G mit minimaler Länge $d(C) := \sum_{e \in C} d(e)$



Gute untere Schranke an optimale TSP Länge?

Satz

Die Kosten einer optimalen TSP Tour in $G = (V, E, c)$ sind mindestens so hoch wie die Kosten eines MST in G.

Beweis.

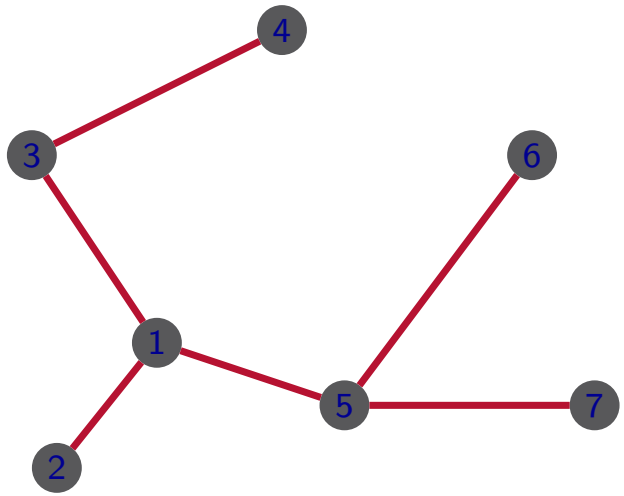
- Betrachte eine optimale TSP Tour in G mit Kosten OPT.
- Entfernen einer beliebigen Kante ergibt einen aufspannenden Baum mit Kosten $c \leq \text{OPT}$.
- Die Kosten eines MST sind von oben beschränkt durch $c \leq \text{OPT}$. □

Algorithmus Double-Tree

Algorithmus baut auf der unteren Schranke auf.

Algorithmus Double-Tree

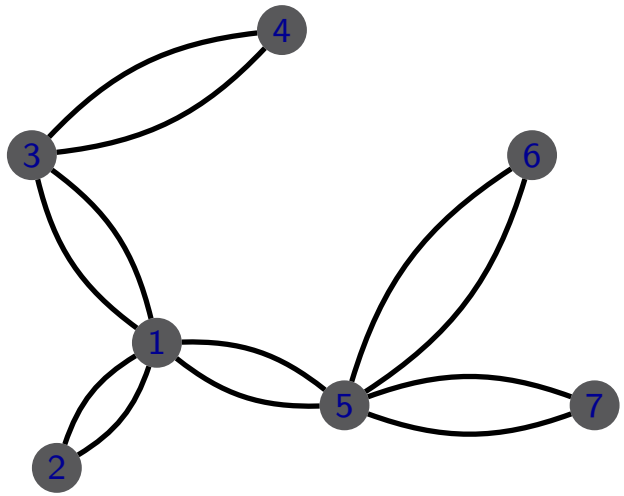
Algorithmus baut auf der unteren Schranke auf.



1. Finde einen MST T in G

Algorithmus Double-Tree

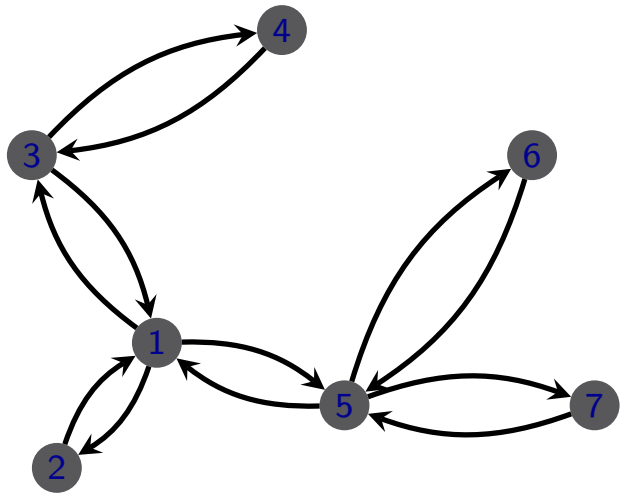
Algorithmus baut auf der unteren Schranke auf.



1. Finde einen MST T in G
2. Verdoppele alle Kanten in T

Algorithmus Double-Tree

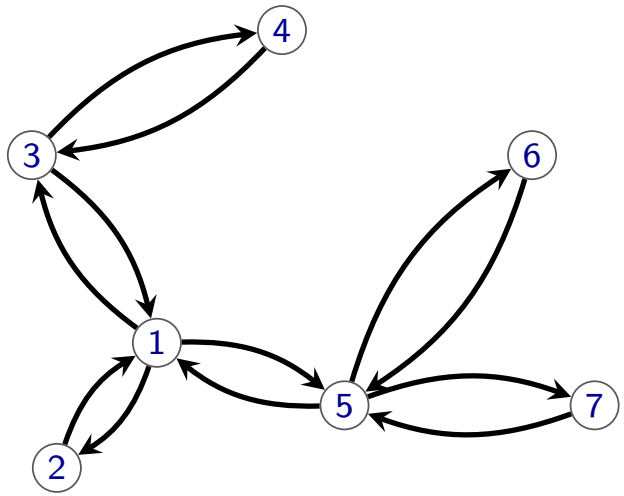
Algorithmus baut auf der unteren Schranke auf.



