VL-17: Jenseits von P und NP

(Berechenbarkeit und Komplexität, WS 2018)

Gerhard Woeginger

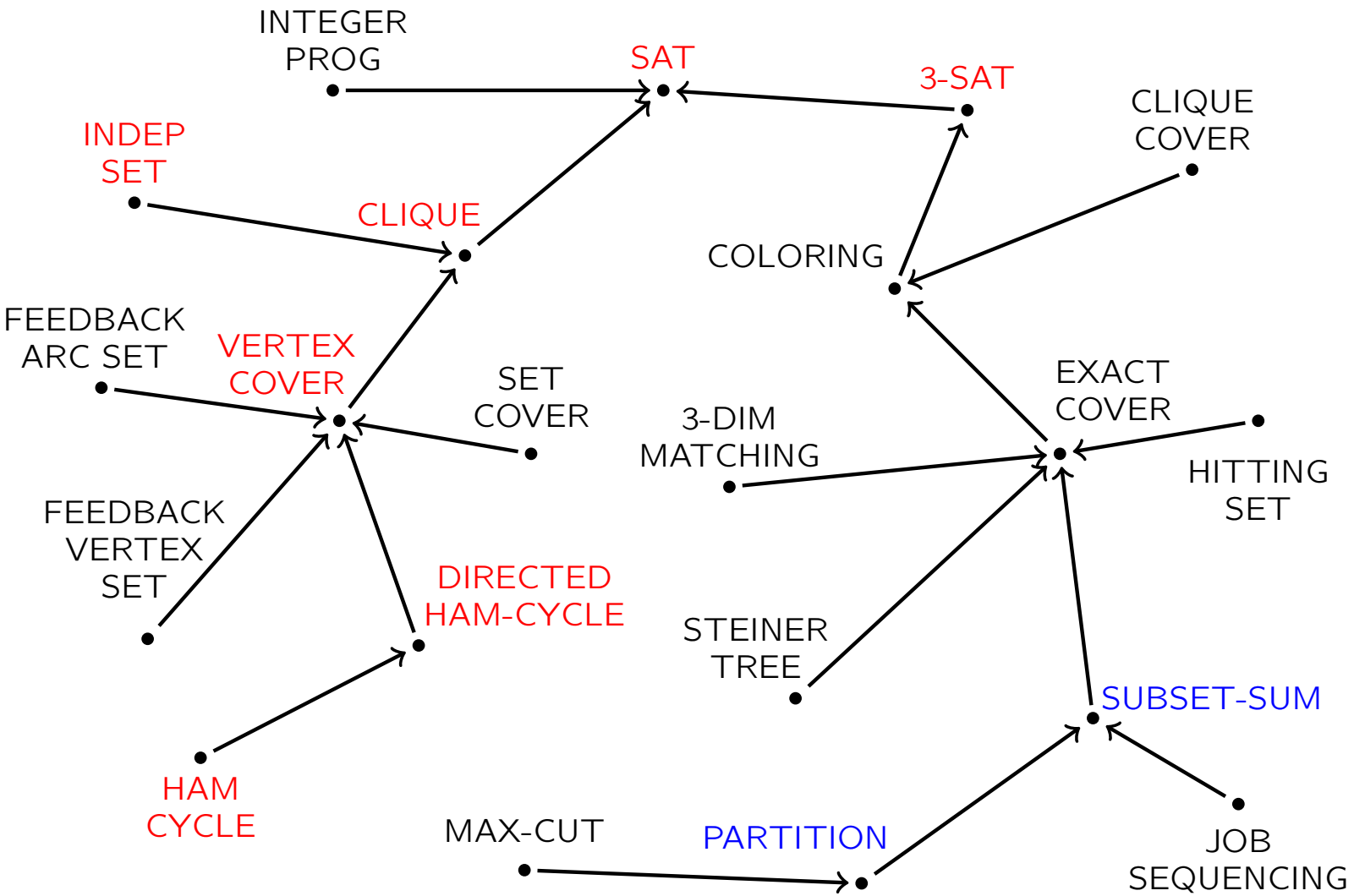
WS 2018, RWTH

Organisatorisches

- Nächste (letzte) Vorlesung:
  Donnerstag, Januar 24, 12:30–14:00 Uhr, Aula

- Webseite:
  http://algo.rwth-aachen.de/Lehre/WS1819/BuK.php
  ($\longrightarrow$ Arbeitsheft zur Berechenbarkeit)
  ($\longrightarrow$ Arbeitsheft zur NP-Vollständigkeit)

Wdh.: Landkarte mit Karp's 20 Reduktionen



Wdh.: SUBSET-SUM

Problem: SUBSET-SUM

Eingabe: Positive ganze Zahlen $a_1, \ldots, a_n$; eine ganze Zahl b

Frage: Existiert eine Indexmenge $I \subseteq \{1, \ldots, n\}$ mit $\sum_{i \in I} a_i = b$?

