

Komplexität von Algorithmen

Sommersemester 2015

Dr. Arne Meier

Institut für Theoretische Informatik
Fakultät für Elektrotechnik und Informatik
Gottfried Wilhelm Leibniz Universität Hannover

Stand: 14. April 2015

Organisatorisches

Übungsbetrieb: Organisiert durch Maurice Chandoo
(chandoo@thi.uni-hannover.de).

Tutorien: ≤ 25 Studierende, auch selber Aufgaben lösen
Wo? hier im Gebäude 2. Stock, Raum 224

Anmeldung: heute ab 11:00 Uhr über das Stud.IP

Fragen? Mein Büro: hier im Gebäude, 2. Stock,
Raum 219, eMail: meier@thi.uni-hannover.de

Bonusregelung

Art: 3 Kurzklausuren und 1 Testat

Kurzklausur: 15min, zu Beginn der Vorlesung,
1 Woche vorher via Stud.IP angekündigt,
Richtig/Falsch-Fragen + Definitionen

Testat: 2 Wochen Bearbeitungszeit (über Pfingstwoche),
mündliche Vorstellung

Bonus: 0.3/0.4 auf *bestandene* Klausur

Voraussetzung: alle Kurzklausuren und Testat bestanden

Gültigkeit: eine angetretene Klausur, maximal bis zur nächsten
Vorlesung im SS 2016

Literaturhinweise (i)



Hopcroft, Motwani, Ullman. *Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie*

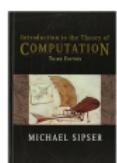
Themenschnittmenge: Alphabet, Sprachen, Probleme,
Einführung in Turingmaschinen, Klassen P und NP,
NP-Vollständigkeit

Literaturhinweise (i)



Hopcroft, Motwani, Ullman. *Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie*

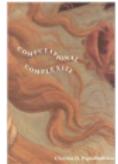
Themenschnittmenge: Alphabet, Sprachen, Probleme, Einführung in Turingmaschinen, Klassen P und NP, NP-Vollständigkeit



Sipser. *Introduction to the Theory of Computation*

Themenschnittmenge: Ländausymbole, Komplexitätsklassen-Beziehungen, Klassen P und NP, NP-Vollständigkeit, Hierarchiesätze, Approximationsalgorithmen

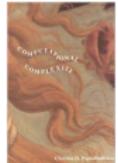
Literaturhinweise (ii)



Papadimitriou. *Computational Complexity*

Themenschnittmenge: Turingmaschinen,
Komplexitätsklassen-Beziehungen, Klassen P und NP,
NP-Vollständigkeit, Hierarchiesätze, Approximationsalgorithmen

Literaturhinweise (ii)



Papadimitriou. *Computational Complexity*

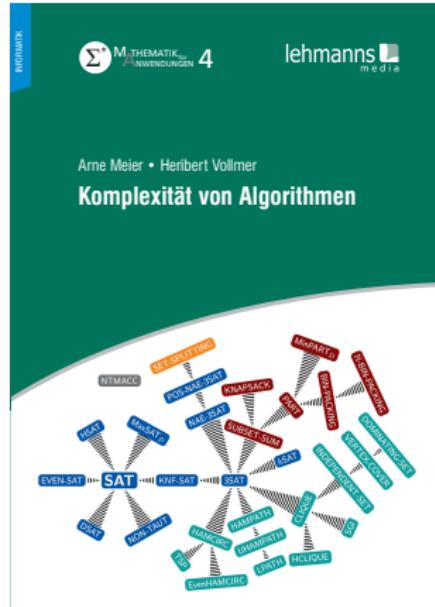
Themenschnittmenge: Turingmaschinen,
Komplexitätsklassen-Beziehungen, Klassen P und NP,
NP-Vollständigkeit, Hierarchiesätze, Approximationsalgorithmen



Ausiello und weitere. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*

Themenschnittmenge: Komplexitätsklassen, Klassen P und NP,
Optimizerungsprobleme, Klassen PO und NPO,
Approximationsalgorithmen, APX, PTAS

Literaturhinweise (iii)

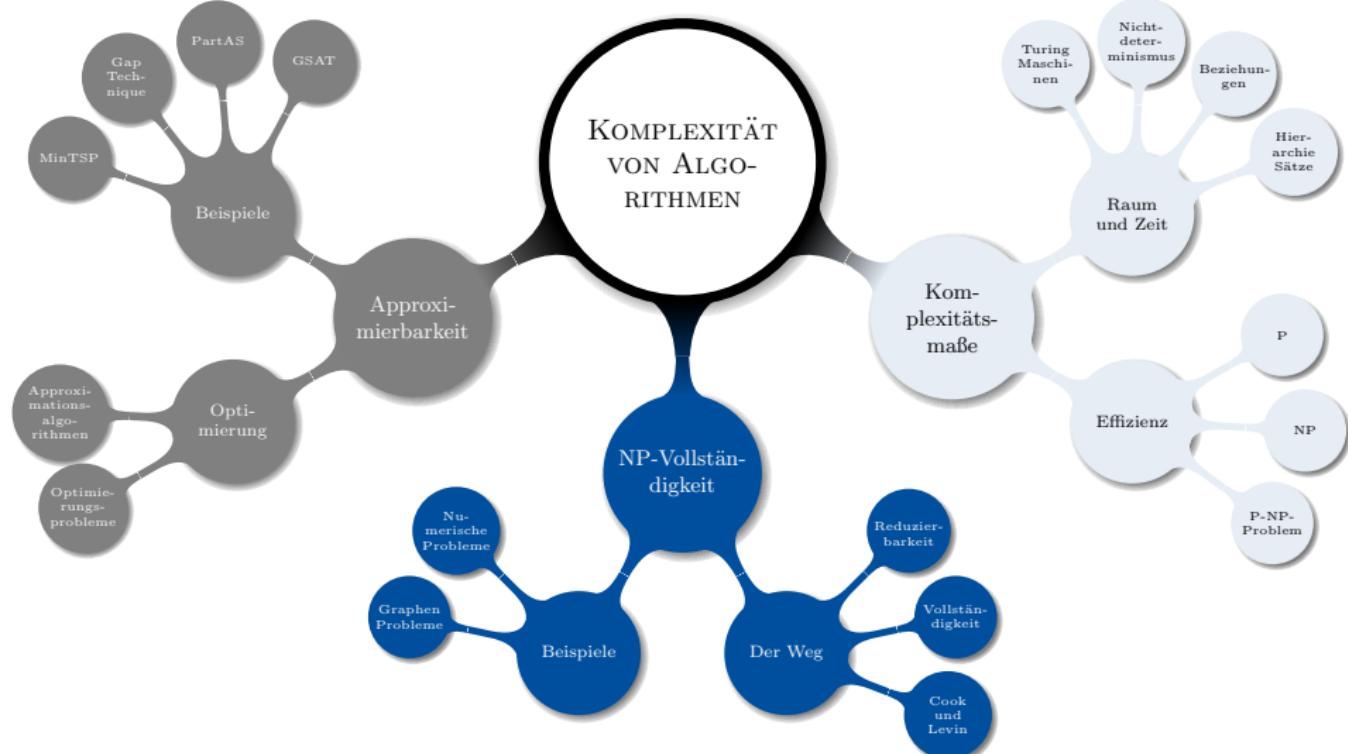


Arne Meier und Heribert Vollmer.
Komplexität von Algorithmen,
lehmanns media, 2015.

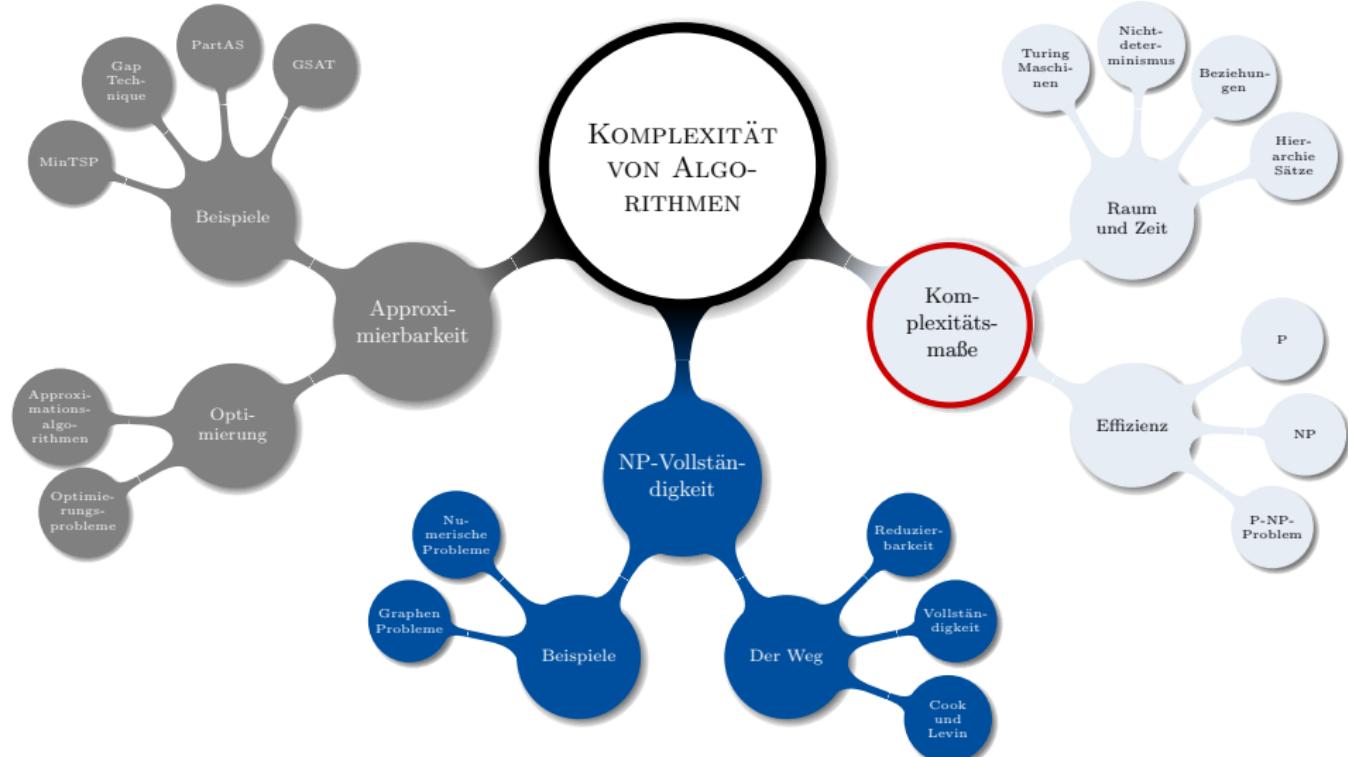
Voraussichtlich: Ende April.

Themenschnittmenge: Obermenge von dieser Veranstaltung.

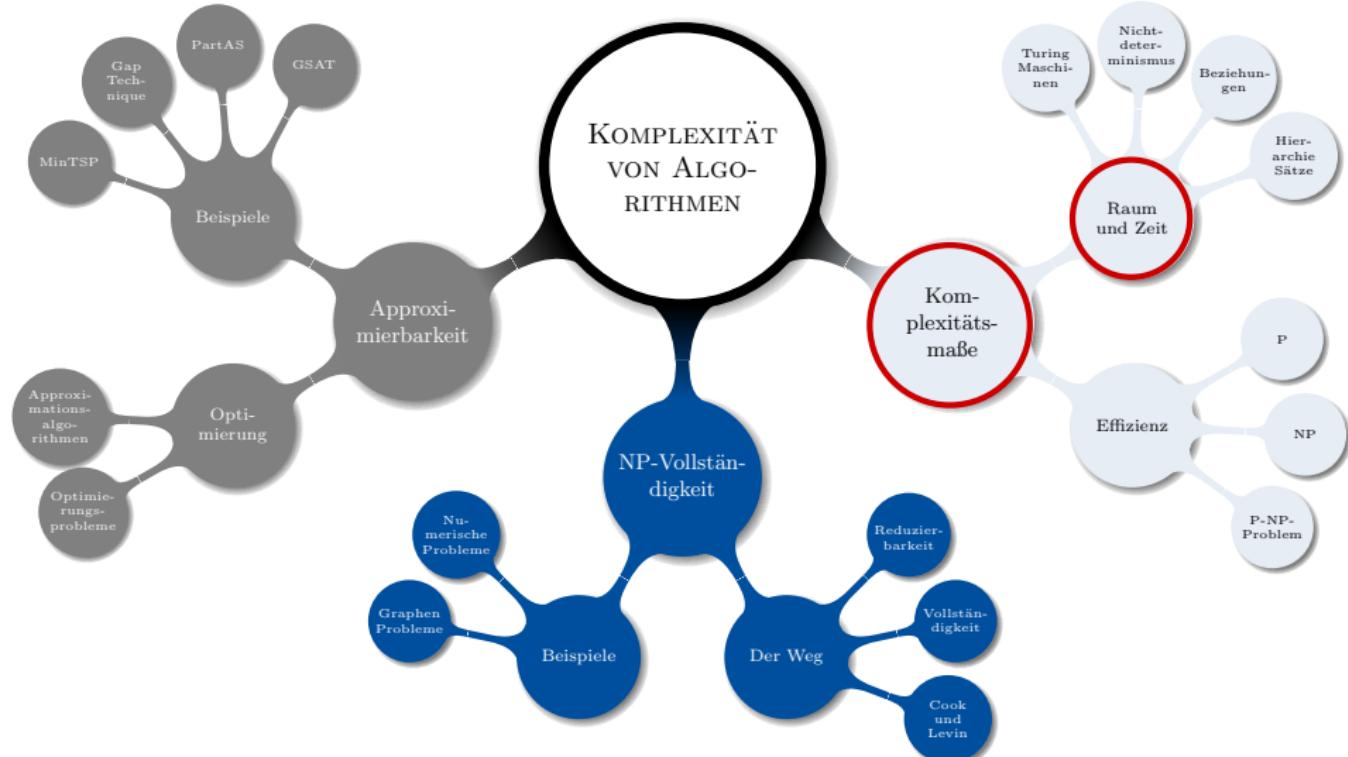
Inhalt



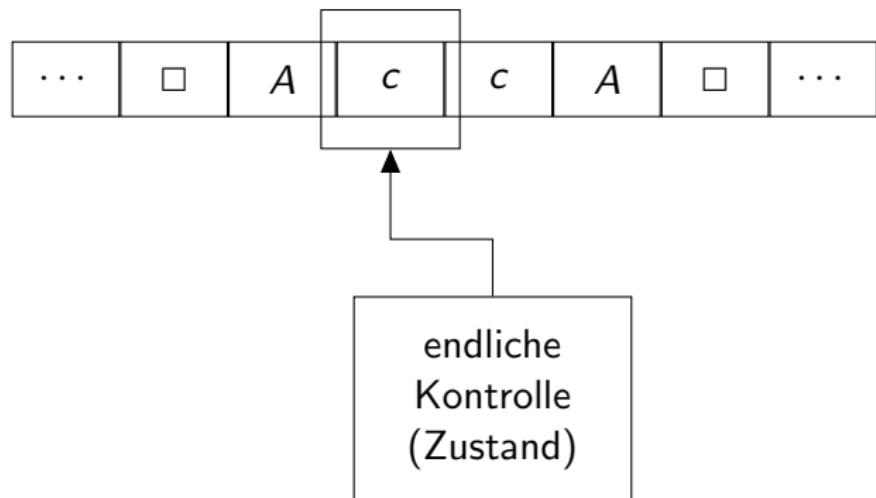
Inhalt



Inhalt



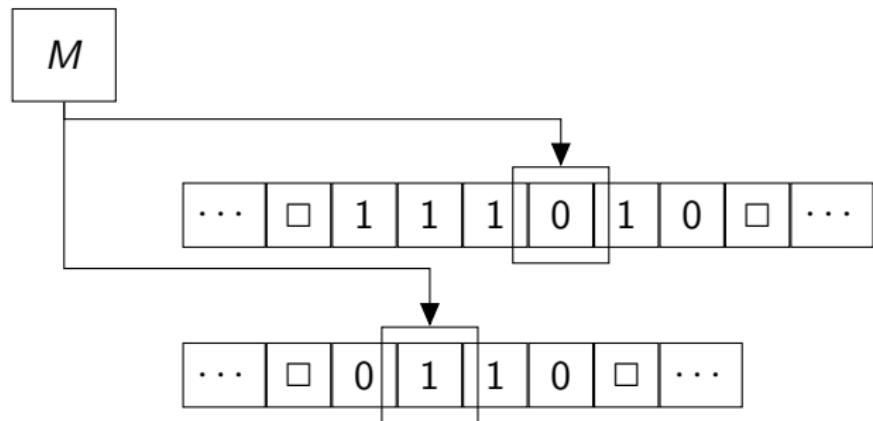
Turing Maschinen



Alan Turing,

wikimedia, public domain

Turing Maschinen



Alan Turing,

wikimedia, public domain

Turing Maschinen: Wie schreibt man sie auf?

$$M = (Z, \Gamma, \delta, z_0, E),$$

wobei

Turing Maschinen: Wie schreibt man sie auf?

$$M = (Z, \Gamma, \delta, z_0, E),$$

wobei

- Z : Menge der Zustände,

Turing Maschinen: Wie schreibt man sie auf?

$$M = (Z, \Gamma, \delta, z_0, E),$$

wobei

- Z : Menge der Zustände,
- Γ : Menge der Bandsymbole,

Turing Maschinen: Wie schreibt man sie auf?

$$M = (Z, \Gamma, \delta, z_0, E),$$

wobei

- Z : Menge der Zustände,
- Γ : Menge der Bandsymbole,
- $\delta : Z \times \Gamma \rightarrow Z \times \Gamma \times \{L, R, N\}$,

Turing Maschinen: Wie schreibt man sie auf?

$$M = (Z, \Gamma, \delta, z_0, E),$$

wobei

- Z : Menge der Zustände,
- Γ : Menge der Bandsymbole,
- $\delta : Z \times \Gamma \rightarrow Z \times \Gamma \times \{L, R, N\}$,
- z_0 : Startzustand,

Turing Maschinen: Wie schreibt man sie auf?

$$M = (Z, \Gamma, \delta, z_0, E),$$

wobei

- Z : Menge der Zustände,
- Γ : Menge der Bandsymbole,
- $\delta : Z \times \Gamma \rightarrow Z \times \Gamma \times \{L, R, N\}$,
- z_0 : Startzustand,
- E : Menge der akzeptierenden Zustände.

Turing Maschinen: Wie schreibt man sie auf?

$$M = (Z, \Gamma, \delta, z_0, E),$$

wobei

- Z : Menge der Zustände,
- Γ : Menge der Bandsymbole,
- $\delta : Z \times \Gamma \rightarrow Z \times \Gamma \times \{L, R, N\}$,
- z_0 : Startzustand,
- E : Menge der akzeptierenden Zustände.

Nun ein Beispiel: Erhöhe Zählerstand um eins.

Zeitbedarf und Platzbedarf

Definition (Zeit und Platz)

Sei M eine Turingmaschine. Sei $f: \mathbb{N} \rightarrow \mathbb{N}$.

M arbeitet in *Zeit f*, falls für alle n und für alle Wörter w der Länge n die Laufzeit von M bei Eingabe w durch $f(n)$ beschränkt ist.

M arbeitet in *Platz f*, falls für alle n und für alle Wörter w der Länge n der Speicherbedarf von M bei Eingabe w durch $f(n)$ beschränkt ist.

Abschätzung des Zeitbedarfs

Definition (O -Notation)

Seien $f, g: \mathbb{N} \rightarrow \mathbb{N}$ zwei Funktionen. Dann ist:

$f \in O(g)$, falls es $c, n_0 \in \mathbb{N}$ gibt, sodass für alle $n \geq n_0$ gilt:

$$f(n) \leq c \cdot g(n).$$

Es gilt $f \in o(g)$, falls

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0.$$



Don Knuth,

wikimedia, public domain

Deterministische Klassen für Raum und Zeit

Definition (TIME)

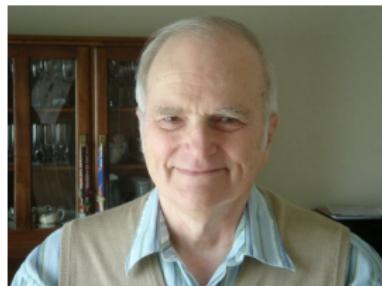
Sei $t: \mathbb{N} \rightarrow \mathbb{N}$. Die Komplexitätsklasse **TIME(t)** besteht aus allen Sprachen A , für die es eine Mehrband-Turingmaschine gibt, die A entscheidet und **in Zeit $O(t)$ arbeitet**.

Definition (SPACE)

Sei $s: \mathbb{N} \rightarrow \mathbb{N}$. Die Komplexitätsklasse **SPACE(s)** besteht aus allen Sprachen A , für die es eine Mehrband-Turingmaschine mit Eingabeband gibt, die A entscheidet und **in Platz $O(s)$ arbeitet**.



Harmanis
und
Stearns



Wie viel besser sind Mehrband-Maschinen?

Satz 1

Sei $t(n) \geq n$ eine Funktion. Jede Mehrband-Turingmaschine M , die in Zeit t arbeitet, kann von einer **1-Band-Turingmaschine M'** simuliert werden, die **in Zeit $O(t^2)$** arbeitet.

Es geht noch ein Stück besser

Satz 2

Sei $t(n) \geq n$ eine Funktion. Jede Mehrband-Turingmaschine M , die in Zeit t arbeitet, kann von einer 2-Band-Turingmaschine M' simuliert werden, die in Zeit $O(t \cdot \log(t))$ arbeitet.

Hier ohne Beweis.

Definition

Sei $t: \mathbb{N} \rightarrow \mathbb{N}$. Die Komplexitätsklasse $\text{NTIME}(t)$ besteht aus allen Sprachen A , für die es eine Mehrband-NTM gibt, die A entscheidet und **in Zeit** $O(t)$ arbeitet.

Definition

Sei $s: \mathbb{N} \rightarrow \mathbb{N}$. Die Komplexitätsklasse $\text{NSPACE}(s)$ besteht aus allen Sprachen A , für die es eine Mehrband-NTM mit Eingabeband gibt, die A entscheidet und **in Platz** $O(s)$ arbeitet.

NTM-Zeit in DTM-Zeit

Satz 3

Sei $t(n) \geq n$ eine Funktion. Dann gilt

$$\text{NTIME}(t(n)) \subseteq \text{TIME}(2^{O(t(n))}).$$

NTM-Zeit in DTM-Platz

Satz 4

Sei $t(n) \geq n$ eine Funktion. Dann gilt

$$\text{NTIME}(t(n)) \subseteq \text{SPACE}(t(n)).$$

Satz 5

Sei $s(n) \geq \log(n)$ eine Funktion. Dann gilt

$$\text{SPACE}(s(n)) \subseteq \text{TIME}(2^{O(s(n))}).$$

Definition

Sei $f: \mathbb{N} \rightarrow \mathbb{N}$ eine Funktion. Wir nennen f **raumkonstruierbar**, falls es eine deterministische Turingmaschine gibt, die bei Eingabe eines Wortes x einen **Platzbedarf von genau $f(|x|)$** hat.

Satz 7

Sei $s(n) \geq \log(n)$ raumkonstruierbar. Dann gilt

$$\text{NSPACE}(s(n)) \subseteq \text{TIME}(2^{O(s(n))}).$$



Satz 8 (Satz von Savitch, 1970)

Sei $s(n) \geq \log(n)$ raumkonstruierbar. Dann gilt

$$\text{NSPACE}(s(n)) \subseteq \text{SPACE}((s(n))^2).$$

Walter Savitch,

<http://cseweb.ucsd.edu/users/savitch/>

Platzhierarchiesatz

Satz 9

Sei $s(n) \geq \log(n)$ eine raumkonstruierbare Funktion. Dann gibt es eine Sprache in $\text{SPACE}(s)$, die **nicht** mit Platzbedarf $o(s)$ entschieden werden kann.

Korollar

Sei $s_1 \in o(s_2)$ und s_2 raumkonstruierbar. Dann ist

$$\text{SPACE}(s_1) \subsetneq \text{SPACE}(s_2).$$

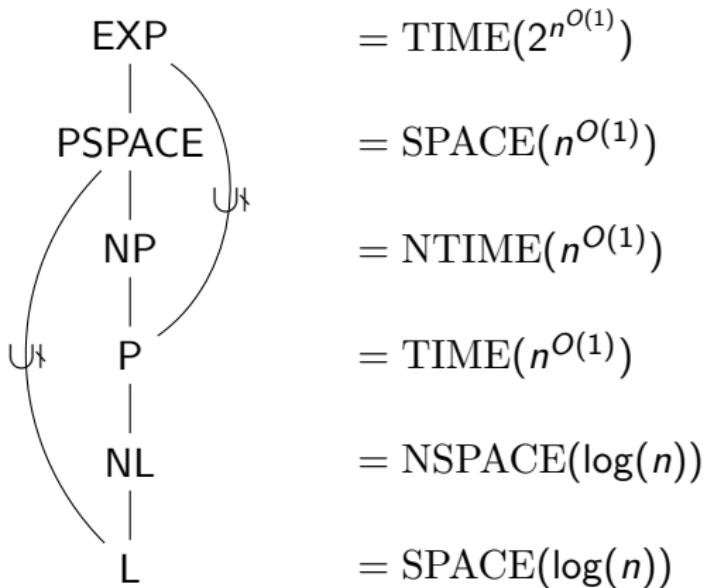
Zeithierarchiesatz

Satz 10

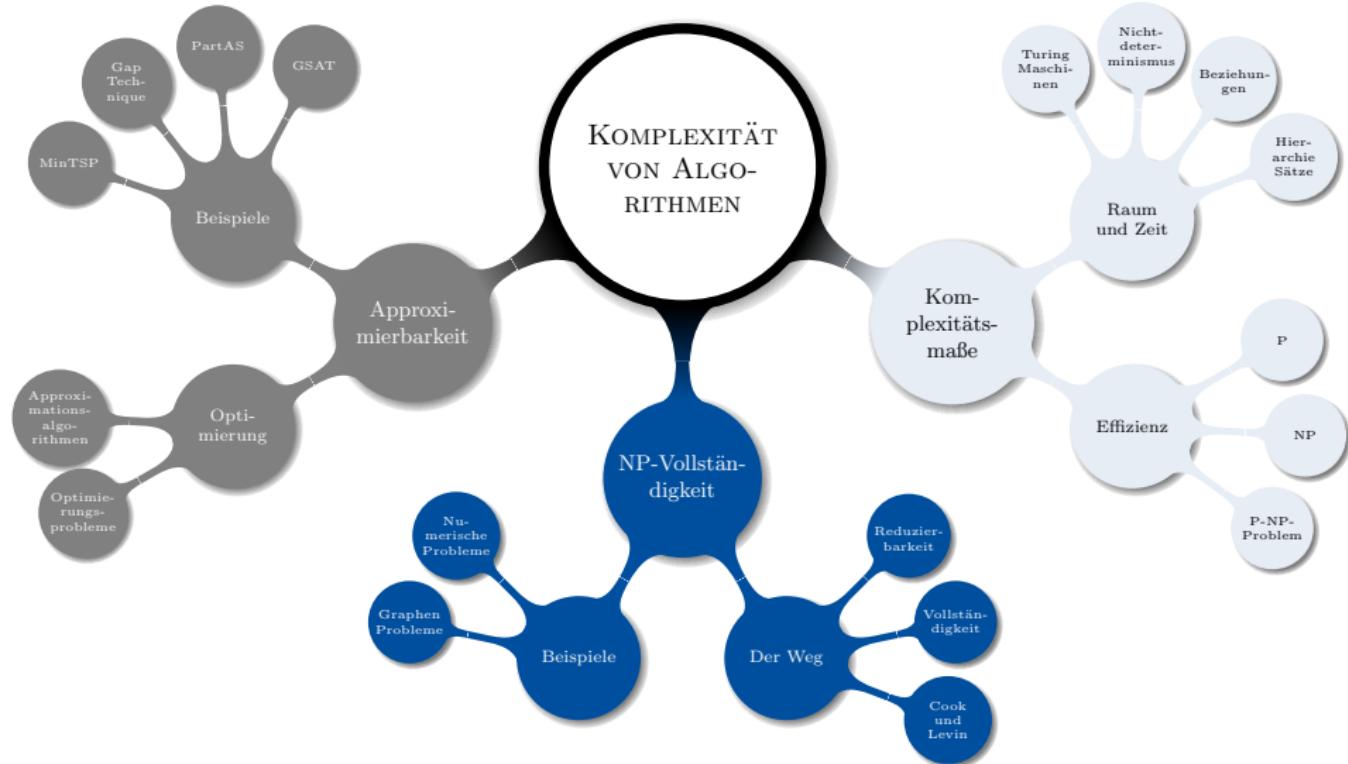
Sei $t_1(n) \in o\left(\frac{t_2(n)}{\log(t_2(n))}\right)$, t_2 „zeitkonstruierbar“. Dann ist

$$\text{TIME}(t_1) \subsetneq \text{TIME}(t_2).$$

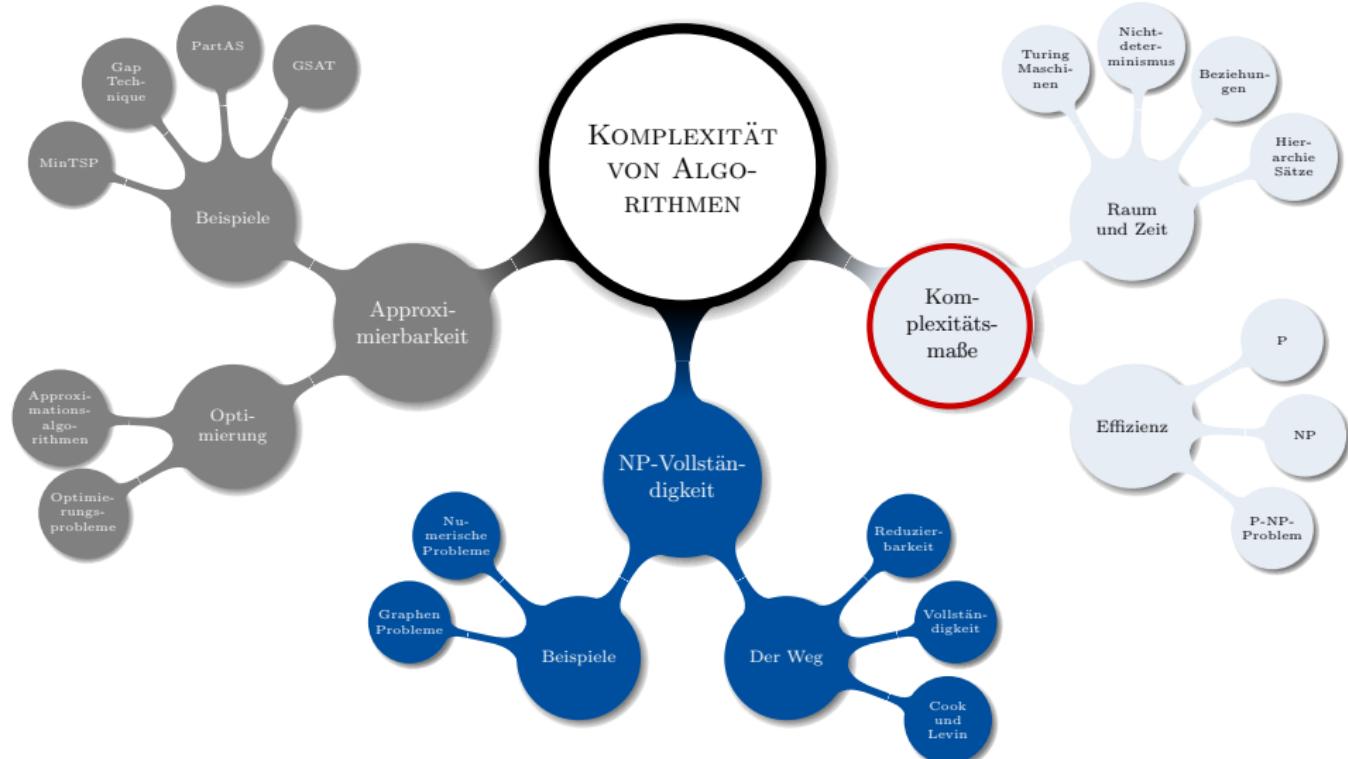
Spezielle Komplexitätsklassen



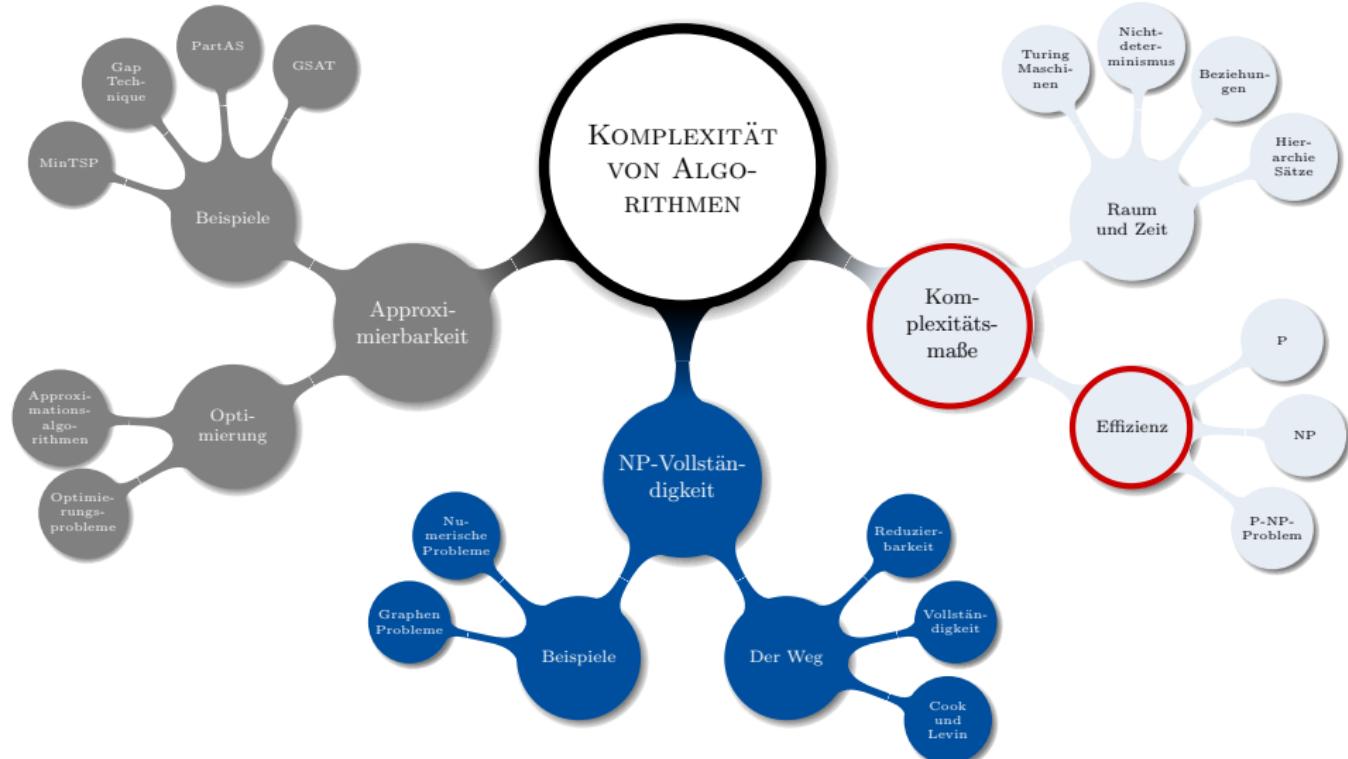
Wie geht es weiter?



Wie geht es weiter?



Wie geht es weiter?



Was bedeutet Polynomialzeit wirklich?

Laufzeit	Eingabegröße $n =$		
	10	20	30
n	10^{-9} s	$2 \cdot 10^{-9}$ s	$3 \cdot 10^{-9}$ s
n^3	10^{-7} s	$8 \cdot 10^{-7}$ s	$2.7 \cdot 10^{-6}$ s
n^5	10^{-5} s	$3.2 \cdot 10^{-4}$ s	$2.43 \cdot 10^{-3}$ s
2^n	$1.024 \cdot 10^{-7}$ s	10^{-4} s	0.1 s
3^n	$5.9 \cdot 10^{-6}$ s	0.35 s	5.7 h

Was bedeutet Polynomialzeit wirklich?

Eingabegröße $n =$

Laufzeit	10	20	30
n	10^{-9} s	$2 \cdot 10^{-9}$ s	$3 \cdot 10^{-9}$ s
n^3	10^{-7} s	$8 \cdot 10^{-7}$ s	$2.7 \cdot 10^{-6}$ s
n^5	10^{-5} s	$3.2 \cdot 10^{-4}$ s	$2.43 \cdot 10^{-3}$ s
2^n	$1.024 \cdot 10^{-7}$ s	10^{-4} s	0.1 s
3^n	$5.9 \cdot 10^{-6}$ s	0.35 s	5.7 h

Eingabegröße $n =$

Laufzeit	40	50	60
n	$4 \cdot 10^{-9}$ s	$5 \cdot 10^{-9}$ s	$6 \cdot 10^{-9}$ s
n^3	$6.4 \cdot 10^{-6}$ s	$1.25 \cdot 10^{-5}$ s	$2.16 \cdot 10^{-5}$ s
n^5	$1.024 \cdot 10^{-2}$ s	$3.125 \cdot 10^{-2}$ s	$7.776 \cdot 10^{-2}$ s
2^n	110 s	31 h	3.7 a
3^n	38 a	$2 \cdot 10^6$ a	$1.3 \cdot 10^{11}$ a

Was können wir in einer Stunde lösen?