1. Finde einen MST T in G
2. Verdoppele alle Kanten in T
3. Bestimme eine Eulertour T'

Algorithmus Double-Tree

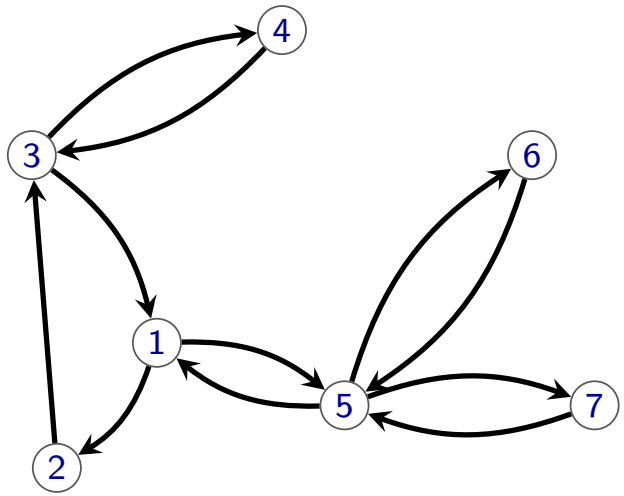
Algorithmus baut auf der unteren Schranke auf.



1. Finde einen MST T in G
2. Verdoppele alle Kanten in T
3. Bestimme eine Eulertour T'
4. Tour in Reihenfolge von T' mit Shortcuts wenn Knoten schon besucht

Algorithmus Double-Tree

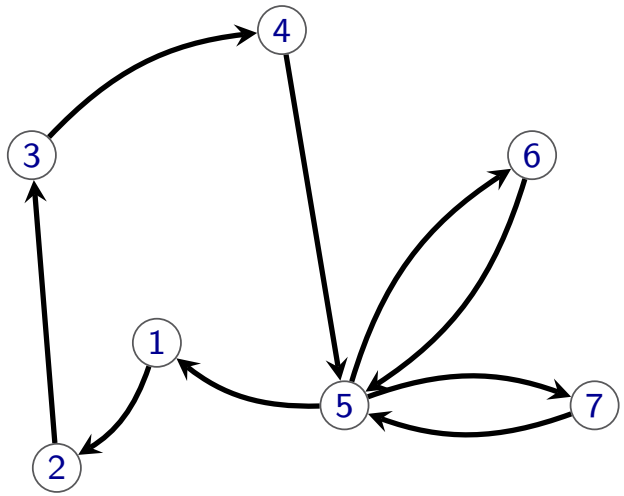
Algorithmus baut auf der unteren Schranke auf.



1. Finde einen MST T in G
2. Verdoppele alle Kanten in T
3. Bestimme eine Eulertour T'
4. Tour in Reihenfolge von T' mit Shortcuts wenn Knoten schon besucht

Algorithmus Double-Tree

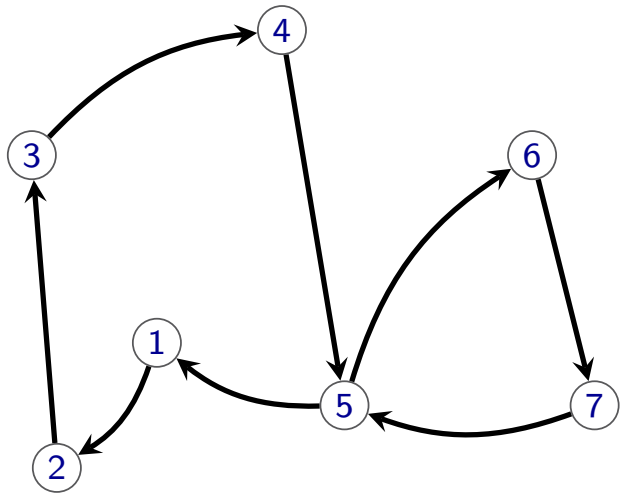
Algorithmus baut auf der unteren Schranke auf.



1. Finde einen MST T in G
2. Verdoppele alle Kanten in T
3. Bestimme eine Eulertour T'
4. Tour in Reihenfolge von T' mit Shortcuts wenn Knoten schon besucht

Algorithmus Double-Tree

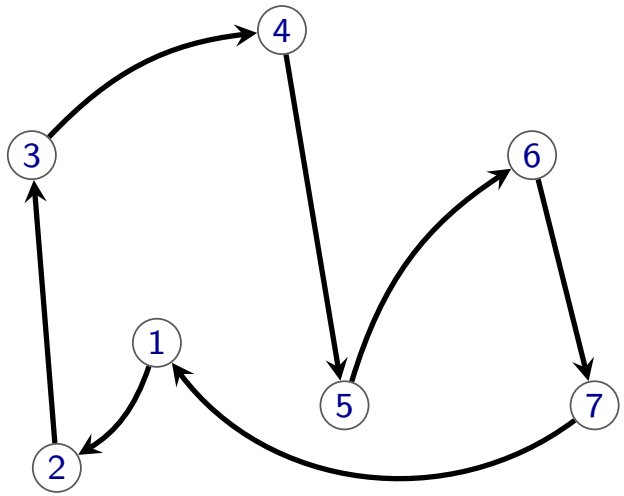
Algorithmus baut auf der unteren Schranke auf.



1. Finde einen MST T in G
2. Verdoppele alle Kanten in T
3. Bestimme eine Eulertour T'
4. Tour in Reihenfolge von T' mit Shortcuts wenn Knoten schon besucht

Algorithmus Double-Tree

Algorithmus baut auf der unteren Schranke auf.



1. Finde einen MST T in G
2. Verdoppele alle Kanten in T
3. Bestimme eine Eulertour T'
4. Tour in Reihenfolge von T' mit Shortcuts wenn Knoten schon besucht

Satz

Double-Tree ist ein 2-Approximationsalgorithmus für metrisches TSP.

Zusammenfassung

- Für viele interessante Probleme können wir nicht alles haben (Optimal, Schnell, Universell).
- Häufig reicht es ein Ziel aufzugeben.
- Methoden aus dieser Vorlesung tauchen häufig als Subroutine auf.

Weiterführende Vorlesungen:

Approximation Algorithms, Prof. Nicole Megow

Parametrisierte Algorithmen, Prof. Sebastian Siebertz