Satz

SUBSET-SUM ist NP-vollständig.

Wdh.: PARTITION, Rucksack, Bin Packing

Problem: PARTITION

Eingabe: Positive ganze Zahlen $a'_1, \ldots, a'_n$; mit $\sum_{i=1}^{n} a'_i = 2A'$

Frage: Existiert eine Indexmenge $I \subseteq \{1, \ldots, n\}$ mit $\sum_{i \in I} a'_i = A'$?

Satz

PARTITION ist NP-vollständig.

Satz

Rucksack ist NP-vollständig.
Bin Packing ist NP-vollständig.

Wdh.: Pseudo-polynomiell versus Stark NP-schwer (1)

Definition: Pseudo-polynomielle Zeit

Ein Algorithmus A löst ein Problem X in pseudo-polynomieller Zeit, falls die Laufzeit von A auf Instanzen I von X polynomiell in $|I|$ und $Number(I)$ beschränkt ist.

Satz

Die Probleme SUBSET-SUM, PARTITION und Rucksack sind pseudo-polynomiell lösbar.

Wdh.: Pseudo-polynomiell versus Stark NP-schwer (2)

Definition: Stark NP-schwer (engl.: NP-hard in the strong sense)

Ein Entscheidungsproblem X ist stark NP-schwer,
wenn es ein Polynom $q : \mathbb{N} \to \mathbb{N}$ gibt, sodass die Restriktion von X auf Instanzen I mit $Number(I) \leq q(|I|)$ NP-schwer ist.

Also: Das Problem X ist sogar dann NP-schwer, wenn alle Zahlenwerte in der Instanz I nur polynomiell gross (gemessen in $|I|$) sind.

Satz

Es sei X ein stark NP-schweres Entscheidungsproblem.
Falls X pseudo-polynomiell lösbar ist, so gilt P=NP.

Also: Pseudo-polynomiell und stark NP-schwer schliessen einander aus (unter unserer Standardannahme P≠NP)

Vorlesung VL-18
Jenseits von P und NP

- Was tun mit NP-schweren Problemen?

- Die Komplexitätsklasse coNP
- Zwischen P und NPC: NP-intermediate
- Die Komplexitätsklassen EXPTIME und PSPACE

Was tun mit NP-schweren Problemen?

Was tun mit NP-schweren Problemen?

- Viele Optimierungsprobleme aus der Praxis sind NP-schwer.
- Beispiel: Bin Packing (BPP), Set Cover, Rucksack, Travelling Salesman Problem (TSP).
- In der Praxis müssen diese Probleme dennoch behandelt werden.

Aus dem Buch von Garey und Johnson (1)

"I can't find an efficient algorithm, I guess I'm just too dumb."

Aus dem Buch von Garey und Johnson (2)

"I can't find an efficient algorithm, because no such algorithm is possible!"

Aus dem Buch von Garey und Johnson (3)

"I can't find an efficient algorithm, but neither can all these famous people."

Was tun mit NP-schweren Problemen? (1)

- Viele Optimierungsprobleme aus der Praxis sind NP-schwer.
- In der Praxis müssen diese Probleme dennoch behandelt werden.

Wichtige Strategien sind:

- Ausnutzen der Eingabestruktur durch spezielle Algorithmen
- Parametrisierte Algorithmen
- Approximationsalgorithmen
- Heuristiken (ohne irgendwelche Garantien)
- Zerteilen in geeignete Unterprobleme

Was tun mit NP-schweren Problemen? (2)

Beispiel: Ausnutzen der Eingabestruktur

- Strassennetze lassen sich durch planare Graphen modellieren
- Es gibt effiziente Graphalgorithmen, die planare Strukturen ausnützen

Beispiel: Approximationsalgorithmen

- Instanzen des TSP erfüllen in der Praxis häufig die Dreiecksungleichung (Δ-TSP)
- Für das Δ-TSP gibt es gute Approximationsalgorithmen:
  In polynomieller Zeit kann man eine Rundreise berechnen, deren Länge höchstens $3/2$-mal die optimale Länge beträgt.

Was tun mit NP-schweren Problemen? (3)

In der Praxis hängt die Effizienz eines Algorithmus oft nicht allein von der Eingabegrösse ab.
Eine verfeinerte Analyse, die andere Eingabeparameter berücksichtigt, kann zu besseren Resultaten führen.