Laufzeit	mit heutigem Computer	mit $100 \times$ schnellerem Computer	mit $1000 \times$ schnellerem Computer
n	N_1	$100N_1$	$1000N_1$
n^3	N_2	$4.64N_2$	$10N_2$
n^5	N_3	$2.5N_3$	$3.98N_3$
2^n	N_4	$N_4 + 6.64$	$N_4 + 9.97$
3^n	N_5	$N_5 + 4.19$	$N_5 + 6.29$

Die Komplexitätsklasse P ist die Klasse der effizient lösbarer Probleme.

Ein Problem aus P

$$\text{PATH} = \left\{ \langle G, s, t \rangle \mid \begin{array}{l} G = (V, E) \text{ ist ein gerichteter Graph,} \\ s, t \in V \text{ und es gibt einen Pfad von } s \\ \text{nach } t \end{array} \right\}.$$

Algorithmus für PATH $\in \text{P}$

Eingabe: $G = (V, E)$ gerichteter Graph, Knoten $s, t \in V$

Algorithmus für PATH $\in \text{P}$

Eingabe: $G = (V, E)$ gerichteter Graph, Knoten $s, t \in V$

- 1: Markiere s .

Algorithmus für PATH $\in \text{P}$

Eingabe: $G = (V, E)$ gerichteter Graph, Knoten $s, t \in V$

- 1: Markiere s .
- 2: **while** es wurde ein Knoten markiert **do**

Algorithmus für PATH $\in \text{P}$

Eingabe: $G = (V, E)$ gerichteter Graph, Knoten $s, t \in V$

1: Markiere s .

2: **while** es wurde ein Knoten markiert **do**

3: **for** jede Kante $(u, v) \in E$, u markiert, v unmarkiert **do**

Algorithmus für PATH $\in \text{P}$

Eingabe: $G = (V, E)$ gerichteter Graph, Knoten $s, t \in V$

- 1: Markiere s .
- 2: **while** es wurde ein Knoten markiert **do**
- 3: **for** jede Kante $(u, v) \in E$, u markiert, v unmarkiert **do**
- 4: Markiere v .

Algorithmus für PATH $\in \text{P}$

Eingabe: $G = (V, E)$ gerichteter Graph, Knoten $s, t \in V$

- 1: Markiere s .
- 2: **while** es wurde ein Knoten markiert **do**
- 3: **for** jede Kante $(u, v) \in E$, u markiert, v unmarkiert **do**
- 4: Markiere v .
- 5: **if** t ist markiert **then**
- 6: **akzeptiere**

Algorithmus für PATH $\in \text{P}$

Eingabe: $G = (V, E)$ gerichteter Graph, Knoten $s, t \in V$

- 1: Markiere s .
- 2: **while** es wurde ein Knoten markiert **do**
- 3: **for** jede Kante $(u, v) \in E$, u markiert, v unmarkiert **do**
- 4: Markiere v .
- 5: **if** t ist markiert **then**
- 6: **akzeptiere**
- 7: **verwerfe**

NP – the class of dashed hopes and idle dreams

Definition (Polynomielle Überprüfbarkeit)

Eine Sprache A ist **polynomiell-überprüfbar**, falls es einen Algorithmus V gibt, sodass für alle Eingaben w gilt:

$w \in A$ gdw. es gibt ein x , sodass V bei Eingabe $\langle w, x \rangle$ akzeptiert.

Die Laufzeit von V bei Eingabe $\langle w, x \rangle$ ist dabei durch ein Polynom in $|w|$ beschränkt.

Sprechweise: x ist ein **Zertifikat** oder **Beweis** für w , falls V Eingabe $\langle w, x \rangle$ akzeptiert.

NP – Überprüfbarkeit: ja, Entscheidbarkeit: ?

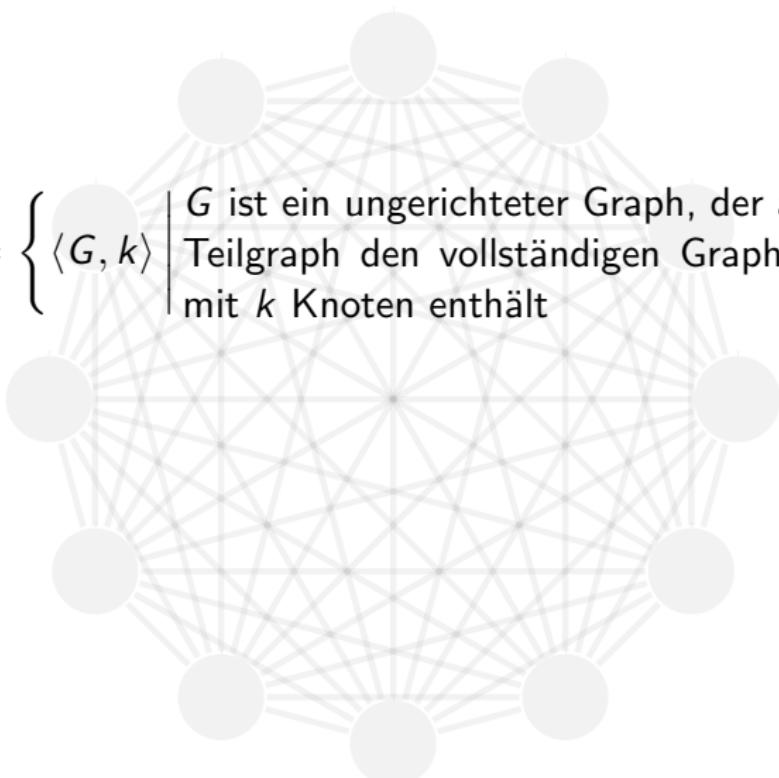
Satz 11

Eine Sprache ist in NP gdw. sie polynomiell-überprüfbar ist.

Die Komplexitätsklasse NP ist die Klasse der effizient überprüfbaren Probleme.

CLIQUE: ein Problem in der Klasse NP

CLIQUE= $\left\{ \langle G, k \rangle \mid \begin{array}{l} G \text{ ist ein ungerichteter Graph, der als} \\ \text{Teilgraph den vollständigen Graphen} \\ \text{mit } k \text{ Knoten enthält} \end{array} \right\}$



CLIQUE: ein Problem in der Klasse NP

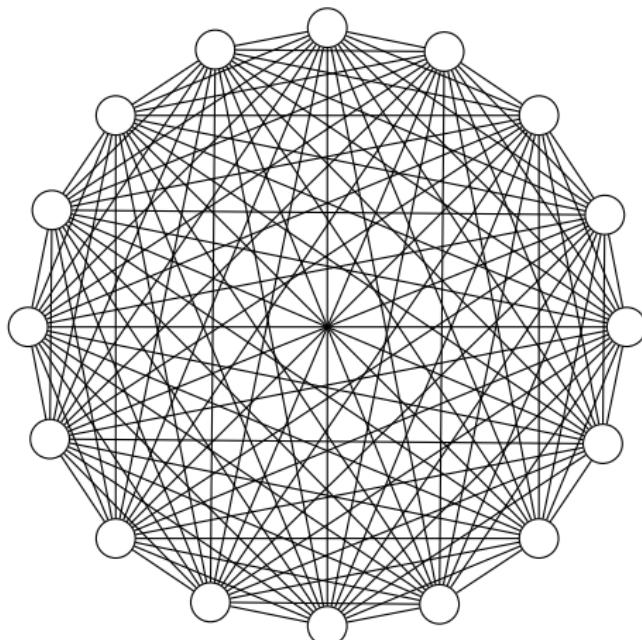
CLIQUE= $\left\{ \langle G, k \rangle \mid \begin{array}{l} G \text{ ist ein ungerichteter Graph, der als} \\ \text{Teilgraph den vollständigen Graphen} \\ \text{mit } k \text{ Knoten enthält} \end{array} \right\}$

Eingabe: $G = (V, E)$ ungerichteter Graph, $k \in \mathbb{N}$, $X \subseteq V$

- 1: Überprüfe, ob $|X| \geq k$.
- 2: Überprüfe, ob je zwei Knoten in X durch Kante in G verbunden sind.
- 3: Falls immer ja: akzeptieren. Sonst: ablehnen.

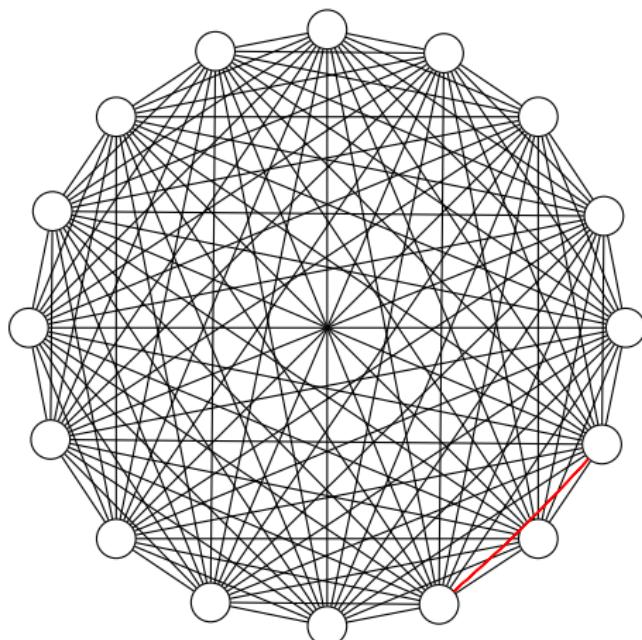
CLIQUE ist einfach? Eher nicht...

Frage: $\langle G, 16 \rangle \in \text{CLIQUE}?$



CLIQUE ist einfach? Eher nicht...

Frage: $\langle G, 16 \rangle \in \text{CLIQUE}?$



Hamilton'sche Pfade liegen auch in NP

Hamilton'scher Pfad:

Ein Pfad, der jeden Knoten **genau** einmal besucht.

$$\text{HAMPATH} = \left\{ \langle G, s, t \rangle \mid \begin{array}{l} G \text{ ist ein gerichteter} \\ \text{Graph, der einen Hamilton'schen Pfad von } s \\ \text{nach } t \text{ besitzt} \end{array} \right\}$$



William Rowan
Hamilton,

wikimedia.org, public domain

Hamilton'sche Pfade liegen auch in NP

Hamilton'scher Pfad:

Ein Pfad, der jeden Knoten **genau** einmal besucht.

$$\text{HAMPATH} = \left\{ \langle G, s, t \rangle \mid \begin{array}{l} G \text{ ist ein gerichteter} \\ \text{Graph, der einen Hamilton'schen Pfad von } s \\ \text{nach } t \text{ besitzt} \end{array} \right\}$$



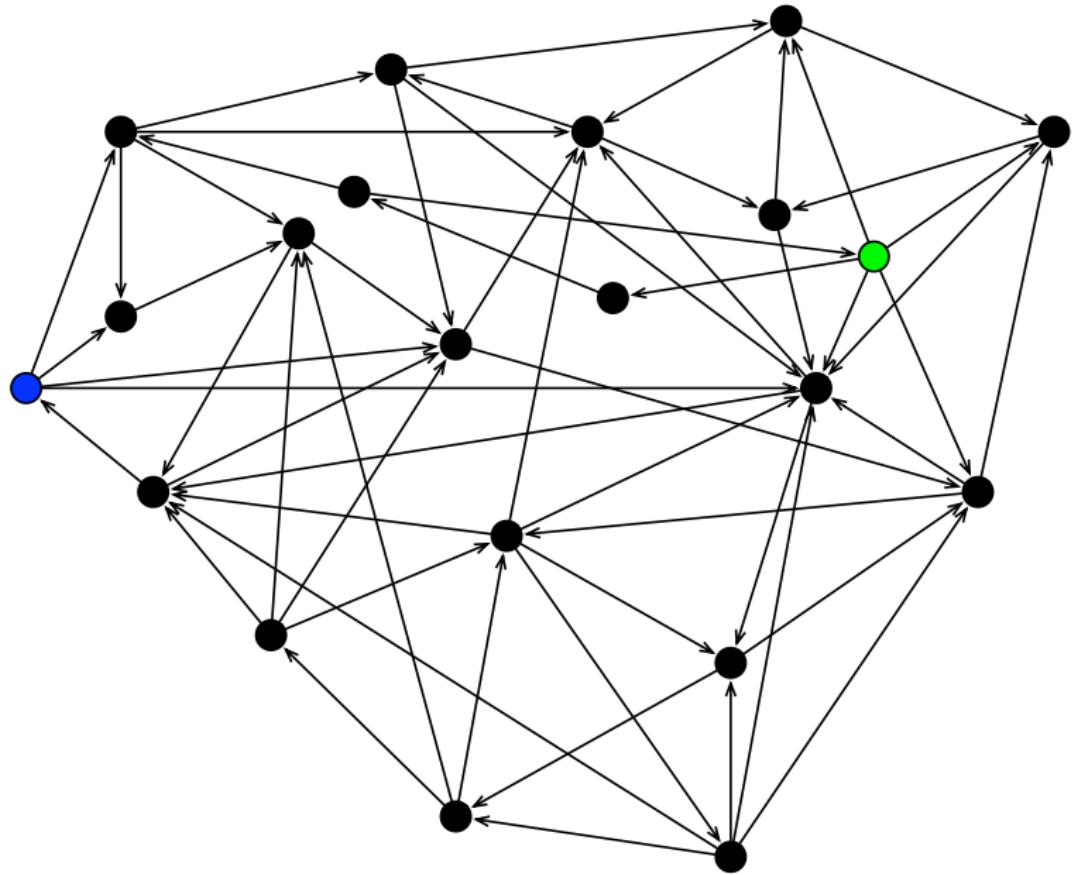
William Rowan
Hamilton,

wikimedia.org, public domain

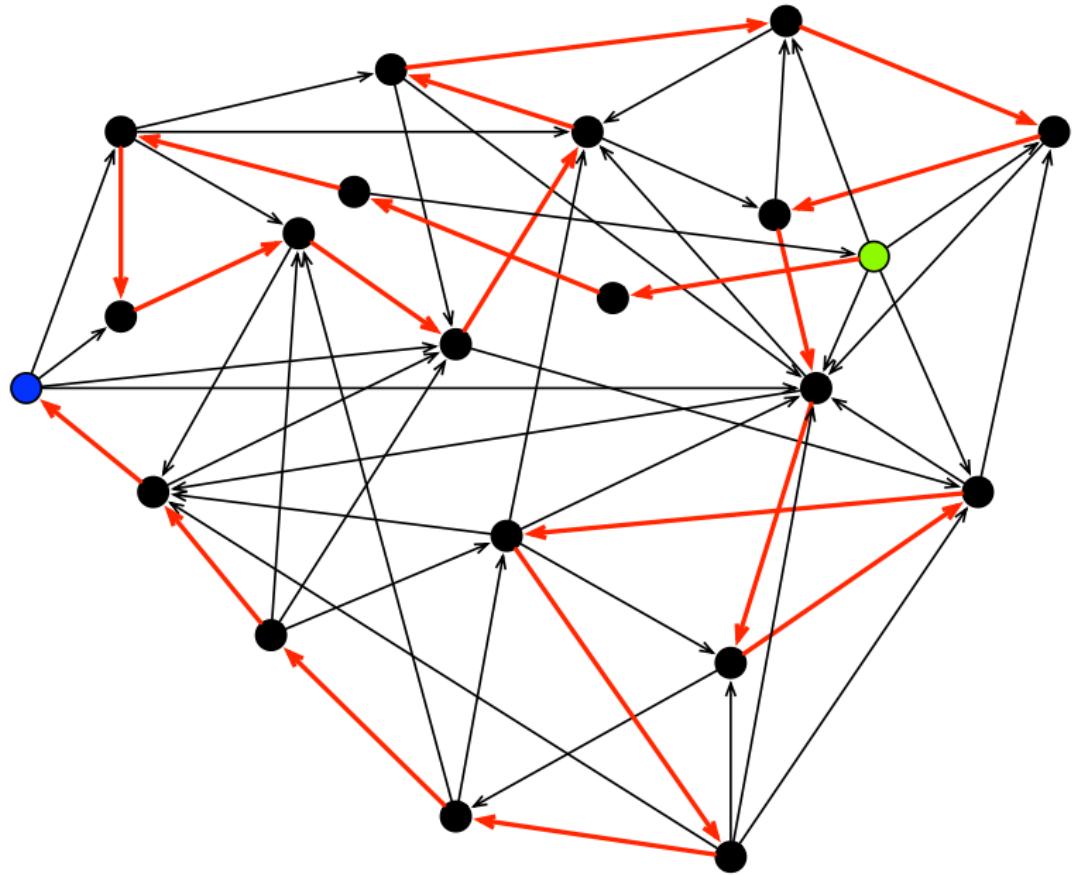
Eingabe: Ger. Graph $G = (V, E)$, Knoten $s, t \in V$, Pfad X

- 1: Überprüfe, ob X einen Pfad in G von s nach t kodiert, der jeden Knoten in G genau einmal enthält.
- 2: Falls immer ja: akzeptieren. Sonst: ablehnen.

$\langle G, \text{●}, \text{○} \rangle \in \text{HAMPATH?}$



$\langle G, \text{●}, \text{○} \rangle \in \text{HAMPATH?}$



Das Bezahlproblem liegt auch in NP

$$\text{SUBSET-SUM} = \left\{ \langle x_1, \dots, x_n, t \rangle \mid \begin{array}{l} x_1, \dots, x_n, t \in \mathbb{N} \text{ und es} \\ \text{existiert eine Teilmenge} \\ I \subseteq \{1, \dots, n\} \text{ mit } \sum_{i \in I} x_i = t \end{array} \right\}$$

Das Bezahlproblem liegt auch in NP

$$\text{SUBSET-SUM} = \left\{ \langle x_1, \dots, x_n, t \rangle \mid \begin{array}{l} x_1, \dots, x_n, t \in \mathbb{N} \text{ und es} \\ \text{existiert eine Teilmenge} \\ I \subseteq \{1, \dots, n\} \text{ mit } \sum_{i \in I} x_i = t \end{array} \right\}$$

Eingabe: Liste von natürlichen Zahlen x_1, \dots, x_n natürliche Zahl t , Menge $I \subseteq \{1, \dots, n\}$

- 1: Prüfe, dass $|I| \leq n$ gilt.
- 2: Prüfe, dass nur Indizes aus $\{1, \dots, n\}$ in I auftauchen.
- 3: Berechne $\sum_{i \in I} x_i$ und teste ob sie gleich t ist.
- 4: Falls immer ja: akzeptieren. Sonst: ablehnen.

Kurzes Logik Repetitorium

- Variablen nehmen Werte aus $\{0, 1\}$ an.
- **Aussagenlogische Formel φ** : Ausdruck über Variablen und Verknüpfungen \wedge, \vee, \neg .
- Beispiel: $\varphi = (\neg x \vee \neg y) \wedge (x \vee y)$
- **Belegung**: Zuweisung von Wahrheitswerten an Variablen.
- Belegung I erfüllt Formel φ , wenn φ zu 1 evaluiert, wenn die Variablen durch ihre zugewiesenen Werte ersetzt werden.
Schreibweise: $I \models \varphi$.

SAT: Das wichtigste Problem in NP

$\text{SAT} = \{\langle \varphi \rangle \mid \varphi \text{ ist aussagenlogische, erfüllbare Formel}\}$

Das P-NP-Problem

- Ist $P = NP$?
- **Anders formuliert:** Ist jedes effizient überprüfbare Problem auch effizient lösbar?



Stephen Cook*



Leonid Levin[§]

Das P-NP-Problem

- Ist $P = NP$?
- **Anders formuliert:** Ist jedes effizient überprüfbare Problem auch effizient lösbar?
- Offen seit 1971: S. A. Cook, *The complexity of theorem-proving procedures.*;
bzw. 1973: L. A. Levin, *Universal sorting problems.*



Stephen Cook*



Leonid Levin[§]

Das P-NP-Problem

- Ist $P = NP$?
- **Anders formuliert:** Ist jedes effizient überprüfbare Problem auch effizient lösbar?
- Offen seit 1971: S. A. Cook, *The complexity of theorem-proving procedures.*;
bzw. 1973: L. A. Levin, *Universal sorting problems.*
- Die folgenden Aussagen sind alle äquivalent:
 - CLIQUE $\in P$.
 - HAMPATH $\in P$.
 - SUBSET-SUM $\in P$.
 - SAT $\in P$.
 - $P = NP$.

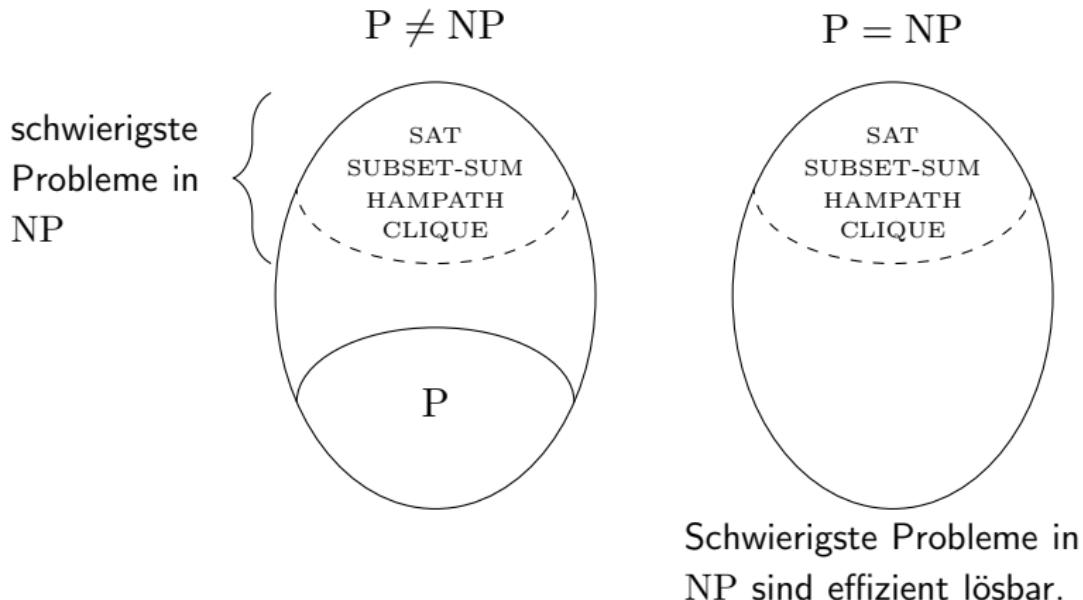


Stephen Cook*

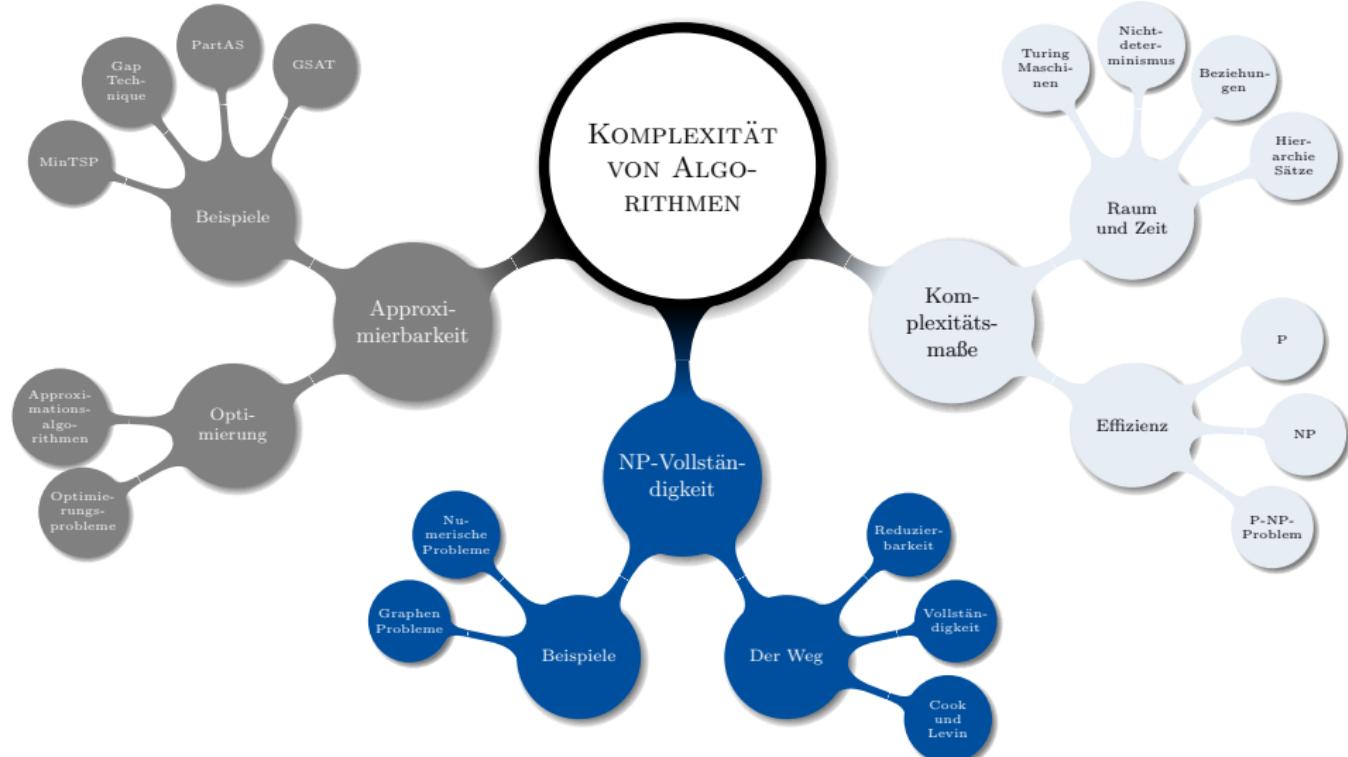


Leonid Levin§

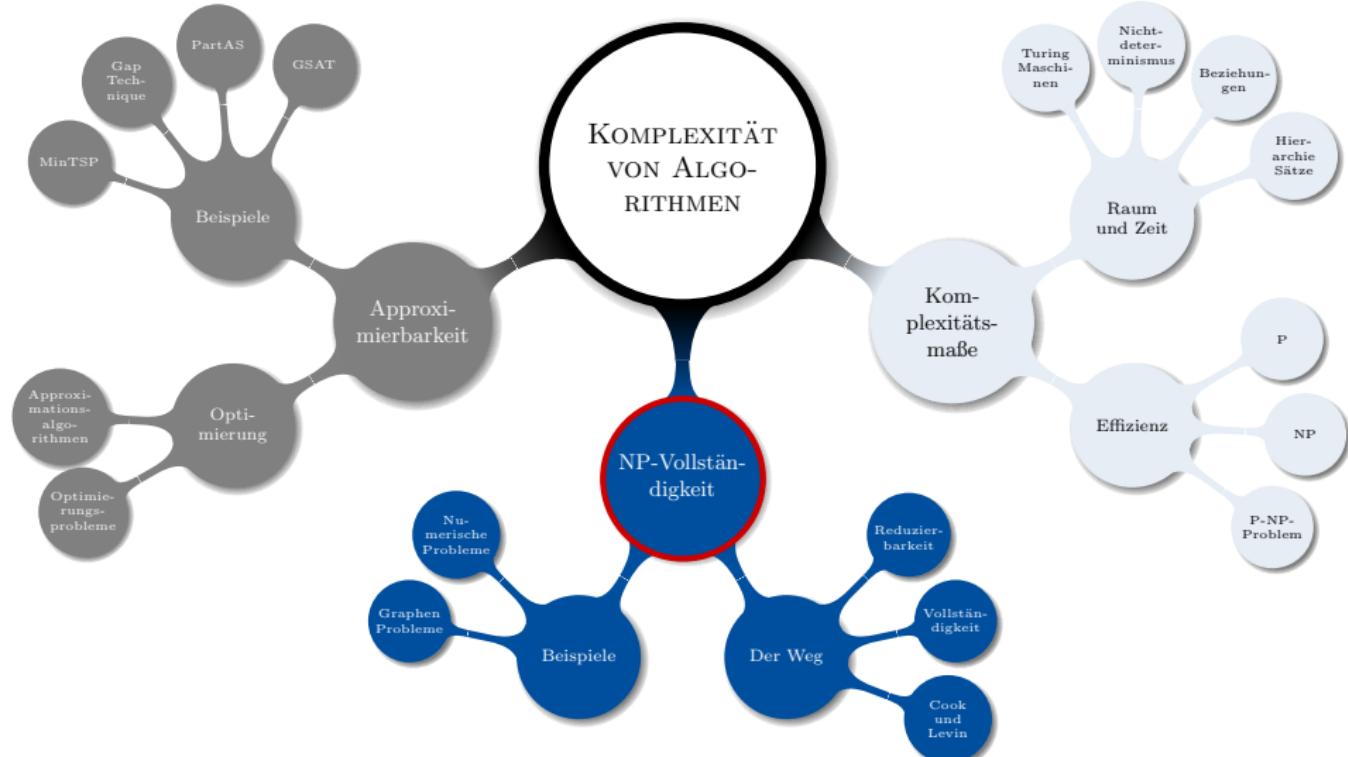
Struktur von NP



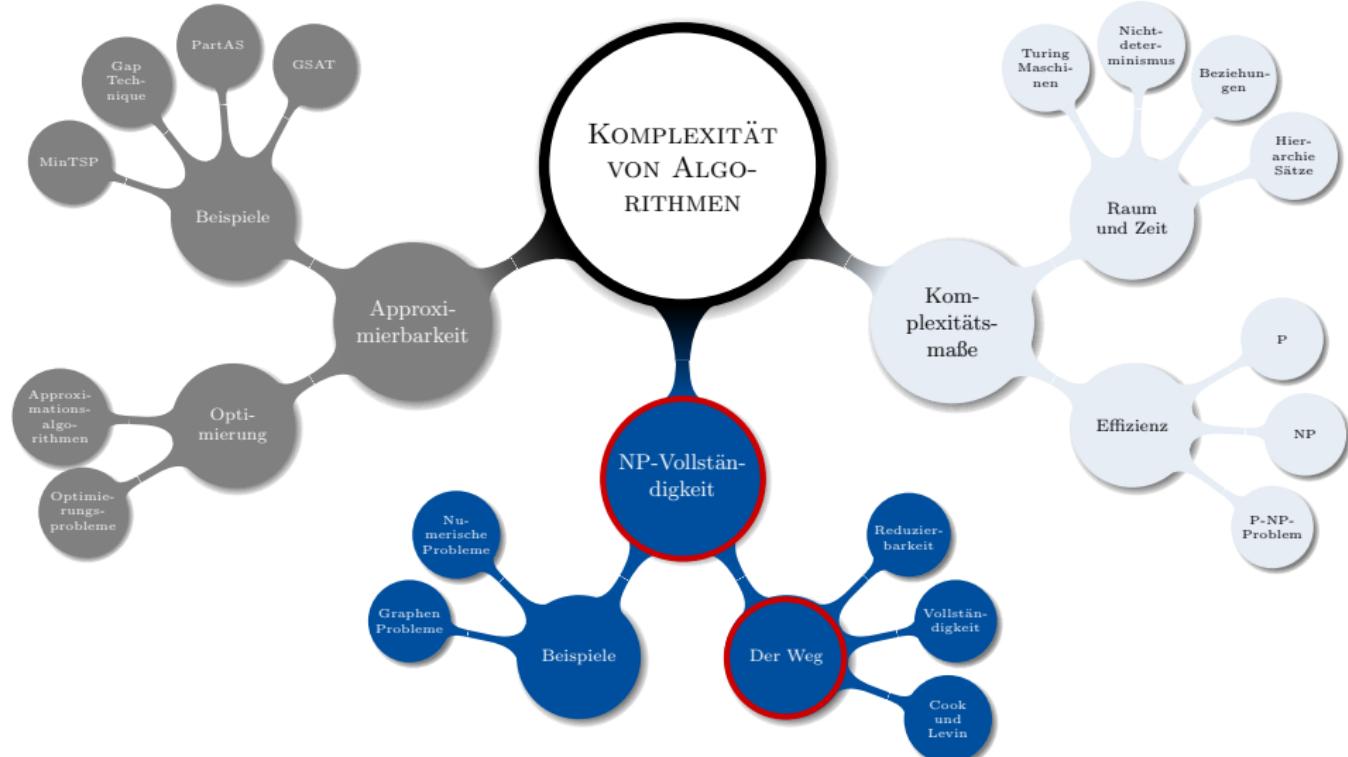
Wie geht es weiter?



Wie geht es weiter?



Wie geht es weiter?



Reduzierbarkeit

Definition (\leq_m^P -Reduktion)

Seien $A \subseteq \Sigma^*$, $B \subseteq \Delta^*$ zwei Sprachen. A heißt auf B in **Polynomialzeit m -reduzierbar**, falls es eine Funktion $f: \Sigma^* \rightarrow \Delta^*$ mit folgenden Eigenschaften gibt:

- Es gibt eine DTM M , die in Polynomialzeit arbeitet und bei Eingabe eines Wortes $x \in \Sigma^*$ das Wort $f(x) \in \Delta^*$ produziert. M hat dazu ein spezielles Ausgabeband, auf dem sich am Ende der Rechnung das Ausgabewort befindet.
Mit anderen Worten: Die Funktion f ist in **Polynomialzeit berechenbar**.
- Für alle $x \in \Sigma^*$ gilt: $x \in A$ gdw. $f(x) \in B$.



Richard Karp,

wikimedia.org, CC

BY-SA 2.0 FR

Wir nennen f auch (Polynomialzeit-)Reduktion von A auf B , in Zeichen, $A \leq_m^P B$.

Satz 12

Sei $A \leq_m^P B$. Dann gilt:

- ① Ist $B \in \text{NP}$, so ist auch $A \in \text{NP}$.
- ② Ist $B \in \text{P}$, so ist auch $A \in \text{P}$.

Weitere Eigenschaften von \leq_m^P -Reduktionen

Satz 13

\leq_m^P ist reflexiv und transitiv.

Vollständigkeit – das (vorerst) letzte Wort

Definition (NP-vollständig)

- ① Eine Sprache B ist **NP-schwer**, falls für alle $A \in \text{NP}$ gilt:
$$A \leq_m^P B.$$
- ② Eine Sprache B ist **NP-vollständig**, falls B NP-schwer ist und $B \in \text{NP}$.



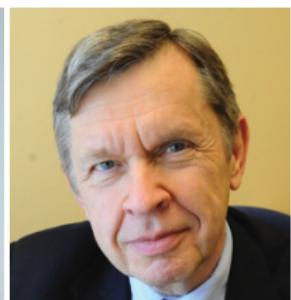
Jeffrey Ullman,

stanford.edu



John Hopcroft,

[wikimedia.org](https://commons.wikimedia.org), public domain



Alfred Aho,

engineering.columbia.edu

Konsequenzen aus der NP-Vollständigkeit

Satz 14

Sei B NP-vollständig. Dann gilt: $B \in P$ gdw. $P = NP$.

Wie zeigt man, dass ein Problem NP-vollständig ist?

Satz 15

Sei B NP-vollständig und $C \in \text{NP}$.

Falls $B \leq_m^P C$, so ist auch C NP-vollständig.

