Proseminar Graphenalgorithmen (WS 2007/08)

Traveling Salesman Problem

Einführung zum Problem des Handlungsreisenden

Florian Weingarten

Betreuer:
Dr. Walter Unger
Lehrstuhl für Informatik 1
Algorithmen und Komplexität
RWTH Aachen

Aachen, 23. Januar 2008

Inhalt

Problemstellung

Problem

Handlungsreisender soll Rundreise durch n Städte planen, dabei

- jede Stadt genau einmal besuchen
- zum Ausgangsort zurück kehren
- zurückgelegte Strecke soll möglichst kurz sein

Anwendung

- „Stadt“ und „Strecke“ nicht wörtlich gemeint
- Tourenplanung
- Logistik
- Entwurf von Mikrochips
- ...

Problemstellung

Problem

Handlungsreisender soll Rundreise durch n Städte planen, dabei

- jede Stadt genau einmal besuchen
- zum Ausgangsort zurück kehren
- zurückgelegte Strecke soll möglichst kurz sein

Anwendung

- „Stadt“ und „Strecke“ nicht wörtlich gemeint
- Tourenplanung
- Logistik
- Entwurf von Mikrochips
- ...

Problemstellung

Problem

Handlungsreisender soll Rundreise durch n Städte planen, dabei

- jede Stadt genau einmal besuchen
- zum Ausgangsort zurück kehren
- zurückgelegte Strecke soll möglichst kurz sein

Anwendung

- „Stadt“ und „Strecke“ nicht wörtlich gemeint
- Tourenplanung
- Logistik
- Entwurf von Mikrochips
- ...

Modellierung

Definition

- $G = (V, E)$ ungerichteter gewichteter Graph (V endlich)
- Kostenfunktion $c : E \to \mathbb{N}$
- TSP-*Tour* (oder *Rundreise*) π ist ein *Hamiltonkreis* in G
- π heißt optimal, wenn die Summe der Kantengewichte minimal ist

Optimierungsvariante

Finde eine optimale TSP-Tour

Entscheidungsvariante

Feststellen, ob TSP-Tour existiert mit Kosten $\leq b$

Modellierung

Definition

- $G = (V, E)$ ungerichteter gewichteter Graph (V endlich)
- Kostenfunktion $c : E \to \mathbb{N}$
- TSP-*Tour* (oder *Rundreise*) π ist ein *Hamiltonkreis* in G
- π heißt optimal, wenn die Summe der Kantengewichte minimal ist

Optimierungsvariante

Finde eine optimale TSP-Tour

Entscheidungsvariante

Feststellen, ob TSP-Tour existiert mit Kosten $\leq b$

Modellierung

Definition

- $G = (V, E)$ ungerichteter gewichteter Graph (V endlich)
- Kostenfunktion $c : E \to \mathbb{N}$
- TSP-*Tour* (oder *Rundreise*) π ist ein *Hamiltonkreis* in G
- π heißt optimal, wenn die Summe der Kantengewichte minimal ist

Optimierungsvariante

Finde eine optimale TSP-Tour

Entscheidungsvariante

Feststellen, ob TSP-Tour existiert mit Kosten $\leq b$

Modellierung

Definition

- $G = (V, E)$ ungerichteter gewichteter Graph (V endlich)
- Kostenfunktion $c : E \to \mathbb{N}$
- TSP-*Tour* (oder *Rundreise*) π ist ein *Hamiltonkreis* in G
- π heißt optimal, wenn die Summe der Kantengewichte minimal ist

Optimierungsvariante

Finde eine optimale TSP-Tour

Entscheidungsvariante

Feststellen, ob TSP-Tour existiert mit Kosten $\leq b$

Lösungen

Anzahl der möglichen Lösungen

Anzahl ungerichteter Hamiltonkreise auf n Knoten: $\frac{1}{2}(n-1)!$

Kosten von optimalen Lösungen

Jede Lösung ist mindestens so teuer wie ein MST

Lösungen

Anzahl der möglichen Lösungen

Anzahl ungerichteter Hamiltonkreise auf n Knoten: $\frac{1}{2}(n-1)!$

Kosten von optimalen Lösungen

Jede Lösung ist mindestens so teuer wie ein MST

Lösungen

Anzahl der möglichen Lösungen

Anzahl ungerichteter Hamiltonkreise auf n Knoten: $\frac{1}{2}(n-1)!$

Kosten von optimalen Lösungen

Jede Lösung ist mindestens so teuer wie ein MST

Beispiel

(a) A-B-C-D-E-A

(b) A-B-C-E-D-A

(c) A-B-D-C-E-A

(d) A-B-D-E-C-A

(e) A-B-E-C-D-A

(f) A-B-E-D-C-A

(g) A-C-B-D-E-A

(h) A-C-B-E-D-A

(i) A-C-D-B-E-A

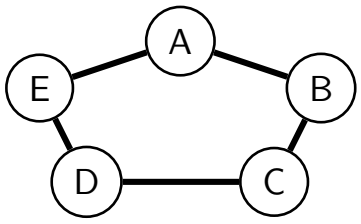
(j) A-C-E-B-D-A

(k) A-D-B-C-E-A

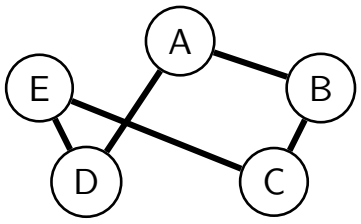
(l) A-D-C-B-E-A

Abbildung: Alle $\frac{1}{2} \cdot (5-1)! = 12$ Hamiltonkreise in K_5 mit fixem Startpunkt A

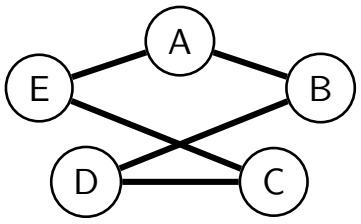
Beispiel



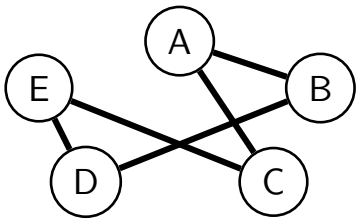
(a) A-B-C-D-E-A



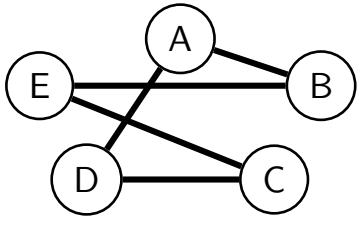
(b) A-B-C-E-D-A



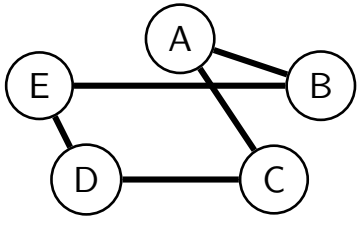
(c) A-B-D-C-E-A



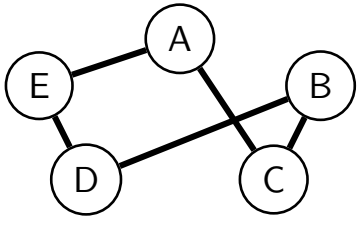
(d) A-B-D-E-C-A



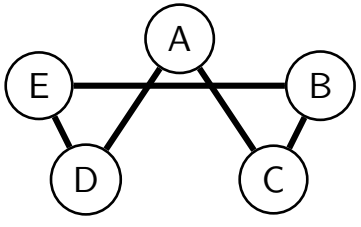
(e) A-B-E-C-D-A



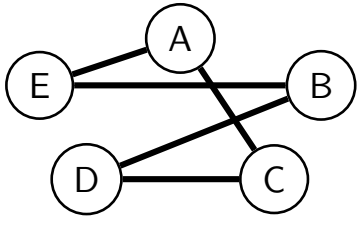
(f) A-B-E-D-C-A



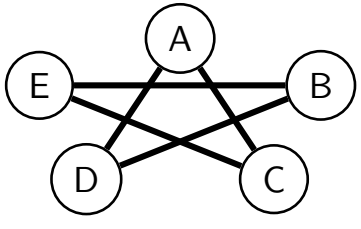
(g) A-C-B-D-E-A



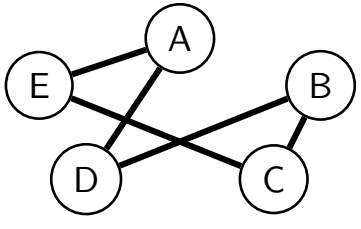
(h) A-C-B-E-D-A



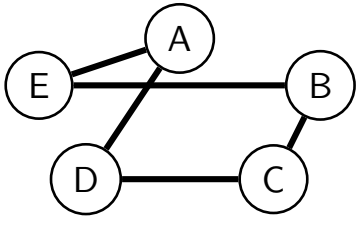
(i) A-C-D-B-E-A



(j) A-C-E-B-D-A



(k) A-D-B-C-E-A



(l) A-D-C-B-E-A

Abbildung: Alle $\frac{1}{2} \cdot (5-1)! = 12$ Hamiltonkreise in K_5 mit fixem Startpunkt A

Brute Force

Naiver Ansatz

- Alle möglichen Lösungen generieren
- Für jede Lösung die Kosten bestimmen
- Lösung mit minimalen Kosten wählen

n	Anzahl möglicher Lösungen	Benötigte Zeit
5	12	
10	181440	
15	$4.36 \cdot 10^{10}$	
20	$6.08 \cdot 10^{16}$	
25	$3.10 \cdot 10^{23}$	
30	$4.42 \cdot 10^{30}$	4.42 Nanosekunden
35	$1.48 \cdot 10^{38}$	0.15 Sekunden
40	$1.02 \cdot 10^{46}$	118 Tage
45	$1.33 \cdot 10^{54}$	$4.21 \cdot 10^{7}$ **Jahre**
50	$3.04 \cdot 10^{62}$	$9.6 \cdot 10^{15}$ **Jahre**

(Annahme: 10^{30} Lösungen pro Nanosekunde generieren)

Brute Force

Naiver Ansatz

- Alle möglichen Lösungen generieren
- Für jede Lösung die Kosten bestimmen
- Lösung mit minimalen Kosten wählen

n	Anzahl möglicher Lösungen	Benötigte Zeit
5	12	
10	181440	
15	$4.36 \cdot 10^{10}$	
20	$6.08 \cdot 10^{16}$	
25	$3.10 \cdot 10^{23}$	
30	$4.42 \cdot 10^{30}$	4.42 Nanosekunden
35	$1.48 \cdot 10^{38}$	0.15 Sekunden
40	$1.02 \cdot 10^{46}$	118 Tage
45	$1.33 \cdot 10^{54}$	$4.21 \cdot 10^{7}$ **Jahre**
50	$3.04 \cdot 10^{62}$	$9.6 \cdot 10^{15}$ **Jahre**

(Annahme: 10^{30} Lösungen pro Nanosekunde generieren)

Brute Force

Naiver Ansatz

- Alle möglichen Lösungen generieren
- Für jede Lösung die Kosten bestimmen
- Lösung mit minimalen Kosten wählen

n	Anzahl möglicher Lösungen	Benötigte Zeit
5	12	
10	181440	
15	$4.36 \cdot 10^{10}$	
20	$6.08 \cdot 10^{16}$	
25	$3.10 \cdot 10^{23}$	
30	$4.42 \cdot 10^{30}$	4.42 Nanosekunden
35	$1.48 \cdot 10^{38}$	0.15 Sekunden
40	$1.02 \cdot 10^{46}$	118 Tage
45	$1.33 \cdot 10^{54}$	$4.21 \cdot 10^{7}$ **Jahre**
50	$3.04 \cdot 10^{62}$	$9.6 \cdot 10^{15}$ **Jahre**

(Annahme: 10^{30} Lösungen pro Nanosekunde generieren)

Die Komplexitätsklasse NP

Definition

NP ist die Klasse der **Entscheidungsprobleme**, die von einer NTM in Polynomialzeit entschieden werden können

Alternative Definition

- Lösungen lassen sich als Zertifikat polynomieller Länge kodieren
- es existiert Polynomialzeitverifizierer

Also:

Für Probleme in NP kann man

- Mögliche Lösungen „kurz“ kodieren
- Effizient testen, ob mögliche Lösung tatsächlich korrekt ist

Die Komplexitätsklasse NP

Definition

NP ist die Klasse der **Entscheidungsprobleme**, die von einer NTM in Polynomialzeit entschieden werden können

Alternative Definition

- Lösungen lassen sich als Zertifikat polynomieller Länge kodieren
- es existiert Polynomialzeitverifizierer

Also:

Für Probleme in NP kann man

- Mögliche Lösungen „kurz“ kodieren
- Effizient testen, ob mögliche Lösung tatsächlich korrekt ist

Die Komplexitätsklasse NP

Definition

NP ist die Klasse der **Entscheidungsprobleme**, die von einer NTM in Polynomialzeit entschieden werden können

Alternative Definition

- Lösungen lassen sich als Zertifikat polynomieller Länge kodieren
- es existiert Polynomialzeitverifizierer

Also:

Für Probleme in NP kann man

- Mögliche Lösungen „kurz“ kodieren
- Effizient testen, ob mögliche Lösung tatsächlich korrekt ist

Die Komplexitätsklasse NP

Definition

NP ist die Klasse der **Entscheidungsprobleme**, die von einer NTM in Polynomialzeit entschieden werden können

Alternative Definition

- Lösungen lassen sich als Zertifikat polynomieller Länge kodieren
- es existiert Polynomialzeitverifizierer

Also:

Für Probleme in NP kann man

- Mögliche Lösungen „kurz“ kodieren
- Effizient testen, ob mögliche Lösung tatsächlich korrekt ist

Definitionen

Definition

L' auf L *reduzierbar* ($L' \leq L$), wenn f existiert, mit $w \in L' \Leftrightarrow f(w) \in L$

Definition

L heißt NP-*hart* (oder NP-*schwer*), wenn $L' \leq_p L$ für alle $L' \in \mathrm{NP}$

Definition

L heißt NP-*vollständig* wenn $L \in \mathrm{NP}$ und L zusätzlich NP-schwer ist

Definitionen

Definition

L' auf L *reduzierbar* ($L' \leq L$), wenn f existiert, mit $w \in L' \Leftrightarrow f(w) \in L$

Definition

L heißt NP-*hart* (oder NP-*schwer*), wenn $L' \leq_p L$ für alle $L' \in \mathrm{NP}$

Definition

L heißt NP-*vollständig* wenn $L \in \mathrm{NP}$ und L zusätzlich NP-schwer ist

Definitionen

Definition

L' auf L *reduzierbar* ($L' \leq L$), wenn f existiert, mit $w \in L' \Leftrightarrow f(w) \in L$

Definition

L heißt NP-*hart* (oder NP-*schwer*), wenn $L' \leq_p L$ für alle $L' \in \mathrm{NP}$

Definition

L heißt NP-*vollständig* wenn $L \in \mathrm{NP}$ und L zusätzlich NP-schwer ist

Definitionen

Definition

L' auf L *reduzierbar* ($L' \leq L$), wenn f existiert, mit $w \in L' \Leftrightarrow f(w) \in L$

Definition

L heißt NP-*hart* (oder NP-*schwer*), wenn $L' \leq_p L$ für alle $L' \in \mathrm{NP}$

Definition

L heißt NP-*vollständig* wenn $L \in \mathrm{NP}$ und L zusätzlich NP-schwer ist

NP-Vollständigkeit

Ziel

Entscheidungsvariante von TSP ist NP-vollständig: $\text{TSP} \in \text{NPC}$

Dazu:

- $\text{TSP} \in \text{NP}$
- TSP ist NP-hart

NP-Vollständigkeit

Ziel

Entscheidungsvariante von TSP ist NP-vollständig: $\text{TSP} \in \text{NPC}$

Dazu:

- $\text{TSP} \in \text{NP}$
- TSP ist NP-hart

NP-Vollständigkeit

Ziel

Entscheidungsvariante von TSP ist NP-vollständig: $\text{TSP} \in \text{NPC}$

Dazu:

- $\text{TSP} \in \text{NP}$
- TSP ist NP-hart

TSP ist in NP

Kodierung möglicher Lösungen

- Eine Lösung ist ein Hamiltonkreis
- Eindeutig bestimmt durch die (geordnete) Angabe der Knoten
- Alle Knoten durchnummerieren: $V = \{1, ..., n\}$
- Pro Knoten werden $\lceil \log_2(n) \rceil$ Bits benötigt
- Kodierung des Kreises in $O(n \cdot \log n)$ möglich

Testen ob Lösung korrekt ist

- Jeweils zwei Knoten einlesen
- Kosten von Kante in Kostenmatrix abrufen
- Kosten aller Kanten addieren
- Testen ob Summe kleiner als Schranke b ist