Beispiel: Parametrisierte Algorithmen

Wir wollen eine Anfrage in einer Datenbank $\mathcal{D}$ auswerten.

- Es sei ℓ die Grösse der Anfrage und es sei m die Grösse der Datenbank $\mathcal{D}$. Die Eingabegrösse ist dann $n = \ell + m$.
- Häufig ist m sehr gross, während ℓ relativ klein ist. Ein Algorithmus mit Laufzeit $2^\ell m$ kann deswegen durchaus gut sein, ein Algorithmus mit Laufzeiten wie m^ℓ oder gar $2^m \ell$ hingegen kaum.
- Analysieren wir die Algorithmen nur nach der Eingabegrösse n, so wird dieser wichtige Unterschied nicht sichtbar.

Die Komplexitätsklasse coNP

Die Klasse coNP

Definition: Klasse NP (zur Erinnerung)

Ein Entscheidungsproblem $X \subseteq \Sigma^*$ liegt in NP,
wenn für jedes Wort $x \in X$ ein polynomiell langes Zertifikat y existiert,
das (zusammen mit x) in polynomieller Zeit verifiziert werden kann.

Definition: Klasse coNP

Ein Entscheidungsproblem $X \subseteq \Sigma^*$ liegt in coNP,
wenn für jedes Wort $x \notin X$ ein polynomiell langes Zertifikat y existiert,
das (zusammen mit x) in polynomieller Zeit verifiziert werden kann.

Intuition:

- Wenn X in NP, dann gibt es für JA-Instanzen $x \in X$ kurze und einfach zu überprüfende Beweise.
- Wenn X in coNP, dann gibt es für NEIN-Instanzen $x \notin X$ kurze und einfach zu überprüfende Widerlegungen.

Beispiel (1)

Problem: Non-Hamiltonkreis (Non-Ham-Cycle)

Eingabe: Ein ungerichteter Graph $G = (V, E)$

Frage: Besitzt G keinen Hamiltonkreis?

Frage: Wie sieht das coNP-Zertifikat für Non-Ham-Cycle aus?

Beispiel (2)

Problem: Unsatisfiability (UNSAT)

Eingabe: Eine Boole'sche Formel φ in CNF über der Boole'schen Variablenmenge $X = \{x_1, \ldots, x_n\}$

Frage: Gibt es keine Wahrheitsbelegung für X, die φ erfüllt?

Problem: TAUTOLOGY

Eingabe: Eine Boole'sche Formel φ in DNF über der Boole'schen Variablenmenge $X = \{x_1, \ldots, x_n\}$

Frage: Wird φ von allen Wahrheitsbelegungen von X erfüllt?

Frage: Wie sehen coNP-Zertifikate für UNSAT und TAUTOLOGY aus?

Beispiel (3a): Lineare Programmierung

Ein primales Lineares Programm (P):

$$\begin{array}{lrcll} \max & \sum_{j=1}^{n} c_j x_j & & & \\ \text{s.t.} & \sum_{j=1}^{n} a_{ij} x_j & \leq & b_i & \text{für } i = 1, \ldots, m \\ & x_j & \geq & 0 & \end{array}$$

Das entsprechende duale Lineare Programm (D):

$$\begin{array}{lrcll} \min & \sum_{i=1}^{m} b_i y_i & & & \\ \text{s.t.} & \sum_{i=1}^{m} a_{ij} y_i & \geq & c_j & \text{für } j = 1, \ldots, n \\ & y_i & \geq & 0 & \end{array}$$

Satz (Starker Dualitätssatz)

Wenn beide LPs zulässige Lösungen haben,
so haben beide den selben optimalen Zielfunktionswert.

Beispiel (3b): Lineare Programmierung

Primal= "$\max cx$ s.t. $Ax \leq b$" Dual= "$\min by$ s.t. $yA \geq c$"

Problem: Lineare Programmierung (LP)

Eingabe: Eine reelle $m \times n$ Matrix A; Vektoren $b \in \mathbb{R}^m$ und $c \in \mathbb{R}^n$; eine Schranke $\gamma \in \mathbb{R}$

Frage: Existiert ein Vektor $x \in \mathbb{R}^n$, der $Ax \leq b$ und $x \geq 0$ erfüllt, und dessen Zielfunktionswert $cx \geq \gamma$ ist?

Beobachtung

LP liegt in NP.

NP-Zertifikat = Vektor x fürs primale LP mit $cx \geq \gamma$

Beobachtung

LP liegt in coNP.

coNP-Zertifikat = Vektor y fürs duale LP mit $by < \gamma$

Beispiel (3c): Lineare Programmierung

Zusammenfassung

LP liegt in $\mathrm{NP} \cap \mathrm{coNP}$.