Das erste NP-vollständige Problem

Satz 16 (Satz von Cook und Levin)

SAT ist NP-vollständig.

$$\text{SAT} = \{\langle \varphi \rangle \mid \varphi \text{ ist aussagenlogische, erfüllbare Formel}\}$$

Konstruktionsidee Satz von Cook und Levin

- Generische Reduktion, Pfad im Berechnungsbaum simulieren
- $p(n)$ Laufzeit der NTM

	1	2	\cdots	$p(n) + 1$	\cdots	$p(n) + n + 2$	\cdots	$2p(n) + 4$
1								
:								
$p(n) + 1$								

Konstruktionsidee Satz von Cook und Levin

- Generische Reduktion, Pfad im Berechnungsbaum simulieren
- $p(n)$ Laufzeit der NTM

	1	2	\cdots	$p(n) + 1$	\cdots		$p(n) + n + 2$	\cdots		$2p(n) + 4$	
1	#	□	\cdots	□	q_0	x_1	\cdots	x_n	□	\cdots	□
:											
$p(n) + 1$											#

Konstruktionsidee Satz von Cook und Levin

- Generische Reduktion, Pfad im Berechnungsbaum simulieren
- $p(n)$ Laufzeit der NTM

	1	2	\cdots	$p(n) + 1$	\cdots		$p(n) + n + 2$	\cdots	$2p(n) + 4$			
1	#	□	\cdots	□	q_0	x_1	\cdots	x_n	□	\cdots	□	#
\vdots												
	$p(n) + 1$											

Konstruktionsidee Satz von Cook und Levin

- Generische Reduktion, Pfad im Berechnungsbaum simulieren
- $p(n)$ Laufzeit der NTM

	1	2	\cdots	$p(n) + 1$	\cdots		$p(n) + n + 2$	\cdots	$2p(n) + 4$	
1	#	□	\cdots	□	q_0	x_1	\cdots	x_n	□	\cdots
:										
$p(n) + 1$										

□

Konstruktionsidee Satz von Cook und Levin

- Generische Reduktion, Pfad im Berechnungsbaum simulieren
- $p(n)$ Laufzeit der NTM

	1	2	...	$p(n) + 1$...		$p(n) + n + 2$...		$2p(n) + 4$	
1	#	□	...	□	q_0	x_1	...	x_n	□	...	□
:											
$p(n) + 1$							q_a				
	$q_a \in E$	akzeptierender Zustand									

3KNF – eine gleichmäßige Struktur

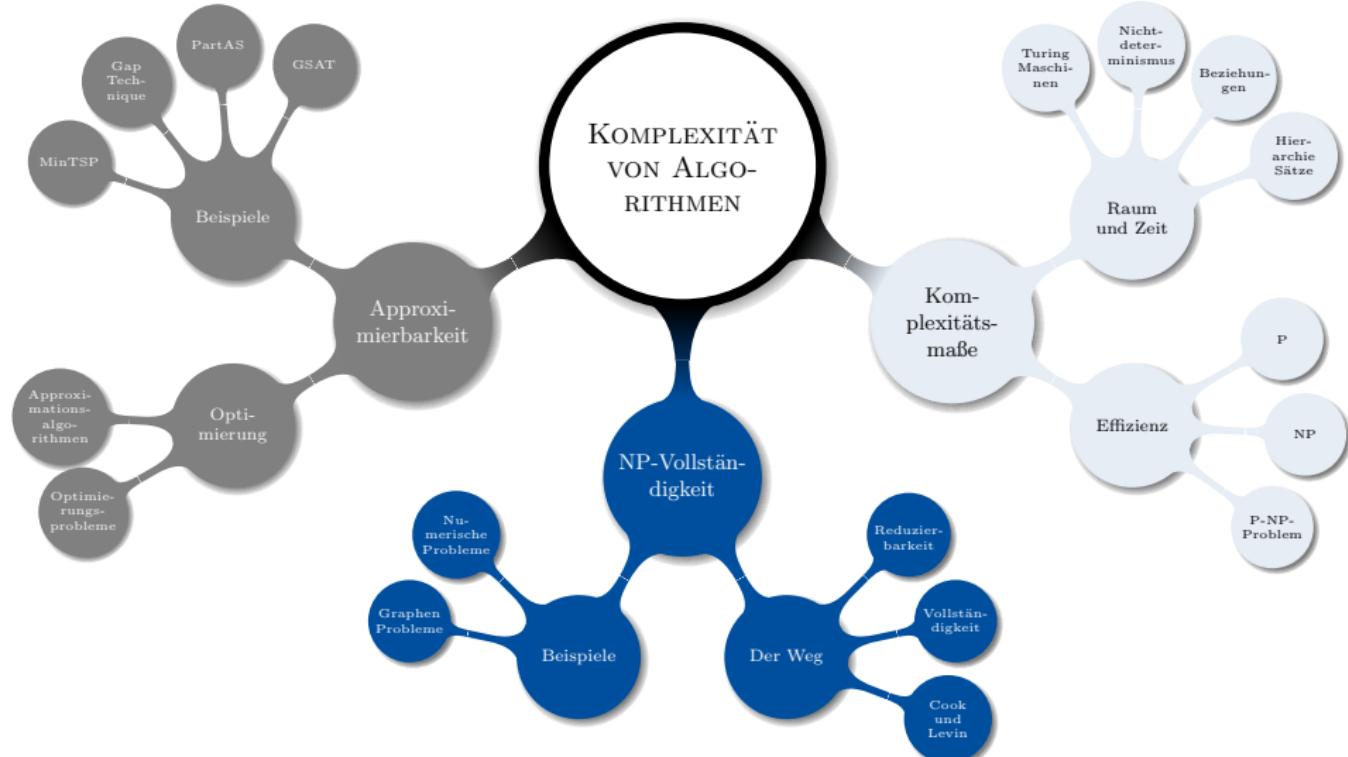
Satz 17

Das Problem

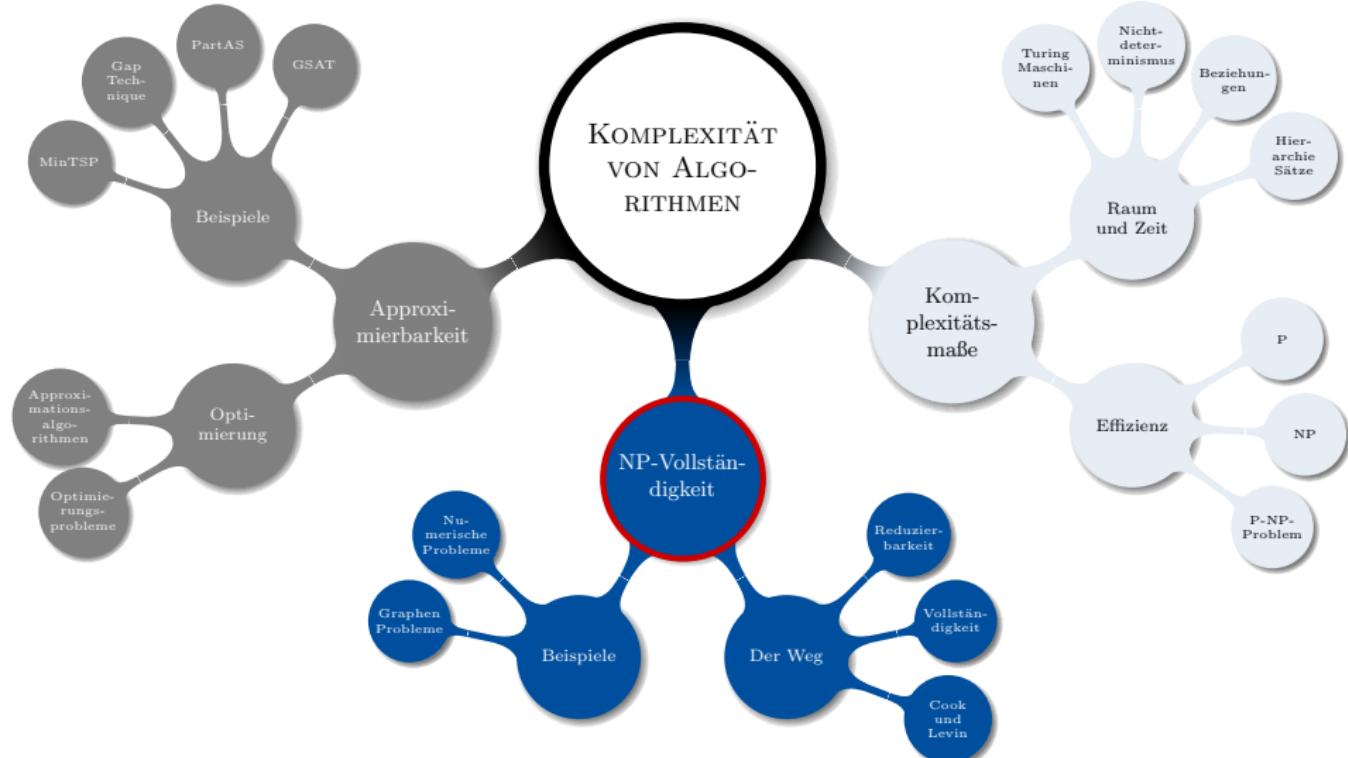
$3SAT = \{\langle \varphi \rangle \mid \varphi \text{ ist eine erfüllbare aussagenlogische Formel in 3KNF}\}$

ist NP-vollständig.

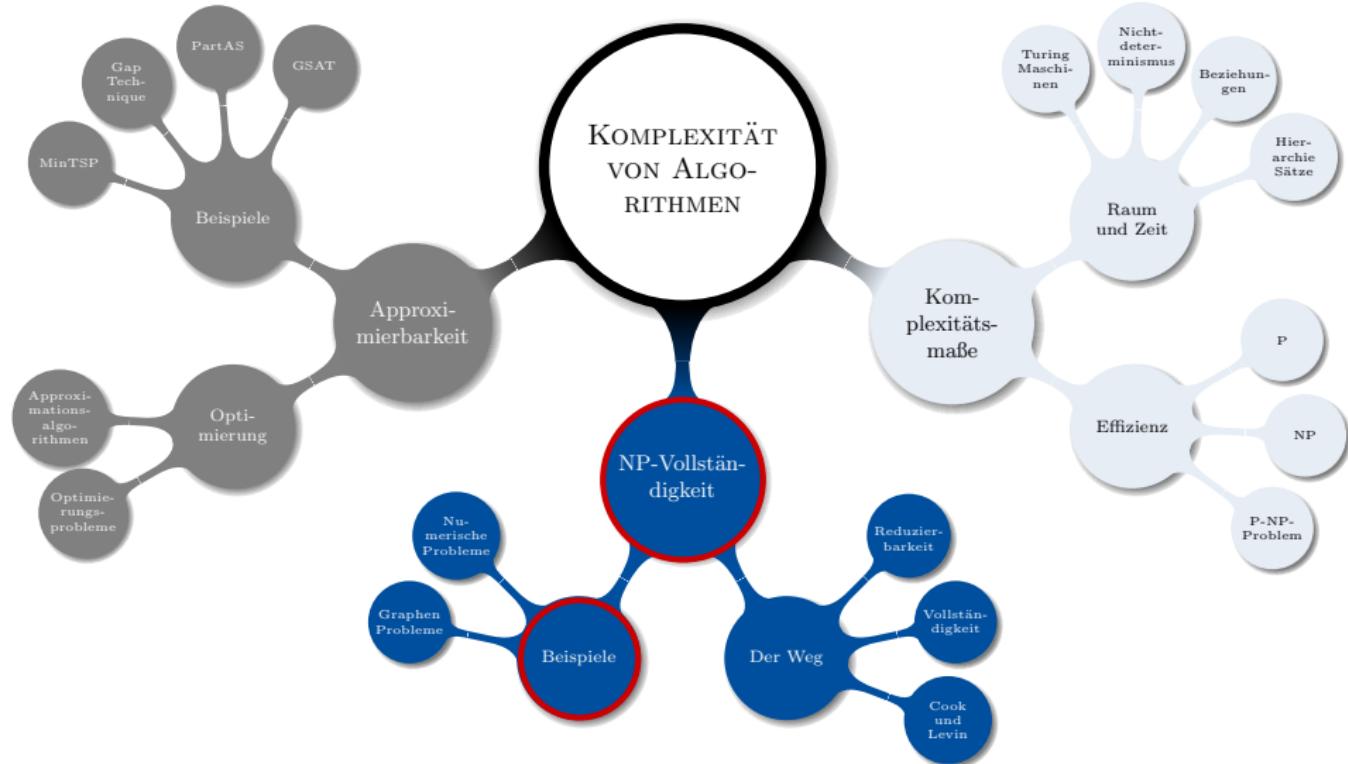
Wie geht es weiter?



Wie geht es weiter?



Wie geht es weiter?



Auch CLIQUE gehört zu den schweren Problemen

Satz 18

CLIQUE ist NP-vollständig.

Auch CLIQUE gehört zu den schweren Problemen

Satz 18

CLIQUE ist NP-vollständig.

→ zunächst schauen wir uns mal ein Beispiel an

$3\text{SAT} \leq_m^P \text{CLIQUE}$

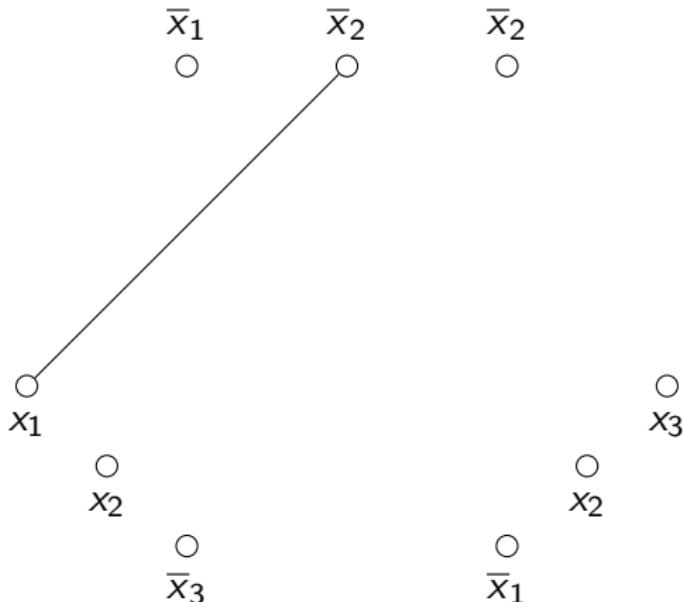
$$\varphi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$$