TSP ist in NP

Kodierung möglicher Lösungen

- Eine Lösung ist ein Hamiltonkreis
- Eindeutig bestimmt durch die (geordnete) Angabe der Knoten
- Alle Knoten durchnummerieren: $V = \{1, ..., n\}$
- Pro Knoten werden $\lceil \log_2(n) \rceil$ Bits benötigt
- Kodierung des Kreises in $O(n \cdot \log n)$ möglich

Testen ob Lösung korrekt ist

- Jeweils zwei Knoten einlesen
- Kosten von Kante in Kostenmatrix abrufen
- Kosten aller Kanten addieren
- Testen ob Summe kleiner als Schranke b ist

TSP ist in NP

Kodierung möglicher Lösungen

- Eine Lösung ist ein Hamiltonkreis
- Eindeutig bestimmt durch die (geordnete) Angabe der Knoten
- Alle Knoten durchnummerieren: $V = \{1, ..., n\}$
- Pro Knoten werden $\lceil \log_2(n) \rceil$ Bits benötigt
- Kodierung des Kreises in $O(n \cdot \log n)$ möglich

Testen ob Lösung korrekt ist

- Jeweils zwei Knoten einlesen
- Kosten von Kante in Kostenmatrix abrufen
- Kosten aller Kanten addieren
- Testen ob Summe kleiner als Schranke b ist

TSP ist in NP: Beispiel einer Kodierung

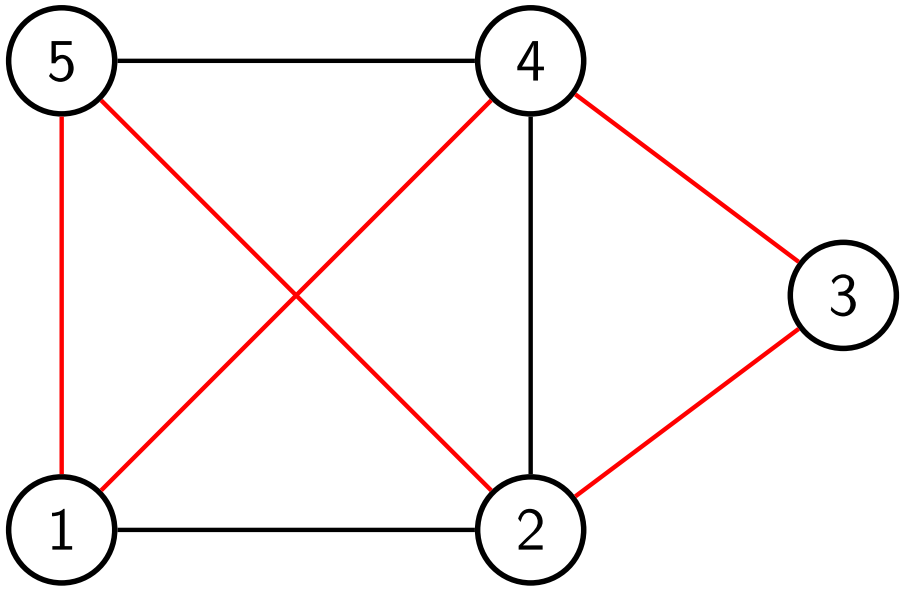
Hamiltonkreis (5-2-3-4-1-5):

$$\underbrace{101}_{5}\ \underbrace{010}_{2}\ \underbrace{011}_{3}\ \underbrace{100}_{4}\ \underbrace{001}_{1}$$

Länge:

$$\underbrace{n \cdot \lceil log_2(n) \rceil}_{\in O(n \cdot log\ n)} = 5 \cdot 3 = 15 \text{ Bit}$$

TSP ist in NP: Beispiel einer Kodierung



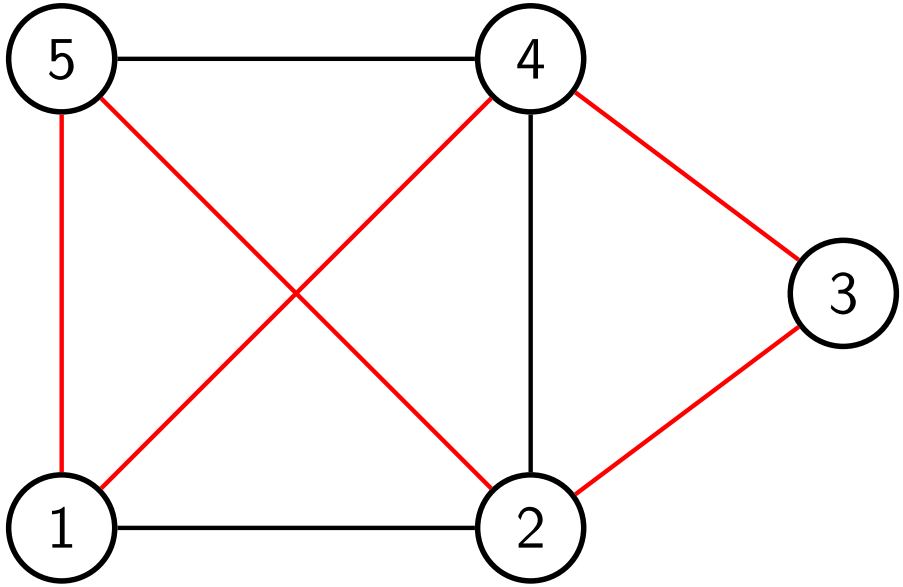
Hamiltonkreis (5-2-3-4-1-5):

$$\underbrace{101}_{5}\ \underbrace{010}_{2}\ \underbrace{011}_{3}\ \underbrace{100}_{4}\ \underbrace{001}_{1}$$

Länge:

$$\underbrace{n \cdot \lceil log_2(n) \rceil}_{\in O(n \cdot log\ n)} = 5 \cdot 3 = 15 \text{ Bit}$$

TSP ist in NP: Beispiel einer Kodierung



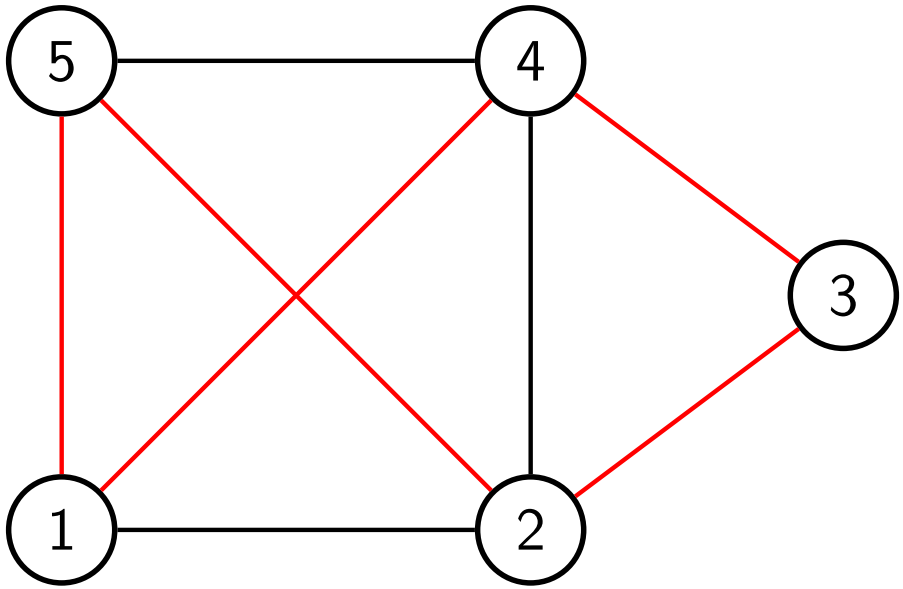
Hamiltonkreis (5-2-3-4-1-5):

$$\underbrace{101}_{5}\ \underbrace{010}_{2}\ \underbrace{011}_{3}\ \underbrace{100}_{4}\ \underbrace{001}_{1}$$

Länge:

$$\underbrace{n \cdot \lceil log_2(n) \rceil}_{\in O(n \cdot log\ n)} = 5 \cdot 3 = 15 \text{ Bit}$$

TSP ist in NP: Beispiel einer Kodierung



Hamiltonkreis (5-2-3-4-1-5):

$$\underbrace{101}_{5}\ \underbrace{010}_{2}\ \underbrace{011}_{3}\ \underbrace{100}_{4}\ \underbrace{001}_{1}$$

Länge:

$$\underbrace{n \cdot \lceil log_2(n) \rceil}_{\in O(n \cdot log\ n)} = 5 \cdot 3 = 15 \text{ Bit}$$

TSP ist in NP

Folgerung

- Lösungen lassen sich polynomiell kodieren
- Test auf Korrektheit in Polynomialzeit

$$\Rightarrow \text{TSP} \in \text{NP}$$

Bleibt zu zeigen

NP-Härte von TSP, d.h. $\forall L \in \text{NP} : L \leq_p \text{TSP}$

TSP ist in NP

Folgerung

- Lösungen lassen sich polynomiell kodieren
- Test auf Korrektheit in Polynomialzeit

$$\Rightarrow \text{TSP} \in \text{NP}$$

Bleibt zu zeigen

NP-Härte von TSP, d.h. $\forall L \in \text{NP} : L \leq_p \text{TSP}$

TSP ist in NP

Folgerung

- Lösungen lassen sich polynomiell kodieren
- Test auf Korrektheit in Polynomialzeit

$$\Rightarrow \text{TSP} \in \text{NP}$$

Bleibt zu zeigen

NP-Härte von TSP, d.h. $\forall L \in \text{NP} : L \leq_p \text{TSP}$

TSP ist NP-hart

Feststellung

polynomielle Reduktion ist transitiv: $L \leq_p L' \wedge L' \leq_p L'' \Rightarrow L \leq_p L''$

Folgerung

L ist NP-hart, wenn anderes NP-hartes Problem L' existiert mit $L' \leq_p L$

Bekannte Reduktionen

- Richard P. Karp: $SAT \leq_p ... \leq_p HC$
- Satz von Cook: $SAT \in \mathrm{NPC}$
- also: HC ist NP-schwer

TSP ist NP-hart

Feststellung

polynomielle Reduktion ist transitiv: $L \leq_p L' \wedge L' \leq_p L'' \Rightarrow L \leq_p L''$

Folgerung

L ist NP-hart, wenn anderes NP-hartes Problem L' existiert mit $L' \leq_p L$

Bekannte Reduktionen

- Richard P. Karp: $SAT \leq_p ... \leq_p HC$
- Satz von Cook: $SAT \in \mathrm{NPC}$
- also: HC ist NP-schwer

TSP ist NP-hart

Feststellung

polynomielle Reduktion ist transitiv: $L \leq_p L' \wedge L' \leq_p L'' \Rightarrow L \leq_p L''$

Folgerung

L ist NP-hart, wenn anderes NP-hartes Problem L' existiert mit $L' \leq_p L$

Bekannte Reduktionen

- Richard P. Karp: $SAT \leq_p ... \leq_p HC$
- Satz von Cook: $SAT \in \mathrm{NPC}$
- also: HC ist NP-schwer

TSP ist NP-hart

Feststellung

polynomielle Reduktion ist transitiv: $L \leq_p L' \wedge L' \leq_p L'' \Rightarrow L \leq_p L''$

Folgerung

L ist NP-hart, wenn anderes NP-hartes Problem L' existiert mit $L' \leq_p L$

Bekannte Reduktionen

- Richard P. Karp: $SAT \leq_p ... \leq_p HC$
- Satz von Cook: $SAT \in \mathrm{NPC}$
- also: HC ist NP-schwer

TSP ist NP-hart

Ziel

Reduktion des Hamiltonkreisproblems auf TSP

Vorgehen

Polynomialzeit-Algorithmus finden, um

- Eingaben von HC zu Eingaben von TSP zu transformieren
- HC-Instanz ist Ja-Instanz **genau dann, wenn** die Transformation eine Ja-Instanz von TSP ist

TSP ist NP-hart

Ziel

Reduktion des Hamiltonkreisproblems auf TSP

Vorgehen

Polynomialzeit-Algorithmus finden, um

- Eingaben von HC zu Eingaben von TSP zu transformieren
- HC-Instanz ist Ja-Instanz **genau dann, wenn** die Transformation eine Ja-Instanz von TSP ist

TSP ist NP-hart

Ziel

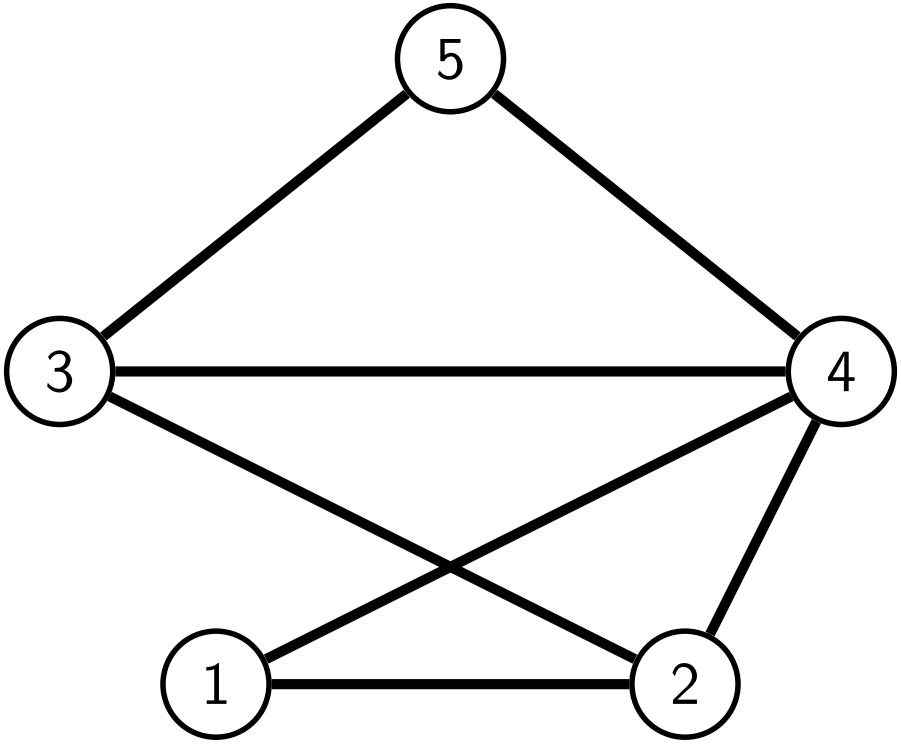
Reduktion des Hamiltonkreisproblems auf TSP

Vorgehen

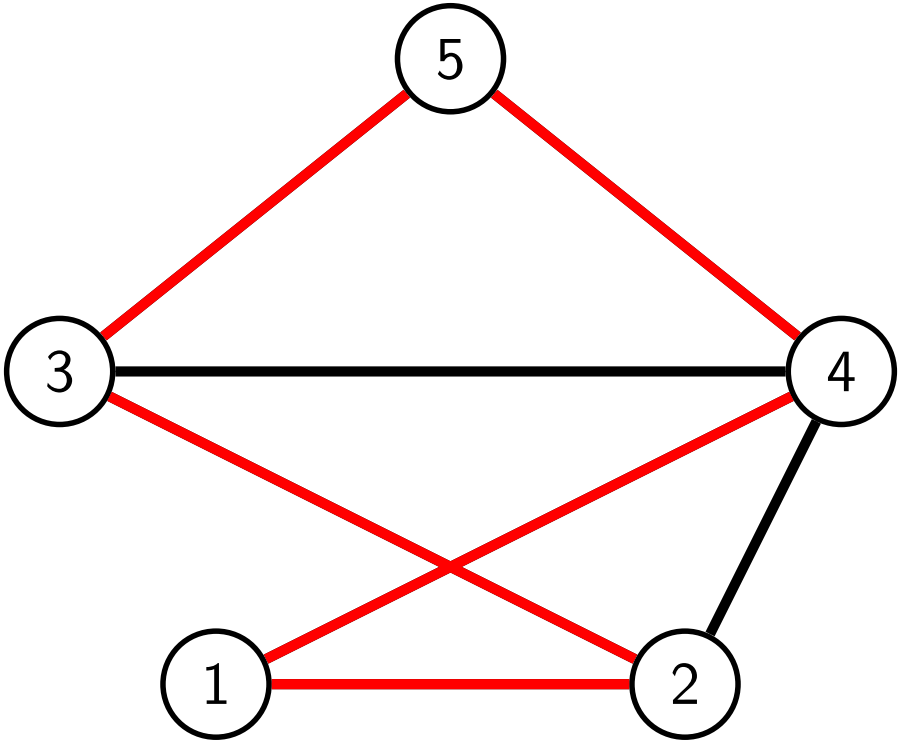
Polynomialzeit-Algorithmus finden, um

- Eingaben von HC zu Eingaben von TSP zu transformieren
- HC-Instanz ist Ja-Instanz **genau dann, wenn** die Transformation eine Ja-Instanz von TSP ist