Anmerkung: Das war bereits in den 1950er Jahren bekannt.

Satz (Leonid Genrikhovich Khachiyan, 1979)

LP liegt in P.

Anmerkung: Khachiyan entwickelte die Ellipsoid-Methode für LP.

coNP-Vollständigkeit (1)

Definition: NP-vollständig (zur Erinnerung)

Ein Entscheidungsproblem X ist NP-vollständig,
wenn $X \in \text{NP}$ und alle $Y \in \text{NP}$ polynomiell auf X reduzierbar sind.

Definition: coNP-vollständig

Ein Entscheidungsproblem X ist coNP-vollständig,
wenn $X \in \text{coNP}$ und alle $Y \in \text{coNP}$ polynomiell auf X reduzierbar sind.

Intuition:

- X ist NP-vollständig,
  wenn es zu den schwierigsten Problemen in NP gehört.
- X ist coNP-vollständig,
  wenn es zu den schwierigsten Problemen in coNP gehört.

coNP-Vollständigkeit (2)

Satz

Non-Ham-Cycle, UNSAT und TAUTOLOGY sind coNP-vollständig.

Beweis: Übung.

Satz

Wenn das Entscheidungsproblem X NP-vollständig ist,
so ist das komplementäre Problem $\overline{X}$ coNP-vollständig.

Komplementäres Problem:

Ja-Instanzen von X werden zu Nein-Instanzen von $\overline{X}$, und
Nein-Instanzen von X werden zu Ja-Instanzen von $\overline{X}$

coNP versus NP und P (1)

Satz

$P \subseteq NP \cap coNP$

Beweis: $P = coP$

coNP versus NP und P (2)

Viele Informatiker denken, dass $\mathrm{NP} \neq \mathrm{coNP}$ gilt.

Satz

Wenn coNP ein NP-vollständiges Problem X enthält, dann $\mathrm{NP} = \mathrm{coNP}$.

Beweis:

- $X \in \mathrm{NPC}$ impliziert: $L \leq_p X$ für alle $L \in \mathrm{NP}$
- $X \in \mathrm{NPC}$ impliziert: $\overline{L} \leq_p \overline{X}$ für alle $L \in \mathrm{NP}$
- $X \in \mathrm{NPC}$ impliziert: $K \leq_p \overline{X}$ für alle $K \in \mathrm{coNP}$
- Ergo: Alle $K \in \mathrm{coNP}$ sind auf $\overline{X} \in \mathrm{NP}$ reduzierbar.
- Ergo: Alle $K \in \mathrm{coNP}$ liegen in NP. $\square$

Daraus erhalten wir das folgende Werkzeug:

"X NP-vollständig"	ist Evidenz für	"$X \notin \mathrm{coNP}$"
"X coNP-vollständig"	ist Evidenz für	"$X \notin \mathrm{NP}$"