\bar{x}_1 \bar{x}_2 \bar{x}_2
○ ○ ○

x_1 x_3
○ ○
 x_2 x_2
○ ○
 \bar{x}_3 \bar{x}_1
○ ○

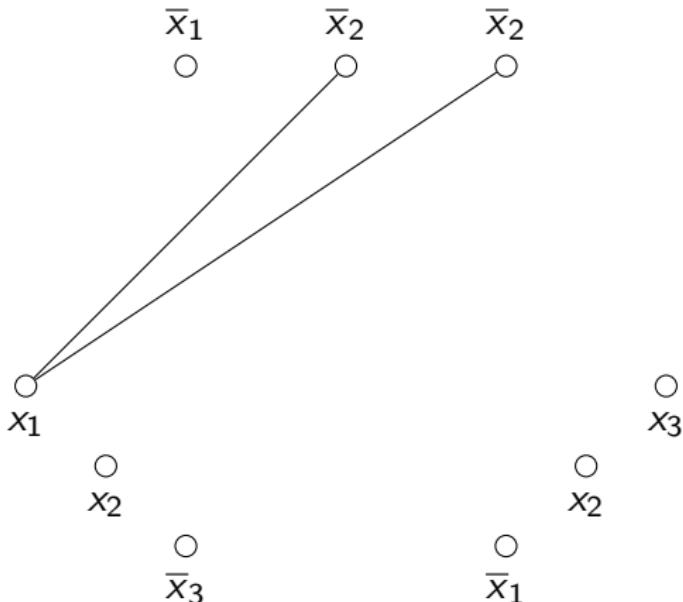
$\text{3SAT} \leq_m^P \text{CLIQUE}$

$$\varphi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$$



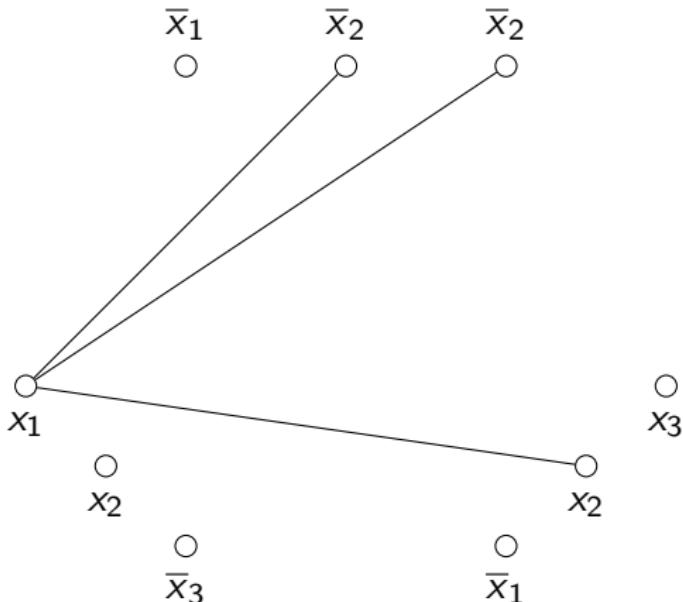
$3\text{SAT} \leq_m^P \text{CLIQUE}$

$$\varphi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$$



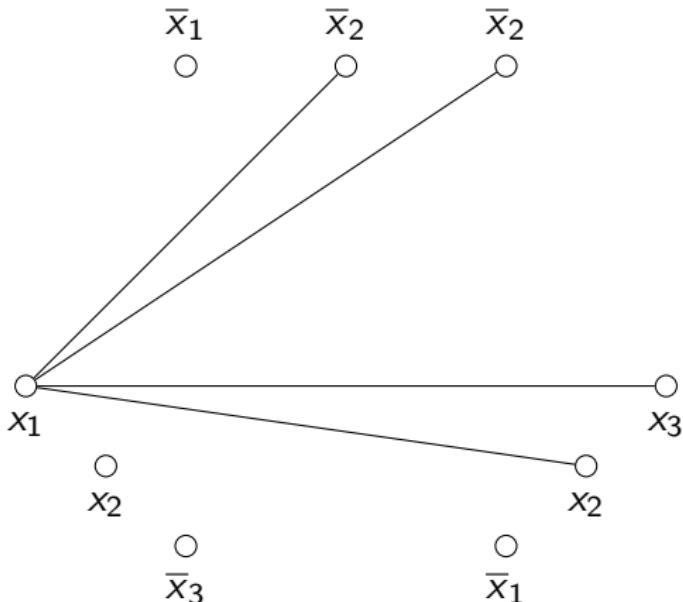
$3\text{SAT} \leq_m^P \text{CLIQUE}$

$$\varphi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$$



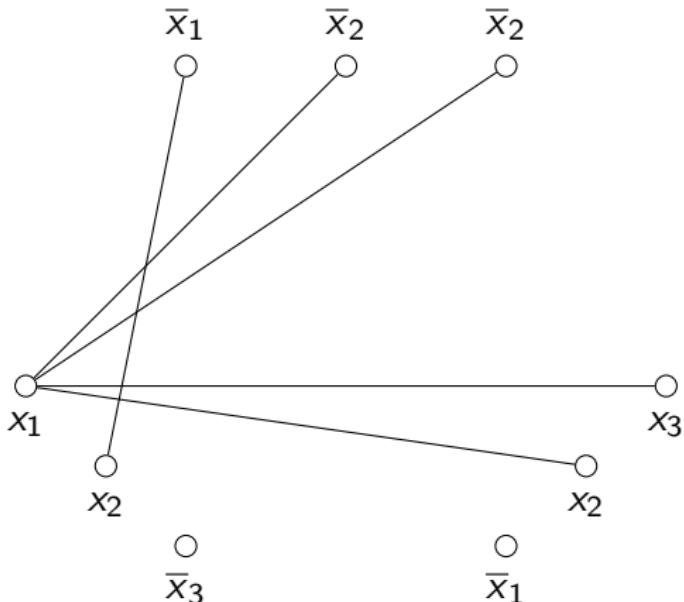
$3\text{SAT} \leq_m^P \text{CLIQUE}$

$$\varphi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$$



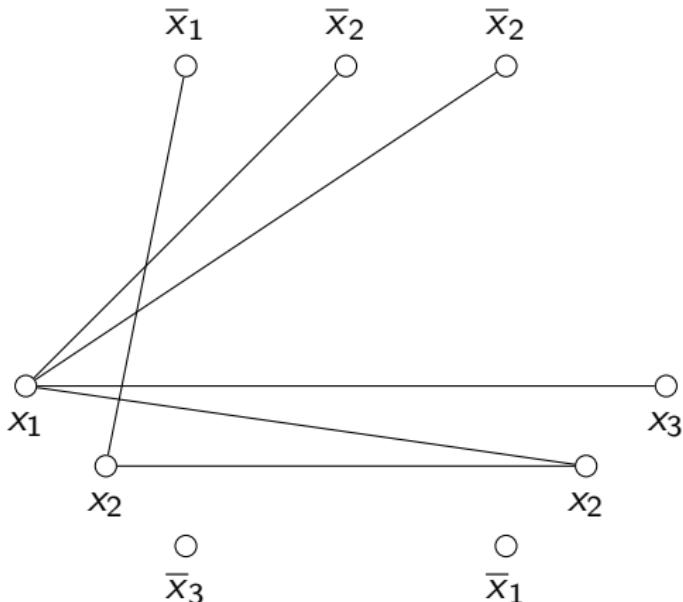
3SAT \leq_m^P CLIQUE

$$\varphi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$$



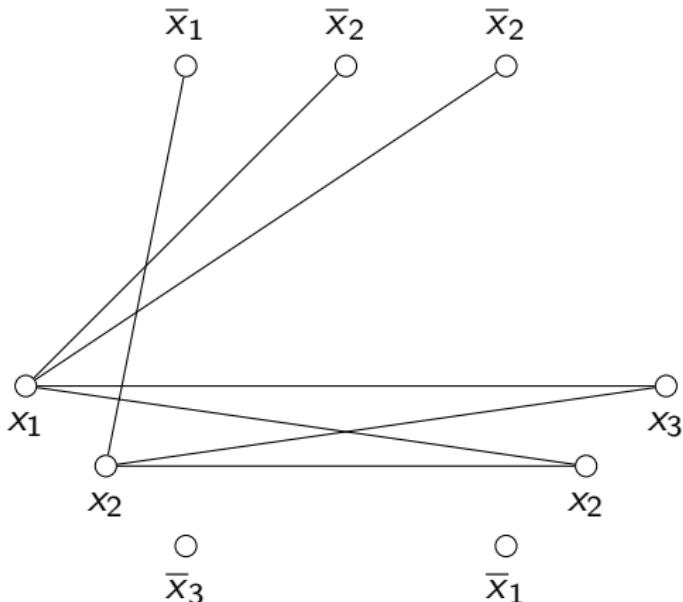
3SAT \leq_m^P CLIQUE

$$\varphi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$$



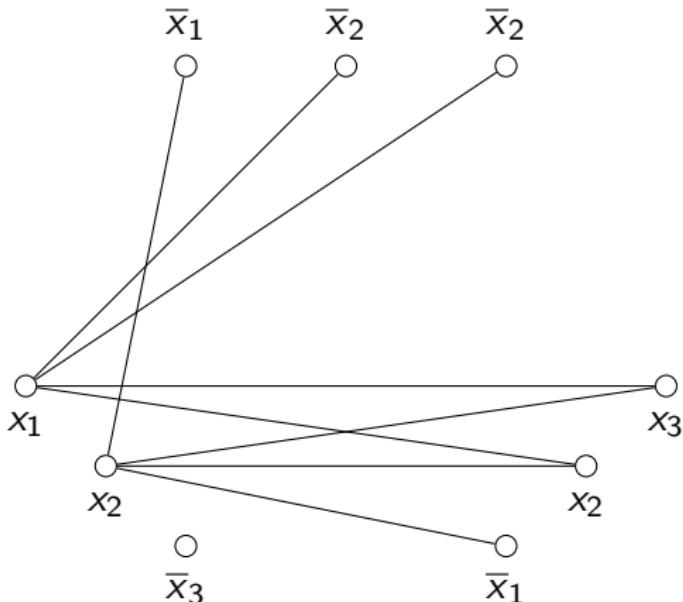
$3\text{SAT} \leq_m^P \text{CLIQUE}$

$$\varphi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$$



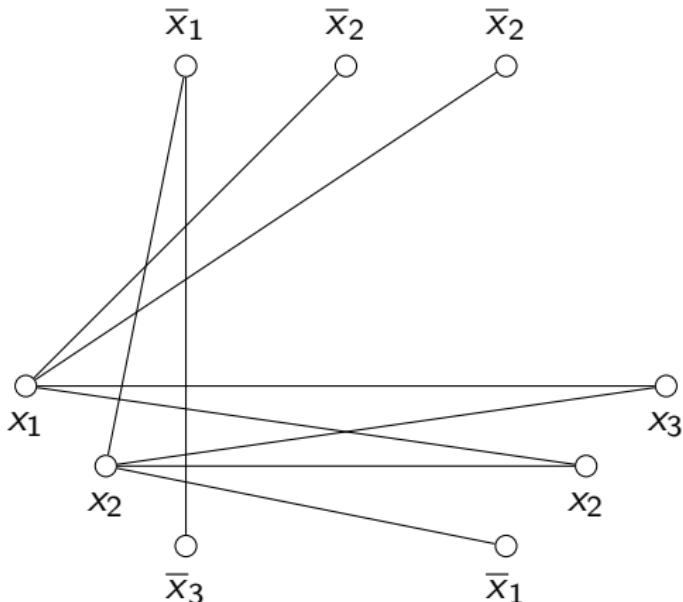
$\text{3SAT} \leq_m^P \text{CLIQUE}$

$$\varphi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$$



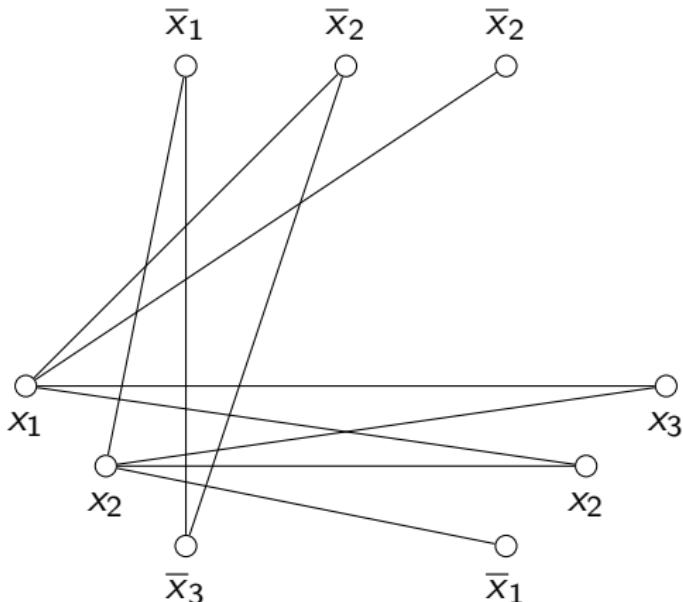
$\text{3SAT} \leq_m^P \text{CLIQUE}$

$$\varphi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$$



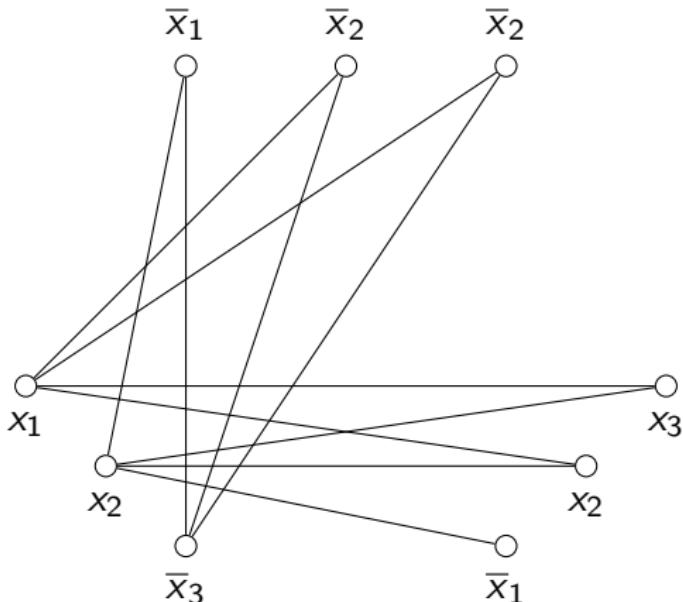
3SAT \leq_m^P CLIQUE

$$\varphi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$$



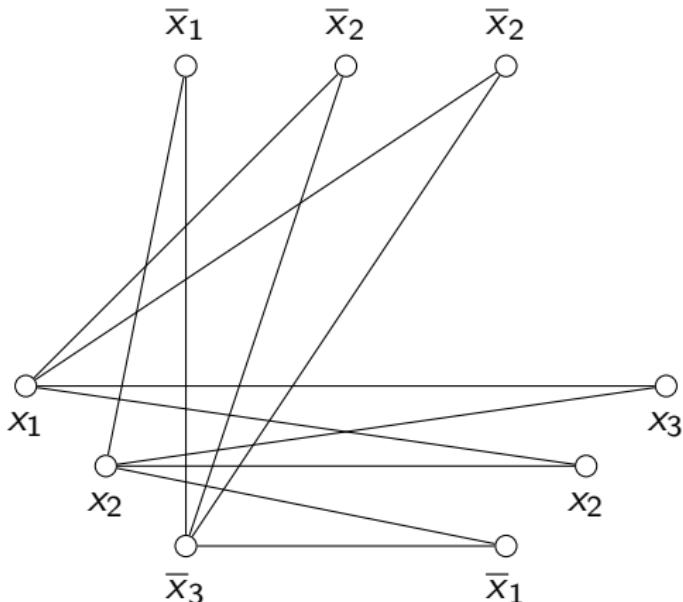
3SAT \leq_m^P CLIQUE

$$\varphi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$$



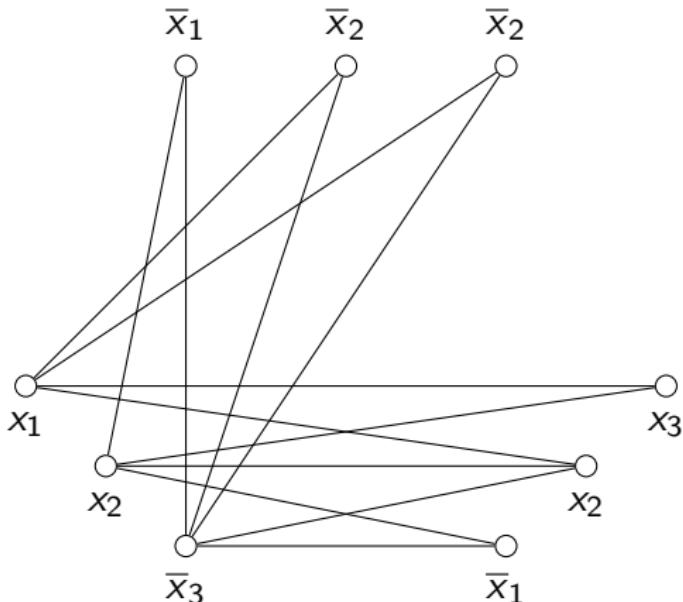
3SAT \leq_m^P CLIQUE

$$\varphi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$$



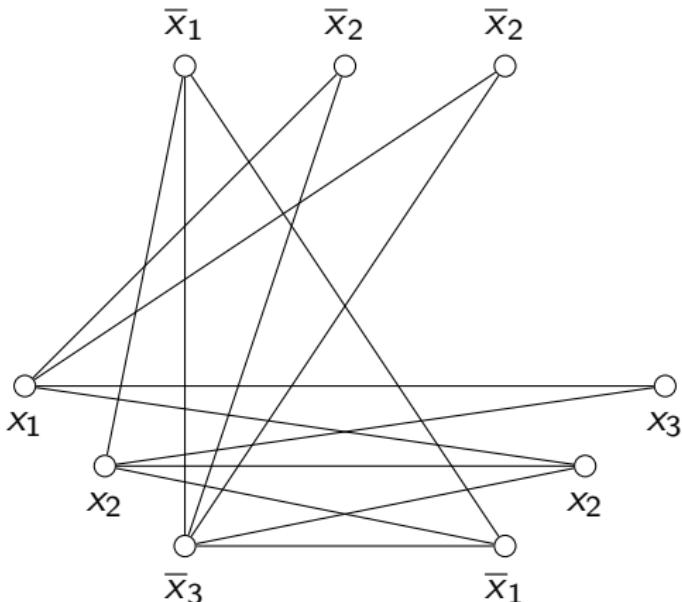
$3\text{SAT} \leq_m^P \text{CLIQUE}$

$$\varphi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$$



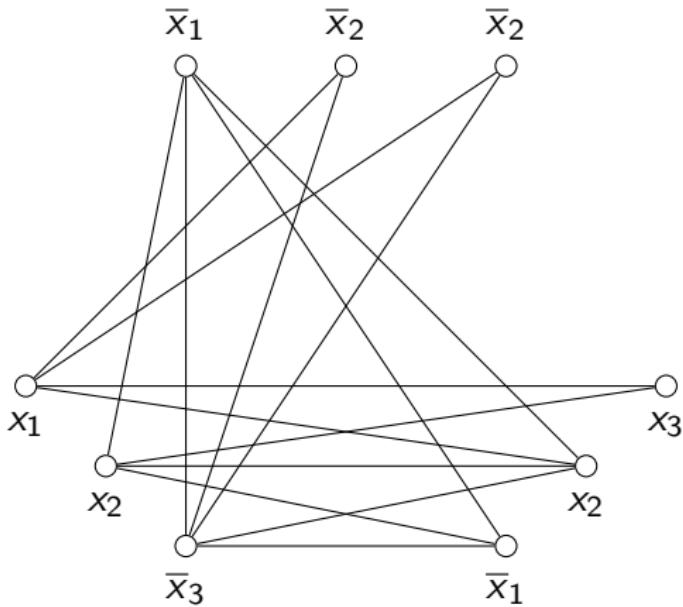
$\text{3SAT} \leq_m^P \text{CLIQUE}$

$$\varphi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$$



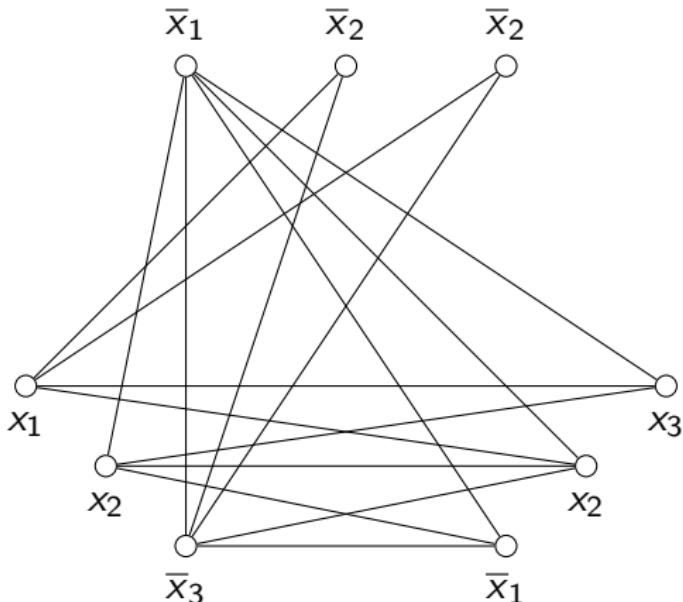
$\text{3SAT} \leq_m^P \text{CLIQUE}$

$$\varphi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$$



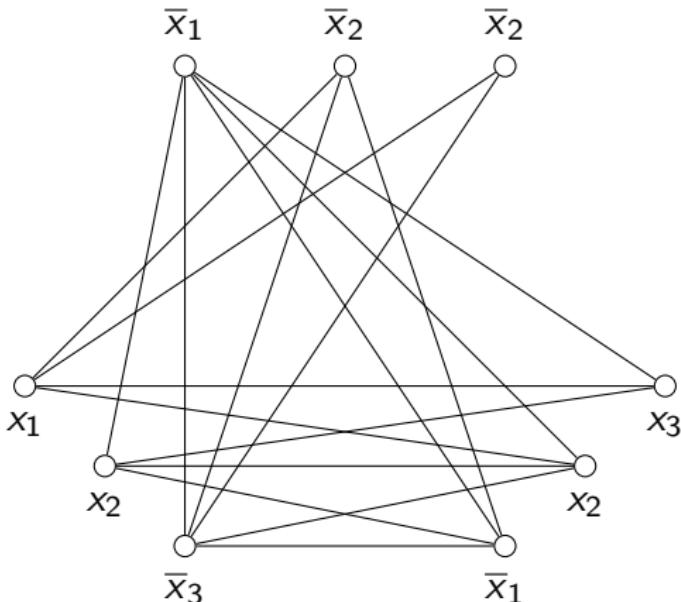
$\text{3SAT} \leq_m^P \text{CLIQUE}$

$$\varphi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$$



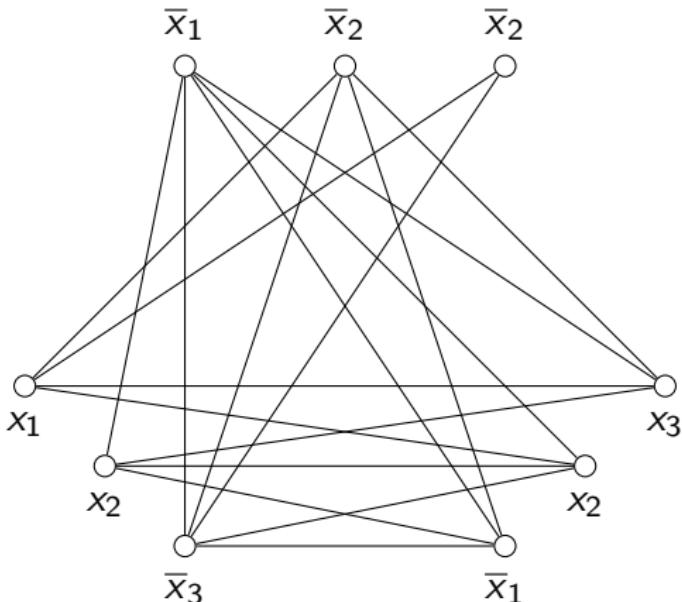
$\text{3SAT} \leq_m^P \text{CLIQUE}$

$$\varphi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$$



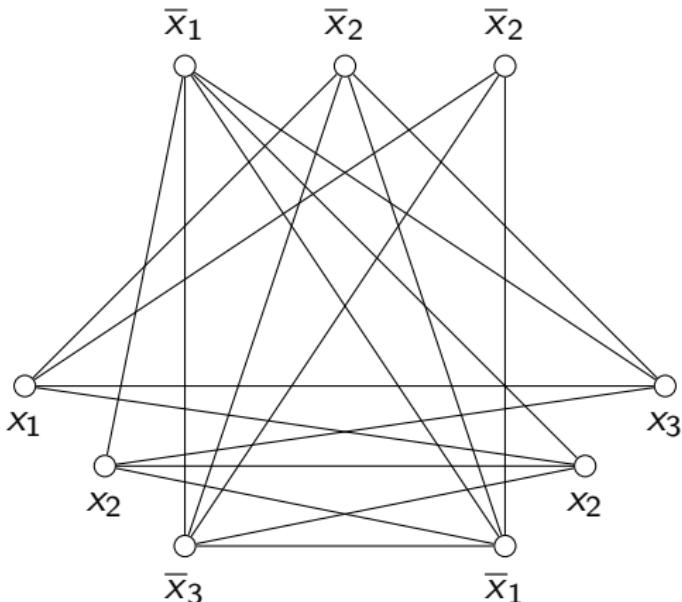
$\text{3SAT} \leq_m^P \text{CLIQUE}$

$$\varphi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$$



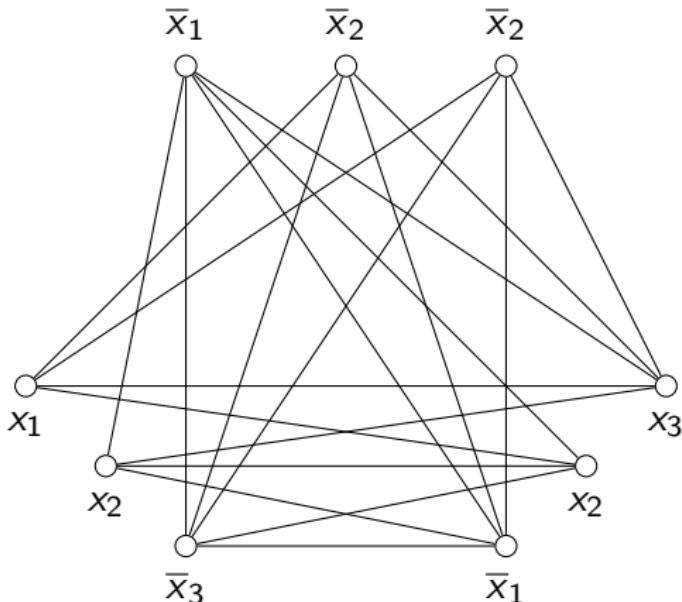
$3\text{SAT} \leq_m^P \text{CLIQUE}$

$$\varphi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$$



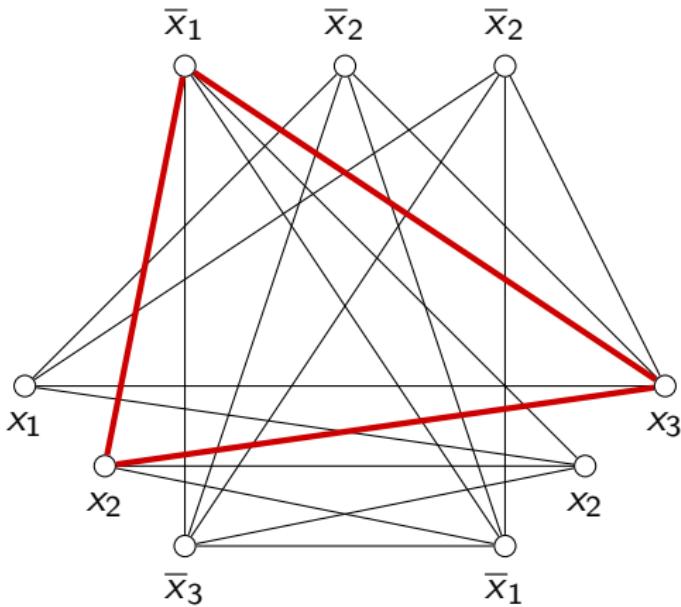
$\text{3SAT} \leq_m^P \text{CLIQUE}$

$$\varphi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$$



$\text{3SAT} \leq_m^P \text{CLIQUE}$

$$\varphi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$$



HAMPATH gehört auch dazu

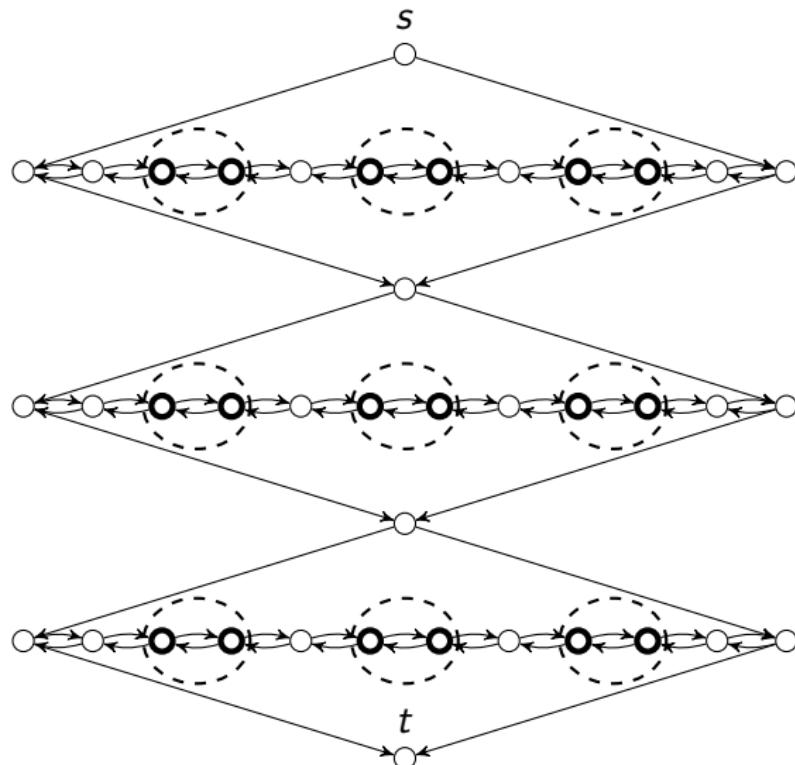
Satz 19

HAMPATH ist NP-vollständig.

$$\text{HAMPATH} = \left\{ \langle G, s, t \rangle \mid \begin{array}{l} G \text{ ist ein gerichteter Graph, der einen} \\ \text{Hamilton'schen Pfad von } s \text{ nach } t \text{ be-} \\ \text{sitzt} \end{array} \right\}$$

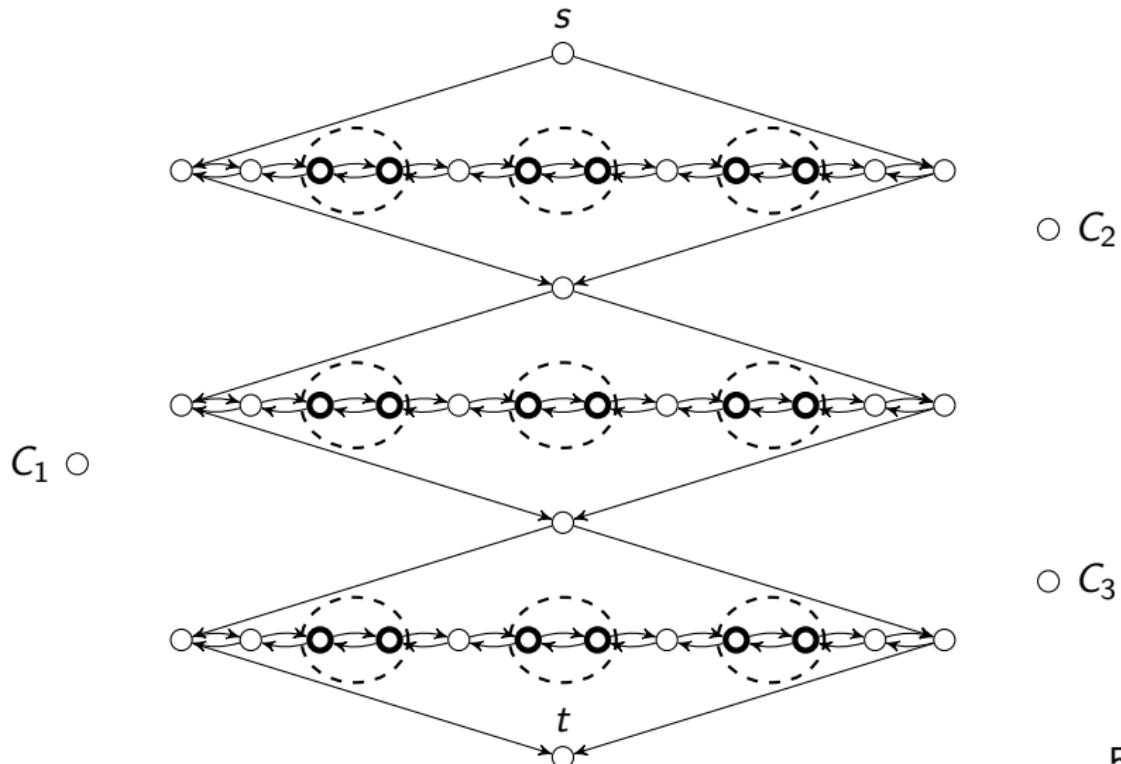
$3\text{SAT} \leq_m^P \text{HAMPATH}$

$$\varphi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$$



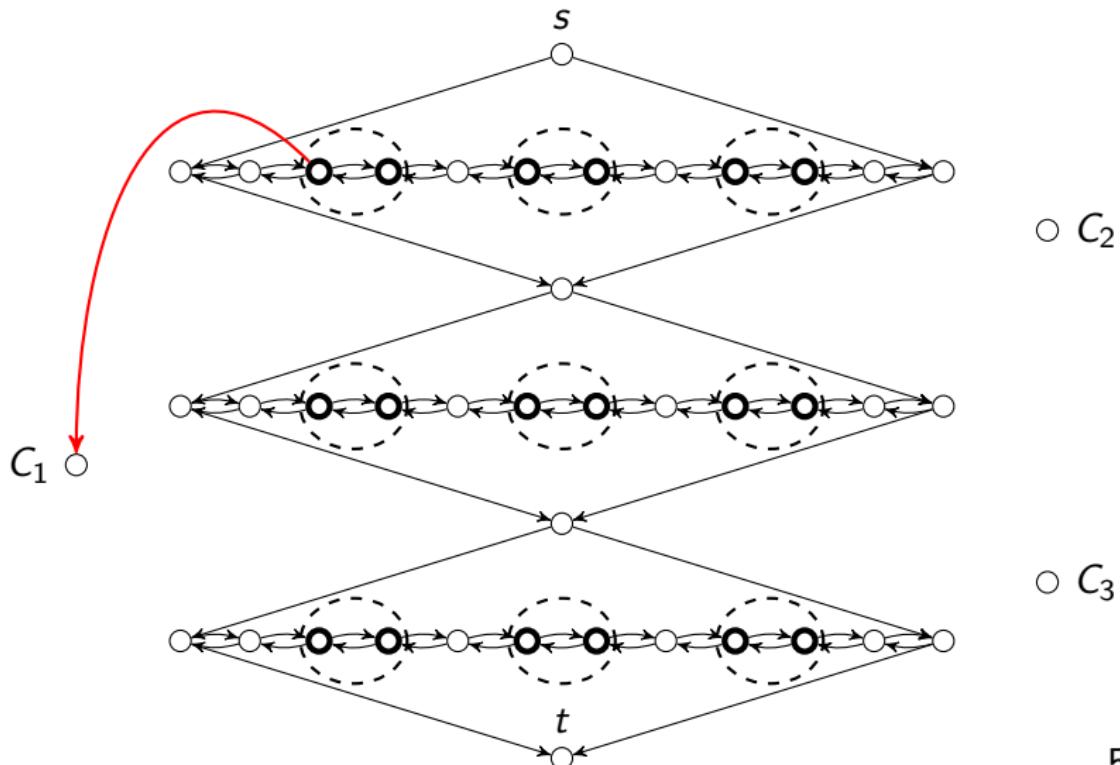
$3\text{SAT} \leq_m^P \text{HAMPATH}$

$$\varphi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$$



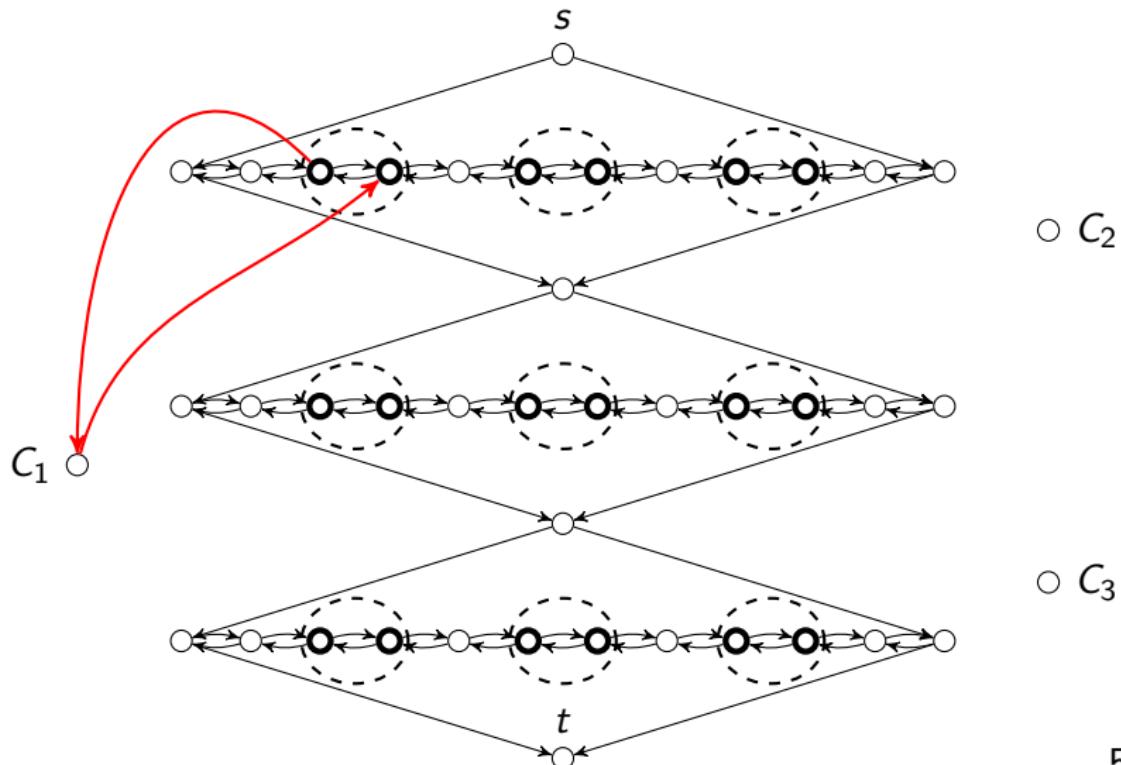
$3\text{SAT} \leq_m^P \text{HAMPATH}$

$$\varphi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$$



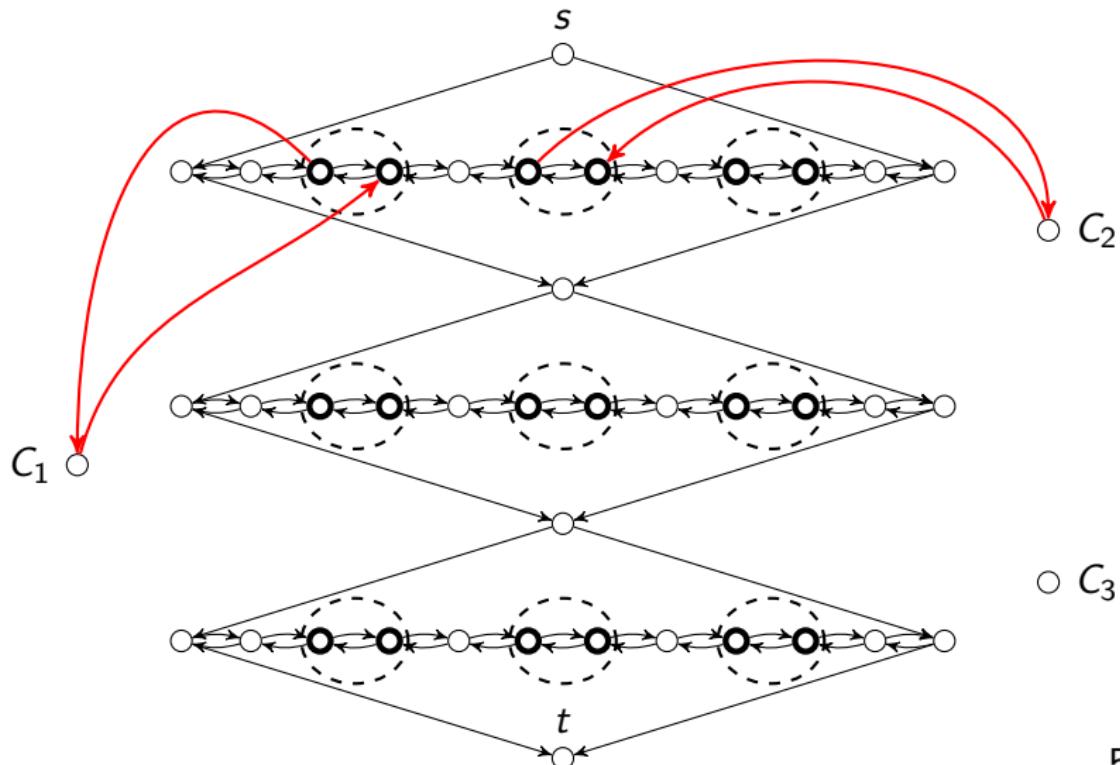
$3\text{SAT} \leq_m^P \text{HAMPATH}$

$$\varphi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$$



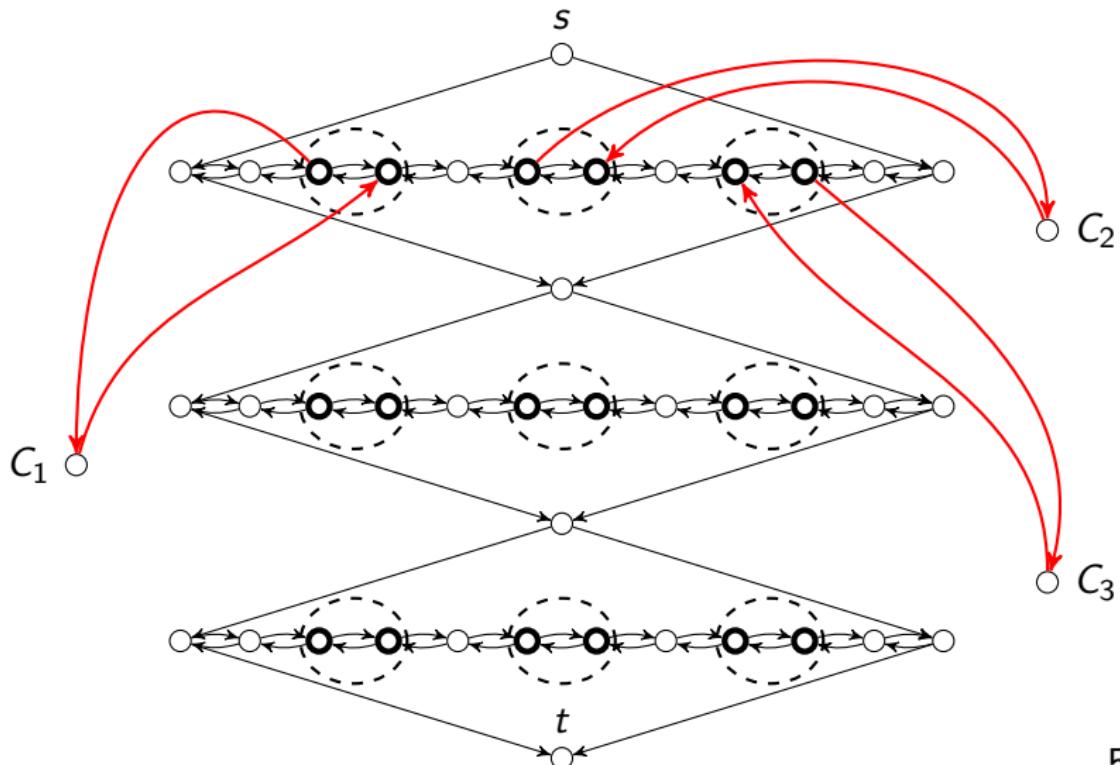
$3\text{SAT} \leq_m^P \text{HAMPATH}$

$$\varphi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$$



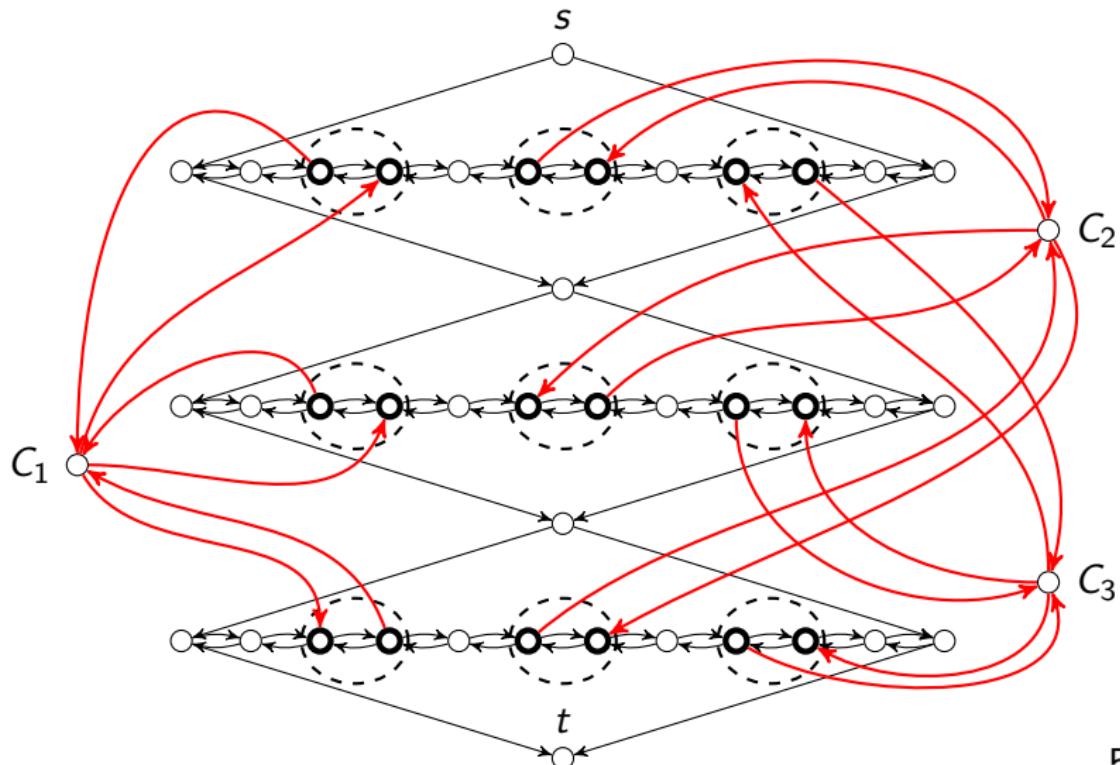
$3\text{SAT} \leq_m^P \text{HAMPATH}$

$$\varphi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$$



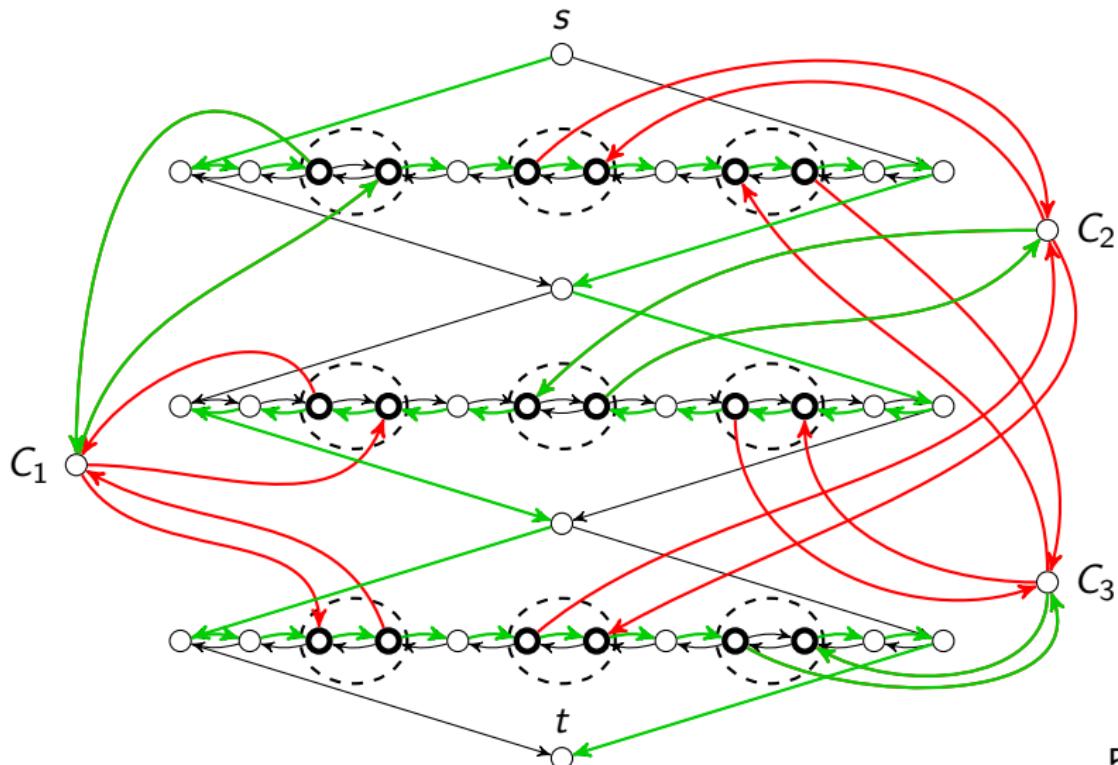
$3\text{SAT} \leq_m^P \text{HAMPATH}$

$$\varphi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$$



$3\text{SAT} \leq_m^P \text{HAMPATH}$

$$\varphi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$$



Hamilton'sche Kreise: Nicht mehr viel zu tun

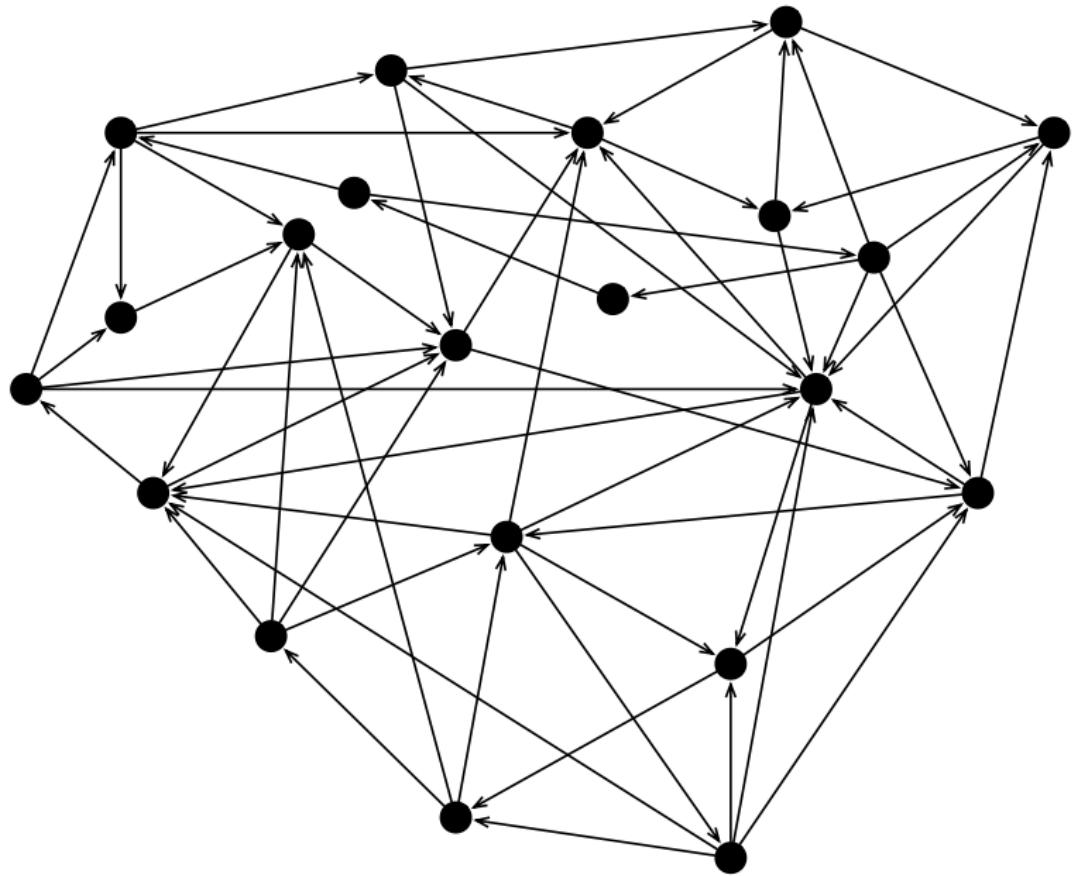
Satz 20

Das Problem

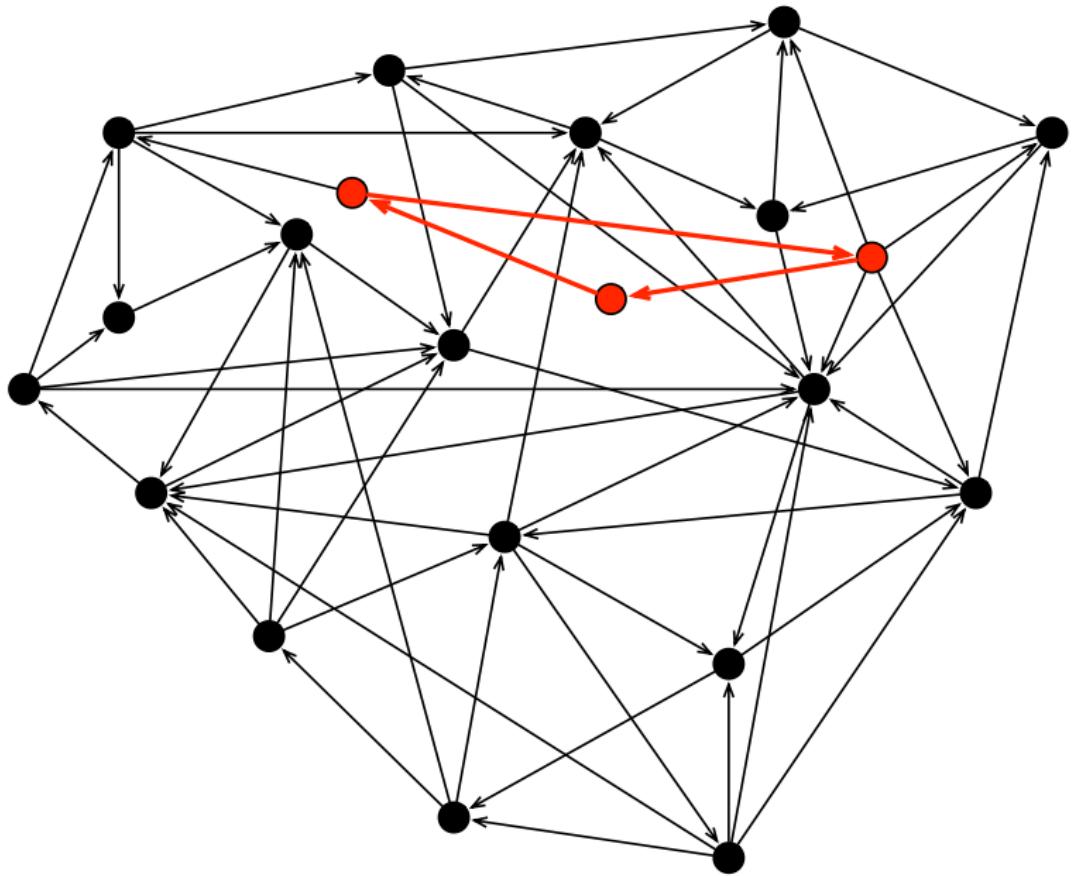
$$\text{HAMCIRC} = \left\{ \langle G \rangle \mid \begin{array}{l} G \text{ ist ein gerichteter Graph der einen} \\ \text{Hamilton'schen Kreis besitzt} \end{array} \right\}$$

ist NP-vollständig.

$\langle G \rangle \in \text{HAMCIRC?}$



$\langle G \rangle \in \text{HAMCIRC?}$



Ein Rezept für NP-Vollständigkeit

Gegeben: Problem A

Aufgabe: Zeige, dass A NP-vollständig ist.

Ein Rezept für NP-Vollständigkeit

Gegeben: Problem A

Aufgabe: Zeige, dass A NP-vollständig ist.

- ① Zeige $A \in \text{NP}$ über

Ein Rezept für NP-Vollständigkeit

Gegeben: Problem A

Aufgabe: Zeige, dass A NP-vollständig ist.

① Zeige $A \in \text{NP}$ über

- Polynomielle Überprüfbarkeit, oder

Ein Rezept für NP-Vollständigkeit

Gegeben: Problem A

Aufgabe: Zeige, dass A NP-vollständig ist.

① Zeige $A \in \text{NP}$ über

- Polynomielle Überprüfbarkeit, oder
- $A \leq_m^P B$ und $B \in \text{NP}$ bekannt, oder

Ein Rezept für NP-Vollständigkeit

Gegeben: Problem A

Aufgabe: Zeige, dass A NP-vollständig ist.

① Zeige $A \in \text{NP}$ über

- Polynomielle Überprüfbarkeit, oder
- $A \leq_m^P B$ und $B \in \text{NP}$ bekannt, oder
- Nichtdeterministische Turing Maschine.

Ein Rezept für NP-Vollständigkeit

Gegeben: Problem A

Aufgabe: Zeige, dass A NP-vollständig ist.

① Zeige $A \in \text{NP}$ über

- Polynomielle Überprüfbarkeit, oder
- $A \leq_m^P B$ und $B \in \text{NP}$ bekannt, oder
- Nichtdeterministische Turing Maschine.

② Zeige A ist NP-schwer über

Ein Rezept für NP-Vollständigkeit

Gegeben: Problem A

Aufgabe: Zeige, dass A NP-vollständig ist.

① Zeige $A \in \text{NP}$ über

- Polynomielle Überprüfbarkeit, oder
- $A \leq_m^P B$ und $B \in \text{NP}$ bekannt, oder
- Nichtdeterministische Turing Maschine.

② Zeige A ist NP-schwer über

- $B \leq_m^P A$ und B ist als NP-schwer bekannt, oder

Ein Rezept für NP-Vollständigkeit

Gegeben: Problem A

Aufgabe: Zeige, dass A NP-vollständig ist.

① Zeige $A \in \text{NP}$ über

- Polynomielle Überprüfbarkeit, oder
- $A \leq_m^P B$ und $B \in \text{NP}$ bekannt, oder
- Nichtdeterministische Turing Maschine.

② Zeige A ist NP-schwer über

- $B \leq_m^P A$ und B ist als NP-schwer bekannt, oder
- $B \leq_m^P A$ für alle $B \in \text{NP}$ als generische Reduktion.

Ein Rezept für NP-Schwere

Gegeben: Problem A

Aufgabe: Zeige, dass A NP-schwer ist.

- ① Finde bekanntes NP-schweres Problem B , „ähnlich“ zu A .

Ein Rezept für NP-Schwere

Gegeben: Problem A

Aufgabe: Zeige, dass A NP-schwer ist.

- ① Finde bekanntes NP-schweres Problem B , „ähnlich“ zu A .
- ② Konstruiere Reduktionsfunktion f , die Instanzen von B auf Instanzen von A umformt.

Ein Rezept für NP-Schwere

Gegeben: Problem A

Aufgabe: Zeige, dass A NP-schwer ist.

- ① Finde bekanntes NP-schweres Problem B , „ähnlich“ zu A .
- ② Konstruiere Reduktionsfunktion f , die Instanzen von B auf Instanzen von A umformt.
 - Zeige, dass f in Polynomialzeit von einer DTM berechnet werden kann.

Ein Rezept für NP-Schwere

Gegeben: Problem A

Aufgabe: Zeige, dass A NP-schwer ist.

- ① Finde bekanntes NP-schweres Problem B , „ähnlich“ zu A .
- ② Konstruiere Reduktionsfunktion f , die Instanzen von B auf Instanzen von A umformt.
 - Zeige, dass f in Polynomialzeit von einer DTM berechnet werden kann.
 - Zeige, dass $x \in B$ genau dann wenn $f(x) \in A$ gilt.

Ein typischer NP-Schwere-Beweis

Wir zeigen, dass A NP-schwer ist über die Reduktion $B \leq_m^P A$.

Ein typischer NP-Schwere-Beweis

Wir zeigen, dass A NP-schwer ist über die Reduktion $B \leq_m^P A$.