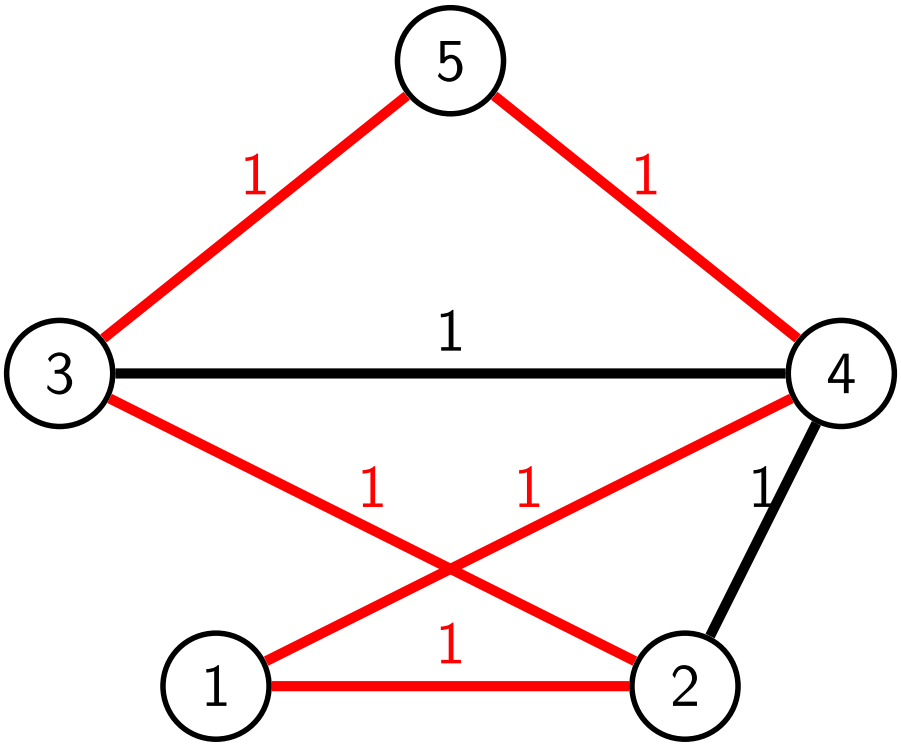
Beweisidee: $\mathrm{HC} \leq_p \mathrm{TSP}$ (Transformation)



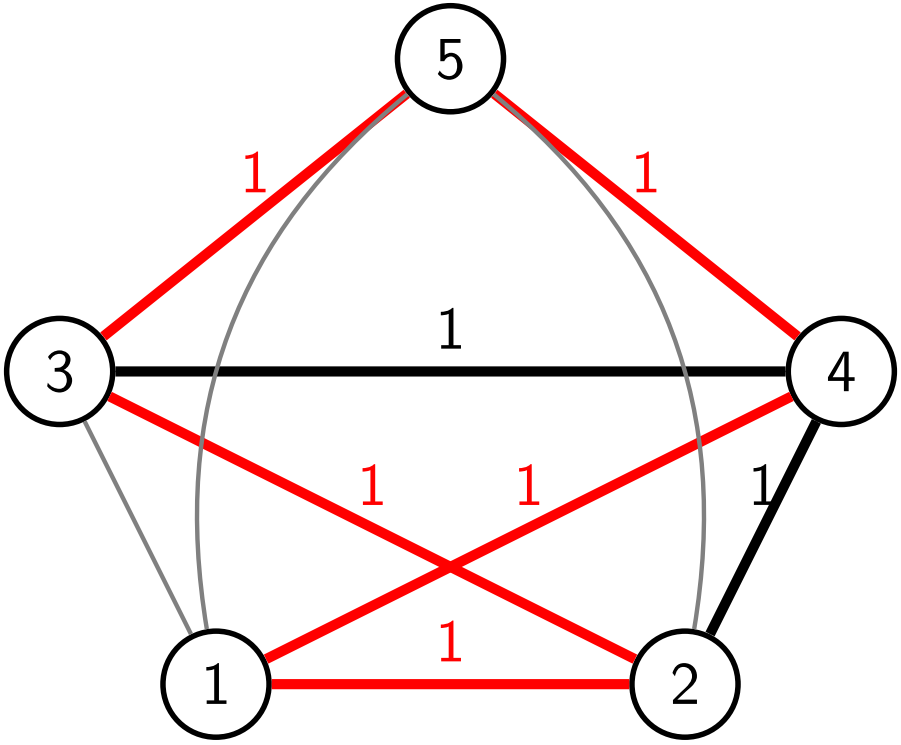
Beweisidee: $\mathrm{HC} \leq_p \mathrm{TSP}$ (Transformation)



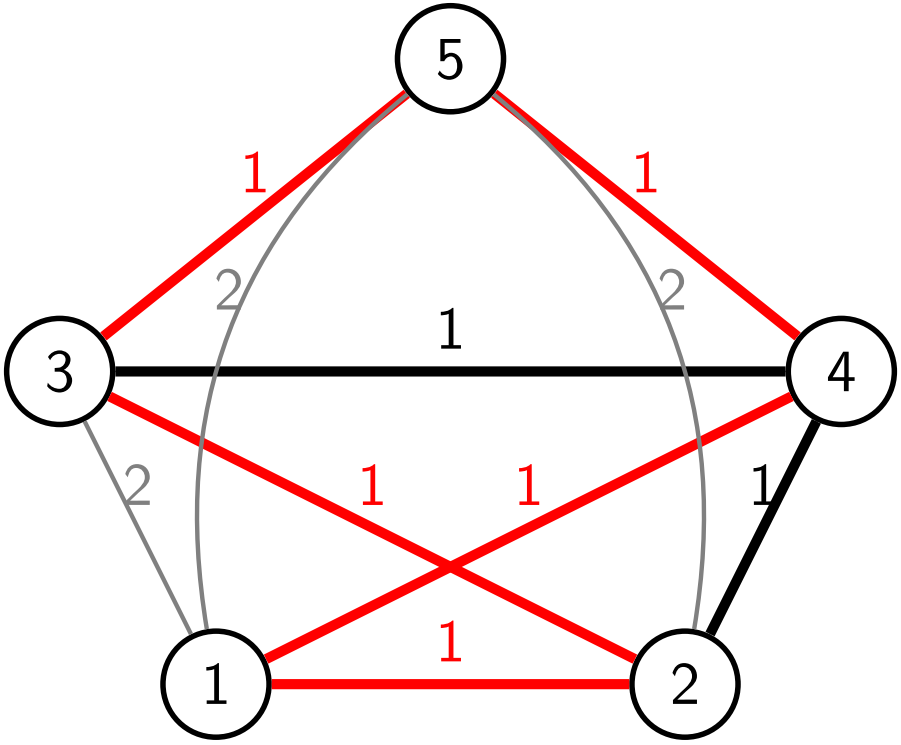
Beweisidee: $\mathrm{HC} \leq_p \mathrm{TSP}$ (Transformation)



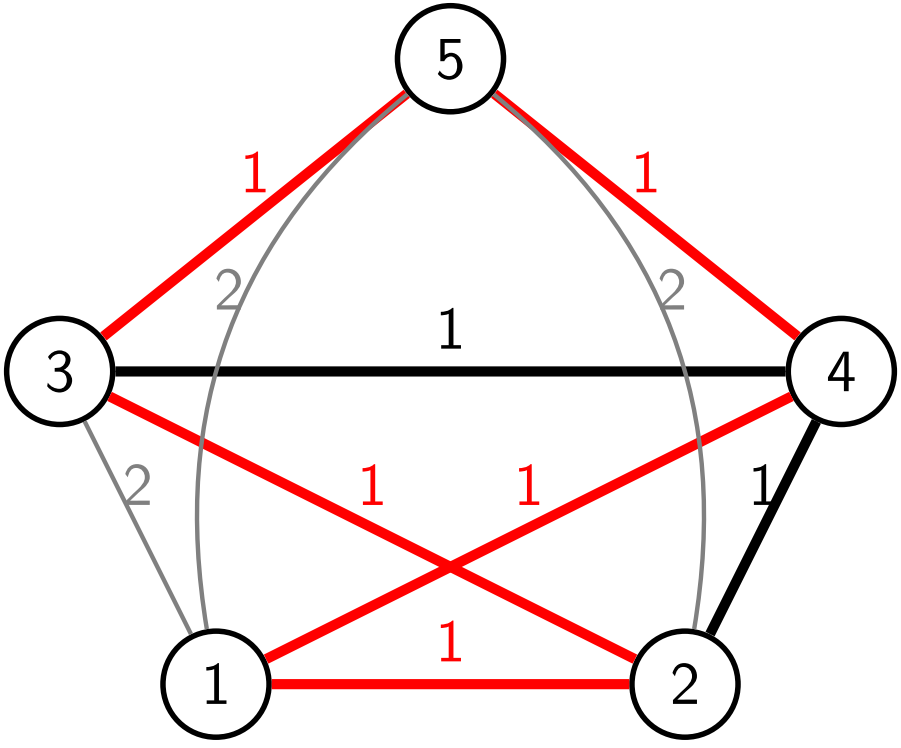
Beweisidee: $\mathrm{HC} \leq_p \mathrm{TSP}$ (Transformation)



Beweisidee: $\mathrm{HC} \leq_p \mathrm{TSP}$ (Transformation)



Beweisidee: $\mathrm{HC} \leq_p \mathrm{TSP}$ (Transformation)



$$b := |V| = 5$$

TSP ist NP-hart (Reduktion von HC)

Zu zeigen

G enthält Hamiltonkreis $\Leftrightarrow$ K_b enthält TSP-Tour mit Kosten $\leq b$

TSP ist NP-hart (Reduktion von HC)

Zu zeigen

G enthält Hamiltonkreis $\Leftrightarrow$ K_b enthält TSP-Tour mit Kosten $\leq b$

TSP ist NP-hart (Reduktion von HC)

„⇒“

- Jeder Hamiltonkreis enthält genau $b = |V|$ Knoten und Kanten
- Jede Kante des Kreises ist in G enthalten
- Jede Kante hat in K_b also Gewicht 1 (nach Def. von c)
- Die Summe der b Kanten mit jeweils Gewicht 1 ist also b
- $b \leq b$, also existiert in K_b ein Hamiltonkreis mit Kosten $\leq b$

TSP ist NP-hart (Reduktion von HC)

„$\Rightarrow$“

- Jeder Hamiltonkreis enthält genau $b = |V|$ Knoten und Kanten
- Jede Kante des Kreises ist in G enthalten
- Jede Kante hat in K_b also Gewicht 1 (nach Def. von c)
- Die Summe der b Kanten mit jeweils Gewicht 1 ist also b
- $b \leq b$, also existiert in K_b ein Hamiltonkreis mit Kosten $\leq b$

TSP ist NP-hart (Reduktion von HC)

„$\Rightarrow$“

- Jeder Hamiltonkreis enthält genau $b = |V|$ Knoten und Kanten
- Jede Kante des Kreises ist in G enthalten
- Jede Kante hat in K_b also Gewicht 1 (nach Def. von c)
- Die Summe der b Kanten mit jeweils Gewicht 1 ist also b
- $b \leq b$, also existiert in K_b ein Hamiltonkreis mit Kosten $\leq b$

TSP ist NP-hart (Reduktion von HC)

„$\Rightarrow$“

- Jeder Hamiltonkreis enthält genau $b = |V|$ Knoten und Kanten
- Jede Kante des Kreises ist in G enthalten
- Jede Kante hat in K_b also Gewicht 1 (nach Def. von c)
- Die Summe der b Kanten mit jeweils Gewicht 1 ist also b
- $b \leq b$, also existiert in K_b ein Hamiltonkreis mit Kosten $\leq b$

TSP ist NP-hart (Reduktion von HC)

„$\Rightarrow$“

- Jeder Hamiltonkreis enthält genau $b = |V|$ Knoten und Kanten
- Jede Kante des Kreises ist in G enthalten
- Jede Kante hat in K_b also Gewicht 1 (nach Def. von c)
- Die Summe der b Kanten mit jeweils Gewicht 1 ist also b
- $b \leq b$, also existiert in K_b ein Hamiltonkreis mit Kosten $\leq b$

TSP ist NP-hart (Reduktion von HC)

„⇒“

- Jeder Hamiltonkreis enthält genau $b = |V|$ Knoten und Kanten
- Jede Kante des Kreises ist in G enthalten
- Jede Kante hat in K_b also Gewicht 1 (nach Def. von c)
- Die Summe der b Kanten mit jeweils Gewicht 1 ist also b
- $b \leq b$, also existiert in K_b ein Hamiltonkreis mit Kosten $\leq b$

TSP ist NP-hart (Reduktion von HC)

„⇒“

- Jeder Hamiltonkreis enthält genau $b = |V|$ Knoten und Kanten
- Jede Kante des Kreises ist in G enthalten
- Jede Kante hat in K_b also Gewicht 1 (nach Def. von c)
- Die Summe der b Kanten mit jeweils Gewicht 1 ist also b
- $b \leq b$, also existiert in K_b ein Hamiltonkreis mit Kosten $\leq b$

TSP ist NP-hart (Reduktion von HC)

„⇐“

- Ang. es ex. TSP-Tour mit Kosten $\leq b$ in K_b
- $c(i,j) \geq 1$ für alle Kanten der Tour
- Summe der Kosten **mindestens** b
- Nach Annahme: Summe ist **nicht größer** als b
- Summe **genau** b
- Jede Kante der Tour in K_b hat Kosten 1, d.h. kommt auch in G vor
- G ist hamiltonsch

TSP ist NP-hart (Reduktion von HC)

„$\Leftarrow$“

- Ang. es ex. TSP-Tour mit Kosten $\leq b$ in K_b
- $c(i,j) \geq 1$ für alle Kanten der Tour
- Summe der Kosten **mindestens** b
- Nach Annahme: Summe ist **nicht größer** als b
- Summe **genau** b
- Jede Kante der Tour in K_b hat Kosten 1, d.h. kommt auch in G vor
- G ist hamiltonsch

TSP ist NP-hart (Reduktion von HC)

„⇐“

- Ang. es ex. TSP-Tour mit Kosten $\leq b$ in K_b
- $c(i,j) \geq 1$ für alle Kanten der Tour
- Summe der Kosten **mindestens** b
- Nach Annahme: Summe ist **nicht größer** als b
- Summe **genau** b
- Jede Kante der Tour in K_b hat Kosten 1, d.h. kommt auch in G vor
- G ist hamiltonsch

TSP ist NP-hart (Reduktion von HC)

„⇐“

- Ang. es ex. TSP-Tour mit Kosten $\leq b$ in K_b
- $c(i,j) \geq 1$ für alle Kanten der Tour
- Summe der Kosten **mindestens** b
- Nach Annahme: Summe ist **nicht größer** als b
- Summe **genau** b
- Jede Kante der Tour in K_b hat Kosten 1, d.h. kommt auch in G vor
- G ist hamiltonsch

TSP ist NP-hart (Reduktion von HC)

„⇐“

- Ang. es ex. TSP-Tour mit Kosten $\leq b$ in K_b
- $c(i,j) \geq 1$ für alle Kanten der Tour
- Summe der Kosten **mindestens** b
- Nach Annahme: Summe ist **nicht größer** als b
- Summe **genau** b
- Jede Kante der Tour in K_b hat Kosten 1, d.h. kommt auch in G vor
- G ist hamiltonsch

TSP ist NP-hart (Reduktion von HC)

„$\Leftarrow$“

- Ang. es ex. TSP-Tour mit Kosten $\leq b$ in K_b
- $c(i,j) \geq 1$ für alle Kanten der Tour
- Summe der Kosten **mindestens** b
- Nach Annahme: Summe ist **nicht größer** als b
- Summe **genau** b
- Jede Kante der Tour in K_b hat Kosten 1, d.h. kommt auch in G vor
- G ist hamiltonsch

TSP ist NP-hart (Reduktion von HC)

„⇐“

- Ang. es ex. TSP-Tour mit Kosten $\leq b$ in K_b
- $c(i,j) \geq 1$ für alle Kanten der Tour
- Summe der Kosten **mindestens** b
- Nach Annahme: Summe ist **nicht größer** als b
- Summe **genau** b
- Jede Kante der Tour in K_b hat Kosten 1, d.h. kommt auch in G vor
- G ist hamiltonsch

TSP ist NP-hart (Reduktion von HC)

„$\Leftarrow$“

- Ang. es ex. TSP-Tour mit Kosten $\leq b$ in K_b
- $c(i,j) \geq 1$ für alle Kanten der Tour
- Summe der Kosten **mindestens** b
- Nach Annahme: Summe ist **nicht größer** als b
- Summe **genau** b
- Jede Kante der Tour in K_b hat Kosten 1, d.h. kommt auch in G vor
- G ist hamiltonsch

TSP ist NP-hart (Reduktion von HC)

„⇐“

- Ang. es ex. TSP-Tour mit Kosten $\leq b$ in K_b
- $c(i,j) \geq 1$ für alle Kanten der Tour
- Summe der Kosten **mindestens** b
- Nach Annahme: Summe ist **nicht größer** als b
- Summe **genau** b
- Jede Kante der Tour in K_b hat Kosten 1, d.h. kommt auch in G vor
- G ist hamiltonsch

Komplexität von TSP: Zusammenfassung

Wir haben also gezeigt:

TSP ist NP-vollständig

Folgerung I

TSP ist deterministisch in Exponentialzeit lösbar (da $\mathrm{NP} \subseteq \mathit{EXPTIME}$)

Folgerung II

TSP ist deterministisch **nicht effizient lösbar** (es sei denn $\mathrm{P} = \mathrm{NP}$)

Komplexität von TSP: Zusammenfassung

Wir haben also gezeigt:

TSP ist NP-vollständig

Folgerung I

TSP ist deterministisch in Exponentialzeit lösbar (da $\mathrm{NP} \subseteq \mathit{EXPTIME}$)

Folgerung II

TSP ist deterministisch **nicht effizient lösbar** (es sei denn $\mathrm{P} = \mathrm{NP}$)

Komplexität von TSP: Zusammenfassung

Wir haben also gezeigt:

TSP ist NP-vollständig

Folgerung I

TSP ist deterministisch in Exponentialzeit lösbar (da NP $\subseteq$ *EXPTIME*)

Folgerung II

TSP ist deterministisch **nicht effizient lösbar** (es sei denn P = NP)

Komplexität von TSP: Zusammenfassung

Wir haben also gezeigt:

TSP ist NP-vollständig

Folgerung I

TSP ist deterministisch in Exponentialzeit lösbar (da $\mathrm{NP} \subseteq \mathit{EXPTIME}$)

Folgerung II

TSP ist deterministisch **nicht effizient lösbar** (es sei denn $\mathrm{P} = \mathrm{NP}$)

Wenn nicht perfekt, dann vielleicht „möglichst gut“?