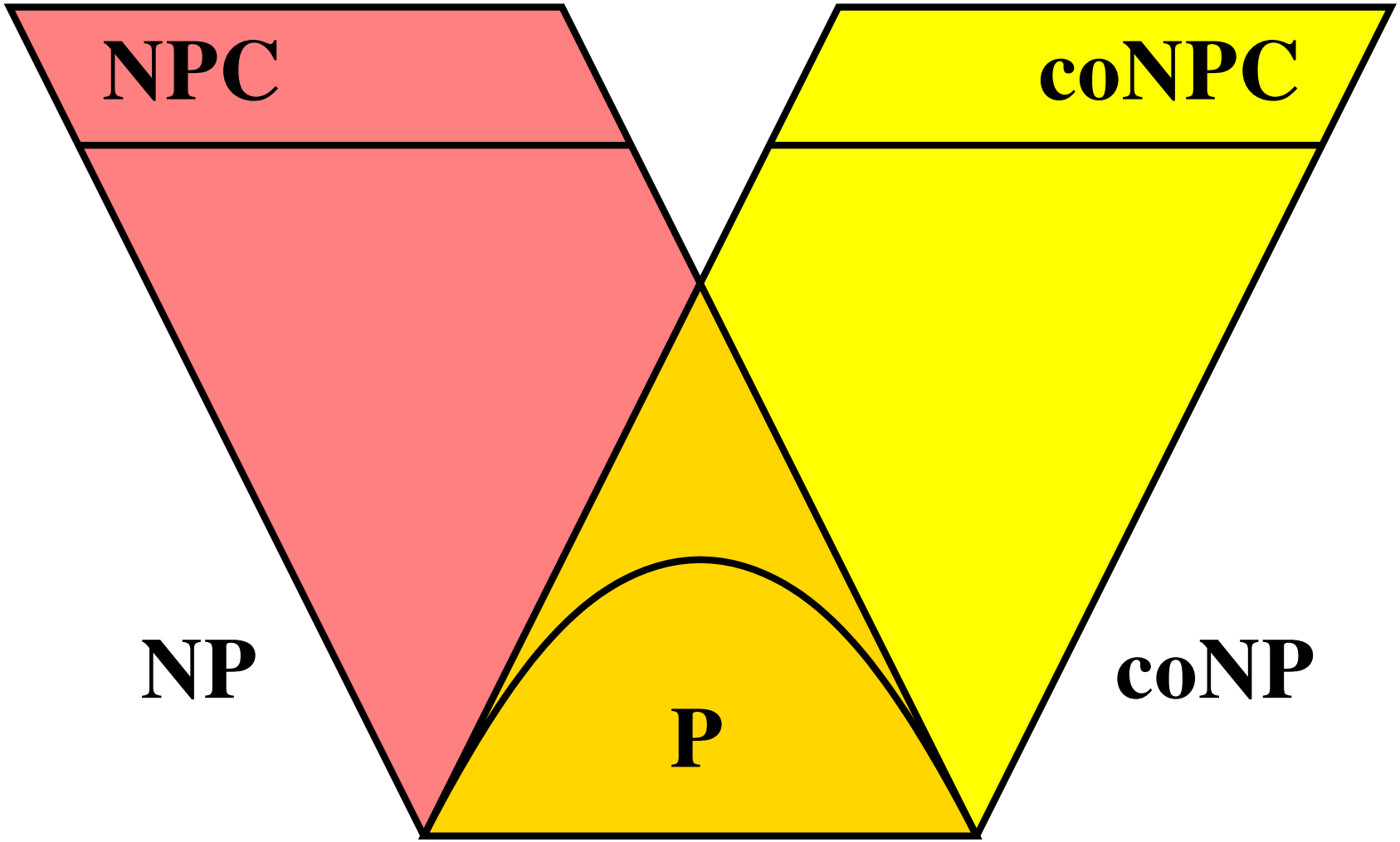
coNP versus NP und P (3)

Wir erhalten das folgende Werkzeug:

"X NP-vollständig"	ist Evidenz für	"$X \notin$ coNP"
"X coNP-vollständig"	ist Evidenz für	"$X \notin$ NP"

Ham-Cycle ist NP-vollständig.
Ham-Cycle hat gute Zertifikate für Ja-Instanzen.
Ergo: Ham-Cycle hat (höchstwahrscheinlich) keine guten Zertifikate für Nein-Instanzen.

SAT ist NP-vollständig.
SAT hat gute Zertifikate für Ja-Instanzen.
Ergo: SAT hat (höchstwahrscheinlich) keine guten Zertifikate für Nein-Instanzen.


NPC
coNPC
NP
P
coNP

coNP versus NP und P (4)

Viele Mathematiker denken, dass $\mathsf{NP} \cap \mathsf{coNP} = \mathsf{P}$ gilt.

Beispiel

LP in $\mathsf{NP} \cap \mathsf{coNP}$ war seit den 1950er Jahren bekannt.
Erst 1979 wurde ein polynomieller Algorithmus für LP gefunden.

Beispiel

PRIMES in $\mathsf{NP} \cap \mathsf{coNP}$ war seit den 1970er Jahren bekannt.
Erst 2002 wurde ein polynomieller Algorithmus für PRIMES gefunden.

Beispiel

PARITY-GAME in $\mathsf{NP} \cap \mathsf{coNP}$ ist seit den 1990er Jahren bekannt.
Ob allerdings PARITY-GAME in P liegt, ist ein offenes Problem.

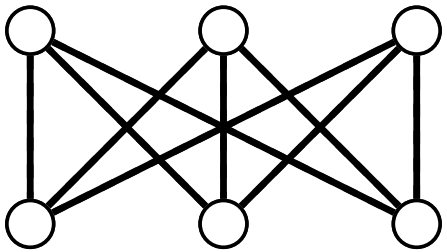
Zwischen P und NPC: NP-intermediate

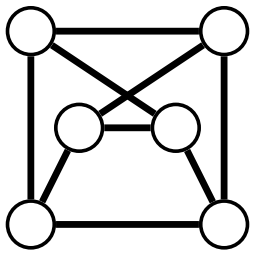
Das Graphisomorphieproblem (1)

Definition

Zwei Graphen $G_1 = (V_1, E_1)$ und $G_2 = (V_2, E_2)$ sind isomorph,
wenn es eine Bijektion $f : V_1 \to V_2$ gibt,
die Adjazenz und Nicht-Adjazenz erhält.