Da B bereits ein NP-schweres Problem ist, folgt aus der Transitivität von \leq_m^P , dass für alle Probleme $C \in \text{NP}$ auch $C \leq_m^P A$ gilt.

Ein typischer NP-Schwere-Beweis

Wir zeigen, dass A NP-schwer ist über die Reduktion $B \leq_m^P A$.

Da B bereits ein NP-schweres Problem ist, folgt aus der Transitivität von \leq_m^P , dass für alle Probleme $C \in \text{NP}$ auch $C \leq_m^P A$ gilt.

Nun geben wir konkret die Reduktionsfunktion f an, welche Instanzen aus B auf Instanzen aus A umformt:

⋮

Ein typischer NP-Schwere-Beweis

Wir zeigen, dass A NP-schwer ist über die Reduktion $B \leq_m^P A$.

Da B bereits ein NP-schweres Problem ist, folgt aus der Transitivität von \leq_m^P , dass für alle Probleme $C \in \text{NP}$ auch $C \leq_m^P A$ gilt.

Nun geben wir konkret die Reduktionsfunktion f an, welche Instanzen aus B auf Instanzen aus A umformt:

⋮

Nun zeigen wir, dass $x \in B \Rightarrow f(x) \in A$ gilt: ...

Ein typischer NP-Schwere-Beweis

Wir zeigen, dass A NP-schwer ist über die Reduktion $B \leq_m^P A$.

Da B bereits ein NP-schweres Problem ist, folgt aus der Transitivität von \leq_m^P , dass für alle Probleme $C \in \text{NP}$ auch $C \leq_m^P A$ gilt.

Nun geben wir konkret die Reduktionsfunktion f an, welche Instanzen aus B auf Instanzen aus A umformt:

⋮

Nun zeigen wir, dass $x \in B \Rightarrow f(x) \in A$ gilt: ...

Nun zeigen wir, dass $x \notin B \Rightarrow f(x) \notin A$ gilt: ...

Ein typischer NP-Schwere-Beweis

Wir zeigen, dass A NP-schwer ist über die Reduktion $B \leq_m^P A$.

Da B bereits ein NP-schweres Problem ist, folgt aus der Transitivität von \leq_m^P , dass für alle Probleme $C \in \text{NP}$ auch $C \leq_m^P A$ gilt.

Nun geben wir konkret die Reduktionsfunktion f an, welche Instanzen aus B auf Instanzen aus A umformt:

⋮

Nun zeigen wir, dass $x \in B \Rightarrow f(x) \in A$ gilt: ⋯

Nun zeigen wir, dass $x \notin B \Rightarrow f(x) \notin A$ gilt: ⋯

Insgesamt folgt also: $x \in A \Leftrightarrow f(x) \in B$.

Ein typischer NP-Schwere-Beweis

Wir zeigen, dass A NP-schwer ist über die Reduktion $B \leq_m^P A$.

Da B bereits ein NP-schweres Problem ist, folgt aus der Transitivität von \leq_m^P , dass für alle Probleme $C \in \text{NP}$ auch $C \leq_m^P A$ gilt.

Nun geben wir konkret die Reduktionsfunktion f an, welche Instanzen aus B auf Instanzen aus A umformt:

⋮

Nun zeigen wir, dass $x \in B \Rightarrow f(x) \in A$ gilt: ⋯

Nun zeigen wir, dass $x \notin B \Rightarrow f(x) \notin A$ gilt: ⋯

Insgesamt folgt also: $x \in A \Leftrightarrow f(x) \in B$.

Die Reduktion f ist in Polynomialzeit berechenbar, weil ⋯.

Ein typischer NP-Schwere-Beweis

Wir zeigen, dass A NP-schwer ist über die Reduktion $B \leq_m^P A$.

Da B bereits ein NP-schweres Problem ist, folgt aus der Transitivität von \leq_m^P , dass für alle Probleme $C \in \text{NP}$ auch $C \leq_m^P A$ gilt.

Nun geben wir konkret die Reduktionsfunktion f an, welche Instanzen aus B auf Instanzen aus A umformt:

⋮

Nun zeigen wir, dass $x \in B \Rightarrow f(x) \in A$ gilt: ⋯

Nun zeigen wir, dass $x \notin B \Rightarrow f(x) \notin A$ gilt: ⋯

Insgesamt folgt also: $x \in A \Leftrightarrow f(x) \in B$.

Die Reduktion f ist in Polynomialzeit berechenbar, weil ⋯.

Insgesamt ist A damit NP-schwer.

□

Satz 21

Das Problem

$$\text{TSP} = \left\{ \langle (d_{i,j})_{1 \leq i,j \leq n}, B \rangle \middle| \begin{array}{l} n, d_{i,j}, B \in \mathbb{N} \text{ und es gibt ein } \pi \in S_n \\ \text{mit } \sum_{i=1}^{n-1} d_{\pi(i), \pi(i+1)} + d_{\pi(n), \pi(1)} \leq B \end{array} \right\}$$

ist NP-vollständig.

Das Bezahl- oder Teilsummenproblem ist auch sehr schwierig

$$\text{SUBSET-SUM} = \left\{ \langle x_1, \dots, x_n, t \rangle \mid \begin{array}{l} x_1, \dots, x_n, t \in \mathbb{N} \text{ und es} \\ \text{existiert eine Teilmenge} \\ I \subseteq \{1, \dots, n\} \text{ mit } \sum_{i \in I} x_i = t \end{array} \right\}$$

Satz 22

SUBSET-SUM ist NP-vollständig

Wie schwierig ist SUBSET-SUM eigentlich?

$\langle 4096,$

$36872,$

$512,$

$32768,$

$33281,$

$584,$

$64,$

$4161,$

$32768,$

$4096,$

$512,$

$64,$

$112329 \rangle \in \text{SUBSET-SUM}?$

$3\text{SAT} \leq_m^P \text{SUBSET-SUM}$

$$(x_1 \vee x_2 \vee x_2) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2})$$

$\text{3SAT} \leq_m^P \text{SUBSET-SUM}$

$$(x_1 \vee x_2 \vee x_2) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2})$$

\rightsquigarrow

$\text{3SAT} \leq_m^P \text{SUBSET-SUM}$

$$(x_1 \vee x_2 \vee x_2) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2})$$

\rightsquigarrow

$$a_1 = (1100\ 10)_8 = (36872)_{10}$$

$\text{3SAT} \leq_m^P \text{SUBSET-SUM}$

$$(x_1 \vee x_2 \vee x_2) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2})$$

\rightsquigarrow

$$a_1 = (1100\ 10)_8 = (36872)_{10}$$

$$a'_1 = (0011\ 10)_8 = (584)_{10}$$

$\text{3SAT} \leq_m^P \text{SUBSET-SUM}$

$$(x_1 \vee x_2 \vee x_2) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2})$$

\rightsquigarrow

$$a_1 = (1100\ 10)_8 = (36872)_{10}$$

$$a'_1 = (0011\ 10)_8 = (584)_{10}$$

$$a_2 = (1010\ 01)_8 = (33281)_{10}$$

$3\text{SAT} \leq_m^P \text{SUBSET-SUM}$

$$(x_1 \vee x_2 \vee x_2) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2})$$

\rightsquigarrow

$$a_1 = (1100\ 10)_8 = (36872)_{10}$$

$$a'_1 = (0011\ 10)_8 = (584)_{10}$$

$$a_2 = (1010\ 01)_8 = (33281)_{10}$$

$$a'_2 = (0101\ 01)_8 = (4161)_{10}$$

$3\text{SAT} \leq_m^P \text{SUBSET-SUM}$

$$(x_1 \vee x_2 \vee x_2) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2})$$

\leadsto

$$a_1 = (1100\ 10)_8 = (36872)_{10}$$

$$a'_1 = (0011\ 10)_8 = (584)_{10}$$

$$b_1 = b'_1 = (1000\ 00)_8 = (32768)_{10}$$

$$b_3 = b'_3 = (0010\ 00)_8 = (512)_{10}$$

$$a_2 = (1010\ 01)_8 = (33281)_{10}$$

$$a'_2 = (0101\ 01)_8 = (4161)_{10}$$

$$b_2 = b'_2 = (0100\ 00)_8 = (4096)_{10}$$

$$b_4 = b'_4 = (0001\ 00)_8 = (64)_{10}$$

$\text{3SAT} \leq_m^P \text{SUBSET-SUM}$

$$(x_1 \vee x_2 \vee x_2) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2})$$

\rightsquigarrow

$$a_1 = (1100\ 10)_8 = (36872)_{10}$$

$$a'_1 = (0011\ 10)_8 = (584)_{10}$$

$$b_1 = b'_1 = (1000\ 00)_8 = (32768)_{10}$$

$$b_3 = b'_3 = (0010\ 00)_8 = (512)_{10}$$

$$a_2 = (1010\ 01)_8 = (33281)_{10}$$

$$a'_2 = (0101\ 01)_8 = (4161)_{10}$$

$$b_2 = b'_2 = (0100\ 00)_8 = (4096)_{10}$$

$$b_4 = b'_4 = (0001\ 00)_8 = (64)_{10}$$

$$t = (3333\ 11)_8 = (112329)_{10}$$

Das Partitionierungsproblem

Satz 23

Das Problem

$$\text{PART} = \left\{ \langle a_1, \dots, a_n \rangle \mid \begin{array}{l} n, a_1, \dots, a_n \in \mathbb{N} \text{ und} \\ \exists I \subseteq \{1, \dots, n\}, \text{ sodass} \\ \sum_{i \in I} a_i = \sum_{i \notin I} a_i \end{array} \right\}$$

ist NP-vollständig.

Ich packe meinen Rucksack...

Satz 24

Das Problem

$$\text{KNAPSACK} = \left\{ \begin{array}{l} \langle a_1, \dots, a_n, \\ v_1, \dots, v_n, s, m \rangle \end{array} \middle| \begin{array}{l} n, a_1, \dots, a_n, v_1, \dots, v_n, s, m \in \mathbb{N} \\ \text{und } \exists I \subseteq \{1, \dots, n\}, \text{ sodass} \\ \sum_{i \in I} a_i \leq s \text{ und } \sum_{i \in I} v_i \geq m \end{array} \right\}$$

ist NP-vollständig.

Umzugskartons packen ist auch NP-schwer

Satz 25

Das Problem

$$\text{BIN-PACKING} = \left\{ \langle a_1, \dots, a_n, K \rangle \mid \begin{array}{l} n, K \in \mathbb{N}, a_1, \dots, a_n \in \mathbb{Q} \cap [0, 1] \\ \text{und es gibt Partitionierung } I_1 \uplus \\ I_2 \uplus \dots \uplus I_K = \{1, \dots, n\} \text{ mit} \\ \sum_{i \in I_j} a_i \leq 1 \text{ f\"ur } 1 \leq j \leq K \end{array} \right\}$$

ist NP-vollstndig.

Mahaney's Theorem

Satz 26

Sei $c \in \mathbb{N}$ und $A \in \text{SPARSE}$. Wenn A NP-vollständig ist, dann folgt $P = NP$.

Mahaney's Theorem

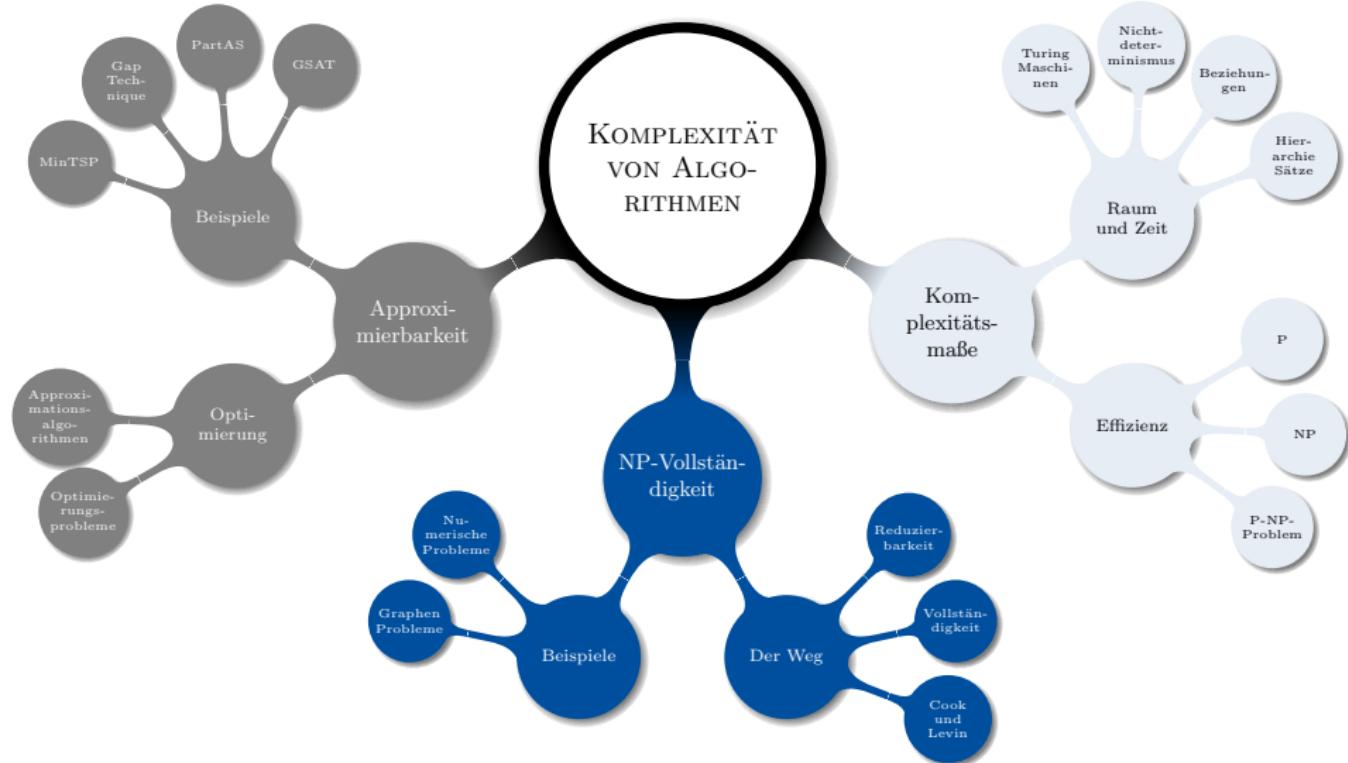
Satz 26

Sei $c \in \mathbb{N}$ und $A \in \text{SPARSE}$. Wenn A NP-vollständig ist, dann folgt $P = NP$.

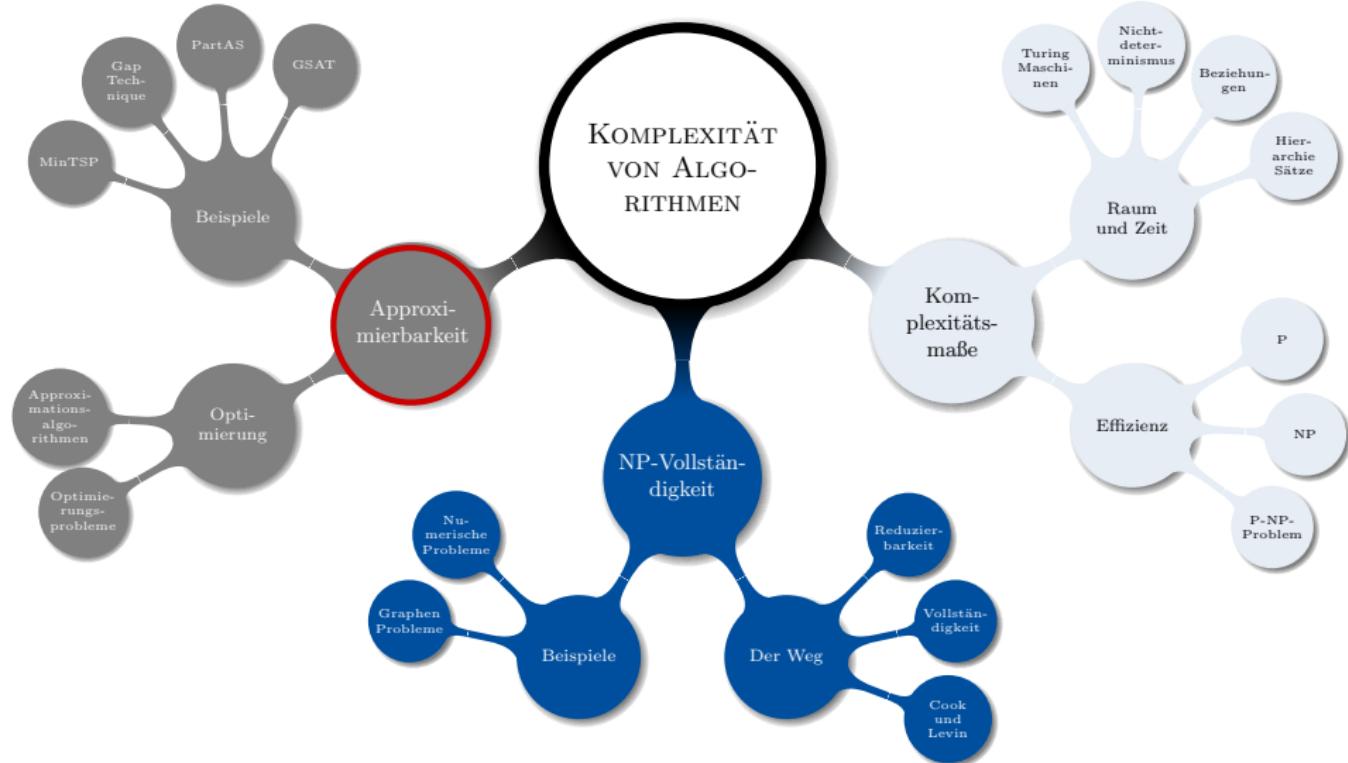
Satz 27 (Cai, Sivakumar)

Wenn es eine Sprache $A \in \text{SPARSE}$ gibt, die P-vollständig ist, dann folgt $P = NL$.

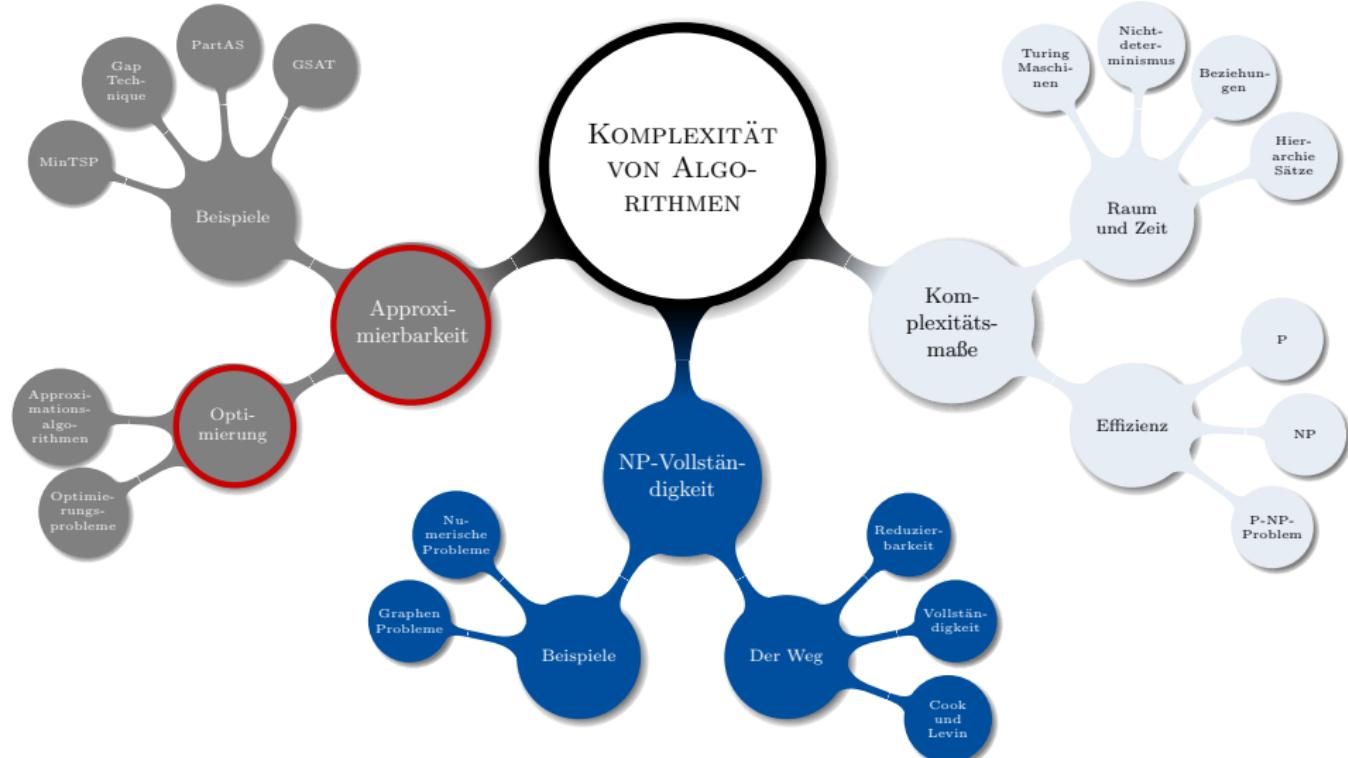
Wie geht es weiter?



Wie geht es weiter?



Wie geht es weiter?



Optimierungsprobleme

Definition (Optimierungsproblem)

Ein Optimierungsproblem \mathcal{P} ist gegeben durch ein Quadrupel

$$\mathcal{P} = (I, s, m, t),$$

wobei:

- I ist die Menge der Instanzen von \mathcal{P} .
- s ist eine Funktion, die ein $x \in I$ abbildet auf die Menge der möglichen Lösungen von x . Sei $\mathcal{S} = \bigcup_{x \in I} s(x)$ die Menge aller möglichen Lösungen.
- $m: I \times \mathcal{S} \rightarrow \mathbb{N}$ ist die Maßfunktion. Dabei ist $m(x, y)$ der Wert der Lösung y zur Instanz $x \in I$, wobei $y \in s(x)$. Es soll $m(x, y)$ immer definiert sein, falls $y \in s(x)$.
- $t \in \{\min, \max\}$ ist das Ziel von \mathcal{P} .

Unsere Optimierungsprobleme sind schön

Sei $\mathcal{P} = (I, s, m, t)$ ein Optimierungsproblem. Dann soll gelten:

- $I \in \mathbb{P}$.
- Es gibt ein Polynom p , sodass für alle $x \in I$ und alle $y \in s(x)$ gilt: $|y| \leq p(|x|)$.
- m ist Polynomialzeit berechenbar.

Optimalität

Definition

Sei \mathcal{P} ein Optimierungsproblem. $y^* \in s_{\mathcal{P}}(x)$ heißt **optimale Lösung** zu x , falls

$$m(x, y^*) = t \{ m(x, y) \mid y \in s(x) \}.$$

Weiterhin definieren wir

$$\begin{aligned}s^*(x) &= \{y^* \in s(x) \mid y^* \text{ ist optimale Lösung zu } x\} \text{ und} \\ m^*(x) &= m(x, y^*) \text{ für ein } y^* \in s^*(x).\end{aligned}$$

NPO – das Optimierungs-Pendant zu NP

Definition

Ein Optimierungsproblem \mathcal{P} gehört zur Klasse NPO, falls gilt:

$$\{\langle x, y \rangle \mid y \in s(x)\} \in P.$$

PO – das Optimierungs-Pendant zu P

Definition

Ein Optimierungsproblem \mathcal{P} gehört zur Klasse PO, falls $\mathcal{P} \in \text{NPO}$ und falls es einen deterministischen Polynomialzeit–Algorithmus gibt, der bei Eingabe von $x \in I_{\mathcal{P}}$ eine (beliebige) Lösung $y^* \in s_{\mathcal{P}}^*(x)$ ausgibt.

Bemerkung

PO ist also die Klasse der effizient lösbarer Optimierungsprobleme.

Eine Brücke zu Entscheidungsproblemen

Definition

Sei \mathcal{P} ein Optimierungsproblem. Das **Entscheidungsproblem** zu \mathcal{P} ist

$$\mathcal{P}_D = \{ \langle x, K \rangle \mid x \in I_{\mathcal{P}}, K \in \mathbb{N} \text{ und } m^*(x) \leq K, \text{ falls } t_{\mathcal{P}} = \min \\ m^*(x) \geq K, \text{ falls } t_{\mathcal{P}} = \max \}.$$

NP-Optimierungsprobleme

Lemma 1

Sei \mathcal{P} ein Optimierungsproblem. Dann gilt: $\mathcal{P} \in \text{NPO}$ genau dann, wenn $\mathcal{P}_D \in \text{NP}$.

P-Optimierungsprobleme

Lemma 2

Sei \mathcal{P} ein Optimierungsproblem. Dann gilt: $\mathcal{P} \in \text{PO}$ genau dann, wenn $\mathcal{P}_D \in \text{P}$.

Analoger Beweis wie zuvor.

Zusammenhang zur Entscheidungswelt



Satz 28 (Paz, Moran 1977)

$P = NP$ gdw. $PO = NPO$.

Azaria Paz*



Shlomo Moran§

NP-Schwere in der Optimierungswelt

Definition

Ein Optimierungsproblem \mathcal{P} ist **NP-schwer**, falls jedes Problem aus NP von einem Polynomialzeit-Algorithmus gelöst werden kann, der elementare Anweisungen der Form

„Sei $y \in s_{\mathcal{P}}^*(x)$ “ oder „ $v := m_{\mathcal{P}}^*(x)$ “

verwenden darf und deren Zeitbedarf nur mit einem Schritt gezählt wird. Formal bedeutet diese Definition, dass jedes Problem aus NP **Turing-reduzierbar** auf \mathcal{P} ist.

NP-Schwere in der Optimierungswelt

Definition

Ein Optimierungsproblem \mathcal{P} ist **NP-schwer**, falls jedes Problem aus NP von einem Polynomialzeit-Algorithmus gelöst werden kann, der elementare Anweisungen der Form

„Sei $y \in s_{\mathcal{P}}^*(x)$ “ oder „ $v := m_{\mathcal{P}}^*(x)$ “

verwenden darf und deren Zeitbedarf nur mit einem Schritt gezählt wird. Formal bedeutet diese Definition, dass jedes Problem aus NP **Turing-reduzierbar** auf \mathcal{P} ist.

Beobachtung

Ist \mathcal{P}_D NP-vollständig, so ist \mathcal{P} auch NP-schwer. Also gilt demnach: Wenn $P \neq NP$ gilt, dann folgt:

\mathcal{P}_D ist NP-vollständig $\Rightarrow \mathcal{P}$ ist NP-schwer $\Rightarrow \mathcal{P} \notin PO$.

Approximationsalgorithmen

Definition

Sei $\mathcal{P} = (I, s, m, t)$ ein Optimierungsproblem. Ein Polynomialzeit-Algorithmus A heißt **Approximationsalgorithmus** für \mathcal{P} , falls A bei Eingabe $x \in I_{\mathcal{P}}$ eine mögliche Lösung $y \in s_{\mathcal{P}}(x)$ berechnet, in Zeichen also $A(x) \in s_{\mathcal{P}}(x)$ gilt.

Schreibweise: $m_A(x) = m_{\mathcal{P}}(x, A(x))$.

Performanzrate – Die Güte einer Lösung

Definition

Sei $\mathcal{P} = (I, s, m, t)$ ein Optimierungsproblem, $x \in I_{\mathcal{P}}$ und $y \in s_{\mathcal{P}}(x)$. Die **Performanz(-rate)** von y bezüglich x ist definiert als

$$R_{\mathcal{P}}(x, y) = \left\{ \begin{array}{ll} \frac{m(x, y)}{m^*(x)} & , \text{ falls } t = \min \\ \frac{m^*(x)}{m(x, y)} & , \text{ falls } t = \max \end{array} \right\} = \max \left\{ \frac{m(x, y)}{m^*(x)}, \frac{m^*(x)}{m(x, y)} \right\}.$$

Die Güte eines Approximationsalgorithmus

Definition

Sei $\mathcal{P} = (I, s, m, t)$ ein Optimierungsproblem und A ein Approximationsalgorithmus für \mathcal{P} . Die **Performanz(-rate)** von A bezüglich \mathcal{P} ist dann

$$R_A(x) = R_{\mathcal{P}}(x, A(x)).$$

Die Güte eines Approximationsalgorithmus

Definition

Sei $\mathcal{P} = (I, s, m, t)$ ein Optimierungsproblem und A ein Approximationsalgorithmus für \mathcal{P} . Die **Performanz(-rate)** von A bezüglich \mathcal{P} ist dann

$$R_A(x) = R_{\mathcal{P}}(x, A(x)).$$

Sei nun $r: \mathbb{N} \rightarrow \mathbb{Q}$ eine Funktion. Wir sagen A ist ein **r -Approximationsalgorithmus für \mathcal{P}** , falls

$$\max_{|x|=n} R_A(x) \leq r(n).$$

Die Güte eines Approximationsalgorithmus

Definition

Sei $\mathcal{P} = (I, s, m, t)$ ein Optimierungsproblem und A ein Approximationsalgorithmus für \mathcal{P} . Die **Performanz(-rate)** von A bezüglich \mathcal{P} ist dann

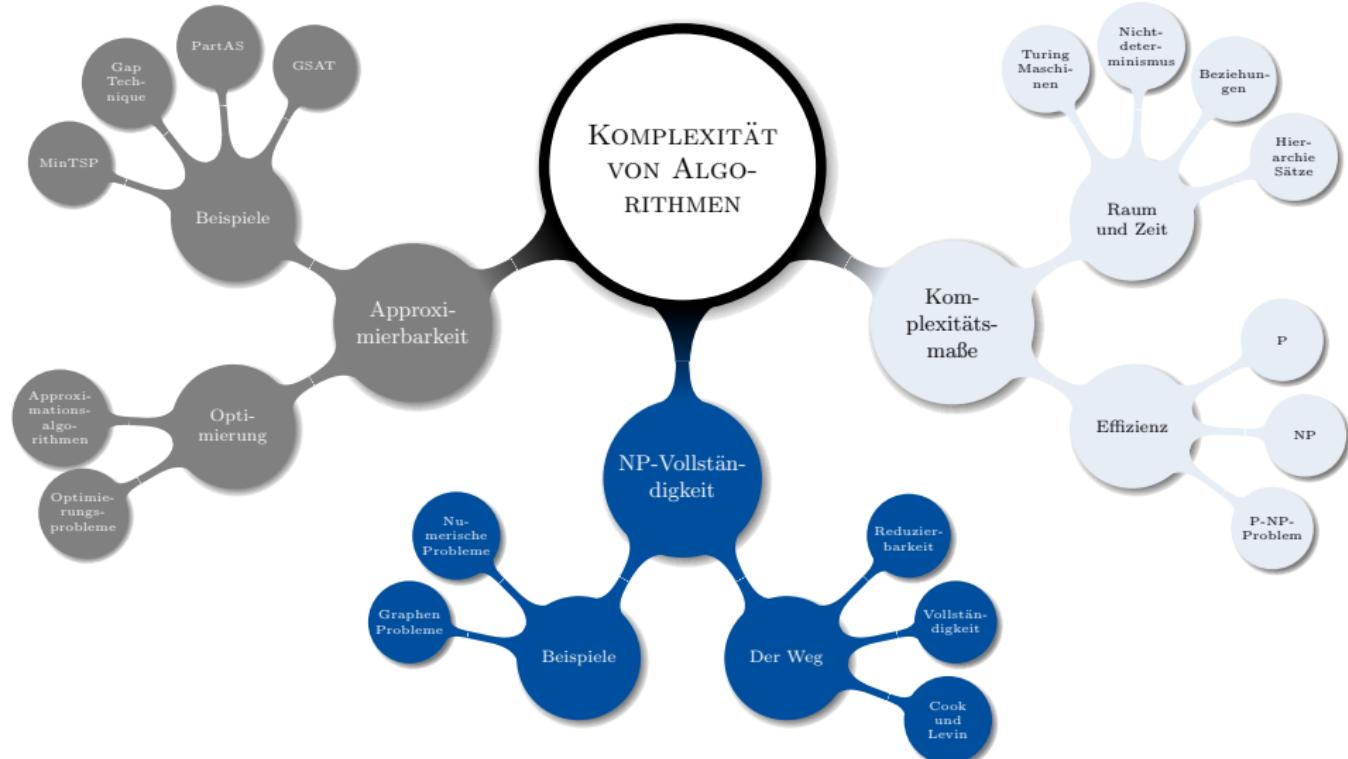
$$R_A(x) = R_{\mathcal{P}}(x, A(x)).$$

Sei nun $r: \mathbb{N} \rightarrow \mathbb{Q}$ eine Funktion. Wir sagen A ist ein **r -Approximationsalgorithmus für \mathcal{P}** , falls

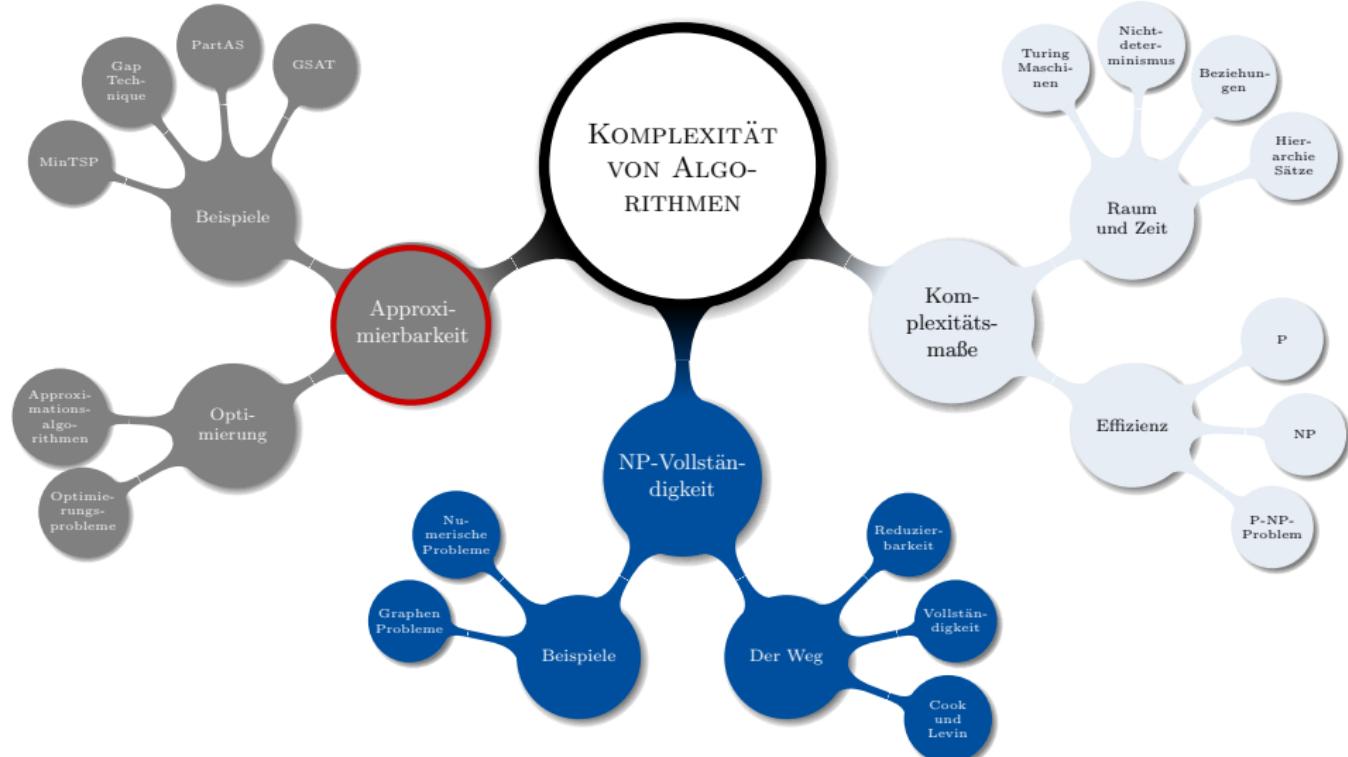
$$\max_{|x|=n} R_A(x) \leq r(n).$$

Spezialfall: Falls r konstant ist, also $r(n) = c$ für $c \in \mathbb{Q}$, so sprechen wir auch von einem **c -Approximationsalgorithmus**.