Frage

Können wir TSP denn wenigstens effizient approximieren?

Antwort

Nein! (zumindest nicht ohne Einschränkungen an die Eingaben)

Satz

Für TSP (in seiner allgemeinen Form!) existiert kein polynomieller α-Approximationsalgorithmus ($\alpha \in \mathbb{R}, \alpha > 1$), es sei denn $\mathrm{P} = \mathrm{NP}$

Wenn nicht perfekt, dann vielleicht „möglichst gut“?

Frage

Können wir TSP denn wenigstens effizient approximieren?

Antwort

Nein! (zumindest nicht ohne Einschränkungen an die Eingaben)

Satz

Für TSP (in seiner allgemeinen Form!) existiert kein polynomieller α-Approximationsalgorithmus ($\alpha \in \mathbb{R}, \alpha > 1$), es sei denn $\mathrm{P} = \mathrm{NP}$

Wenn nicht perfekt, dann vielleicht „möglichst gut“?

Frage

Können wir TSP denn wenigstens effizient approximieren?

Antwort

Nein! (zumindest nicht ohne Einschränkungen an die Eingaben)

Satz

Für TSP (in seiner allgemeinen Form!) existiert kein polynomieller α-Approximationsalgorithmus ($\alpha \in \mathbb{R}, \alpha > 1$), es sei denn $\mathrm{P} = \mathrm{NP}$

Wenn nicht perfekt, dann vielleicht „möglichst gut“?

Frage

Können wir TSP denn wenigstens effizient approximieren?

Antwort

Nein! (zumindest nicht ohne Einschränkungen an die Eingaben)

Satz

Für TSP (in seiner allgemeinen Form!) existiert kein polynomieller α-Approximationsalgorithmus ($\alpha \in \mathbb{R}, \alpha > 1$), es sei denn $\mathrm{P} = \mathrm{NP}$

Wenn nicht perfekt, dann vielleicht „möglichst gut“?

Frage

Können wir TSP denn wenigstens effizient approximieren?

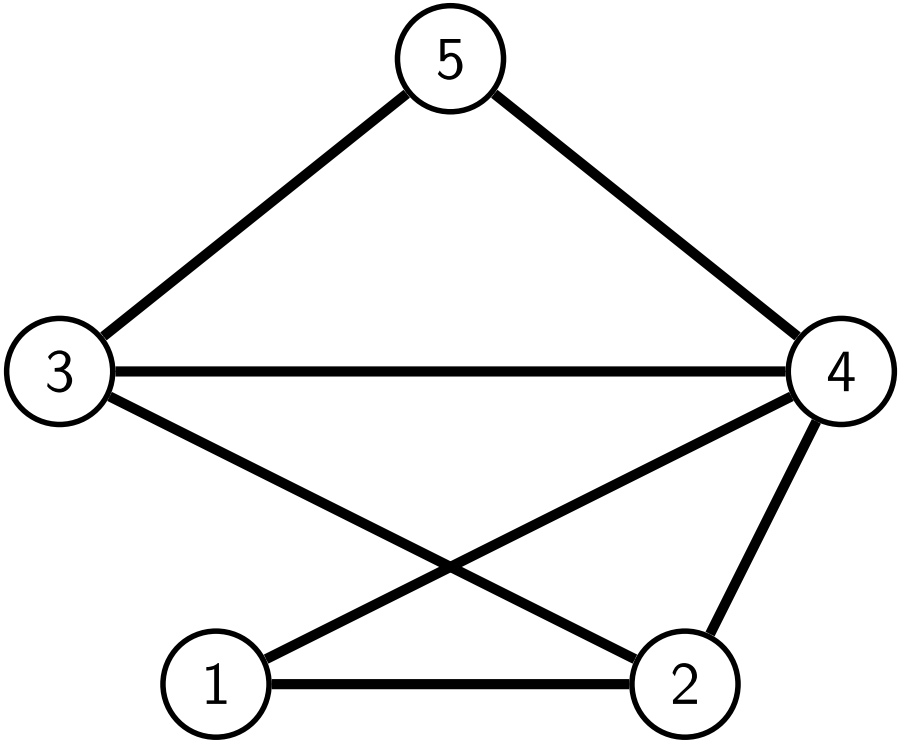
Antwort

Nein! (zumindest nicht ohne Einschränkungen an die Eingaben)

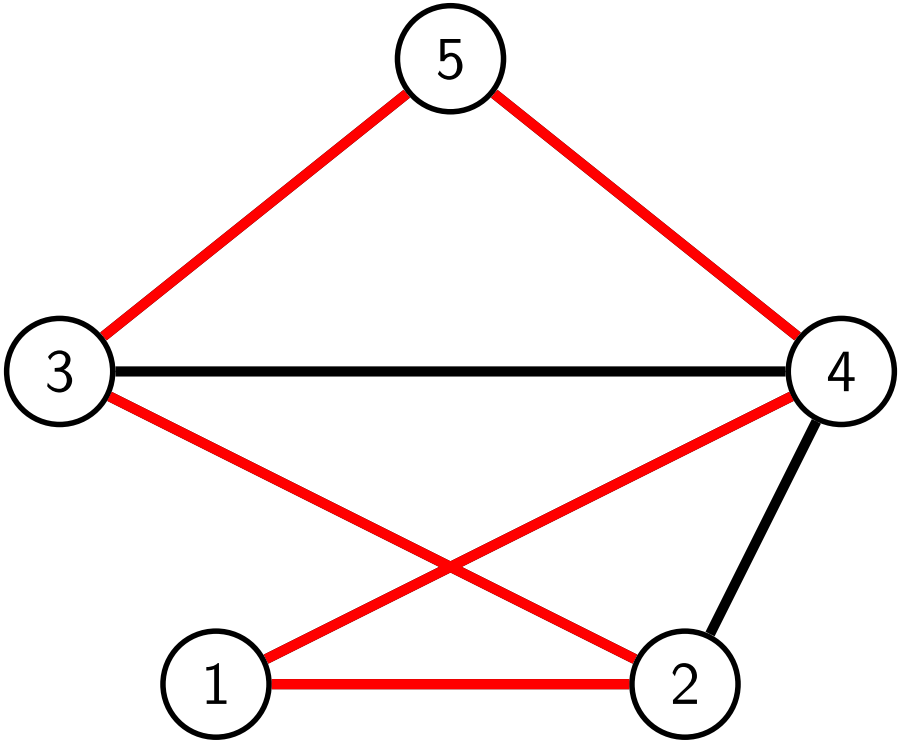
Satz

Für TSP *(in seiner allgemeinen Form!) existiert kein polynomieller* α*-Approximationsalgorithmus* ($\alpha \in \mathbb{R}, \alpha > 1$)*, es sei denn* $\mathrm{P} = \mathrm{NP}$

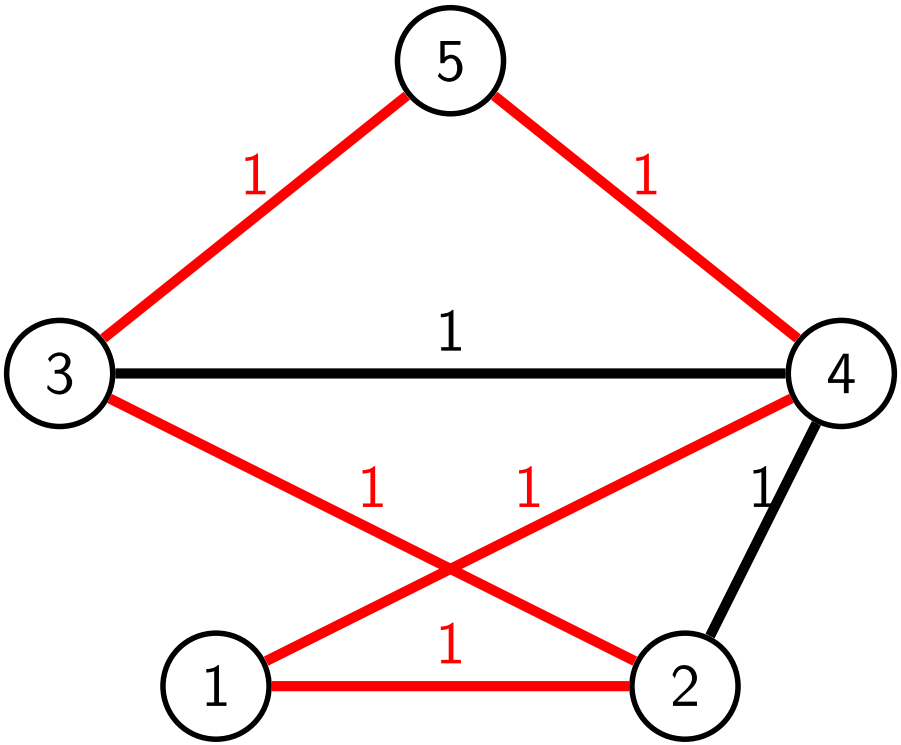
Beweisidee



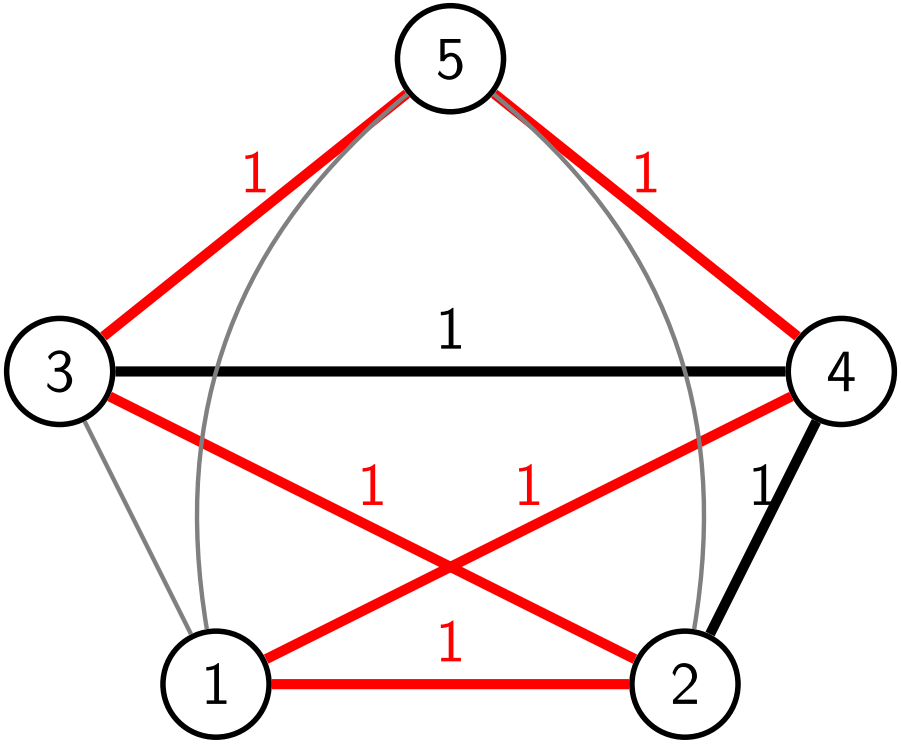
Beweisidee



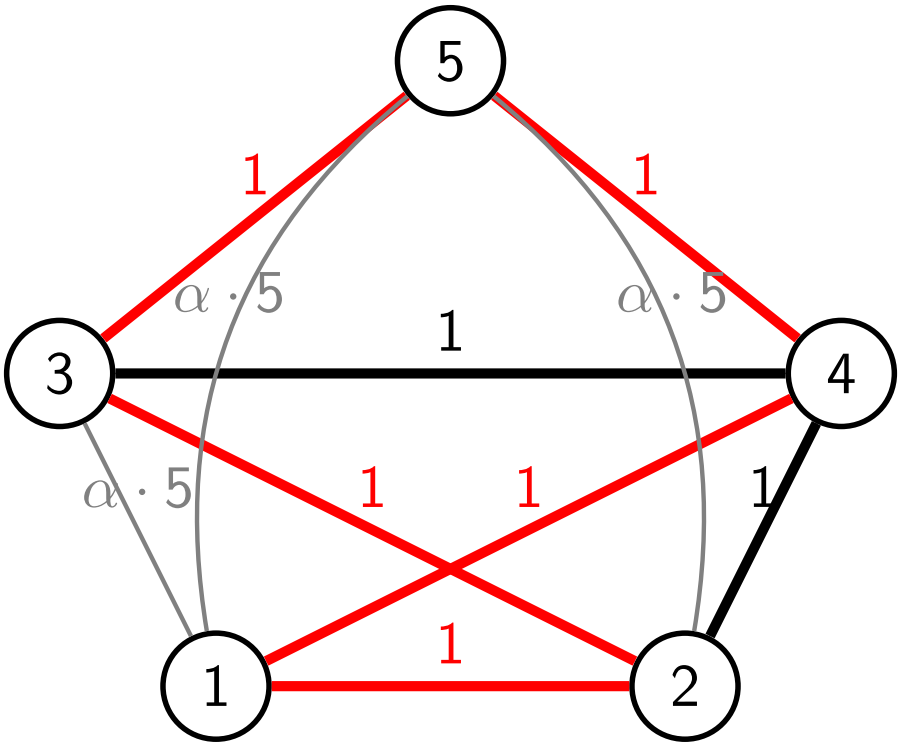
Beweisidee



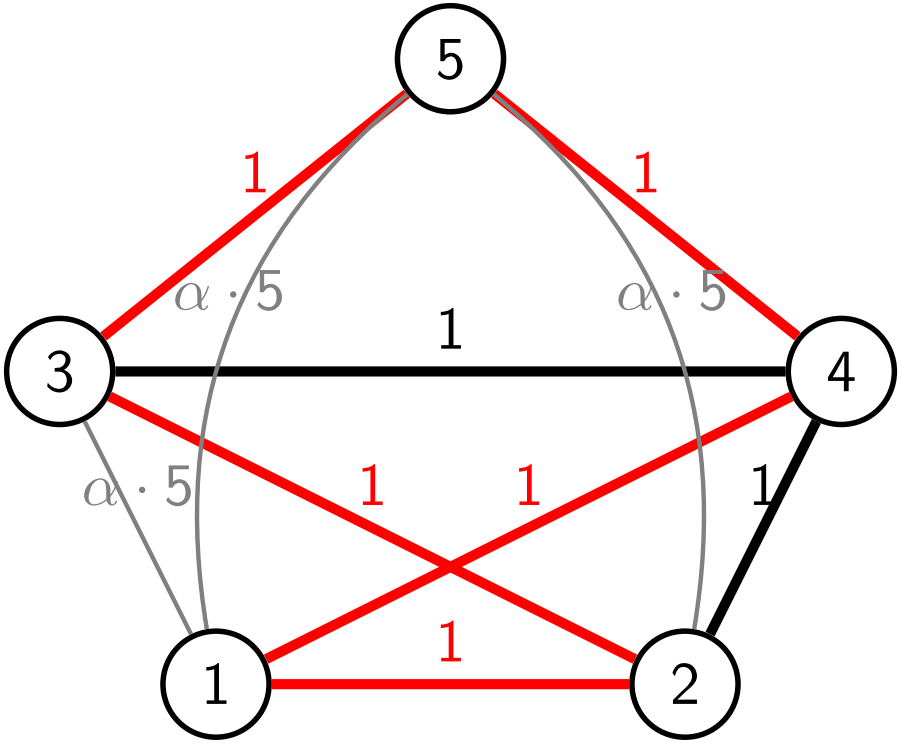
Beweisidee



Beweisidee



Beweisidee



$$b := |V| = 5$$

Angenommen es existiert A_α

Angenommen G ist hamiltonsch

- Es existiert ein Hamiltonkreis (n Knoten, n Kanten) in G
- In TSP Instanz hat dieser Kreis die Kosten n
- A_α findet Tour, deren Kosten höchstens $\alpha \cdot n$ sind