Eine solche Bijektion heisst Isomorphismus.





Das Graphisomorphieproblem (2)

Problem: GRAPH-ISOMORPHUS

Eingabe: Zwei ungerichtete Graphen G_1 und G_2
Frage: Gibt es einen Isomorphismus von G_1 nach G_2?

Satz

GRAPH-ISOMORPHUS liegt in NP.

Beweis: Verwende Isomorphismus als Zertifikat.

Das Graphisomorphieproblem (3)

Folgende Fragen sind derzeit noch ungelöst:

- Liegt GRAPH-ISOMORPHUS in P?
- Ist GRAPH-ISOMORPHUS NP-vollständig?
- Liegt GRAPH-ISOMORPHUS in coNP?

Das Graphisomorphieproblem (4)

Satz (László Babai, 2016)

GRAPH-ISOMORPHUS auf Graphen mit n Knoten
kann in $2^{p(\log n)}$ Zeit gelöst werden (wobei p ein Polynom ist).

Algorithmus verwendet algorithmische und strukturelle Theorie der Permutationsgruppen.

Exponential Time Hypothesis (ETH)

Es existiert eine reelle Zahl $\delta > 0$,
sodass kein Algorithmus 3-SAT in Zeit $O(2^{\delta n})$ löst.

- ETH wurde in den letzten 20 Jahren populär (ist aber unbewiesen!)
- ETH impliziert $P \neq NP$
- Wenn GRAPH-ISOMORPHUS NP-vollständig ist, dann ist ETH falsch

NP-intermediate (1)

Definition

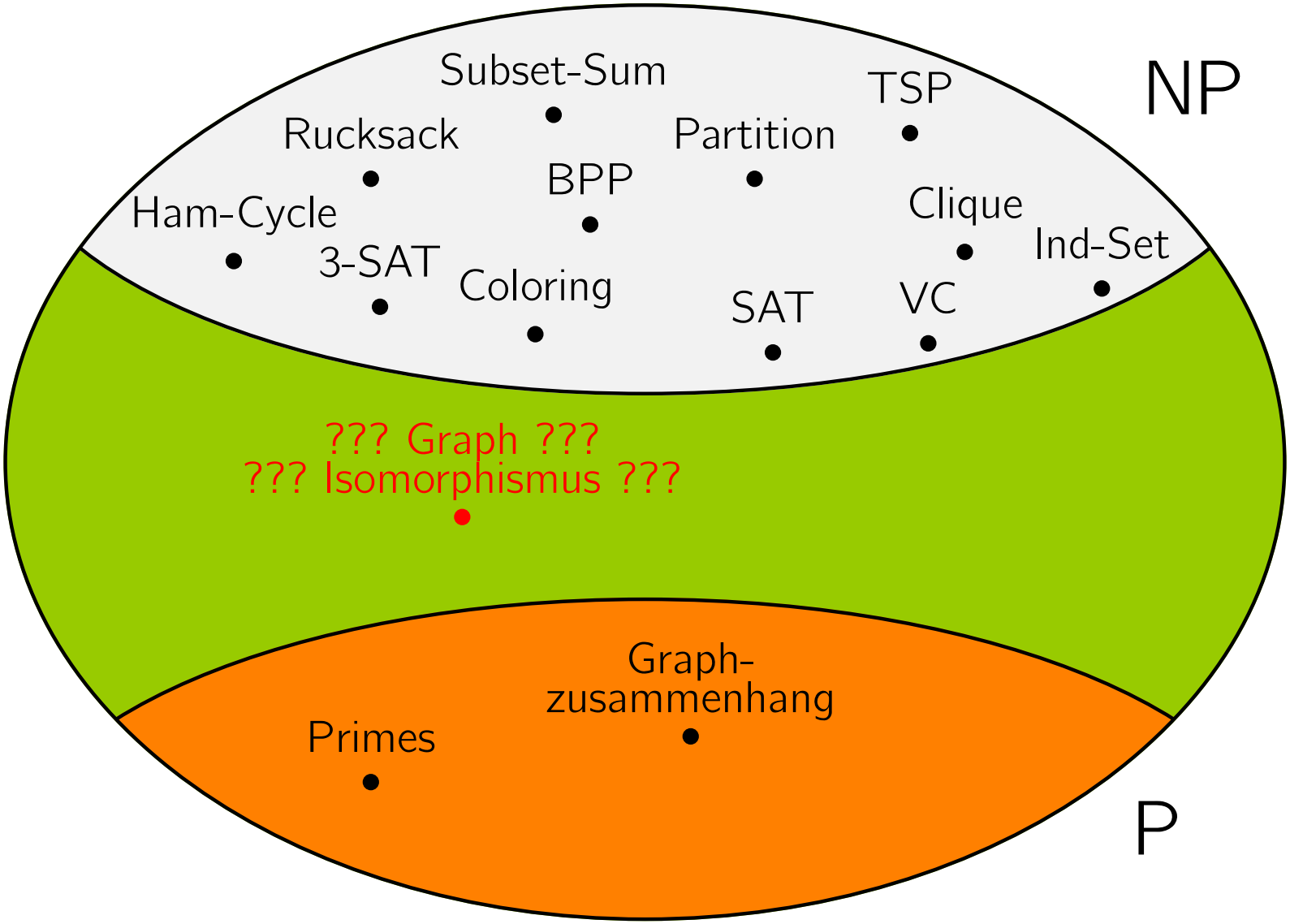
Ein Entscheidungsproblem $X \subseteq \Sigma^*$ heisst NP-intermediate,
wenn $L \in NP$ und wenn sowohl $L \notin P$ als auch $L \notin NPC$ gilt.