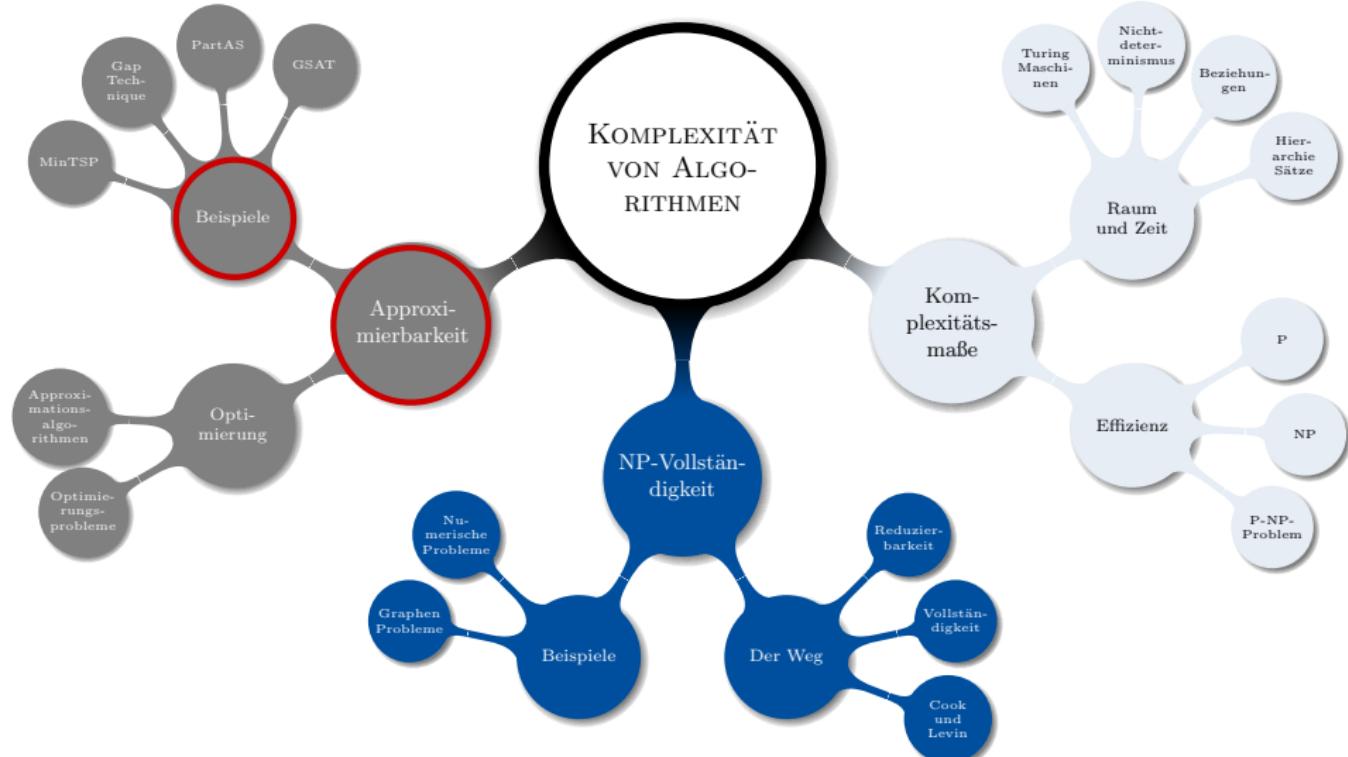
Wie geht es weiter?



Wie geht es weiter?



Wie geht es weiter?



Ein negatives Resultat

Satz 29

Falls $P \neq NP$, so gibt es kein $c \geq 1$, sodass MinTSP einen c -Approximationsalgorithmus besitzt.

Gap-technique: Mut zur Lücke

Satz 30

Sei \mathcal{P} ein Minimierungsproblem aus NPO. Sei L eine NP-schwere Sprache, $L \subseteq \Sigma^*$. Seien f, c in Polynomialzeit berechenbare Funktionen, wobei $f: \Sigma^* \rightarrow I_{\mathcal{P}}$, $c: \Sigma^* \rightarrow \mathbb{N}$ und $g > 0$ eine Konstante, sodass:

$$m_{\mathcal{P}}^*(f(x)) \leq c(x), \text{ falls } x \in L$$

$$m_{\mathcal{P}}^*(f(x)) \geq (1 + g) \cdot c(x), \text{ falls } x \notin L$$

Dann existiert kein r -Approximationsalgorithmus für \mathcal{P} für alle $r < 1 + g$, es sei denn $P = NP$.

Eine Einschränkung von MinTSP

Problem: MinMTSP (metric TSP)

Instanz: $(d_{i,j})_{1 \leq i,j \leq n}$ mit $n, d_{i,j} \in \mathbb{N}$ und für alle $i, j, k \in \{1, \dots, k\}$ gilt:

$$d_{i,j} = d_{j,i} \text{ (Symmetrie)}$$

$$d_{i,j} + d_{j,k} \geq d_{i,k} \text{ (Dreiecksungleichung)}$$

Lösung: $\pi \in S_n$ (= Rundreise)

Maß: Länge von π , also $\sum_{i=1}^{n-1} d_{\pi(i), \pi(i+1)} + d_{\pi(n), \pi(1)}$

Multigraphen und Eulerkreise

Definition

Ein **Multigraph** ist ein Paar $G = (V, F)$, wobei V eine Knotenmenge und F eine Kanten-**Multimenge** ist, d.h. es gibt eventuell mehrere Kanten zwischen Knotenpaaren.

Multigraphen und Eulerkreise



Definition

Ein **Multigraph** ist ein Paar $G = (V, F)$, wobei V eine Knotenmenge und F eine Kanten-**Multimenge** ist, d.h. es gibt eventuell mehrere Kanten zwischen Knotenpaaren.

Definition

Ein **Eulerpfad** in G ist ein Pfad v_1, \dots, v_m mit $\{v_i, v_{i+1}\} \in F$ für $1 \leq i < m$, der jede Kante von G genau einmal besucht. Ein **Eulerkreis** ist ein Eulerpfad, der ein Kreis ist.

Leonhard Euler,

wikimedia, public domain

Das Königsberger Brückenproblem

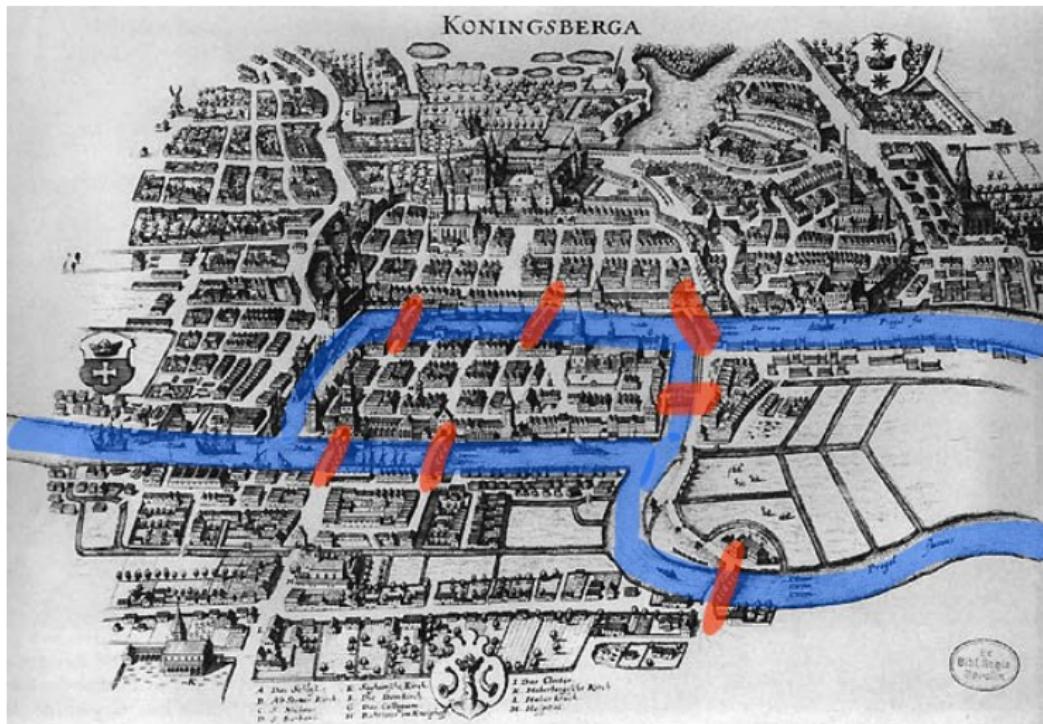


Abbildung von [wikimedia.org](https://commons.wikimedia.org), public domain

Das Königsberger Brückenproblem als Multigraph

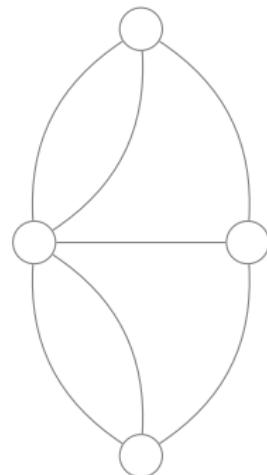
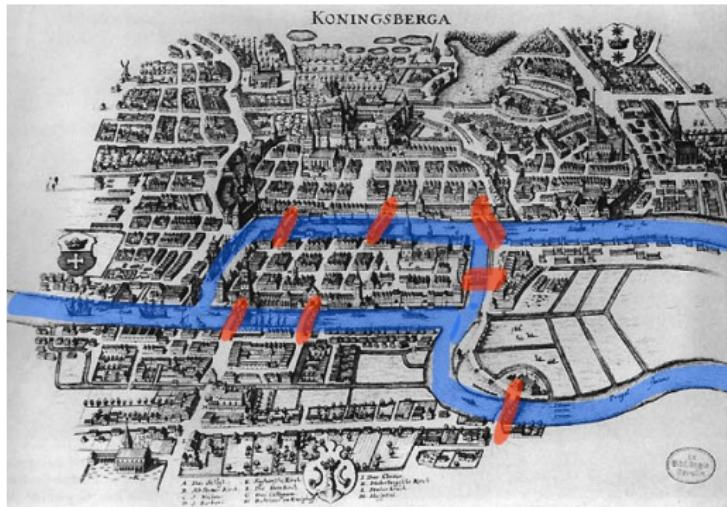


Abbildung von [wikimedia.org](https://commons.wikimedia.org), public domain

Definition

Ein **gewichteter** Graph ist ein Graph mit Gewichten an den Kanten, also ein Tripel $G = (V, E, D)$, wobei (V, E) ein Graph ist und $D = (d_{u,v})_{(u,v) \in E}$ eine Gewichtsmatrix für die Kanten ist, die ihnen ein Gewicht zuordnet.

Spannbäume

Definition

Ein **gewichteter** Graph ist ein Graph mit Gewichten an den Kanten, also ein Tripel $G = (V, E, D)$, wobei (V, E) ein Graph ist und $D = (d_{u,v})_{(u,v) \in E}$ eine Gewichtsmatrix für die Kanten ist, die ihnen ein Gewicht zuordnet.

Definition

Ein **Spannbaum** in einem Graphen $G = (V, E)$ ist ein Teilgraph $G' = (V, E')$, sodass $E' \subseteq E$ gilt und G' außerdem ein Baum ist.

Definition

Ein **gewichteter** Graph ist ein Graph mit Gewichten an den Kanten, also ein Tripel $G = (V, E, D)$, wobei (V, E) ein Graph ist und $D = (d_{u,v})_{(u,v) \in E}$ eine Gewichtsmatrix für die Kanten ist, die ihnen ein Gewicht zuordnet.

Definition

Ein **Spannbaum** in einem Graphen $G = (V, E)$ ist ein Teilgraph $G' = (V, E')$, sodass $E' \subseteq E$ gilt und G' außerdem ein Baum ist.

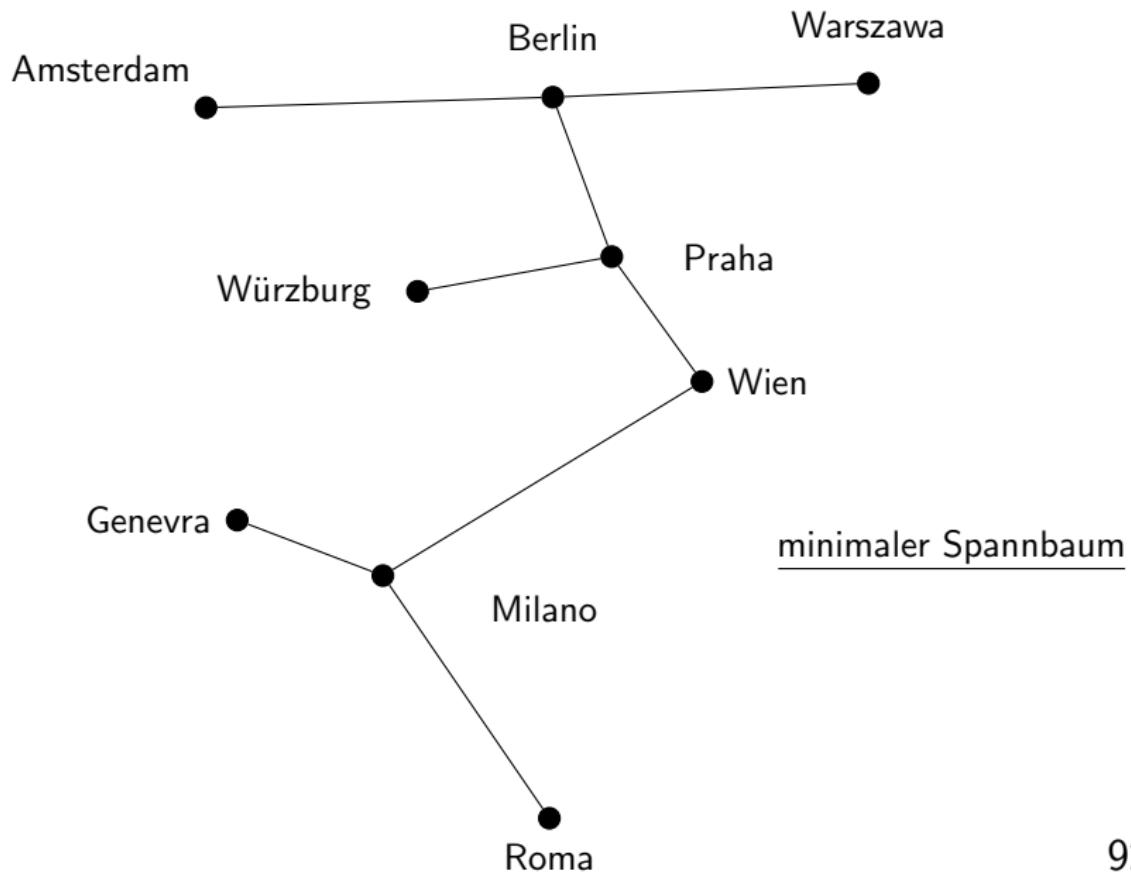
Definition

Sei $G = (V, E, D)$ ein gewichteter Graph. Ein **minimaler Spannbaum** in G ist ein Spannbaum $G' = (V, E')$ im Graphen (V, E) mit minimalem Gewicht. Dabei ist das Gewicht von G' definiert als $\sum_{(u,v) \in E} d_{(u,v)}$.