Angenommen G ist **nicht** hamiltonsch

- In K_n existiert aufjedenfall ein Hamiltonkreis
- Da in G keiner existiert, enthält Kreis eine Kante, die nicht in G enthalten ist
- Kante hat Kosten $\alpha \cdot n$
- Kosten jeder TSP-Tour sind also mindestens $\alpha \cdot n + (n-1) \cdot 1$
- Kosten der Tour, die A_α findet, sind damit auch mindestens $\alpha \cdot n + (n-1) > \alpha \cdot n$

Angenommen es existiert A_α

Angenommen G ist hamiltonsch

- Es existiert ein Hamiltonkreis (n Knoten, n Kanten) in G
- In TSP Instanz hat dieser Kreis die Kosten n
- A_α findet Tour, deren Kosten höchstens $\alpha \cdot n$ sind

Angenommen G ist **nicht** hamiltonsch

- In K_n existiert aufjedenfall ein Hamiltonkreis
- Da in G keiner existiert, enthält Kreis eine Kante, die nicht in G enthalten ist
- Kante hat Kosten $\alpha \cdot n$
- Kosten jeder TSP-Tour sind also mindestens $\alpha \cdot n + (n-1) \cdot 1$
- Kosten der Tour, die A_α findet, sind damit auch mindestens $\alpha \cdot n + (n-1) > \alpha \cdot n$

Angenommen es existiert A_α

Angenommen G ist hamiltonsch

- Es existiert ein Hamiltonkreis (n Knoten, n Kanten) in G
- In TSP Instanz hat dieser Kreis die Kosten n
- A_α findet Tour, deren Kosten höchstens $\alpha \cdot n$ sind

Angenommen G ist **nicht** hamiltonsch

- In K_n existiert aufjedenfall ein Hamiltonkreis
- Da in G keiner existiert, enthält Kreis eine Kante, die nicht in G enthalten ist
- Kante hat Kosten $\alpha \cdot n$
- Kosten jeder TSP-Tour sind also mindestens $\alpha \cdot n + (n-1) \cdot 1$
- Kosten der Tour, die A_α findet, sind damit auch mindestens $\alpha \cdot n + (n-1) > \alpha \cdot n$

Angenommen es existiert A_α

Angenommen G ist hamiltonsch

- Es existiert ein Hamiltonkreis (n Knoten, n Kanten) in G
- In TSP Instanz hat dieser Kreis die Kosten n
- A_α findet Tour, deren Kosten höchstens $\alpha \cdot n$ sind

Angenommen G ist **nicht** hamiltonsch

- In K_n existiert aufjedenfall ein Hamiltonkreis
- Da in G keiner existiert, enthält Kreis eine Kante, die nicht in G enthalten ist
- Kante hat Kosten $\alpha \cdot n$
- Kosten jeder TSP-Tour sind also mindestens $\alpha \cdot n + (n-1) \cdot 1$
- Kosten der Tour, die A_α findet, sind damit auch mindestens $\alpha \cdot n + (n-1) > \alpha \cdot n$

Zusammenfassung

- G ist hamiltonsch $\Rightarrow$ A_α findet Tour mit Kosten $< \alpha \cdot n$
- G ist **nicht** hamiltonsch $\Rightarrow$ A_α findet **keine** Tour mit Kosten $< \alpha \cdot n$

Das heisst: G ist hamiltonsch $\Leftrightarrow$ A_α findet Tour mit Kosten $< \alpha \cdot n$

Folgerung

A_α entscheidet das Hamiltonkreisproblem

Widerspruch

A_α ist **Polynomialzeitalgorithmus**, aber HC ist NP-hart!

Zusammenfassung

- G ist hamiltonsch $\Rightarrow$ A_α findet Tour mit Kosten $< \alpha \cdot n$
- G ist **nicht** hamiltonsch $\Rightarrow$ A_α findet **keine** Tour mit Kosten $< \alpha \cdot n$

Das heisst: G ist hamiltonsch $\Leftrightarrow$ A_α findet Tour mit Kosten $< \alpha \cdot n$

Folgerung

A_α entscheidet das Hamiltonkreisproblem

Widerspruch

A_α ist **Polynomialzeitalgorithmus**, aber HC ist NP-hart!

Zusammenfassung

- G ist hamiltonsch $\Rightarrow$ A_α findet Tour mit Kosten $< \alpha \cdot n$
- G ist **nicht** hamiltonsch $\Rightarrow$ A_α findet **keine** Tour mit Kosten $< \alpha \cdot n$

Das heisst: G ist hamiltonsch $\Leftrightarrow$ A_α findet Tour mit Kosten $< \alpha \cdot n$

Folgerung

A_α entscheidet das Hamiltonkreisproblem

Widerspruch

A_α ist **Polynomialzeitalgorithmus**, aber HC ist NP-hart!

Zusammenfassung

- G ist hamiltonsch $\Rightarrow$ A_α findet Tour mit Kosten $< \alpha \cdot n$
- G ist **nicht** hamiltonsch $\Rightarrow$ A_α findet **keine** Tour mit Kosten $< \alpha \cdot n$

Das heisst: G ist hamiltonsch $\Leftrightarrow$ A_α findet Tour mit Kosten $< \alpha \cdot n$

Folgerung

A_α entscheidet das Hamiltonkreisproblem

Widerspruch

A_α ist **Polynomialzeitalgorithmus**, aber HC ist NP-hart!

Zusammenfassung

- G ist hamiltonsch $\Rightarrow$ A_α findet Tour mit Kosten $< \alpha \cdot n$
- G ist **nicht** hamiltonsch $\Rightarrow$ A_α findet **keine** Tour mit Kosten $< \alpha \cdot n$

Das heisst: G ist hamiltonsch $\Leftrightarrow$ A_α findet Tour mit Kosten $< \alpha \cdot n$

Folgerung

A_α entscheidet das Hamiltonkreisproblem

Widerspruch

A_α ist **Polynomialzeitalgorithmus**, aber HC ist NP-hart!

Was jetzt?

Problem

- TSP ist nicht effizient optimal lösbar
- TSP ist auch nicht effizient approximierbar

Was jetzt?

Problem

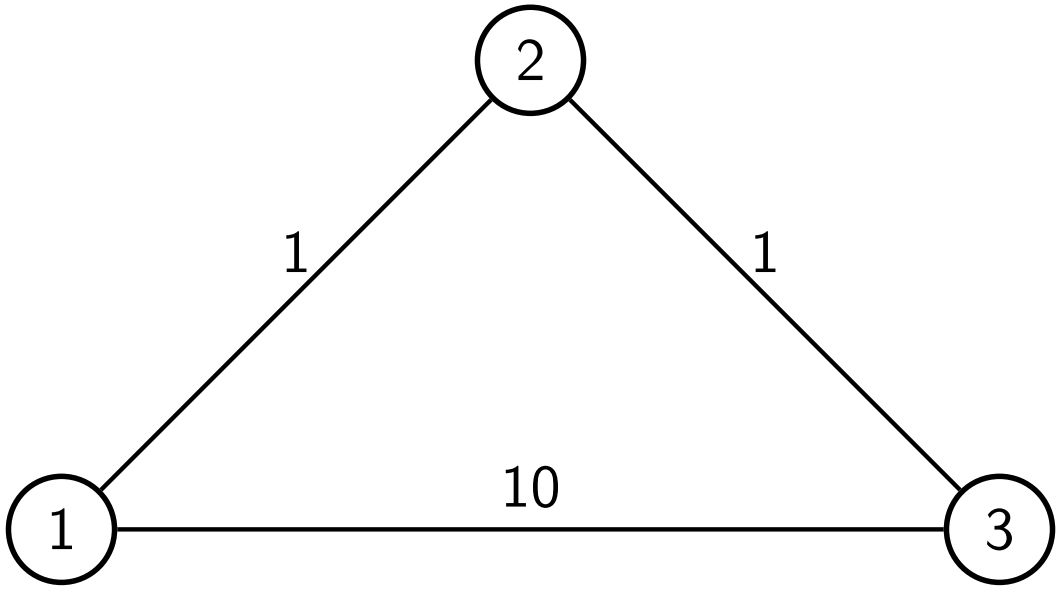
- TSP ist nicht effizient optimal lösbar
- TSP ist auch nicht effizient approximierbar

Metrisches TSP: Idee

Kosten

$$cost(13) = 10 > 2 = 1 + 1 = cost(123)$$

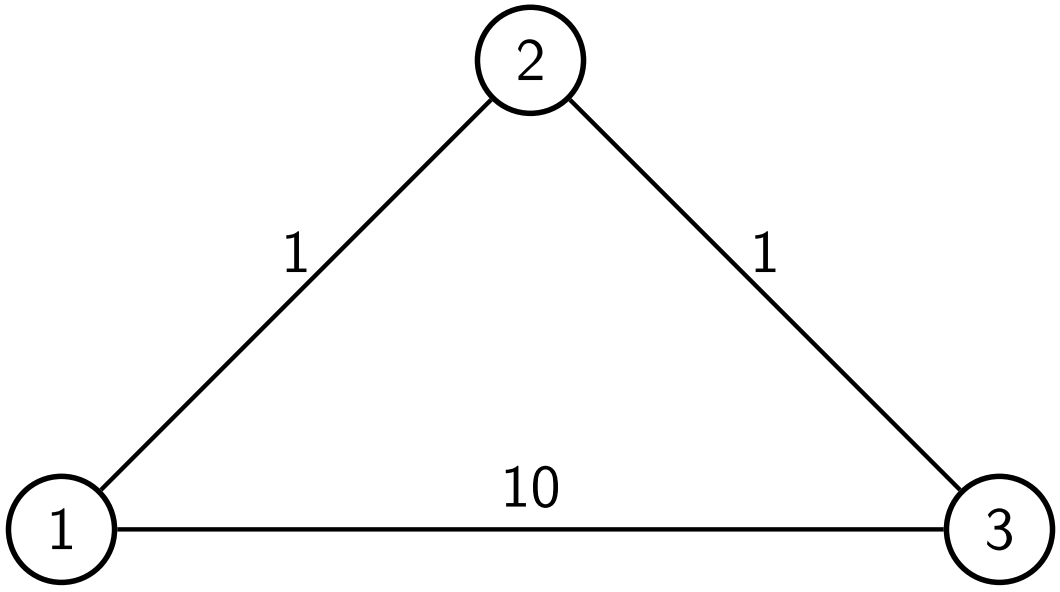
Metrisches TSP: Idee



Kosten

$$cost(13) = 10 > 2 = 1 + 1 = cost(123)$$

Metrisches TSP: Idee



Kosten

$$cost(13) = 10 > 2 = 1 + 1 = cost(123)$$

Metrisches TSP: Idee

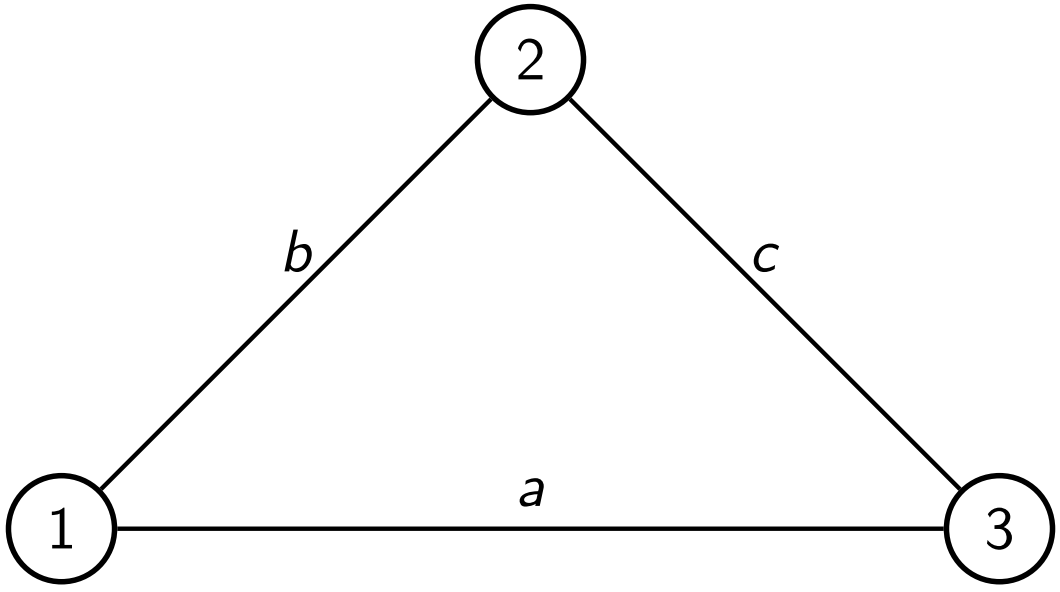
In der Praxi gilt aber oft:

Dreiecksungleichung: $a \leq b + c$

Wichtige Folgerung:

Wir können Knoten „überspringen“ ohne die Kosten zu erhöhen

Metrisches TSP: Idee



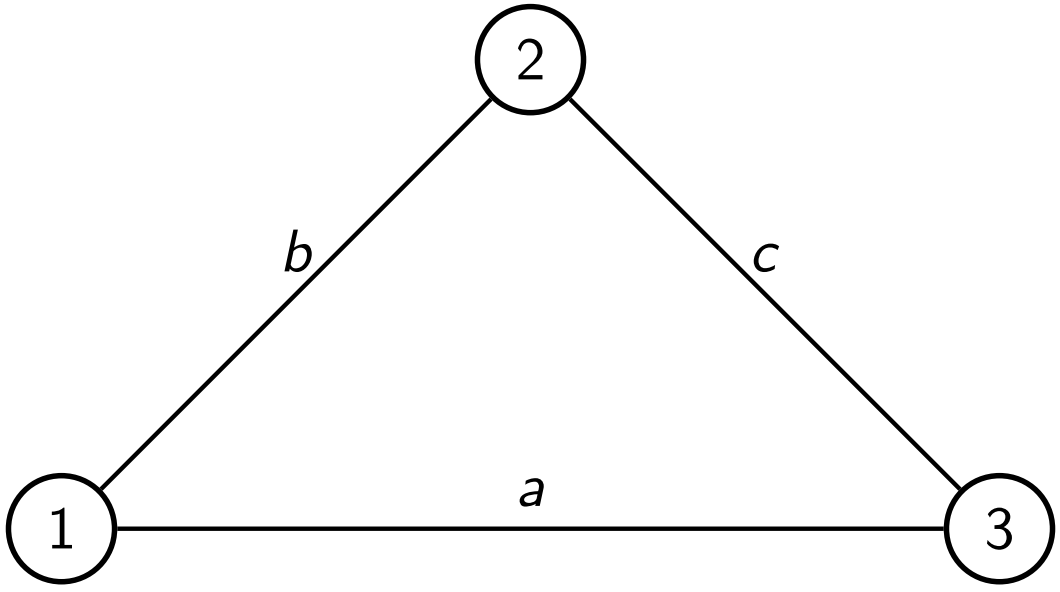
In der Praxi gilt aber oft:

Dreiecksungleichung: $a \leq b + c$

Wichtige Folgerung:

Wir können Knoten „überspringen“ ohne die Kosten zu erhöhen

Metrisches TSP: Idee



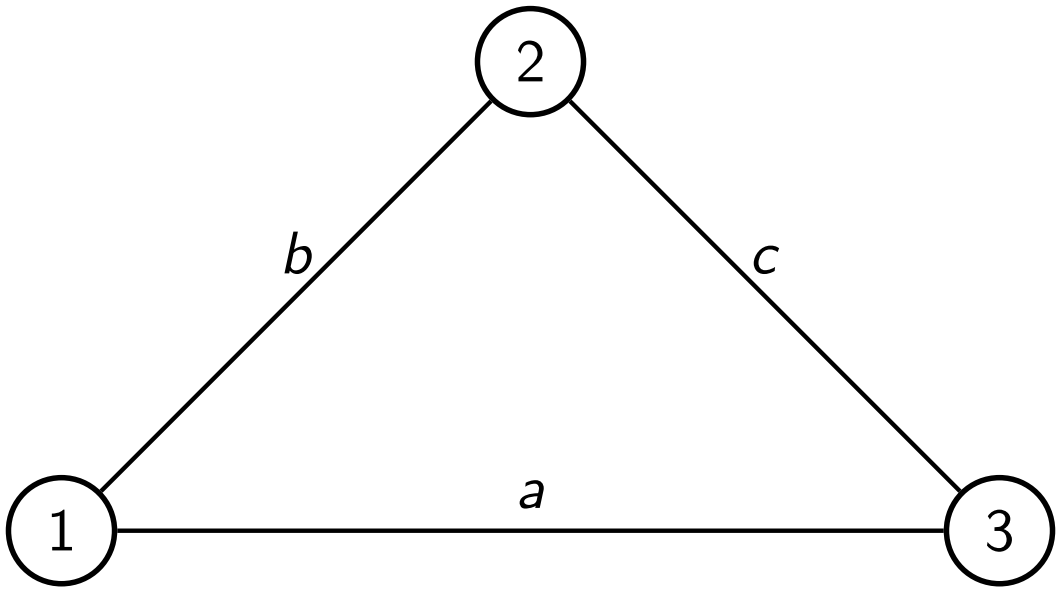
In der Praxi gilt aber oft:

Dreiecksungleichung: $a \leq b + c$

Wichtige Folgerung:

Wir können Knoten „überspringen“ ohne die Kosten zu erhöhen

Metrisches TSP: Idee



In der Praxi gilt aber oft:

Dreiecksungleichung: $a \leq b + c$

Wichtige Folgerung:

Wir können Knoten „überspringen“ ohne die Kosten zu erhöhen

Metrisches TSP

Definition

Abbildung c heißt *Metrik* wenn

- $c(i,i) = 0$ und $c(i,j) > 0$ für alle $i \neq j$ (c ist positiv definit)
- $c(i,j) = c(j,i)$ (c ist symmetrisch)
- $c(i,k) \leq c(i,j) + c(j,k)$ (c erfüllt **Dreiecksungleichung**)

Wenn die Kostenfunktion c eine Metrik ist, so spricht man von *metrischem* TSP (Δ-TSP)

MST-Heuristik

Metrisches TSP ist 2-approximierbar

Christofides-Heuristik

Metrisches TSP ist sogar $\frac{3}{2}$-approximierbar

Metrisches TSP

Definition

Abbildung c heißt *Metrik* wenn

- $c(i,i) = 0$ und $c(i,j) > 0$ für alle $i \neq j$ (c ist positiv definit)
- $c(i,j) = c(j,i)$ (c ist symmetrisch)
- $c(i,k) \leq c(i,j) + c(j,k)$ (c erfüllt **Dreiecksungleichung**)

Wenn die Kostenfunktion c eine Metrik ist, so spricht man von *metrischem* TSP (Δ-TSP)

MST-Heuristik

Metrisches TSP ist 2-approximierbar

Christofides-Heuristik

Metrisches TSP ist sogar $\frac{3}{2}$-approximierbar

Metrisches TSP

Definition

Abbildung c heißt *Metrik* wenn

- $c(i,i) = 0$ und $c(i,j) > 0$ für alle $i \neq j$ (c ist positiv definit)
- $c(i,j) = c(j,i)$ (c ist symmetrisch)
- $c(i,k) \leq c(i,j) + c(j,k)$ (c erfüllt **Dreiecksungleichung**)

Wenn die Kostenfunktion c eine Metrik ist, so spricht man von *metrischem* TSP (Δ-TSP)

MST-Heuristik

Metrisches TSP ist 2-approximierbar

Christofides-Heuristik

Metrisches TSP ist sogar $\frac{3}{2}$-approximierbar

Metrisches TSP

Definition

Abbildung c heißt *Metrik* wenn

- $c(i,i) = 0$ und $c(i,j) > 0$ für alle $i \neq j$ (c ist positiv definit)
- $c(i,j) = c(j,i)$ (c ist symmetrisch)
- $c(i,k) \leq c(i,j) + c(j,k)$ (c erfüllt **Dreiecksungleichung**)

Wenn die Kostenfunktion c eine Metrik ist, so spricht man von *metrischem* TSP (Δ-TSP)

MST-Heuristik

Metrisches TSP ist 2-approximierbar

Christofides-Heuristik

Metrisches TSP ist sogar $\frac{3}{2}$-approximierbar

Grundlagen

Definition

- Kreis, der alle **Kanten** eines Graphen enthält, heißt *Eulertour*
- Graph, der eine Eulertour enthält, heißt *eulersch*

Satz

G ist genau dann eulersch, wenn Grad jedes Knoten gerade ist

Eulerkreisproblem

- Testen ob Graph eulersch ist
- $\textsc{Eulerkreisproblem} \in \mathrm{P}$
- Eulertour in eulerschen Graphen bestimmen auch effizient möglich

Grundlagen

Definition

- Kreis, der alle **Kanten** eines Graphen enthält, heißt *Eulertour*
- Graph, der eine Eulertour enthält, heißt *eulersch*

Satz

G ist genau dann eulersch, wenn Grad jedes Knoten gerade ist

Eulerkreisproblem

- Testen ob Graph eulersch ist
- EULERKREISPROBLEM $\in$ P
- Eulertour in eulerschen Graphen bestimmen auch effizient möglich

Grundlagen

Definition

- Kreis, der alle **Kanten** eines Graphen enthält, heißt *Eulertour*
- Graph, der eine Eulertour enthält, heißt *eulersch*

Satz

G ist genau dann eulersch, wenn Grad jedes Knoten gerade ist

Eulerkreisproblem

- Testen ob Graph eulersch ist
- EULERKREISPROBLEM $\in$ P
- Eulertour in eulerschen Graphen bestimmen auch effizient möglich

Grundlagen

Definition

- Kreis, der alle **Kanten** eines Graphen enthält, heißt *Eulertour*
- Graph, der eine Eulertour enthält, heißt *eulersch*

Satz

G ist genau dann eulersch, wenn Grad jedes Knoten gerade ist

Eulerkreisproblem

- Testen ob Graph eulersch ist
- $\textsc{Eulerkreisproblem} \in \mathrm{P}$
- Eulertour in eulerschen Graphen bestimmen auch effizient möglich

Der MST Algorithmus

Sei c eine Metrik

Algorithmus

1. Erzeuge MST T von G
2. Verdopple jede Kante in T
3. Bestimme einen Eulerkreis K in T
4. Erzeugen neuen Graph K' durch überspringen doppelt besuchter Knoten in K

Der MST Algorithmus

Sei c eine Metrik

Algorithmus

1. Erzeuge MST T von G
2. Verdopple jede Kante in T
3. Bestimme einen Eulerkreis K in T
4. Erzeugen neuen Graph K' durch überspringen doppelt besuchter Knoten in K

Der MST Algorithmus

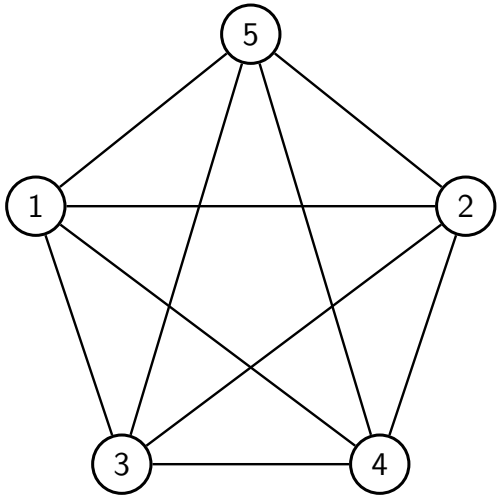
Sei c eine Metrik

Algorithmus

1. Erzeuge MST T von G
2. Verdopple jede Kante in T
3. Bestimme einen Eulerkreis K in T
4. Erzeugen neuen Graph K' durch überspringen doppelt besuchter Knoten in K

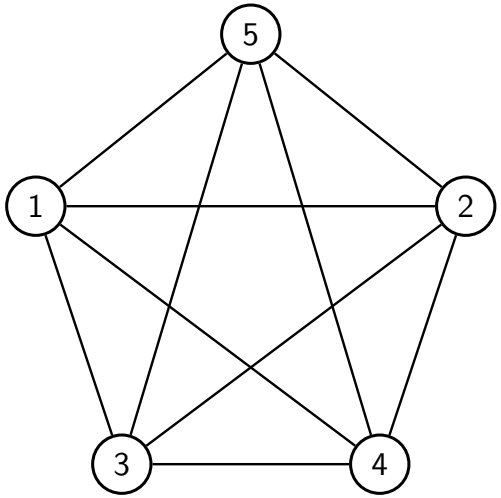
Beispiel

Beispiel

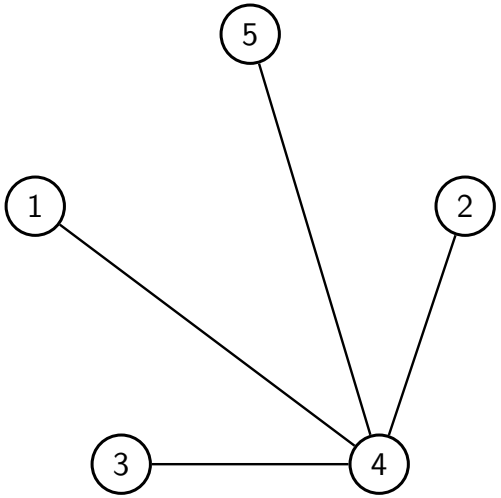


(a) K_5

Beispiel

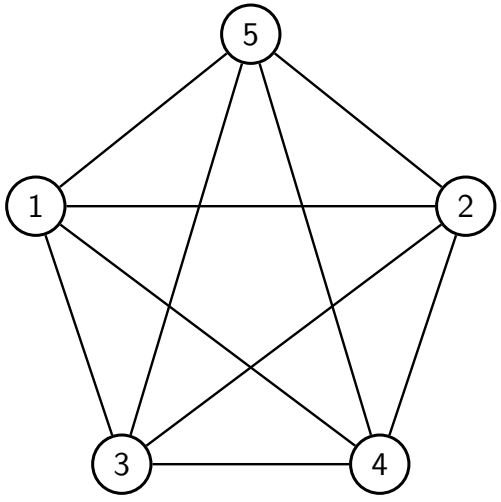


(a) K_5

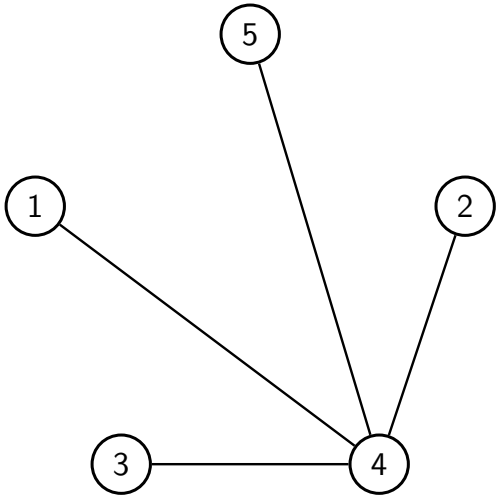


(b) Bestimme MST T von G

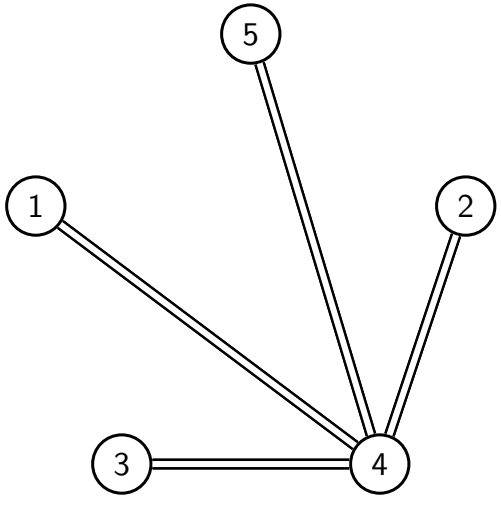
Beispiel



(a) K_5

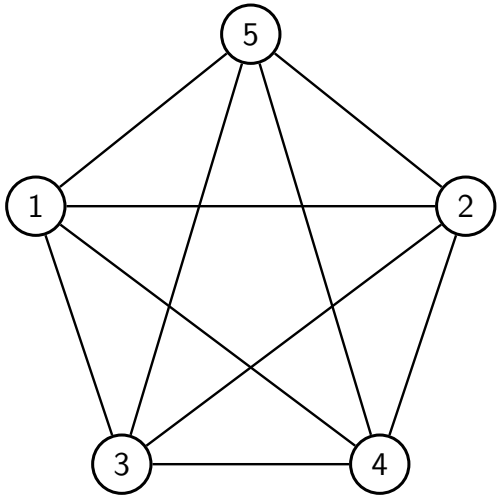


(b) Bestimme MST T von G

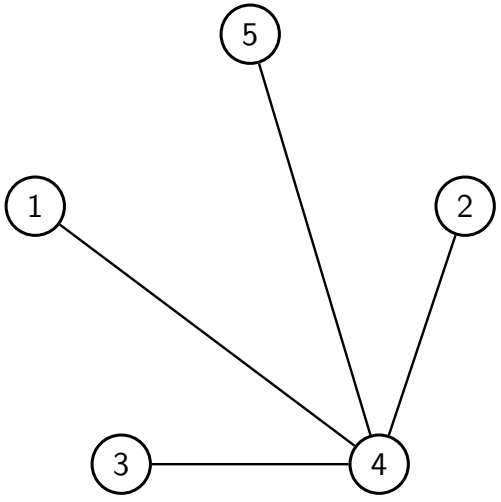


(c) Verdopple alle Kanten

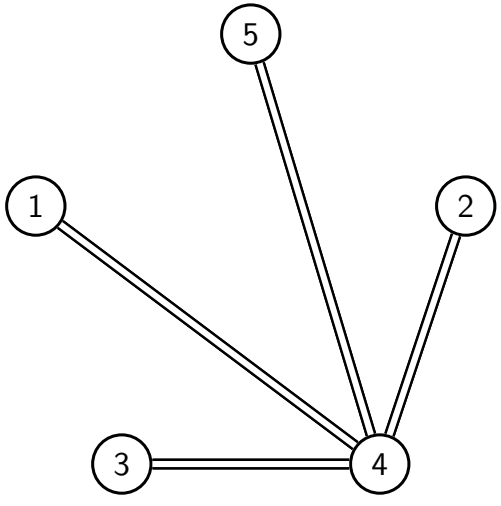
Beispiel



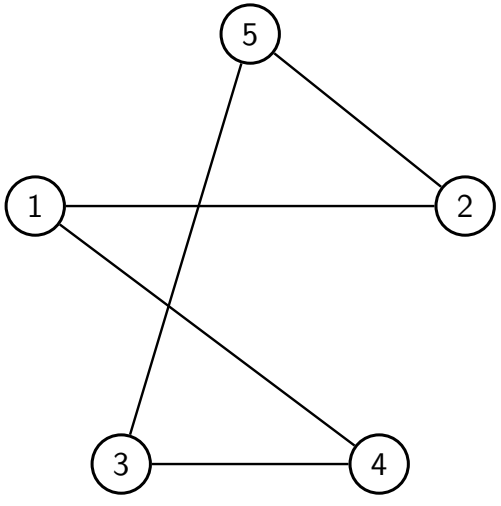
(a) K_5



(b) Bestimme MST T von G



(c) Verdopple alle Kanten



(d) Erzeuge Eulerkreis (K') und überspringe anschließend doppelte Knoten

Approximiationsgüte

Satz

MST-Heuristik ist polynomieller 2-Approximationsalgorithmus für Δ-TSP

Beweis

- K' besucht jeden Knoten genau einmal
- K' ist also Hamiltonkreis und damit TSP-Tour
- $cost(T) \leq opt(G)$, da T minimaler Spannbaum ist
- $cost(K) = 2 \cdot cost(T)$, da jede Kante doppelt vorkommt
- c ist Metrik, also erhöht überspringen von Knoten die Kosten nicht

$$cost(K') \leq cost(K) = 2 \cdot cost(T) \leq 2 \cdot opt(G)$$

Also ist K' eine TSP-Tour, deren Kosten weniger als doppelt so hoch sind, wie das Optimum

Approximiationsgüte

Satz

MST-Heuristik ist polynomieller 2-Approximationsalgorithmus für Δ-TSP

Beweis

- K' besucht jeden Knoten genau einmal
- K' ist also Hamiltonkreis und damit TSP-Tour
- $cost(T) \leq opt(G)$, da T minimaler Spannbaum ist
- $cost(K) = 2 \cdot cost(T)$, da jede Kante doppelt vorkommt
- c ist Metrik, also erhöht überspringen von Knoten die Kosten nicht

$$cost(K') \leq cost(K) = 2 \cdot cost(T) \leq 2 \cdot opt(G)$$

Also ist K' eine TSP-Tour, deren Kosten weniger als doppelt so hoch sind, wie das Optimum

