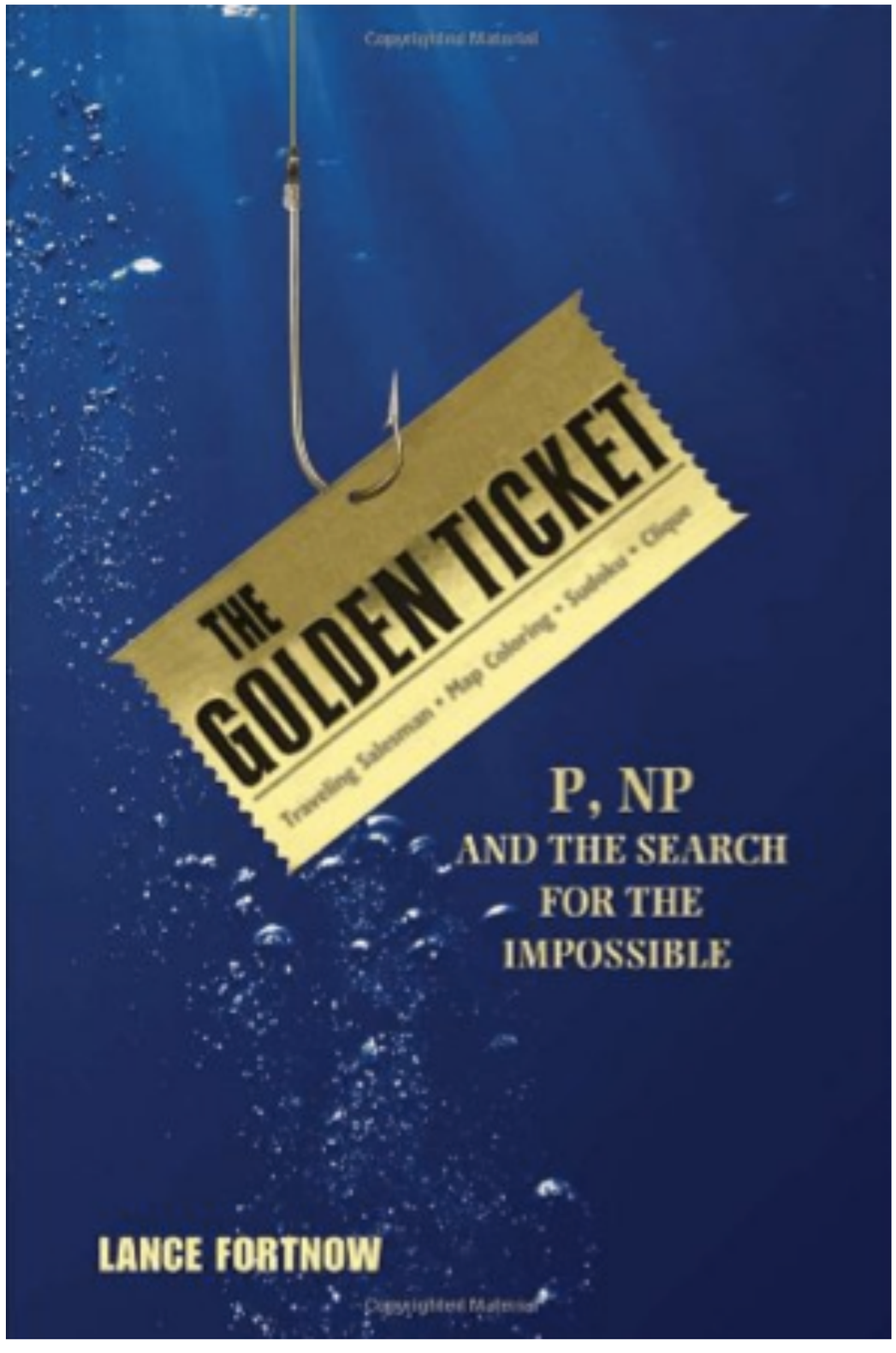
Komplexitätstheorie

Kapitel 3: P vs NP

Wir definieren die wichtigen Komplexitätsklassen P und NP und studieren deren Zusammenhang:

- Definitionen der Klassen, polynomielle Beweissysteme
- Zusammenhang zwischen NP und Nichtdeterminismus
- NP-Vollständigkeit
- Beispiele für NP-vollständige Probleme
- Komplemente und co-NP
- Isomorphie und spärliche Mengen
- Ladner’s Theorem und Constraint Satisfaction Probleme

Einleitung


Copyrighted Material
THE GOLDEN TICKET
Traveling Salesman • Map Coloring • Sudoku • Clique
P, NP
AND THE SEARCH FOR THE IMPOSSIBLE
LANCE FORTNOW
Copyrighted Material

Kapitel 3

Definition von P und NP

Die Klasse P

Traditionelle Sicht:

Problem ist "effizient lösbar" gdw es in polynomieller Zeit lösbar ist

(Polynom von beliebigem Grad!)

Entsprechende Komplexitätsklasse:

$$\mathsf{P} := \bigcup_{i \geq 1} \mathsf{DTime}(n^i) \qquad \text{(oft auch ”PTIME”)}$$

Probleme in P nennt man oft auch tractable

Die Klasse P

Warum ist P eine ”gute” Komplexitätsklasse?

- Zeitkomplexität n^k mit sehr großem k ist auch bedenklich, aber: Polynome mit großem Grad sind fast nie notwendig!
- Nach erweiterter Church-Turing These ist die Klasse P für alle Berechnungsmodelle identisch (z.B. ein vs mehrere Bänder bei TMs)
- P ist abgeschlossen unter den üblichen Kompositionsoperationen auf Algorithmen

Die letzten beiden Eigenschaften werden z.B. von DTime(n) nicht erfüllt.

Die Klasse NP

Sehr viele Probleme in der Informatik haben folgende Charakteristik:

Für alle $w \in \Sigma^*$ gilt:

- wenn $w \in L$ (w ist "Ja"-Instanz), dann gibt es einen einfach zu verifizierenden Beweis dafür, der von kurzer Länge ist (polynomiell in der Länge von w)
- wenn $w \notin L$ (w ist "Nein"-Instanz), dann gibt es keinen solchen Beweis.

Diese Beobachtung ist die Grundlage für die Definition der Komplexitätsklasse NP.

Beispiel 1: Cliquenproblem

Der "Beweis" ist die Clique selbst

- wenn $(G, k) \in$ CLIQUE, dann gibt es Knotenmenge $\{v_1, \ldots, v_k\}$, die Clique in G ist;
- wenn $(G, k) \notin$ CLIQUE, dann gibt es keine solche Menge.

Der Beweis ist einfach zu verifizieren: man kann offensichtlich in polynomieller Zeit entscheiden, ob gegebene Knotenmenge Clique ist.

Der Beweis ist kurz: nicht größer als der Graph.

Beispiel 2: Rucksackproblem

Definition Rucksackproblem

Sei $M = \{a_1, \ldots, a_k\}$ eine Menge von *Gegenständen*, wobei

Gegenstand a_i Gewicht g_i hat und Nutzen n_i.

Sei $G \geq 0$ eine Gewichtsgrenze und N ein intendierter Nutzen.

Lösung für Rucksackproblem (M, G, N) ist Teilmenge $R \subseteq M$ so dass

1. $\sum_{a_i \in R} g_i \leq G$ und
2. $\sum_{a_i \in R} n_i \geq N$.

RP ist die Menge aller Instanzen (M, G, N), für die Lösung existiert.

Beweis für “ja”-Instanz (M, G, N) ist Teilmenge $R \subseteq M$, die 1. + 2. erfüllt

Offenbar nicht länger als Eingabe und leicht zu verifizieren!