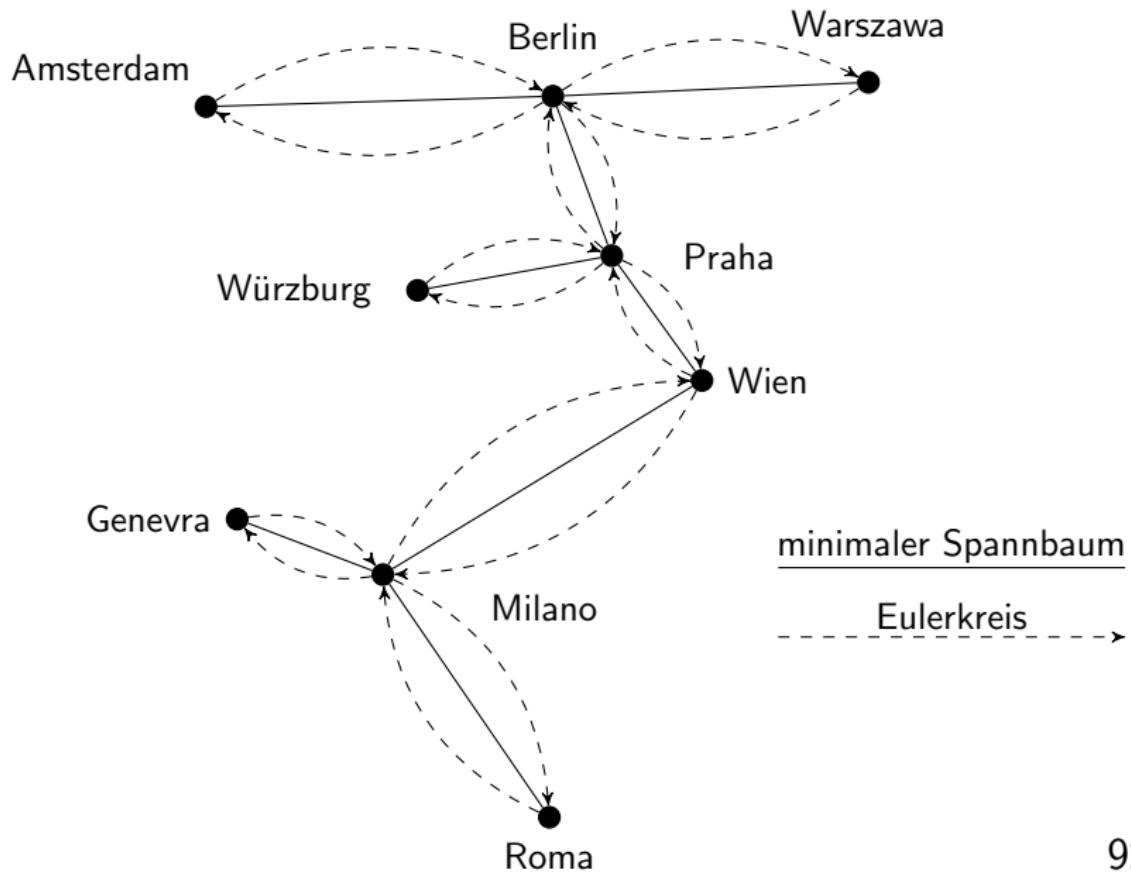
TreeTSP am Beispiel



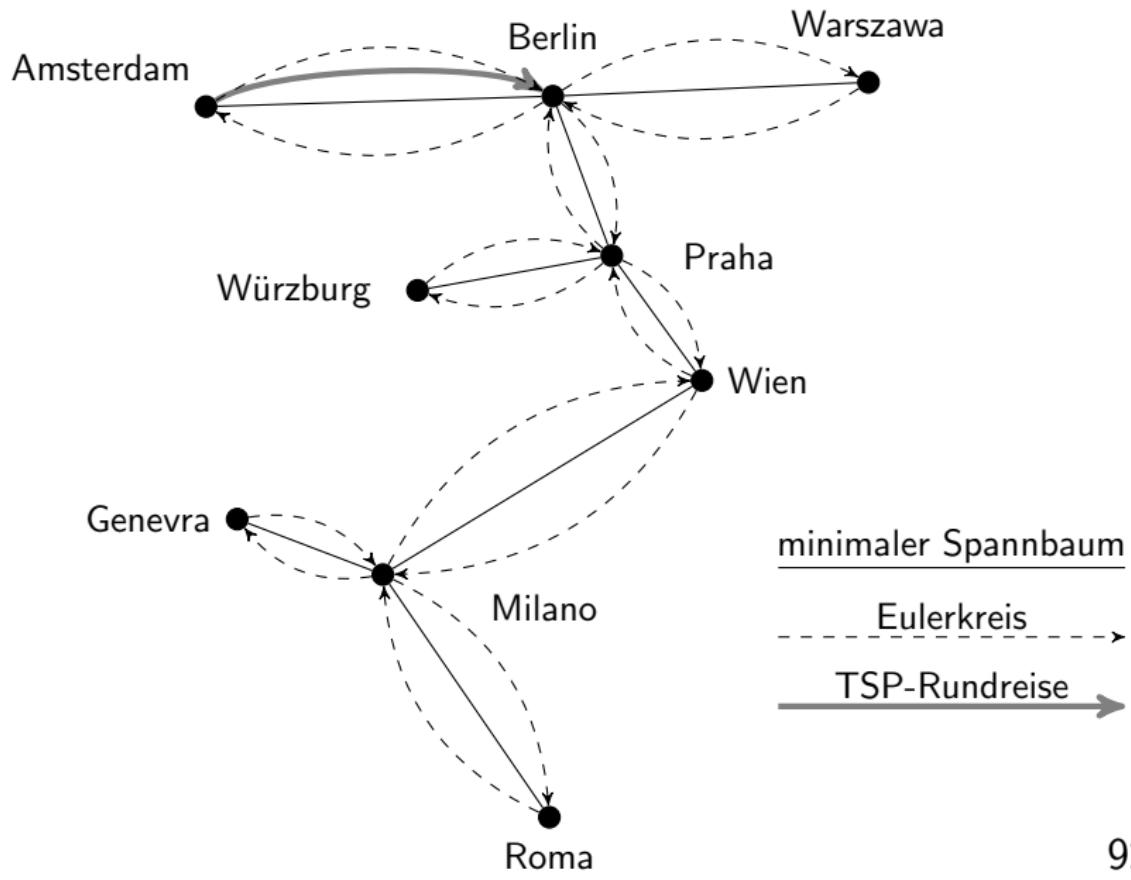
TreeTSP am Beispiel



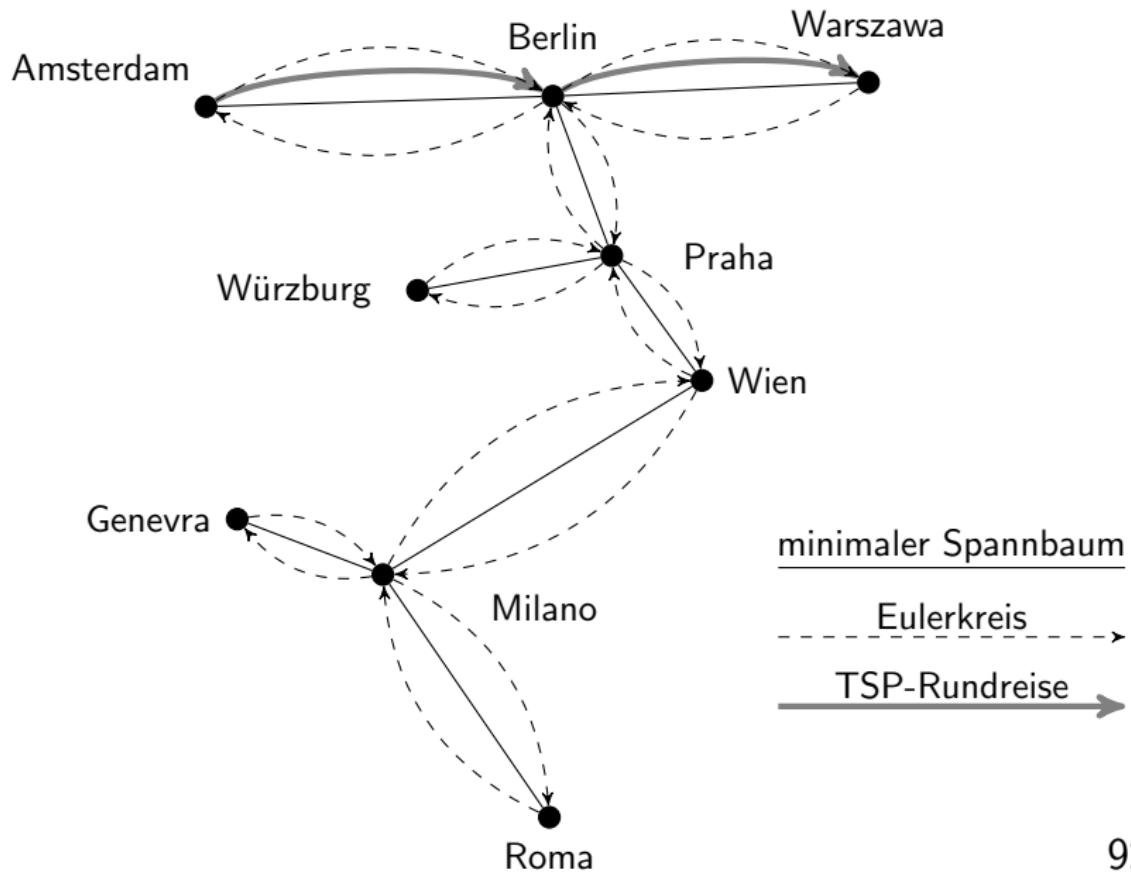
TreeTSP am Beispiel



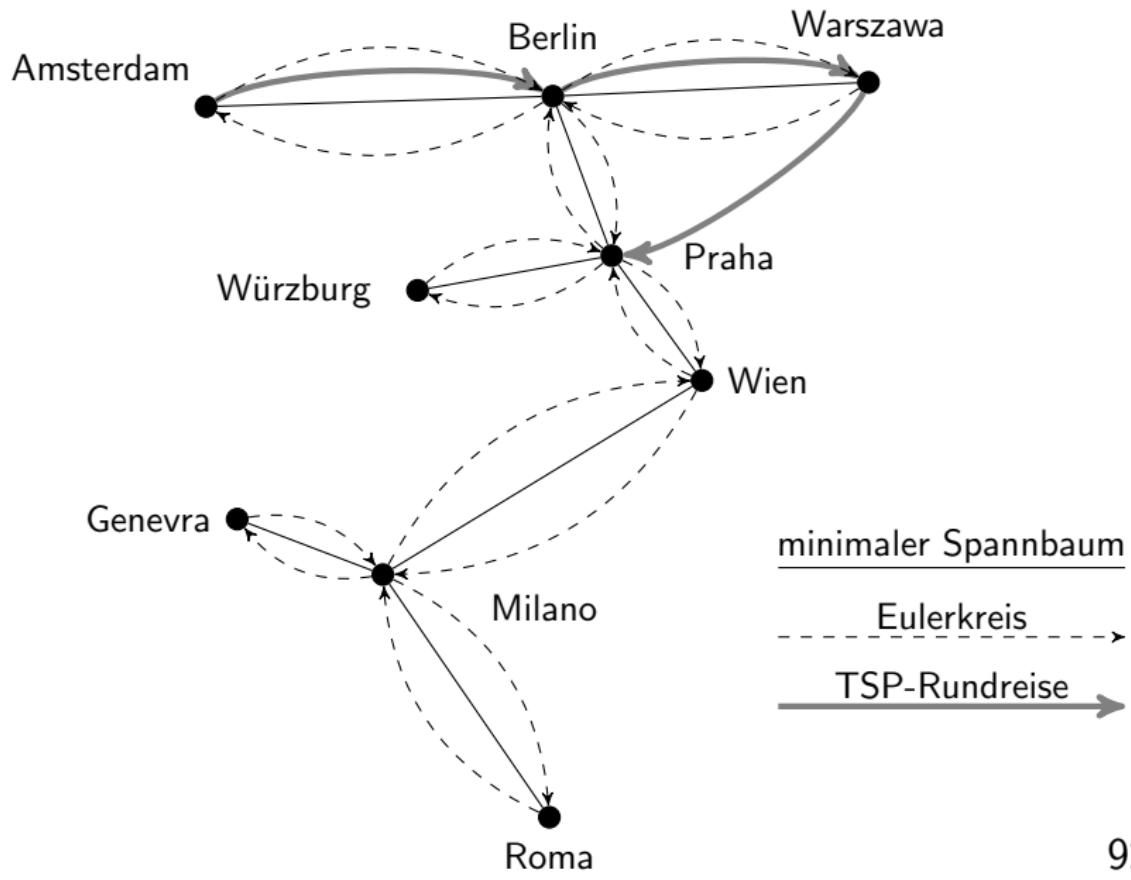
TreeTSP am Beispiel



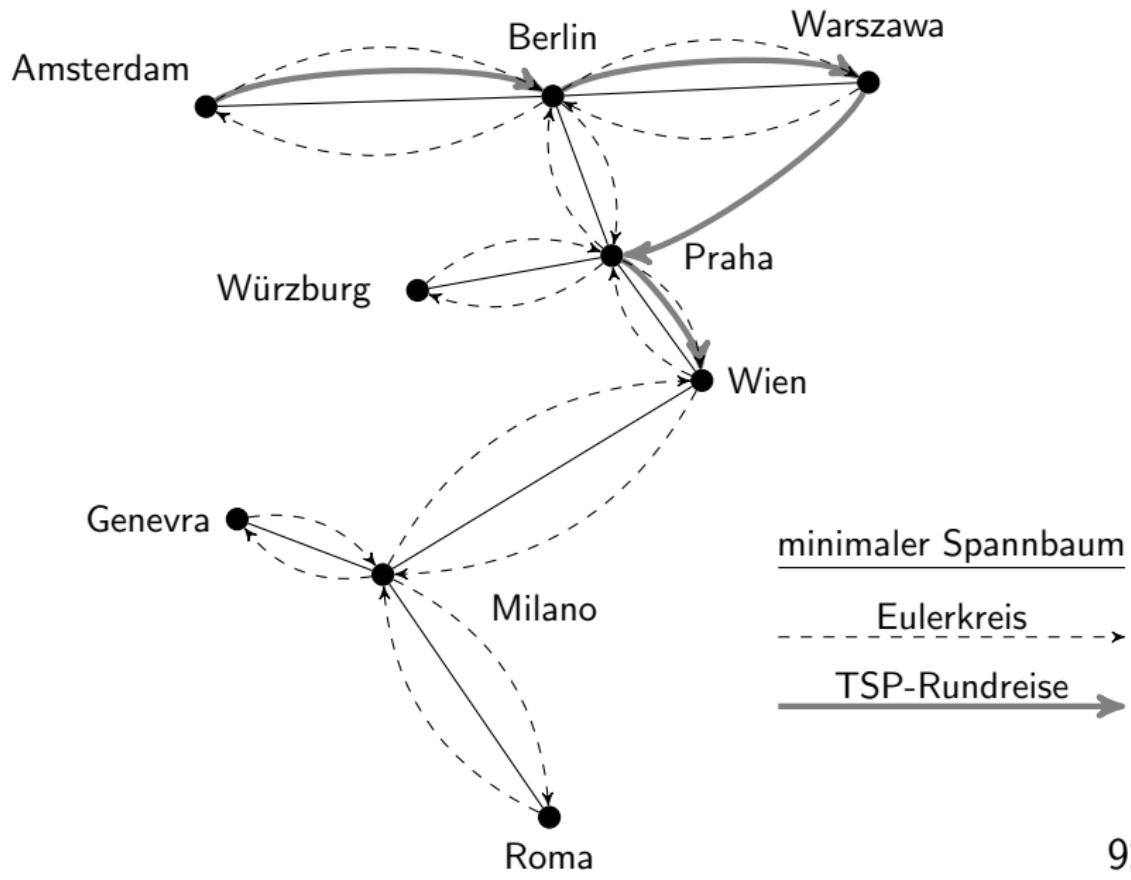
TreeTSP am Beispiel



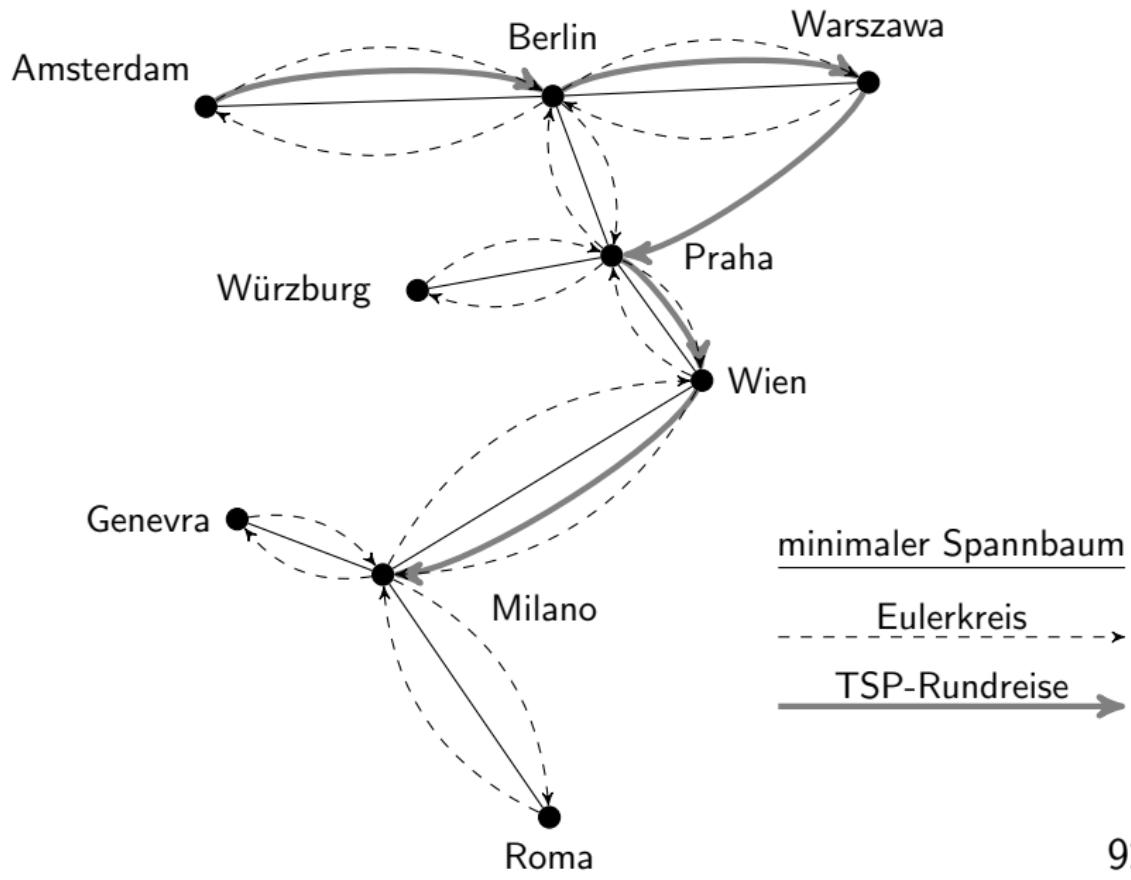
TreeTSP am Beispiel



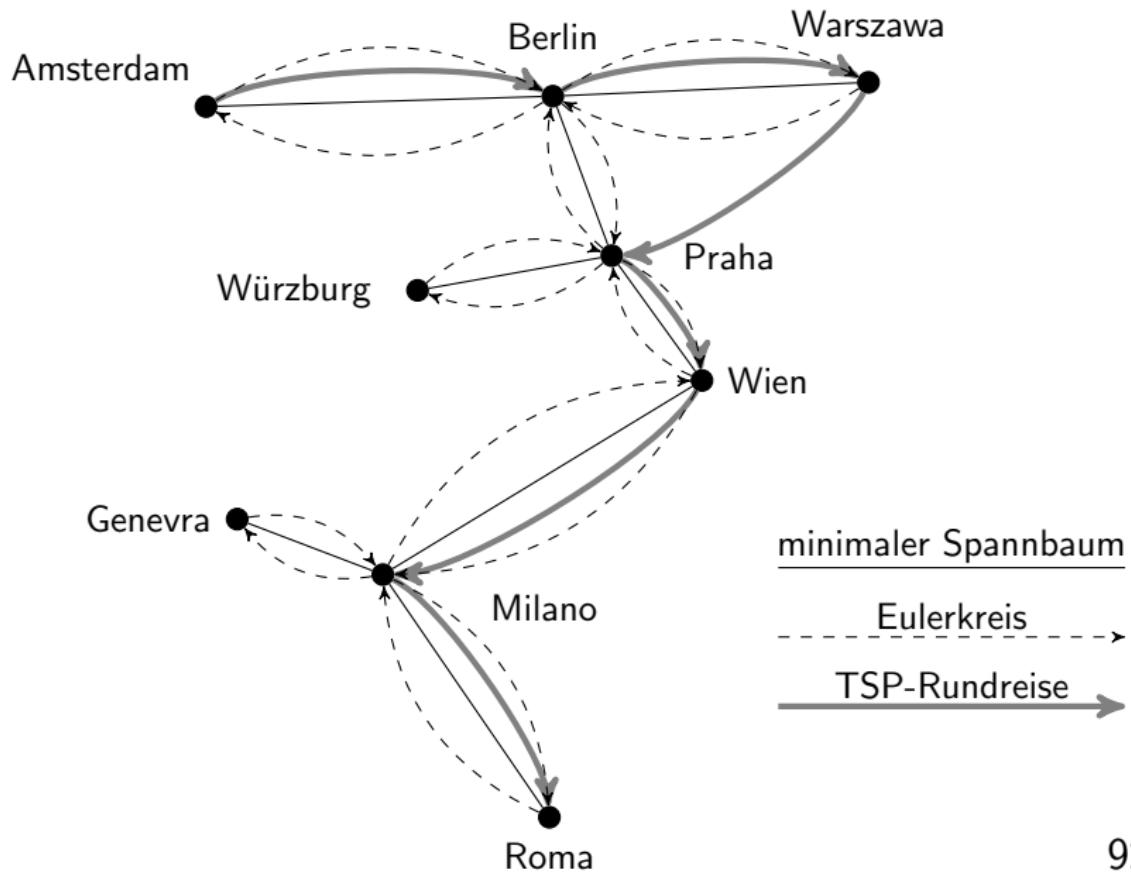
TreeTSP am Beispiel



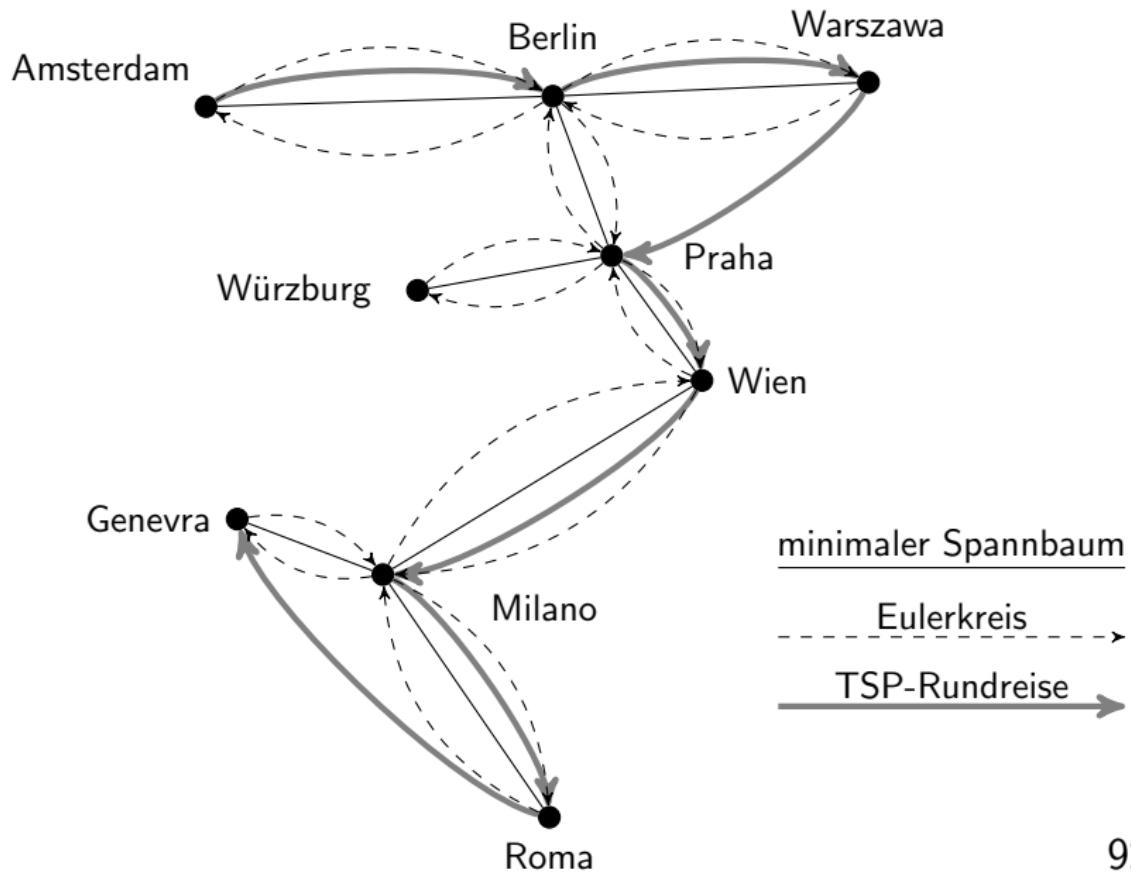
TreeTSP am Beispiel



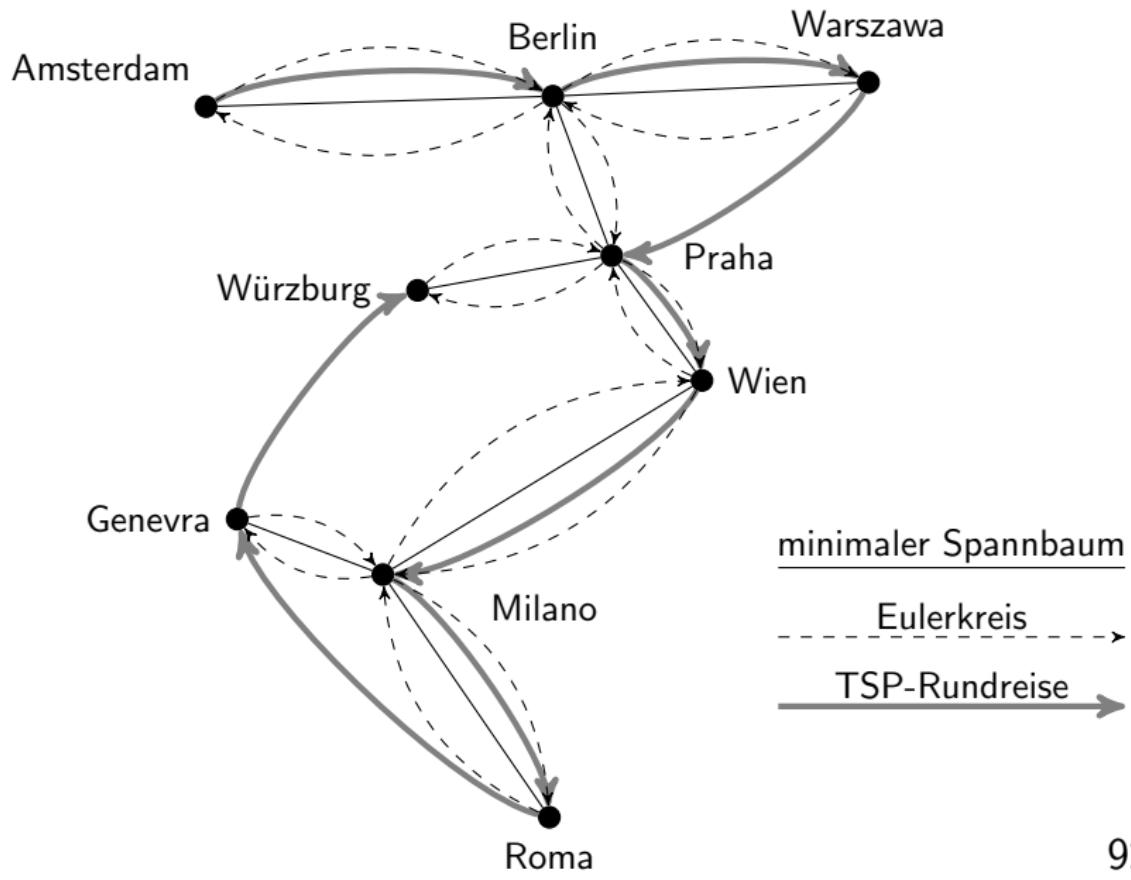
TreeTSP am Beispiel



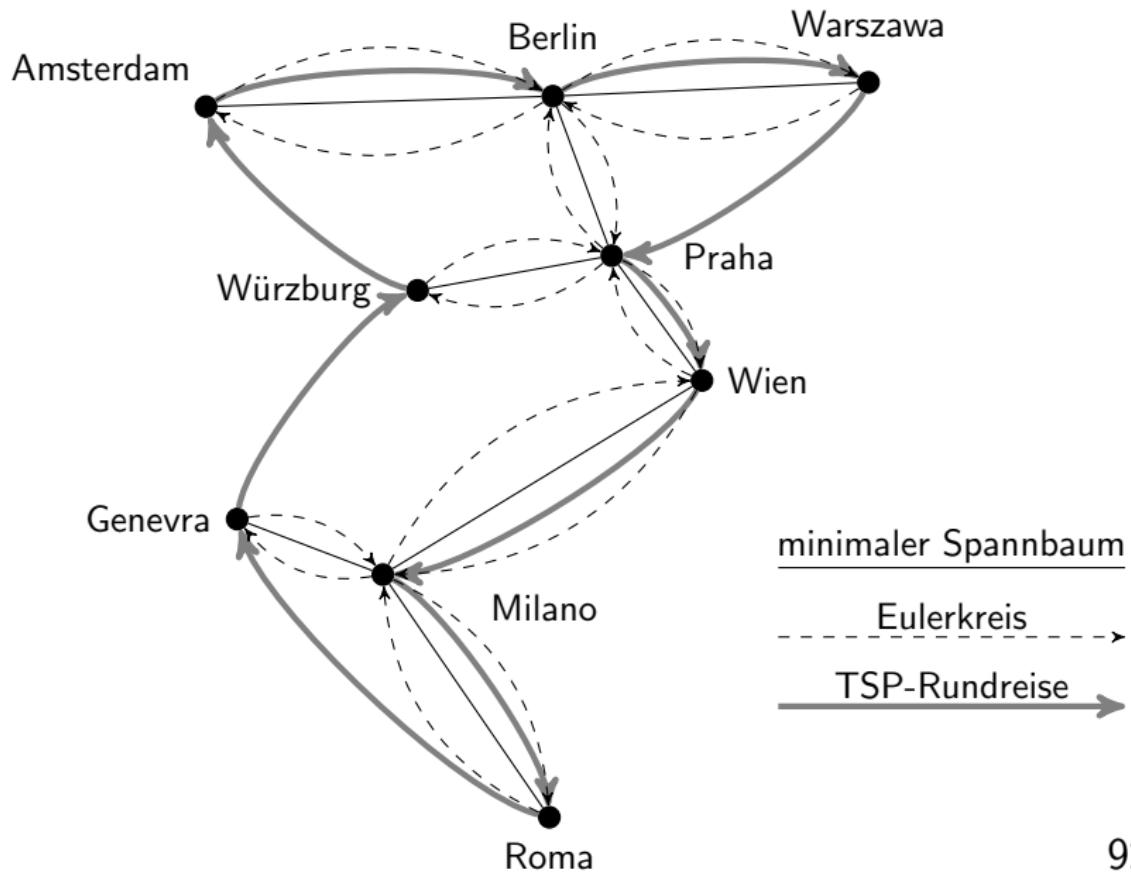
TreeTSP am Beispiel



TreeTSP am Beispiel



TreeTSP am Beispiel



TreeTSPs Lösungen sind halb so gut

Satz 31

TreeTSP ist ein 2-Approximationsalgorithmus für MinMTSP.

Definition (Matching)

Sei $G = (V, E)$ ein Graph. Ein *Matching* von G ist eine Kantenmenge $M \subseteq E$, so dass keine zwei Kanten in M einen gemeinsamen Endpunkt haben,

d.h. $(u, v), (u', v') \in M \Rightarrow \{u, v\} \cap \{u', v'\} = \emptyset$.

M heißt *vollständig* (oder *perfekt*), falls $|M| = \lfloor \frac{1}{2}|V| \rfloor$ (jeder Knoten kommt als Endpunkt einer Kante in M vor).

Nun sind wir nur noch 1.5x so schlecht wie eine beste Lösung



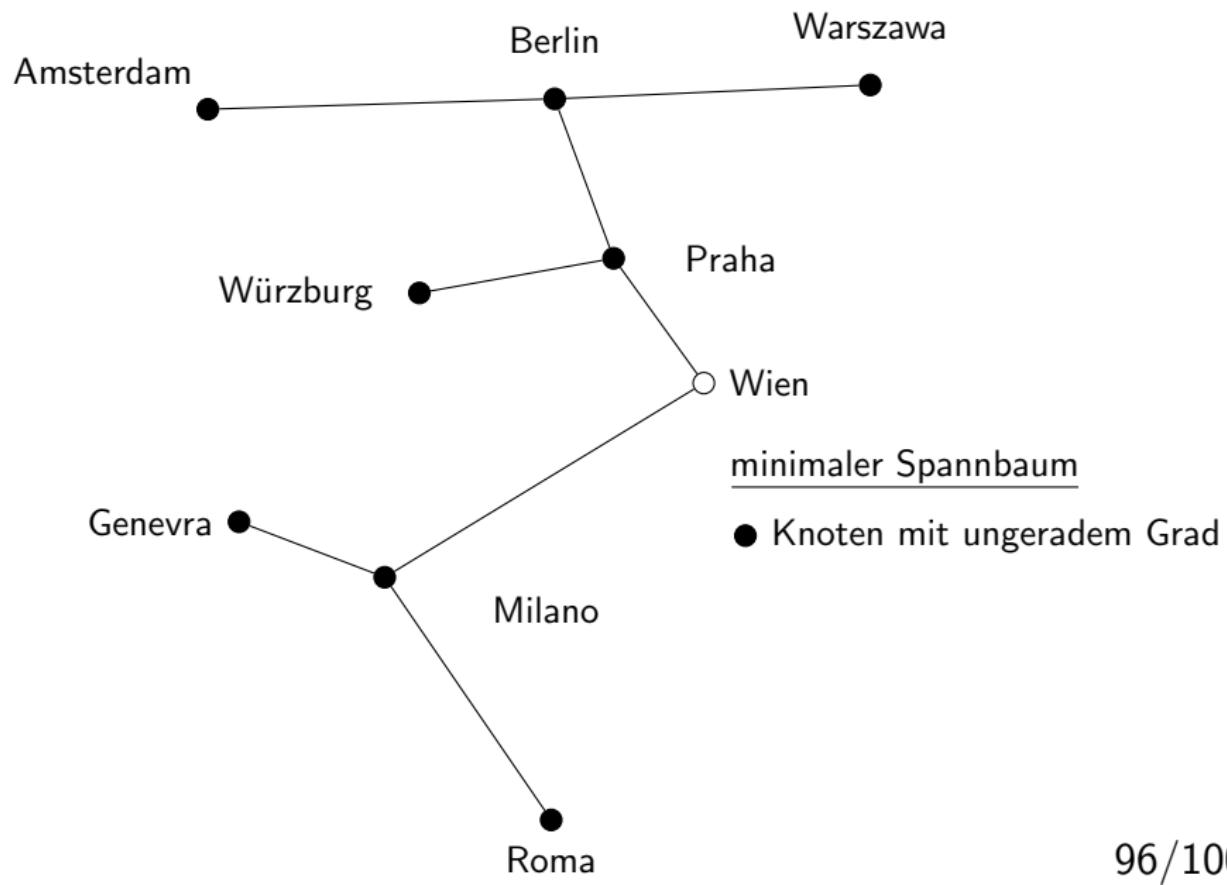
Satz 32

Der Algorithmus von Christofides ist ein $\frac{3}{2}$ -Approximationsalgorithmus für MinMTSP.