Approximiationsgüte

Satz

MST-Heuristik ist polynomieller 2-Approximationsalgorithmus für Δ-TSP

Beweis

- K' besucht jeden Knoten genau einmal
- K' ist also Hamiltonkreis und damit TSP-Tour
- $cost(T) \leq opt(G)$, da T minimaler Spannbaum ist
- $cost(K) = 2 \cdot cost(T)$, da jede Kante doppelt vorkommt
- c ist Metrik, also erhöht überspringen von Knoten die Kosten nicht

$$cost(K') \leq cost(K) = 2 \cdot cost(T) \leq 2 \cdot opt(G)$$

Also ist K' eine TSP-Tour, deren Kosten weniger als doppelt so hoch sind, wie das Optimum

Approximiationsgüte

Satz

MST-Heuristik ist polynomieller 2-Approximationsalgorithmus für Δ-TSP

Beweis

- K' besucht jeden Knoten genau einmal
- K' ist also Hamiltonkreis und damit TSP-Tour
- $cost(T) \leq opt(G)$, da T minimaler Spannbaum ist
- $cost(K) = 2 \cdot cost(T)$, da jede Kante doppelt vorkommt
- c ist Metrik, also erhöht überspringen von Knoten die Kosten nicht

$$cost(K') \leq cost(K) = 2 \cdot cost(T) \leq 2 \cdot opt(G)$$

Also ist K' eine TSP-Tour, deren Kosten weniger als doppelt so hoch sind, wie das Optimum

Approximiationsgüte

Satz

MST-Heuristik ist polynomieller 2-Approximationsalgorithmus für Δ-TSP

Beweis

- K' besucht jeden Knoten genau einmal
- K' ist also Hamiltonkreis und damit TSP-Tour
- $cost(T) \leq opt(G)$, da T minimaler Spannbaum ist
- $cost(K) = 2 \cdot cost(T)$, da jede Kante doppelt vorkommt
- c ist Metrik, also erhöht überspringen von Knoten die Kosten nicht

$$cost(K') \leq cost(K) = 2 \cdot cost(T) \leq 2 \cdot opt(G)$$

Also ist K' eine TSP-Tour, deren Kosten weniger als doppelt so hoch sind, wie das Optimum

Approximiationsgüte

Satz

MST-Heuristik ist polynomieller 2-Approximationsalgorithmus für Δ-TSP

Beweis

- K' besucht jeden Knoten genau einmal
- K' ist also Hamiltonkreis und damit TSP-Tour
- $cost(T) \leq opt(G)$, da T minimaler Spannbaum ist
- $cost(K) = 2 \cdot cost(T)$, da jede Kante doppelt vorkommt
- c ist Metrik, also erhöht überspringen von Knoten die Kosten nicht

$$cost(K') \leq cost(K) = 2 \cdot cost(T) \leq 2 \cdot opt(G)$$

Also ist K' eine TSP-Tour, deren Kosten weniger als doppelt so hoch sind, wie das Optimum

Approximiationsgüte

Satz

MST-Heuristik ist polynomieller 2-Approximationsalgorithmus für Δ-TSP

Beweis

- K' besucht jeden Knoten genau einmal
- K' ist also Hamiltonkreis und damit TSP-Tour
- $cost(T) \leq opt(G)$, da T minimaler Spannbaum ist
- $cost(K) = 2 \cdot cost(T)$, da jede Kante doppelt vorkommt
- c ist Metrik, also erhöht überspringen von Knoten die Kosten nicht

$$cost(K') \leq cost(K) = 2 \cdot cost(T) \leq 2 \cdot opt(G)$$

Also ist K' eine TSP-Tour, deren Kosten weniger als doppelt so hoch sind, wie das Optimum

Approximiationsgüte

Satz

MST-Heuristik ist polynomieller 2-Approximationsalgorithmus für Δ-TSP

Beweis

- K' besucht jeden Knoten genau einmal
- K' ist also Hamiltonkreis und damit TSP-Tour
- $cost(T) \leq opt(G)$, da T minimaler Spannbaum ist
- $cost(K) = 2 \cdot cost(T)$, da jede Kante doppelt vorkommt
- c ist Metrik, also erhöht überspringen von Knoten die Kosten nicht

$$cost(K') \leq cost(K) = 2 \cdot cost(T) \leq 2 \cdot opt(G)$$

Also ist K' eine TSP-Tour, deren Kosten weniger als doppelt so hoch sind, wie das Optimum

Approximiationsgüte

Satz

MST-Heuristik ist polynomieller 2-Approximationsalgorithmus für Δ-TSP

Beweis

- K' besucht jeden Knoten genau einmal
- K' ist also Hamiltonkreis und damit TSP-Tour
- $cost(T) \leq opt(G)$, da T minimaler Spannbaum ist
- $cost(K) = 2 \cdot cost(T)$, da jede Kante doppelt vorkommt
- c ist Metrik, also erhöht überspringen von Knoten die Kosten nicht

$$cost(K') \leq cost(K) = 2 \cdot cost(T) \leq 2 \cdot opt(G)$$

Also ist K' eine TSP-Tour, deren Kosten weniger als doppelt so hoch sind, wie das Optimum

Christofides Algorithmus

Definition

$M \subseteq E$ heißt *perfektes Matching* in $G = (V, E)$, falls **jeder** Knoten aus V zu **genau einer** Kante aus M inzident ist

Algorithmus von Christofides

1. Bestimme MST T in G
2. Bestimme $V' = \{v \in V \mid v \text{ hat ungeraden Grad in } T\} \subseteq V$
3. Finde perfektes Matching mit minimalen Kosten M auf V'
4. Finde Euler-Tour auf den Kanten von $H := T \cup M$
5. Überspringe wiederholt vorkommende Kanten in Euler-Tour

Satz

Christofides ist polynomieller $\frac{3}{2}$-Approximationsalgorithmus für Δ-TSP

Christofides Algorithmus

Definition

$M \subseteq E$ heißt *perfektes Matching* in $G = (V, E)$, falls **jeder** Knoten aus V zu **genau einer** Kante aus M inzident ist

Algorithmus von Christofides

1. Bestimme MST T in G
2. Bestimme $V' = \{v \in V \mid v \text{ hat ungeraden Grad in } T\} \subseteq V$
3. Finde perfektes Matching mit minimalen Kosten M auf V'
4. Finde Euler-Tour auf den Kanten von $H := T \cup M$
5. Überspringe wiederholt vorkommende Kanten in Euler-Tour

Satz

Christofides ist polynomieller $\frac{3}{2}$-Approximationsalgorithmus für Δ-TSP

Christofides Algorithmus

Definition

$M \subseteq E$ heißt *perfektes Matching* in $G = (V, E)$, falls **jeder** Knoten aus V zu **genau einer** Kante aus M inzident ist

Algorithmus von Christofides

1. Bestimme MST T in G
2. Bestimme $V' = \{v \in V \mid v \text{ hat ungeraden Grad in } T\} \subseteq V$
3. Finde perfektes Matching mit minimalen Kosten M auf V'
4. Finde Euler-Tour auf den Kanten von $H := T \cup M$
5. Überspringe wiederholt vorkommende Kanten in Euler-Tour

Satz

Christofides ist polynomieller $\frac{3}{2}$-Approximationsalgorithmus für Δ-TSP

Christofides Algorithmus

Definition

$M \subseteq E$ heißt *perfektes Matching* in $G = (V, E)$, falls **jeder** Knoten aus V zu **genau einer** Kante aus M inzident ist

Algorithmus von Christofides

1. Bestimme MST T in G
2. Bestimme $V' = \{v \in V \mid v \text{ hat ungeraden Grad in } T\} \subseteq V$
3. Finde perfektes Matching mit minimalen Kosten M auf V'
4. Finde Euler-Tour auf den Kanten von $H := T \cup M$
5. Überspringe wiederholt vorkommende Kanten in Euler-Tour

Satz

Christofides ist polynomieller $\frac{3}{2}$*-Approximationsalgorithmus für* Δ-TSP

Beispiel

(a) $G = K_{10}$

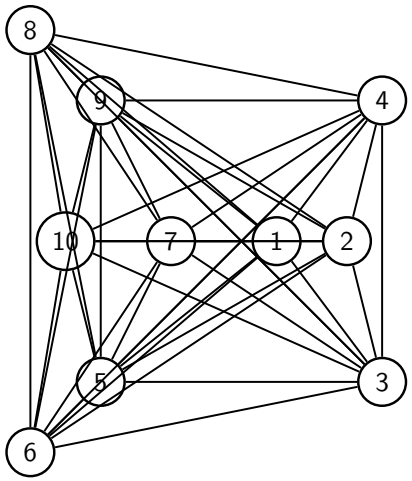
(b) Bestimme MST T in G

(c) Knoten mit ungeradem Grad in T

(d) Perfektes Matching für Knoten mit ungeradem Grad

(e) Euler-Tour (doppelte Knoten uebersprungen)

Beispiel



(a) $G = K_{10}$

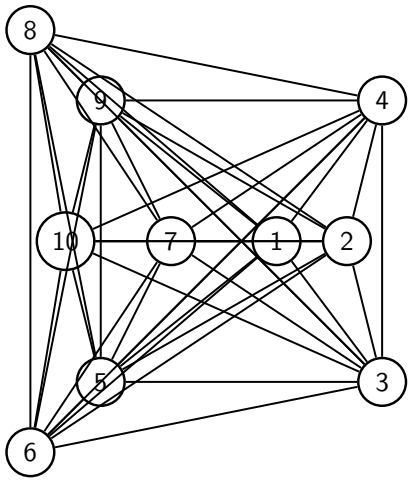
(b) Bestimme MST T in G

(c) Knoten mit ungeradem Grad in T

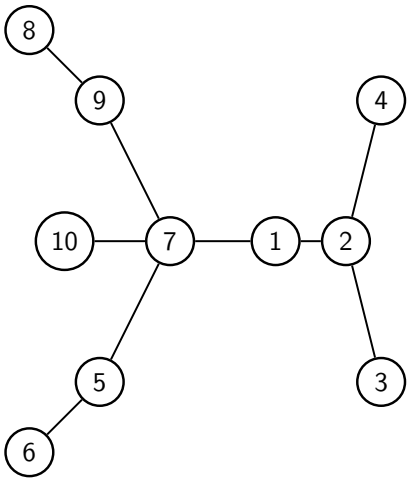
(d) Perfektes Matching für Knoten mit ungeradem Grad

(e) Euler-Tour (doppelte Knoten uebersprungen)

Beispiel



(a) $G = K_{10}$



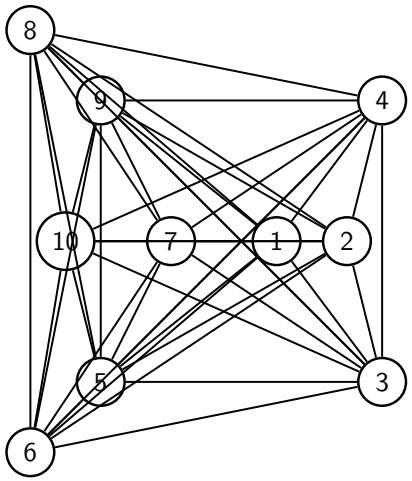
(b) Bestimme MST T in G

(c) Knoten mit ungeradem Grad in T

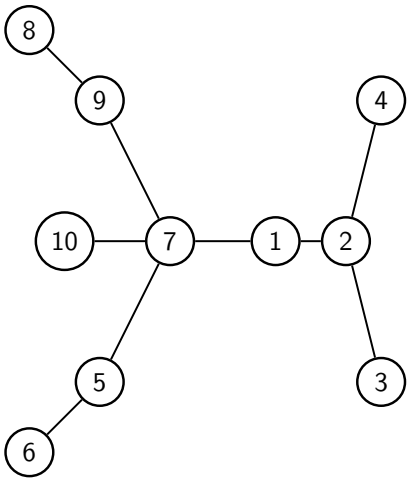
(d) Perfektes Matching für Knoten mit ungeradem Grad

(e) Euler-Tour (doppelte Knoten uebersprungen)

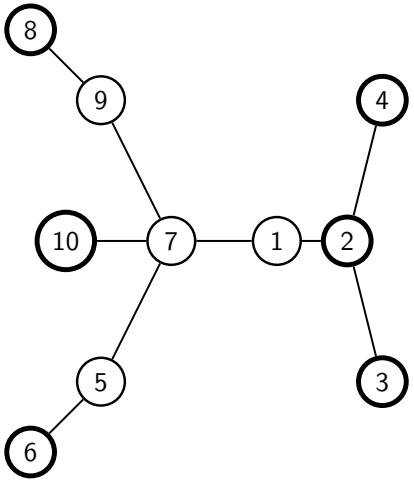
Beispiel



(a) $G = K_{10}$



(b) Bestimme MST T in G

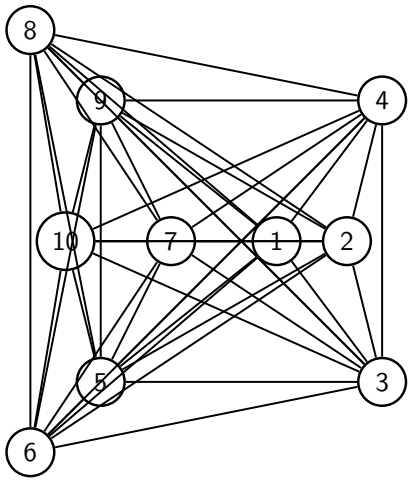


(c) Knoten mit ungeradem Grad in T

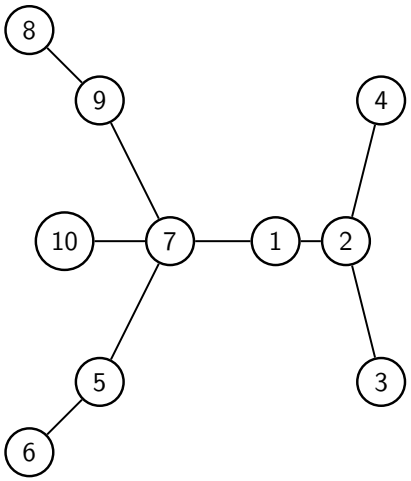
(d) Perfektes Matching für Knoten mit ungeradem Grad

(e) Euler-Tour (doppelte Knoten uebersprungen)

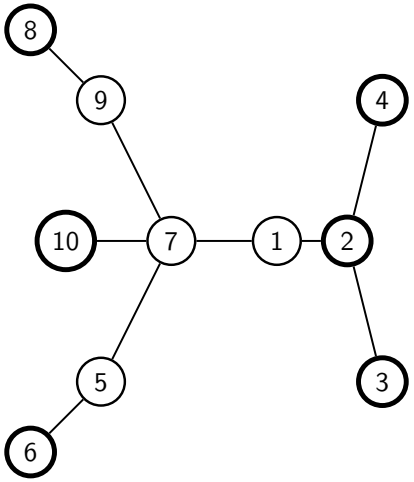
Beispiel



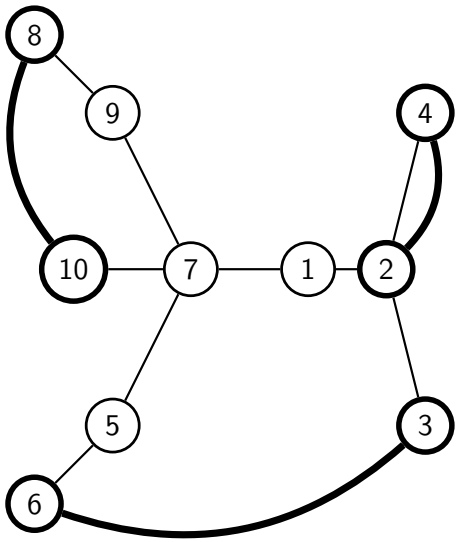
(a) $G = K_{10}$



(b) Bestimme MST T in G



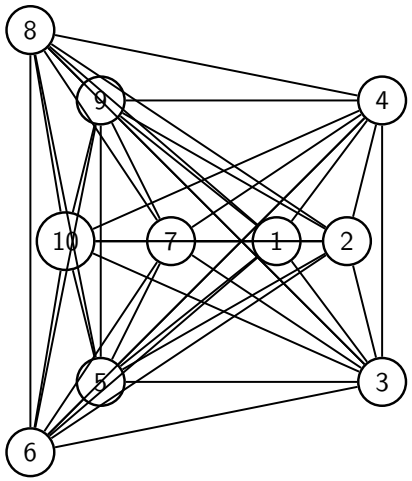
(c) Knoten mit ungeradem Grad in T



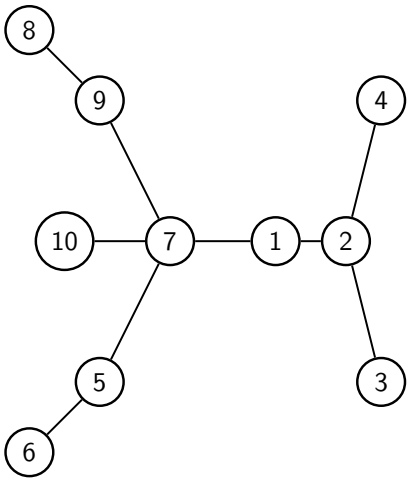
(d) Perfektes Matching für Knoten mit ungeradem Grad