Satz von Ladner (ohne Beweis)

Wenn $P \neq NP$ gilt,
dann existieren Probleme, die NP-intermediate sind.

Viele Informatiker denken,
dass das GRAPH-ISOMORPHUS Problem NP-intermediate ist.

NP-intermediate (2)



Warnung: Dieser Abbildung liegt die Annahme $P \neq NP$ zu Grunde.

Die Komplexitätsklassen PSPACE und EXPTIME

Die Klasse PSPACE (1)

Definition: Komplexitätsklasse PSPACE

PSPACE ist die Klasse aller Entscheidungsprobleme,
die durch eine DTM M entschieden werden, deren Worst Case Speicherplatzbedarf durch $q(n)$ mit einem Polynom q beschränkt ist.

Definition: Komplexitätsklasse NPSPACE

NPSPACE ist die Klasse aller Entscheidungsprobleme,
die durch eine NTM M entschieden werden, deren Worst Case Speicherplatzbedarf durch $q(n)$ mit einem Polynom q beschränkt ist.

Die Klasse PSPACE (2)

Satz von Savitch (ohne Beweis)

$\mathrm{PSPACE} = \mathrm{NPSPACE}$

Wie verhält sich PSPACE zu NP?

Da sich der Kopf einer Turingmaschine in einem Schritt nur um eine Position bewegen kann gilt: $\mathrm{NP} \subseteq \mathrm{NPSPACE} = \mathrm{PSPACE}$

Die Klasse PSPACE (3)

Problem: QUANTIFIED-SAT (Q-SAT)

Eingabe: Eine Boole'sche Formel φ in CNF über der Boole'schen Variablenmenge $\{x_1, \ldots, x_n, y_1, \ldots, y_n\}$

Frage: $\exists x_1 \, \forall y_1 \, \exists x_2 \, \forall y_2 \, \exists x_3 \, \forall y_3 \; \cdots\cdots \; \exists x_n \, \forall y_n \;\; \varphi$

Satz

Q-SAT liegt in PSPACE.

Anmerkung: Q-SAT ist PSPACE-vollständig.

Die Klasse EXPTIME (1)

Definition: Komplexitätsklasse EXPTIME

EXPTIME ist die Klasse aller Entscheidungsprobleme, die durch eine DTM M entschieden werden, deren Worst Case Laufzeit durch $2^{q(n)}$ mit einem Polynom q beschränkt ist.

Laufzeit-Beispiele: $2^{\sqrt{n}}$, 2^n, 3^n, $n!$, n^n. Aber nicht: 2^{2^n}

Die Klasse EXPTIME (2)

Wie verhält sich EXPTIME zu PSPACE?

- Bei einer Speicherplatzbeschränkung $s(n)$ gibt es nur $2^{O(s(n))}$ viele verschiedenen Konfigurationen für eine Turingmaschine.
  Daher ist die Rechenzeit durch $2^{O(s(n))}$ beschränkt.
- Die Probleme in PSPACE können deshalb in Zeit $2^{p(n)}$ gelöst werden: $\text{PSPACE} \subseteq \text{EXPTIME}$

Die Klasse EXPTIME (3)

Problem: k-Schritt-HALTEPROBLEM

Eingabe: Eine deterministische Turingmachine M; eine ganze Zahl k
Frage: Wenn M mit leerem Band gestartet wird, hält M dann nach höchstens k Schritten an?

Die Zahl k ist binär (oder dezimal) kodiert.

Satz

Das k-Schritt-HALTEPROBLEM liegt in EXPTIME.