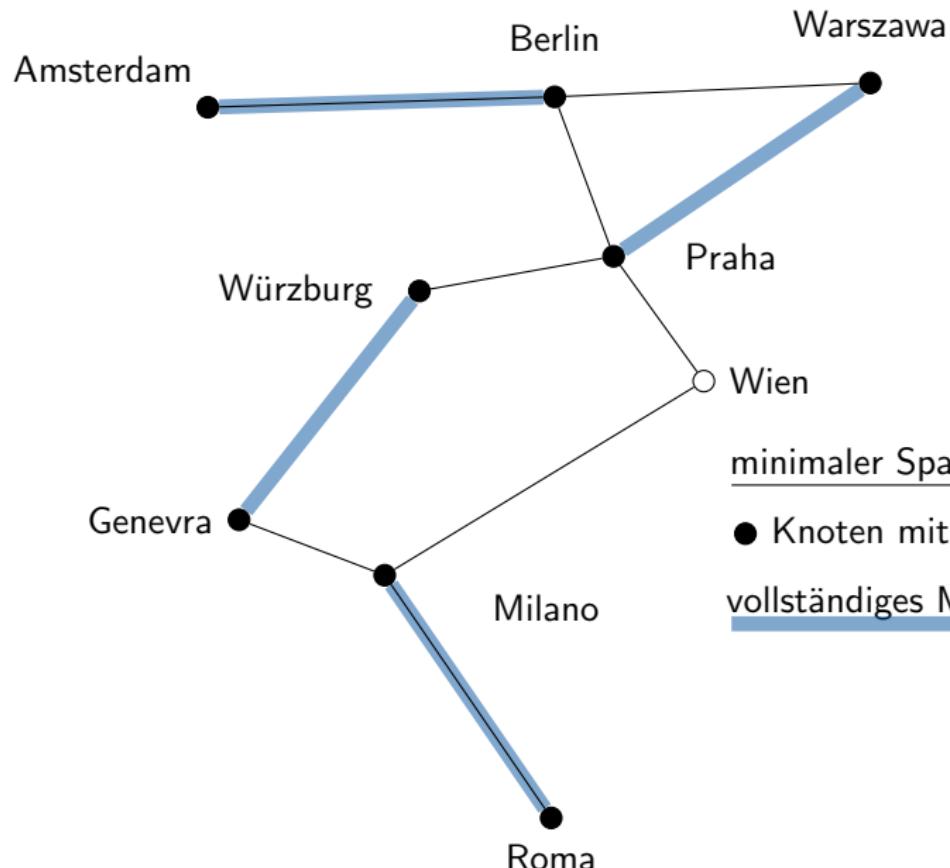
Nicos Christofides,

Quelle: investments-forum.com

Christofides am Beispiel



Christofides am Beispiel

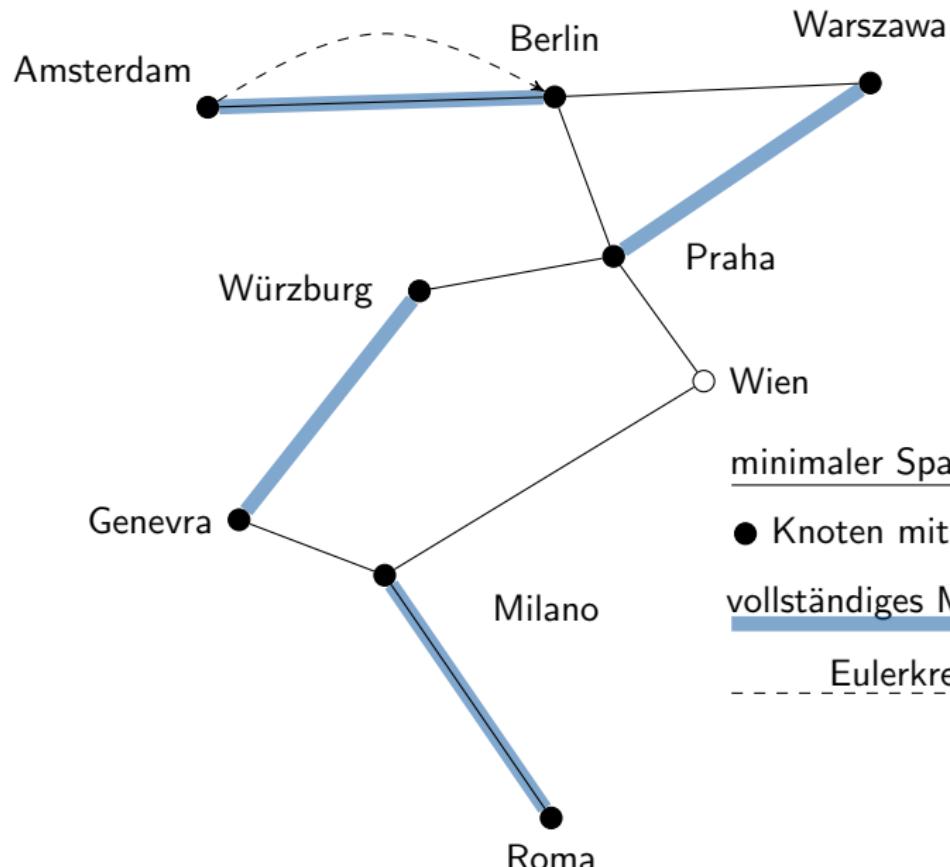


minimaler Spannbaum

● Knoten mit ungeradem Grad

vollständiges Matching

Christofides am Beispiel



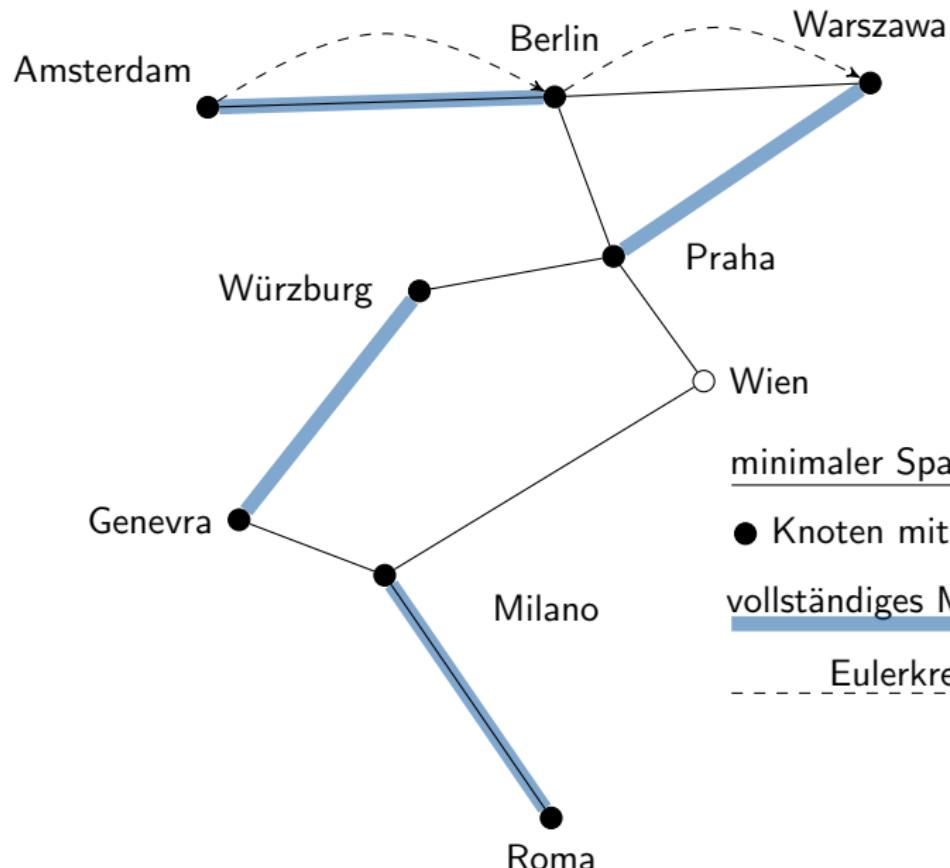
minimaler Spannbaum

● Knoten mit ungeradem Grad

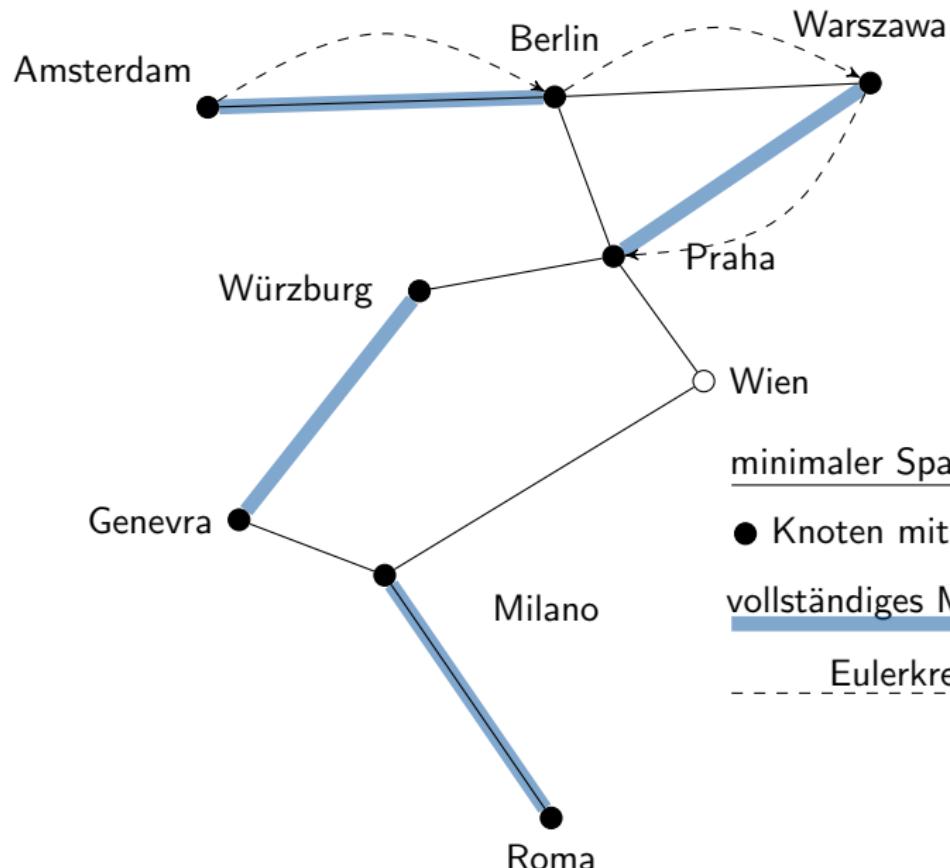
vollständiges Matching

Eulerkreis

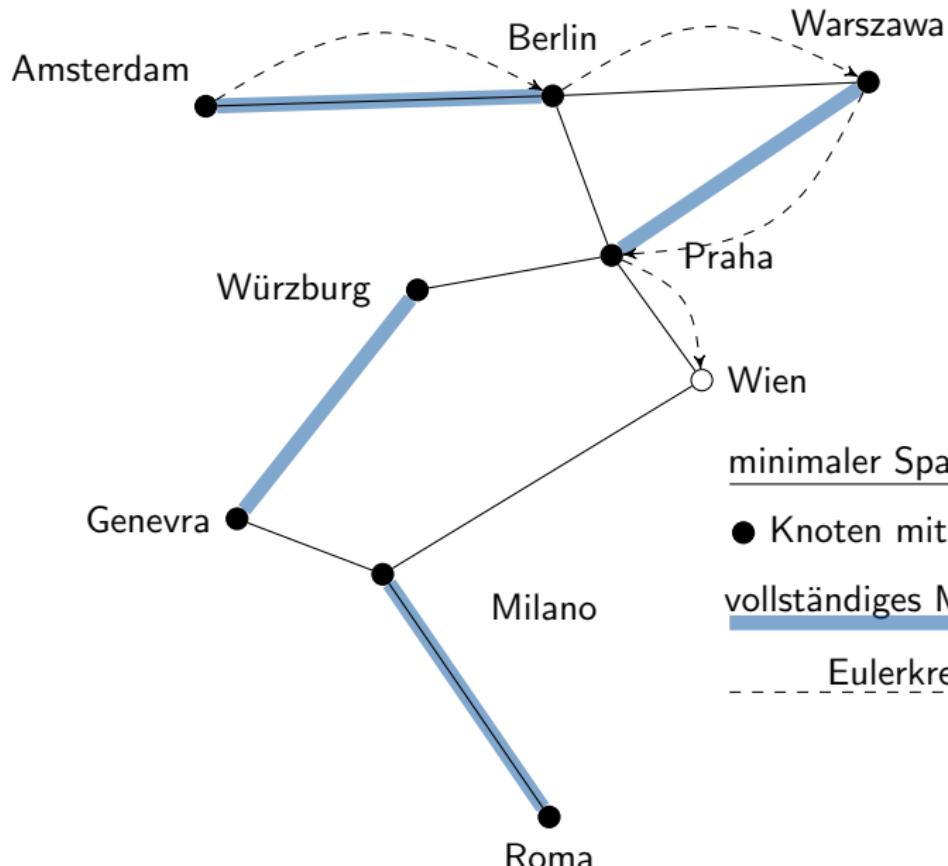
Christofides am Beispiel



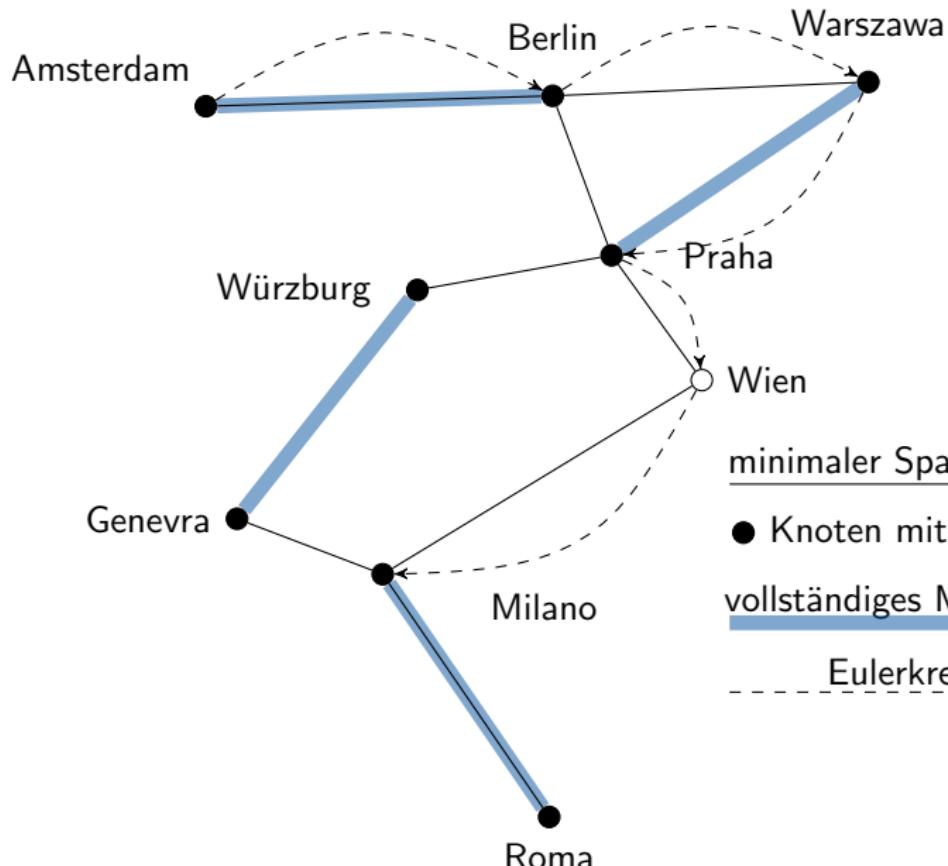
Christofides am Beispiel



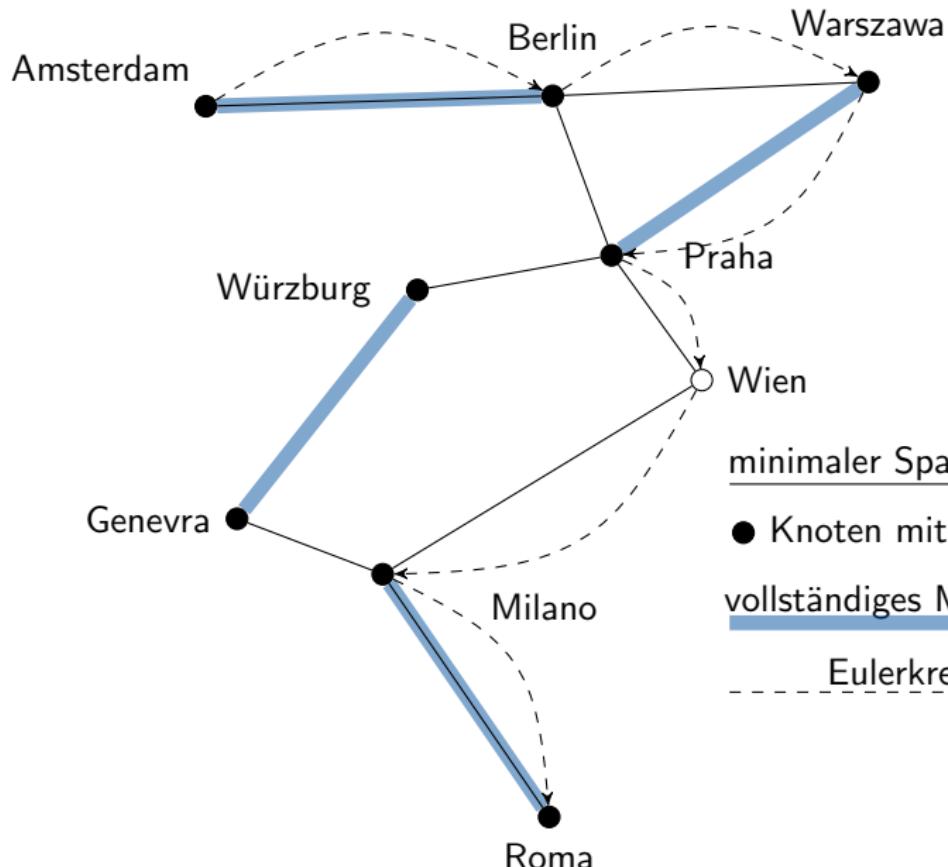
Christofides am Beispiel



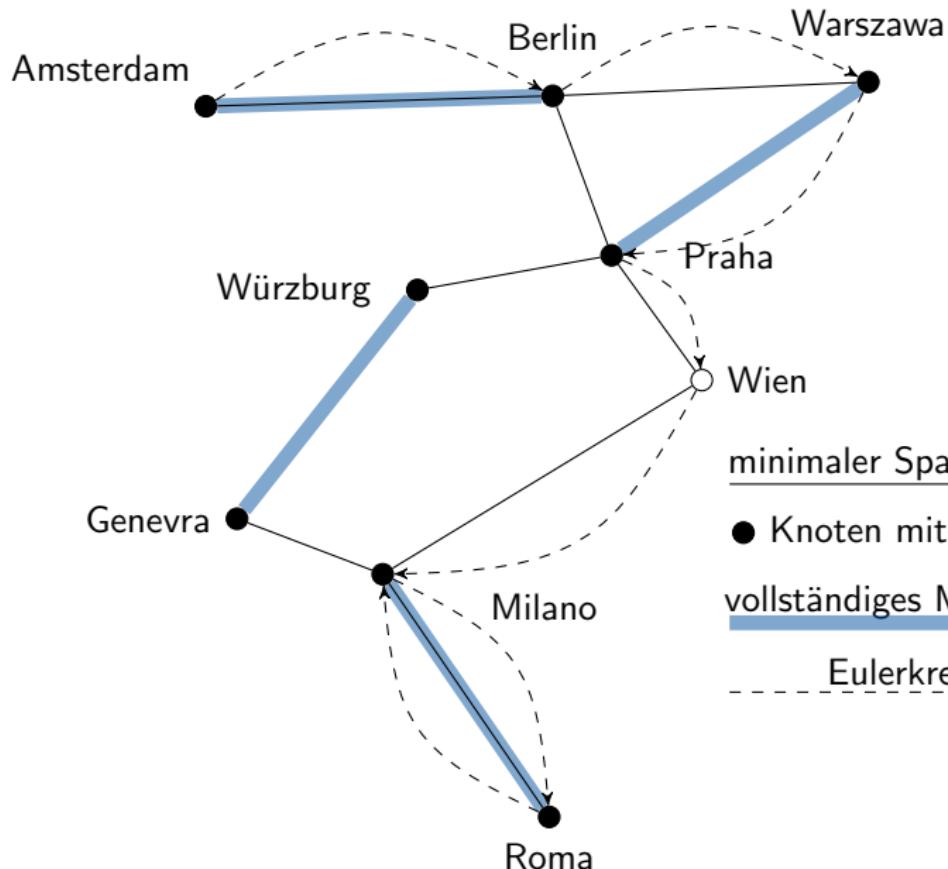
Christofides am Beispiel



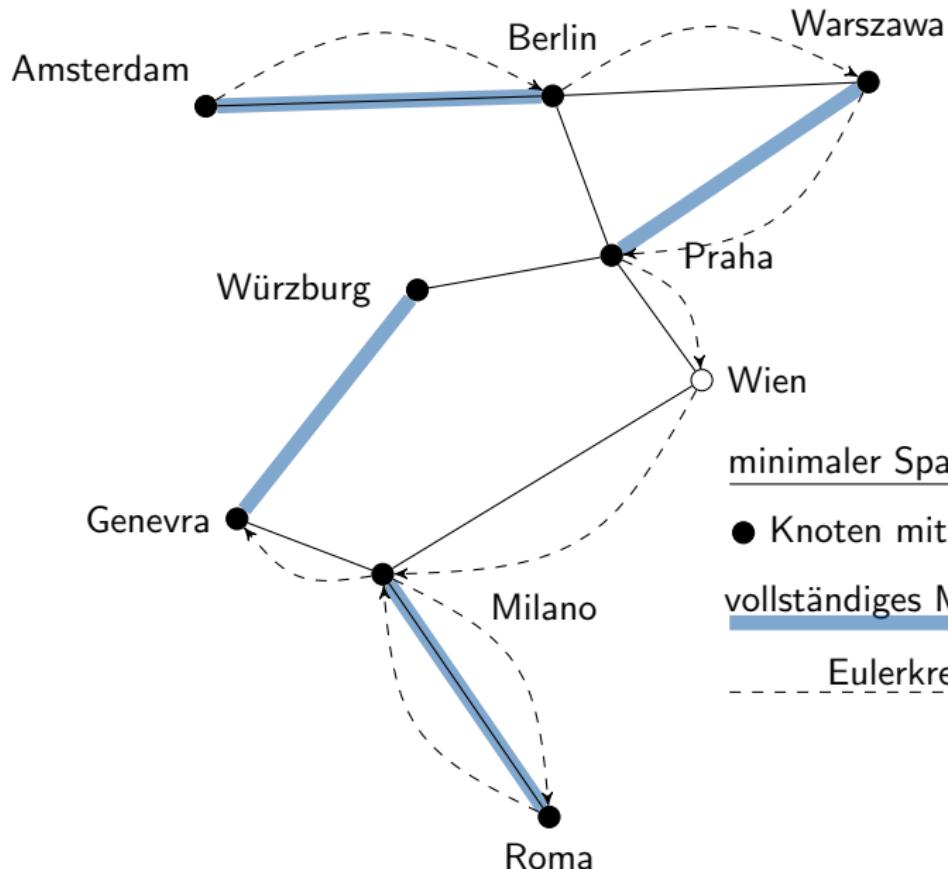
Christofides am Beispiel



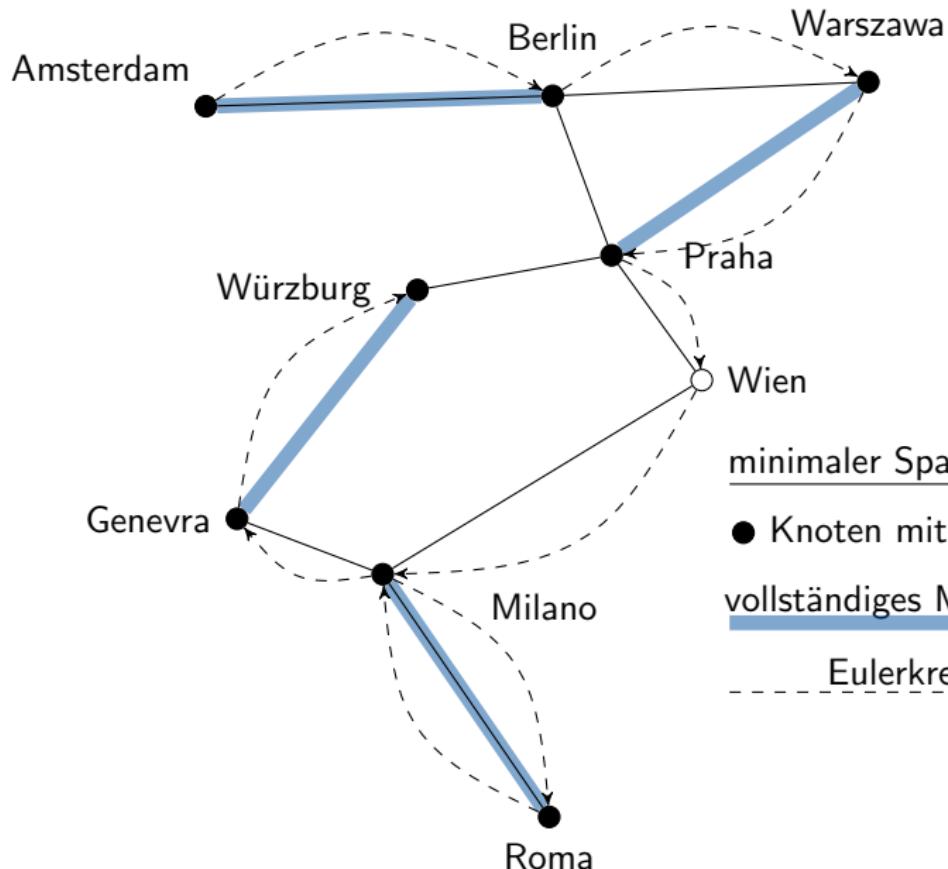
Christofides am Beispiel



Christofides am Beispiel



Christofides am Beispiel



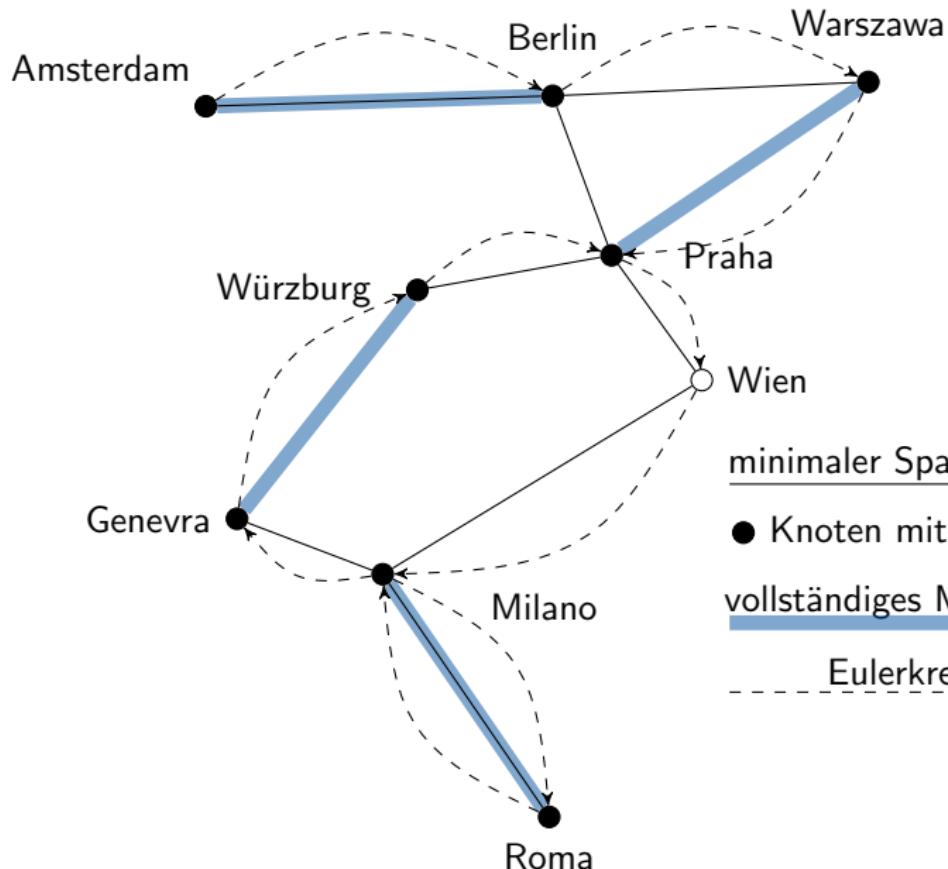
minimaler Spannbaum

● Knoten mit ungeradem Grad

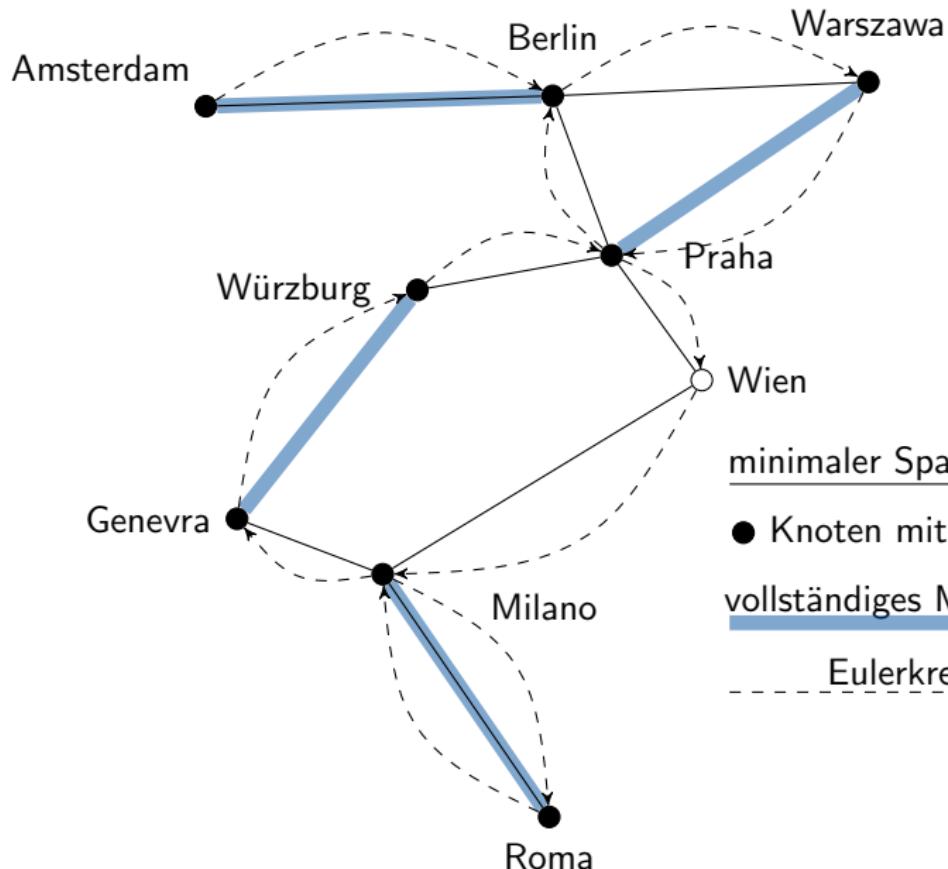
vollständiges Matching

Eulerkreis

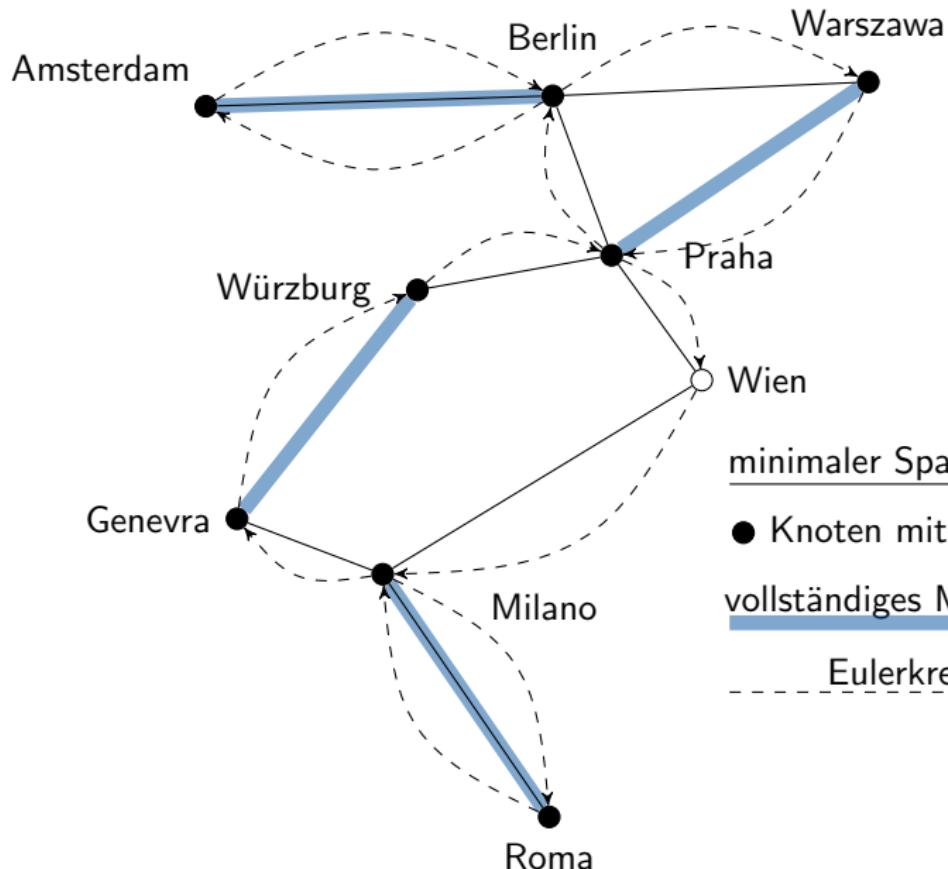
Christofides am Beispiel



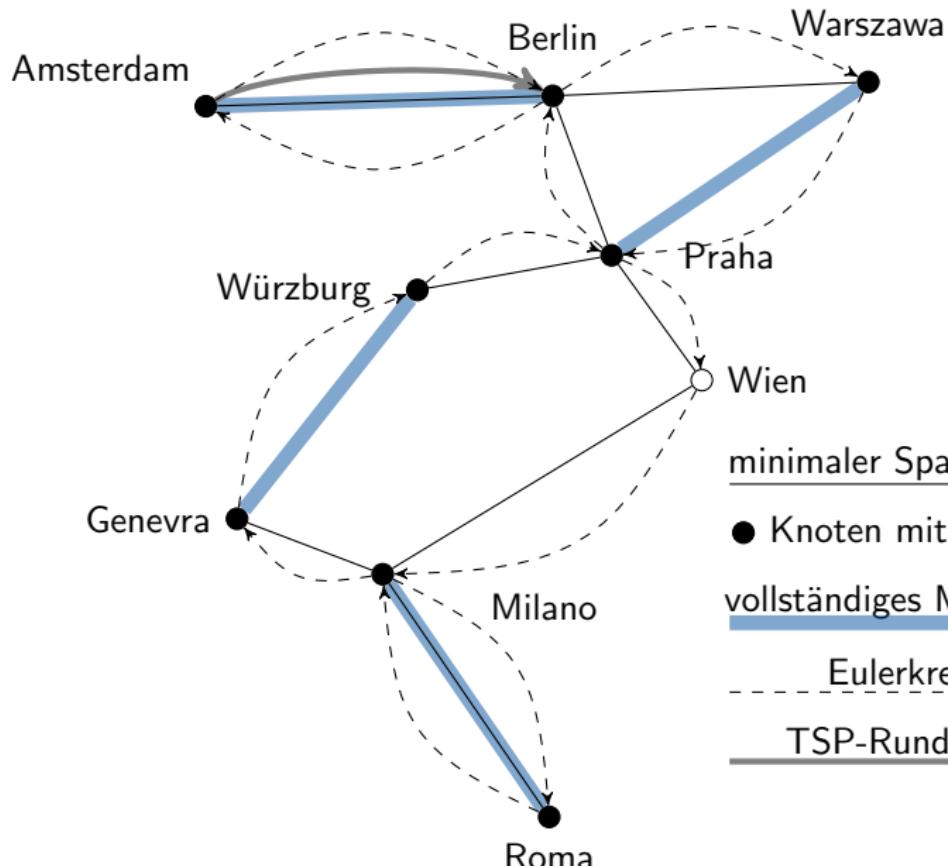
Christofides am Beispiel



Christofides am Beispiel



Christofides am Beispiel



minimaler Spannbaum

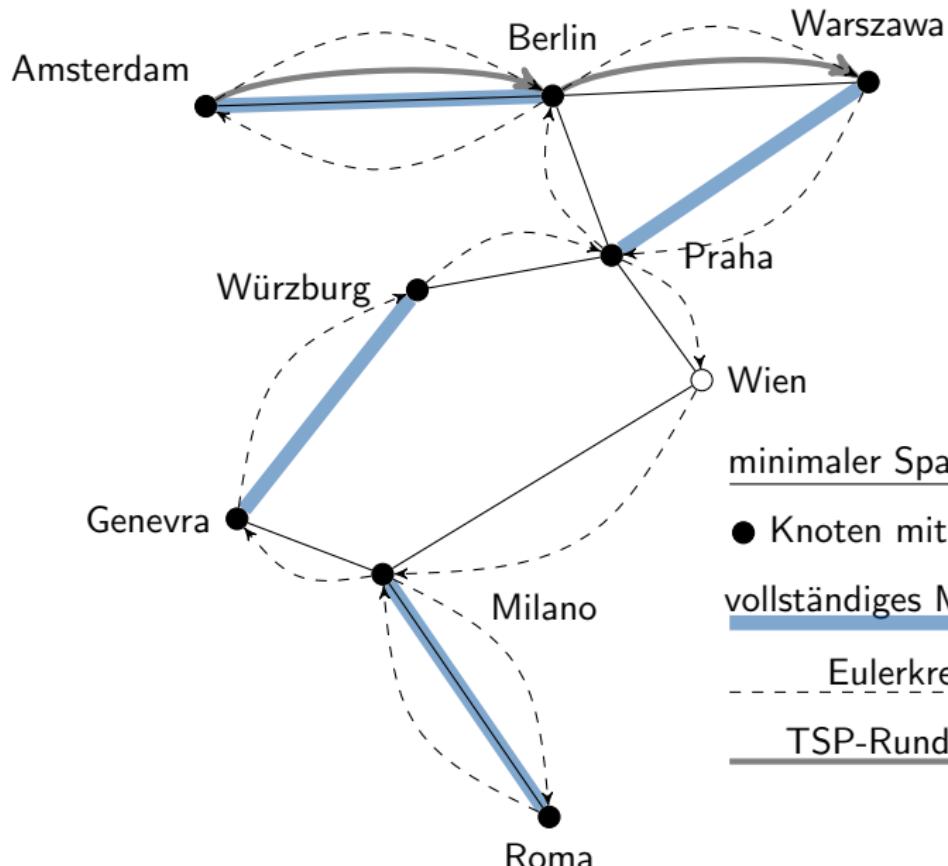
● Knoten mit ungeradem Grad

vollständiges Matching

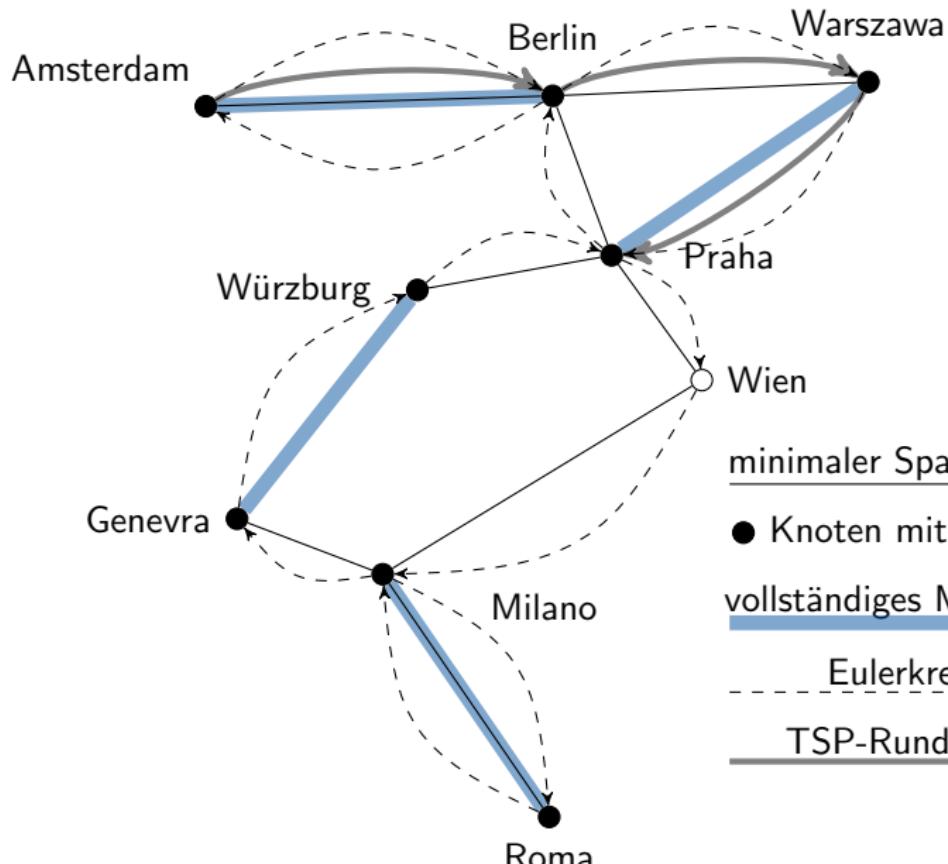
Eulerkreis →

TSP-Rundreise →

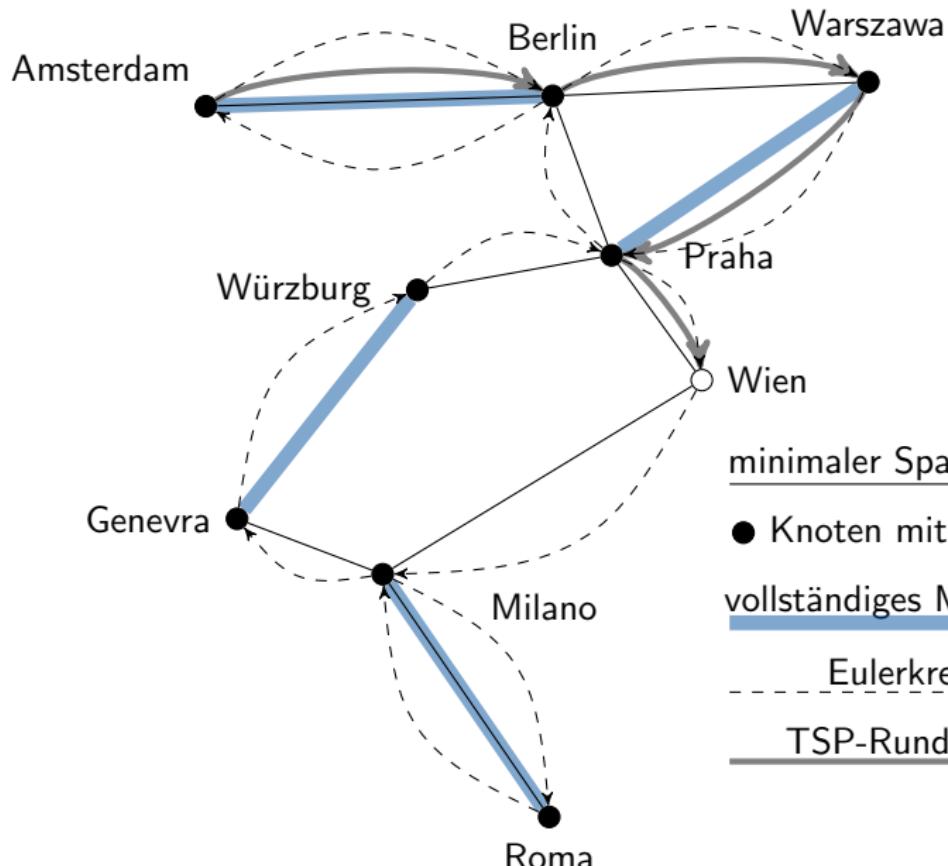
Christofides am Beispiel



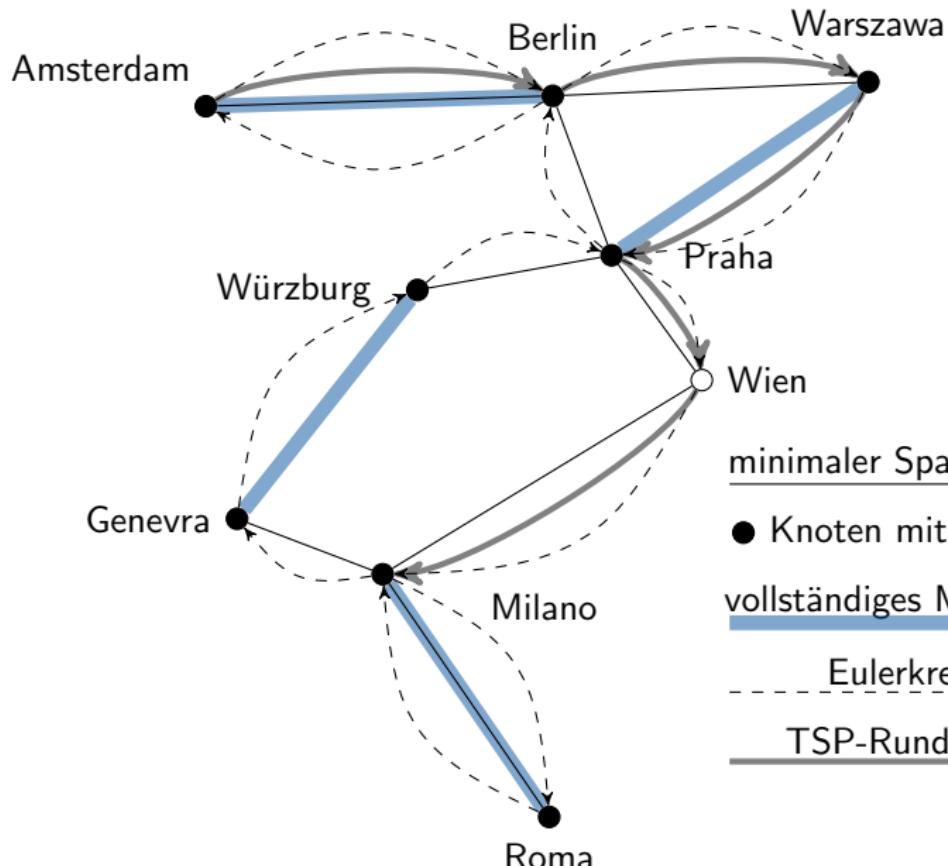
Christofides am Beispiel



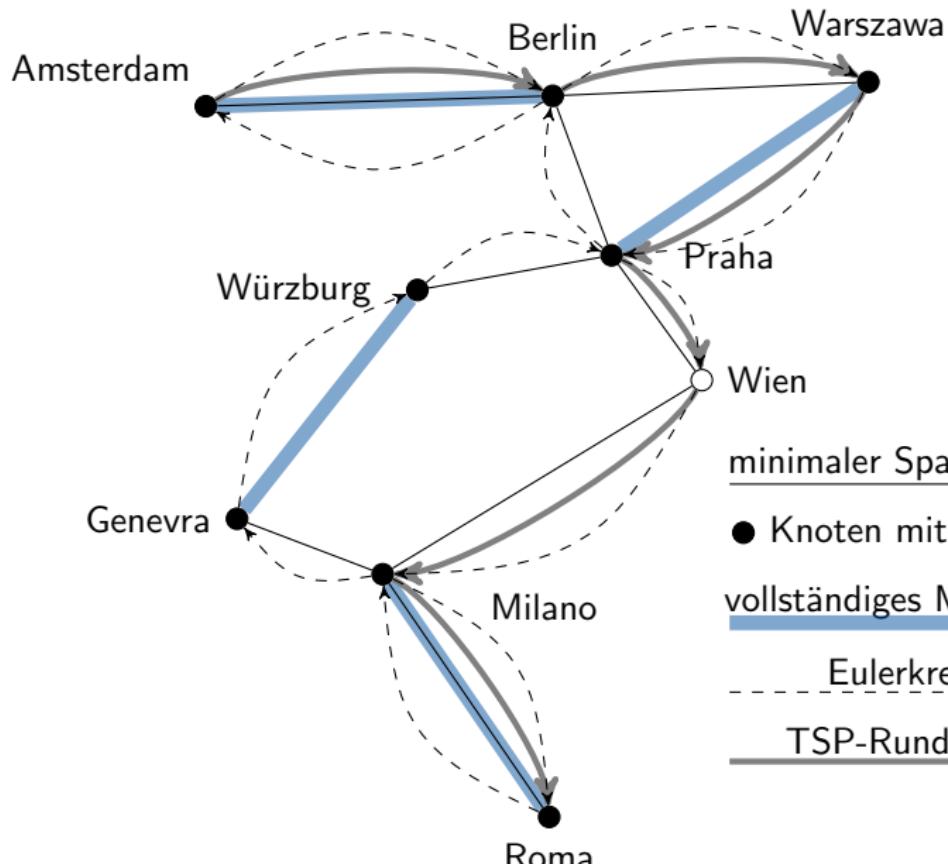
Christofides am Beispiel



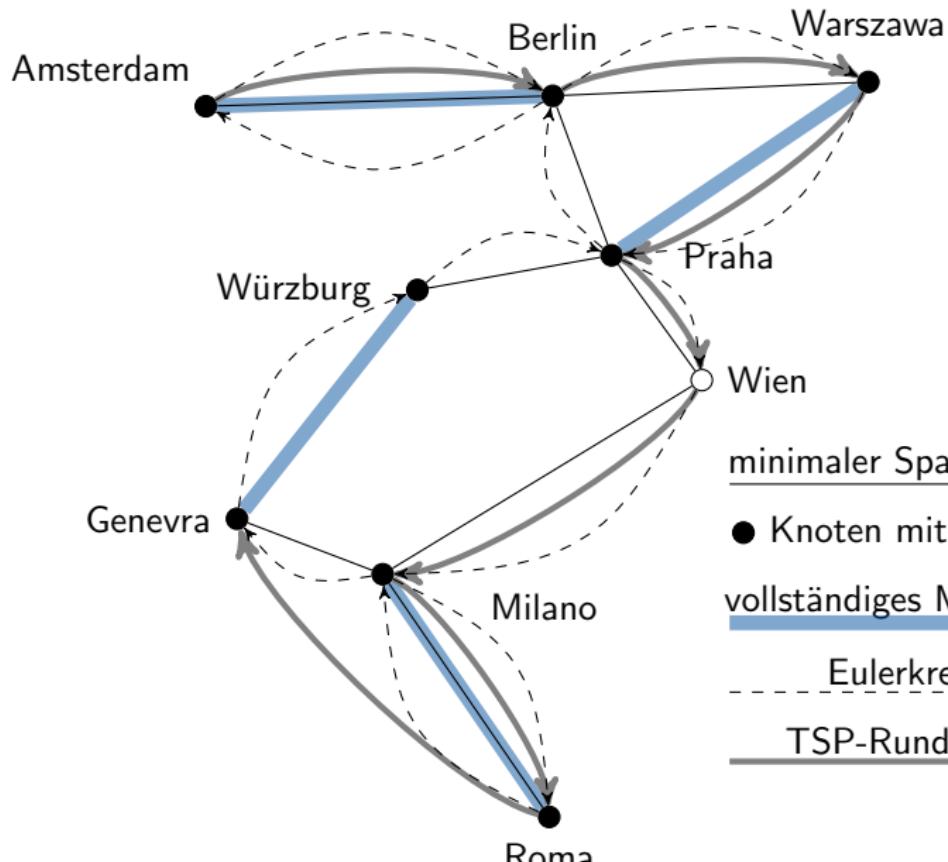
Christofides am Beispiel



Christofides am Beispiel



Christofides am Beispiel



minimaler Spannbaum

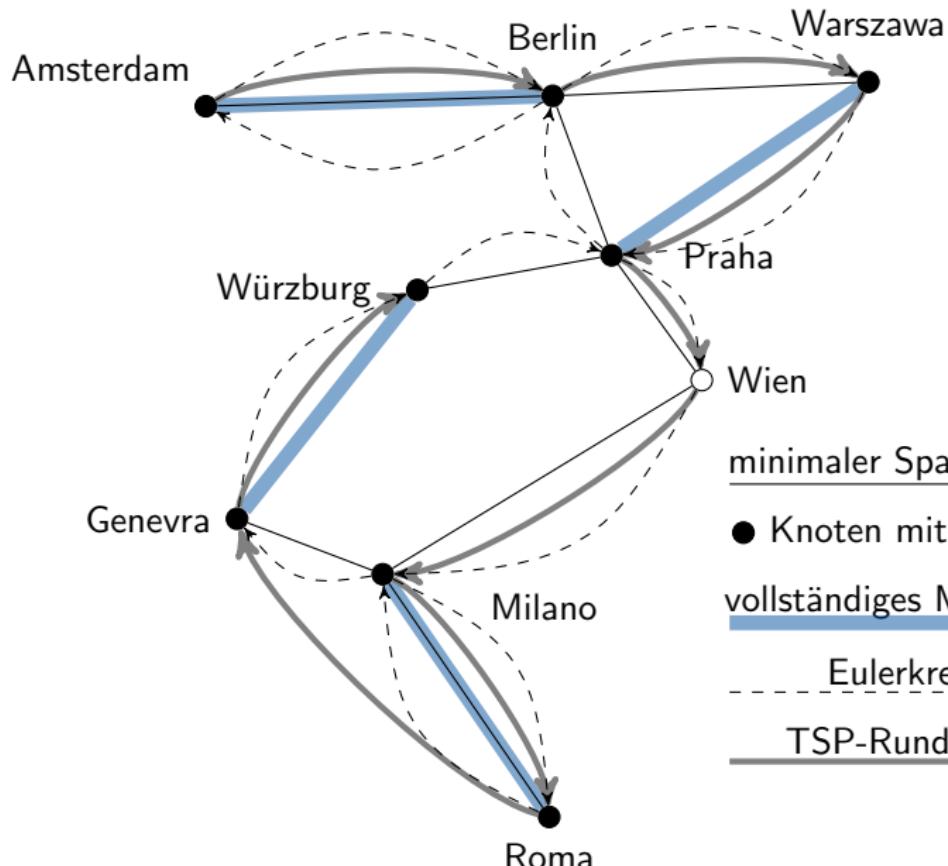
● Knoten mit ungeradem Grad

vollständiges Matching

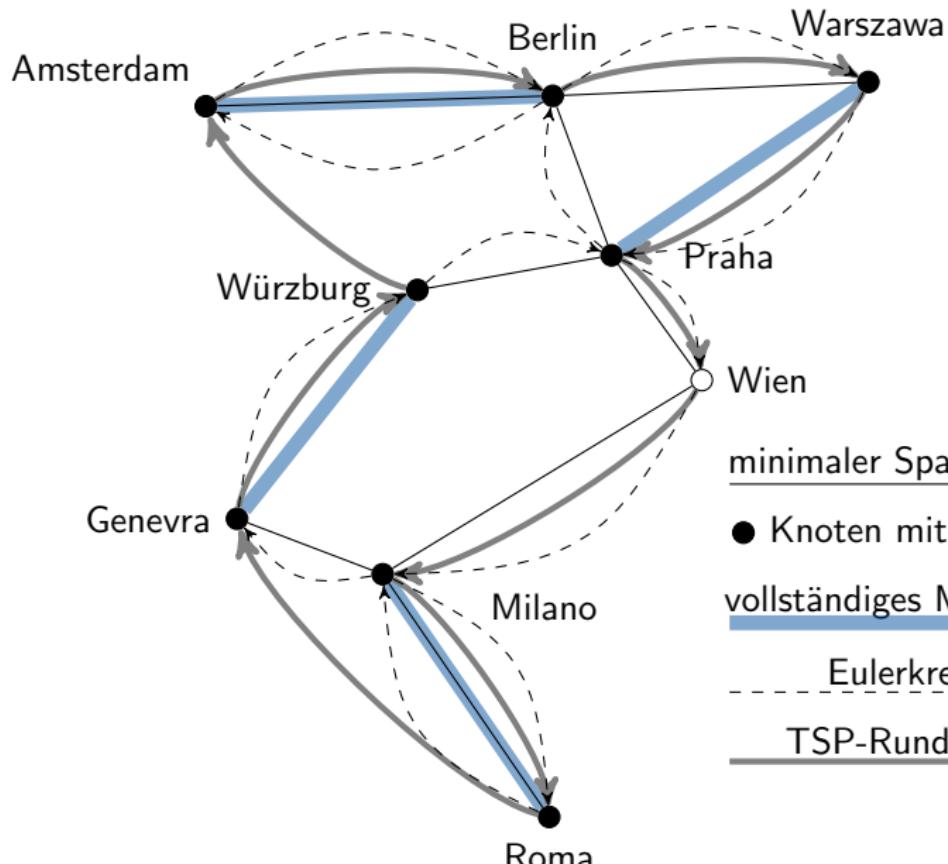
Eulerkreis →

TSP-Rundreise →

Christofides am Beispiel



Christofides am Beispiel



Das Partitionierungsproblem

Problem: Minimum Partition (MinPART)

Instanzen: $\langle a_1, \dots, a_n \rangle$, $a_i \in \mathbb{N}^+$

Lösungen: Mengen Y_1, Y_2 mit $Y_1 \cup Y_2 = \{1, \dots, n\}$

Maß: $\max\{\sum_{i \in Y_1} a_i, \sum_{i \in Y_2} a_i\}$

Ziel: min

Das Partitionierungsproblem

Problem: Minimum Partition (MinPART)

Instanzen: $\langle a_1, \dots, a_n \rangle$, $a_i \in \mathbb{N}^+$

Lösungen: Mengen Y_1, Y_2 mit $Y_1 \cup Y_2 = \{1, \dots, n\}$

Maß: $\max\{\sum_{i \in Y_1} a_i, \sum_{i \in Y_2} a_i\}$

Ziel: min

Satz 33

Für jede feste Zahl $r > 1$ liefert der Algorithmus r -PartAS bei Eingabe einer Instanz $x \in I_{\text{MinPART}}$, $x = \langle a_1, \dots, a_n \rangle$ eine Lösung mit Performanzrate $\leq r$.

Erfüllbarkeit aus Optimierungssicht

Problem: Maximum Satisfiability (MaxSAT)

Instanzen: aussagenlogische Formel F in KNF über Variablenmenge V

Lösungen: Belegung $f: V \rightarrow \{0, 1\}$

Maß: Anzahl der durch f erfüllten Klauseln

Ziel: max

Erfüllbarkeit aus Optimierungssicht

Problem: Maximum Satisfiability (MaxSAT)

Instanzen: aussagenlogische Formel F in KNF über Variablenmenge V

Lösungen: Belegung $f: V \rightarrow \{0, 1\}$

Maß: Anzahl der durch f erfüllten Klauseln

Ziel: max

Satz 34

GSAT ist ein 2-Approximationsalgorithmus für MaxSAT.

Geben wir dem Ganzen einen Namen: Optimierungs-Komplexitätsklassen

Definition (PTAS)

Sei \mathcal{P} ein Optimierungsproblem, sodass für jede Zahl $r > 1$ gilt, dass es einen Polynomialzeit-Approximationsalgorithmus für \mathcal{P} mit Performanzrate $\leq r$ gibt. Diese Eigenschaft nennen wir **polynomial time approximation scheme** und ordnen das Problem der Klasse PTAS zu.

Geben wir dem Ganzen einen Namen: Optimierungs-Komplexitätsklassen

Definition (PTAS)

Sei \mathcal{P} ein Optimierungsproblem, sodass für jede Zahl $r > 1$ gilt, dass es einen Polynomialzeit-Approximationsalgorithmus für \mathcal{P} mit Performanzrate $\leq r$ gibt. Diese Eigenschaft nennen wir **polynomial time approximation scheme** und ordnen das Problem der Klasse PTAS zu.

Definition (APX)

Die Klasse APX umfasst alle Optimierungsprobleme, für die es einen c -Approximationsalgorithmus für ein $c \geq 1, c \in \mathbb{Q}$ gibt.

Die Situation in der Optimierungswelt