(e) Euler-Tour (doppelte Knoten uebersprungen)

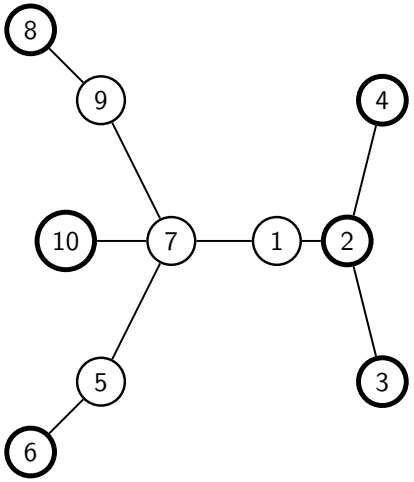
Beispiel



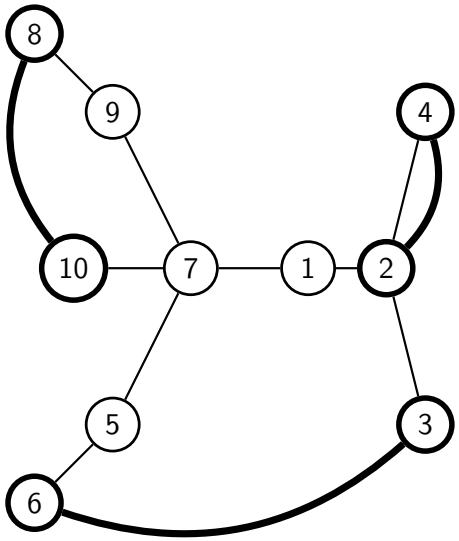
(a) $G = K_{10}$



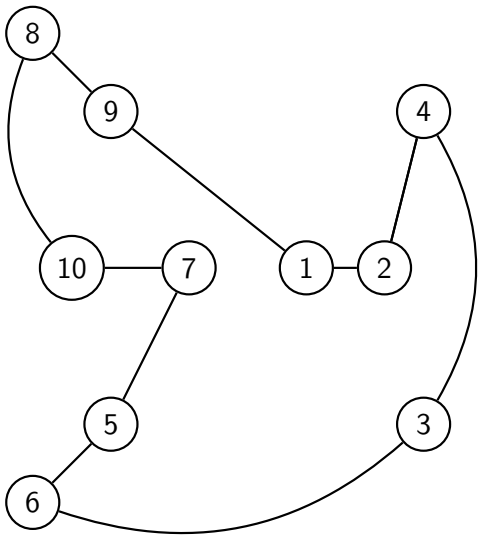
(b) Bestimme MST T in G



(c) Knoten mit ungeradem Grad in T



(d) Perfektes Matching für Knoten mit ungeradem Grad



(e) Euler-Tour (doppelte Knoten uebersprungen)

Analyse

- Euler-Tour in H ist TSP-Tour
- $cost(T) \leq opt$
- Sei τ eine optimale TSP-Tour, d.h. $cost(\tau) = opt$
- Betrachte Kreis τ', der Knoten aus V' verbindet, indem in τ alle Knoten übersprungen werden, die nicht zu V' gehören
- $cost(\tau') \leq cost(\tau)$
- Zerlege τ' in zwei perfekte Matchings
- Günstigere der beiden Matchings hat höchstens die Kosten $\frac{1}{2} \cdot cost(\tau')$
- $\frac{1}{2} cost(\tau') \leq \frac{1}{2} cost(\tau) = \frac{1}{2} opt$
- Günstigstes perfektes Matching auf V' hat höchstens Kosten $\frac{1}{2} \cdot opt$

Analyse

- Euler-Tour in H ist TSP-Tour
- $cost(T) \leq opt$
- Sei τ eine optimale TSP-Tour, d.h. $cost(\tau) = opt$
- Betrachte Kreis τ', der Knoten aus V' verbindet, indem in τ alle Knoten übersprungen werden, die nicht zu V' gehören
- $cost(\tau') \leq cost(\tau)$
- Zerlege τ' in zwei perfekte Matchings
- Günstigere der beiden Matchings hat höchstens die Kosten $\frac{1}{2} \cdot cost(\tau')$
- $\frac{1}{2} cost(\tau') \leq \frac{1}{2} cost(\tau) = \frac{1}{2} opt$
- Günstigstes perfektes Matching auf V' hat höchstens Kosten $\frac{1}{2} \cdot opt$

Analyse

- Euler-Tour in H ist TSP-Tour
- $cost(T) \leq opt$
- Sei τ eine optimale TSP-Tour, d.h. $cost(\tau) = opt$
- Betrachte Kreis τ', der Knoten aus V' verbindet, indem in τ alle Knoten übersprungen werden, die nicht zu V' gehören
- $cost(\tau') \leq cost(\tau)$
- Zerlege τ' in zwei perfekte Matchings
- Günstigere der beiden Matchings hat höchstens die Kosten $\frac{1}{2} \cdot cost(\tau')$
- $\frac{1}{2} cost(\tau') \leq \frac{1}{2} cost(\tau) = \frac{1}{2} opt$
- Günstigstes perfektes Matching auf V' hat höchstens Kosten $\frac{1}{2} \cdot opt$

Analyse

- Euler-Tour in H ist TSP-Tour
- $cost(T) \leq opt$
- Sei τ eine optimale TSP-Tour, d.h. $cost(\tau) = opt$
- Betrachte Kreis τ', der Knoten aus V' verbindet, indem in τ alle Knoten übersprungen werden, die nicht zu V' gehören
- $cost(\tau') \leq cost(\tau)$
- Zerlege τ' in zwei perfekte Matchings
- Günstigere der beiden Matchings hat höchstens die Kosten $\frac{1}{2} \cdot cost(\tau')$
- $\frac{1}{2} cost(\tau') \leq \frac{1}{2} cost(\tau) = \frac{1}{2} opt$
- Günstigstes perfektes Matching auf V' hat höchstens Kosten $\frac{1}{2} \cdot opt$

Analyse

- Euler-Tour in H ist TSP-Tour
- $cost(T) \leq opt$
- Sei τ eine optimale TSP-Tour, d.h. $cost(\tau) = opt$
- Betrachte Kreis τ', der Knoten aus V' verbindet, indem in τ alle Knoten übersprungen werden, die nicht zu V' gehören
- $cost(\tau') \leq cost(\tau)$
- Zerlege τ' in zwei perfekte Matchings
- Günstigere der beiden Matchings hat höchstens die Kosten $\frac{1}{2} \cdot cost(\tau')$
- $\frac{1}{2} cost(\tau') \leq \frac{1}{2} cost(\tau) = \frac{1}{2} opt$
- Günstigstes perfektes Matching auf V' hat höchstens Kosten $\frac{1}{2} \cdot opt$

Analyse

- Euler-Tour in H ist TSP-Tour
- $cost(T) \leq opt$
- Sei τ eine optimale TSP-Tour, d.h. $cost(\tau) = opt$
- Betrachte Kreis τ', der Knoten aus V' verbindet, indem in τ alle Knoten übersprungen werden, die nicht zu V' gehören
- $cost(\tau') \leq cost(\tau)$
- Zerlege τ' in zwei perfekte Matchings
- Günstigere der beiden Matchings hat höchstens die Kosten $\frac{1}{2} \cdot cost(\tau')$
- $\frac{1}{2} cost(\tau') \leq \frac{1}{2} cost(\tau) = \frac{1}{2} opt$
- Günstigstes perfektes Matching auf V' hat höchstens Kosten $\frac{1}{2} \cdot opt$

Analyse

- Euler-Tour in H ist TSP-Tour
- $cost(T) \leq opt$
- Sei τ eine optimale TSP-Tour, d.h. $cost(\tau) = opt$
- Betrachte Kreis τ', der Knoten aus V' verbindet, indem in τ alle Knoten übersprungen werden, die nicht zu V' gehören
- $cost(\tau') \leq cost(\tau)$
- Zerlege τ' in zwei perfekte Matchings
- Günstigere der beiden Matchings hat höchstens die Kosten $\frac{1}{2} \cdot cost(\tau')$
- $\frac{1}{2} cost(\tau') \leq \frac{1}{2} cost(\tau) = \frac{1}{2} opt$
- Günstigstes perfektes Matching auf V' hat höchstens Kosten $\frac{1}{2} \cdot opt$

Analyse

- Euler-Tour in H ist TSP-Tour
- $cost(T) \leq opt$
- Sei τ eine optimale TSP-Tour, d.h. $cost(\tau) = opt$
- Betrachte Kreis τ', der Knoten aus V' verbindet, indem in τ alle Knoten übersprungen werden, die nicht zu V' gehören
- $cost(\tau') \leq cost(\tau)$
- Zerlege τ' in zwei perfekte Matchings
- Günstigere der beiden Matchings hat höchstens die Kosten $\frac{1}{2} \cdot cost(\tau')$
- $\frac{1}{2} cost(\tau') \leq \frac{1}{2} cost(\tau) = \frac{1}{2} opt$
- Günstigstes perfektes Matching auf V' hat höchstens Kosten $\frac{1}{2} \cdot opt$

Analyse

- Euler-Tour in H ist TSP-Tour
- $cost(T) \leq opt$
- Sei τ eine optimale TSP-Tour, d.h. $cost(\tau) = opt$
- Betrachte Kreis τ', der Knoten aus V' verbindet, indem in τ alle Knoten übersprungen werden, die nicht zu V' gehören
- $cost(\tau') \leq cost(\tau)$
- Zerlege τ' in zwei perfekte Matchings
- Günstigere der beiden Matchings hat höchstens die Kosten $\frac{1}{2} \cdot cost(\tau')$
- $\frac{1}{2} cost(\tau') \leq \frac{1}{2} cost(\tau) = \frac{1}{2} opt$
- Günstigstes perfektes Matching auf V' hat höchstens Kosten $\frac{1}{2} \cdot opt$

Analyse

- Euler-Tour in H ist TSP-Tour
- $cost(T) \leq opt$
- Sei τ eine optimale TSP-Tour, d.h. $cost(\tau) = opt$
- Betrachte Kreis τ', der Knoten aus V' verbindet, indem in τ alle Knoten übersprungen werden, die nicht zu V' gehören
- $cost(\tau') \leq cost(\tau)$
- Zerlege τ' in zwei perfekte Matchings
- Günstigere der beiden Matchings hat höchstens die Kosten $\frac{1}{2} \cdot cost(\tau')$
- $\frac{1}{2} cost(\tau') \leq \frac{1}{2} cost(\tau) = \frac{1}{2} opt$
- Günstigstes perfektes Matching auf V' hat höchstens Kosten $\frac{1}{2} \cdot opt$

Approximationsgüte

Die Euler-Tour (= TSP-Tour) hat also die Kosten:

$$cost(H) \leq cost(M) + cost(T) \leq opt + \frac{1}{2} opt = \frac{3}{2} opt$$

□

Approximationsgüte

Die Euler-Tour (= TSP-Tour) hat also die Kosten:

$$cost(H) \leq cost(M) + cost(T) \leq opt + \frac{1}{2} opt = \frac{3}{2} opt$$

□

Approximationsgüte

Die Euler-Tour ($=$ TSP-Tour) hat also die Kosten:

$$cost(H) \leq cost(M) + cost(T) \leq opt + \frac{1}{2}opt = \mathbf{\frac{3}{2}}opt$$

□

Grenzen der Approximierbarkeit

Frage

Geht es **noch** besser?

Antwort

Kein polynomieller Approximationsalgorithmus bekannt mit $\alpha < \frac{3}{2}$

Frage

Kann es überhaupt einen besseren geben?

Antwort

- Für $\alpha < \frac{220}{219}$ kann es **keinen** α-Approximationsalgorithmus geben
- Ob es für $\alpha < \frac{3}{2}$ einen geben kann ist nicht bekannt

Grenzen der Approximierbarkeit

Frage

Geht es **noch** besser?

Antwort

Kein polynomieller Approximationsalgorithmus bekannt mit $\alpha < \frac{3}{2}$

Frage

Kann es überhaupt einen besseren geben?

Antwort

- Für $\alpha < \frac{220}{219}$ kann es **keinen** α-Approximationsalgorithmus geben
- Ob es für $\alpha < \frac{3}{2}$ einen geben kann ist nicht bekannt

Grenzen der Approximierbarkeit

Frage

Geht es **noch** besser?

Antwort

Kein polynomieller Approximationsalgorithmus bekannt mit $\alpha < \frac{3}{2}$

Frage

Kann es überhaupt einen besseren geben?

Antwort

- Für $\alpha < \frac{220}{219}$ kann es **keinen** α-Approximationsalgorithmus geben
- Ob es für $\alpha < \frac{3}{2}$ einen geben kann ist nicht bekannt

Grenzen der Approximierbarkeit

Frage

Geht es **noch** besser?

Antwort

Kein polynomieller Approximationsalgorithmus bekannt mit $\alpha < \frac{3}{2}$

Frage

Kann es überhaupt einen besseren geben?

Antwort

- Für $\alpha < \frac{220}{219}$ kann es **keinen** α-Approximationsalgorithmus geben
- Ob es für $\alpha < \frac{3}{2}$ einen geben kann ist nicht bekannt

Grenzen der Approximierbarkeit

Frage

Geht es **noch** besser?

Antwort

Kein polynomieller Approximationsalgorithmus bekannt mit $\alpha < \frac{3}{2}$

Frage

Kann es überhaupt einen besseren geben?

Antwort

- Für $\alpha < \frac{220}{219}$ kann es **keinen** α-Approximationsalgorithmus geben
- Ob es für $\alpha < \frac{3}{2}$ einen geben kann ist nicht bekannt

Was war wichtig?

- Was ist $\textsc{Tsp}$?
- Entscheidungsvariante von $\textsc{Tsp}$ ist NP-vollständig
- $\textsc{Tsp}$ ist **nicht effizient lösbar** (ausser $\mathrm{P} = \mathrm{NP}$)
- **Nur eingeschränkte** Versionen von $\textsc{Tsp}$ effizient approximierbar
- Beste bekannte Approximation für Δ-$\textsc{Tsp}$: $\alpha = \frac{3}{2}$

Was war wichtig?

- Was ist $\textsc{Tsp}$?
- Entscheidungsvariante von $\textsc{Tsp}$ ist NP-vollständig
- $\textsc{Tsp}$ ist **nicht effizient lösbar** (ausser $\mathrm{P} = \mathrm{NP}$)
- **Nur eingeschränkte** Versionen von $\textsc{Tsp}$ effizient approximierbar
- Beste bekannte Approximation für Δ-$\textsc{Tsp}$: $\alpha = \frac{3}{2}$

Was war wichtig?

- Was ist $\textsc{Tsp}$?
- Entscheidungsvariante von $\textsc{Tsp}$ ist NP-vollständig
- $\textsc{Tsp}$ ist **nicht effizient lösbar** (ausser $\mathrm{P} = \mathrm{NP}$)
- **Nur eingeschränkte** Versionen von $\textsc{Tsp}$ effizient approximierbar
- Beste bekannte Approximation für Δ-$\textsc{Tsp}$: $\alpha = \frac{3}{2}$

Was war wichtig?

- Was ist $\textsc{Tsp}$?
- Entscheidungsvariante von $\textsc{Tsp}$ ist NP-vollständig
- $\textsc{Tsp}$ ist **nicht effizient lösbar** (ausser $\mathrm{P} = \mathrm{NP}$)
- **Nur eingeschränkte** Versionen von $\textsc{Tsp}$ effizient approximierbar
- Beste bekannte Approximation für Δ-$\textsc{Tsp}$: $\alpha = \frac{3}{2}$

Was war wichtig?

- Was ist $\textsc{Tsp}$?
- Entscheidungsvariante von $\textsc{Tsp}$ ist NP-vollständig
- $\textsc{Tsp}$ ist **nicht effizient lösbar** (ausser $\mathrm{P} = \mathrm{NP}$)
- **Nur eingeschränkte** Versionen von $\textsc{Tsp}$ effizient approximierbar
- Beste bekannte Approximation für Δ-$\textsc{Tsp}$: $\alpha = \frac{3}{2}$

Was war wichtig?

- Was ist $\textsc{Tsp}$?
- Entscheidungsvariante von $\textsc{Tsp}$ ist NP-vollständig
- $\textsc{Tsp}$ ist **nicht effizient lösbar** (ausser $\mathrm{P} = \mathrm{NP}$)
- **Nur eingeschränkte** Versionen von $\textsc{Tsp}$ effizient approximierbar
- Beste bekannte Approximation für Δ-$\textsc{Tsp}$: $\alpha = \frac{3}{2}$

Vielen Dank fürs Zuhören!