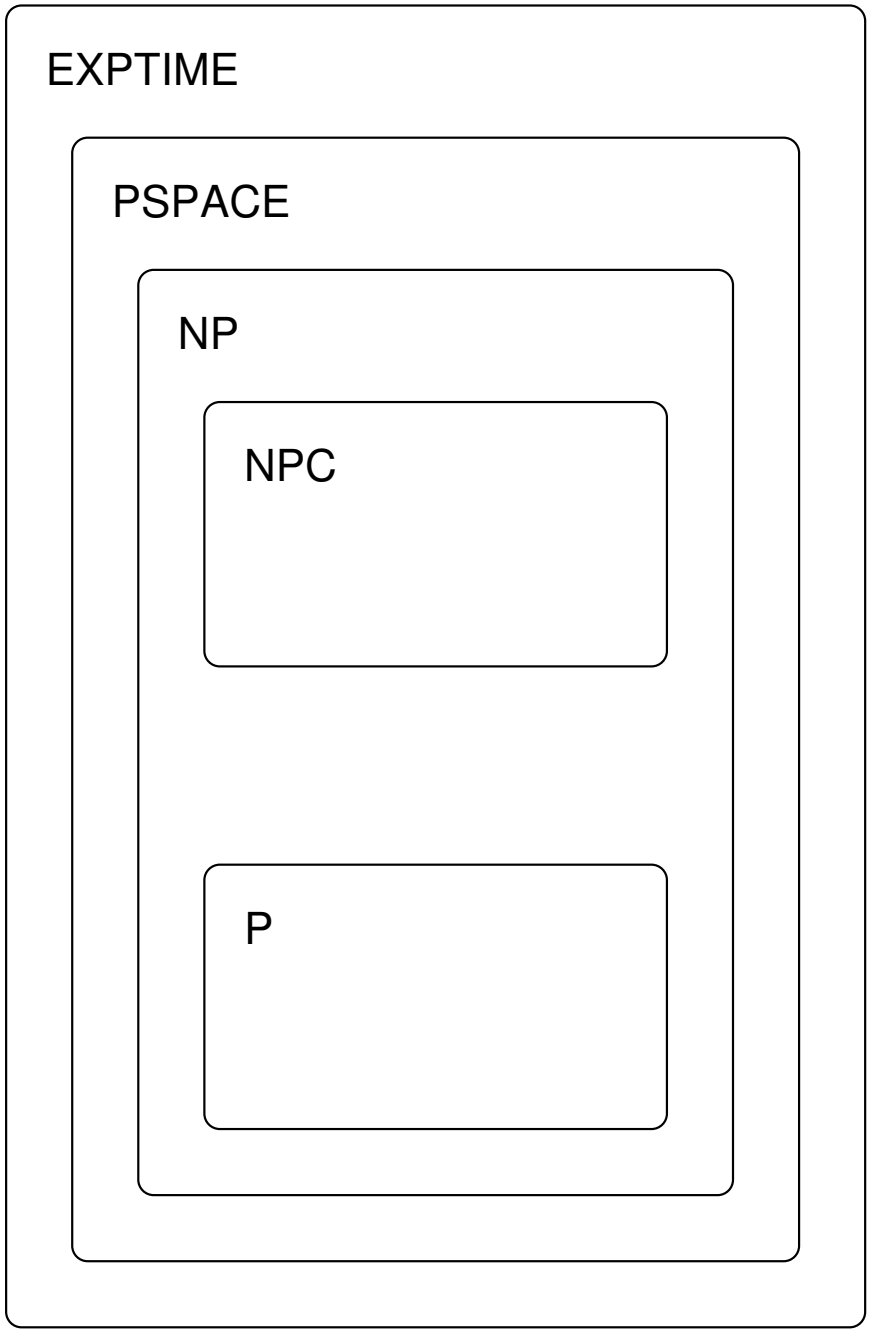
Anmerkung: Das k-Schritt-HALTEPROBLEM ist EXPTIME-vollständig.

PSPACE und EXPTIME

Wir haben gezeigt:

$$\mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PSPACE} \subseteq \mathrm{EXPTIME}$$

- Es ist nicht bekannt, welche dieser Inklusionen strikt sind.
- Möglicherweise gilt $\mathrm{P} = \mathrm{PSPACE}$ oder $\mathrm{NP} = \mathrm{EXPTIME}$.
- Wir wissen allerdings, dass $\mathrm{P} \neq \mathrm{EXPTIME}$ gilt.
  (Das folgt aus dem so-genannten *Zeithierarchiesatz*.)


EXPTIME
PSPACE
NP
NPC
P