Beispiel 3: Integer Programming

Definition Integer Programming

Ein lineares Gleichungssytem G ist eine Menge von Gleichungen der Form

$$c_1 \cdot x_1 + \cdots + c_n \cdot x_n = \alpha$$

wobei $x_1, \ldots, x_n$ aus Variablenmenge V, $c_1, \ldots, c_n \in \mathbb{N}$ und $\alpha \in V \cup \mathbb{N}$.

Lösung ist Abbildung $\tau : V \to \mathbb{N}$, so dass alle Gleichungen erfüllt sind.

IPROG ist Menge aller Gleichunggsysteme G, für die Lösung existiert.

Gegeben einen Lösungskandidaten kann man offensichtlich in polynomieller Zeit überprüfen, ob er Lösung ist (Addition).

Es ist aber nicht offensichtlich, dass es immer kurze Lösungen/ Beweise für IPROG gibt ($\mathbb{N}$ hat Elemente unbeschränkter Größe)

Beispiel 3: Integer Programming

Ohne Beweis:

Theorem (Papadimitriou)

Wenn G Gleichungssystem mit m Gleichungen, n Variablen und Konstanten beschränkt durch a, dann gibt es Lösung gdw. es eine Lösung aus dem Bereich $\{0, \ldots, n(ma)^{2m+1}\}$.

Wir verwenden als "kleine Beweise":

- Lösungen, deren Zahlenwerte wie im o.g. Theorem beschränkt sind
- Zahlenwerte sind exponentiell in der Größe von G, also ist deren binäre Repräsentation nur polynomiell groß ($\log(2^n)$ ist polynomiell!)

Die Klasse NP

Es gibt viele tausend solcher Probleme

Wir setzen:

"einfach zu verifizieren": in polynomieller Zeit

"kurzer Beweis": Länge des Beweises polynomiell in Länge der Instanz

Daraus ergibt sich die Definition der Klasse NP.

Die Klasse NP

Definition Komplexitätsklasse NP

Sei $L \subseteq \Sigma^*$. Relation $R \subseteq \Sigma^* \times \Gamma^*$ ist *Beweissystem* für L wenn

- $(w, b) \in R$ impliziert $w \in L$ und (Korrektheit)
- $w \in L$ impliziert $(w, b) \in R$ für ein $b \in \Gamma^*$ (Vollständigkeit)
  so ein b heißt *Beweis* oder *Zeuge* für w

R ist *polynomiell* wenn

- es gibt Polynom p so dass $|b| \leq p(|w|)$ für alle $(w, b) \in R$
- $R \in$ P (also: gegeben $w \in \Sigma^*$ und $b \in \Gamma^*$ kann DTM in polynomieller Zeit entscheiden, ob $(w, b) \in R$)

NP ist Klasse aller Probleme L, für die es polynomielles Beweissystem gibt.

Die Beispiele haben gezeigt:

CLIQUE, RP, IPROG sind in NP.

Kapitel 3

P vs NP

P vs NP

Theorem

$\mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{ExpTime}$, wobei $\mathsf{ExpTime} := \bigcup_{i \geq 1} \mathsf{DTIME}(2^{\mathcal{O}(n^i)})$.

Sehr unbefriedigend:

Für viele wichtige Probleme in NP (z.B. CLIQUE, RP, IPROG) ist unbekannt, ob sie in P (also effizient lösbar) sind!

Es ist sogar unbekannt, ob P ≠ NP (und NP ≠ ExpTime)

P vs NP

Intuitive Formulierung der P vs NP Frage:

Ist das Finden eines Beweises schwieriger als das Überprüfen?

Unsere Intuition sagt "Ja", also P ≠ NP!

In der Tat glaubt eigentlich niemand, das P = NP, aber ein Beweis konnte seit 50 Jahren nicht geführt werden!

Die P=NP Frage:

- ist eines der wichtigsten offenen Probleme der Informatik
- Lösung wird vom Clay Mathematics Institute mit US$1.000.000 belohnt (Millenium Prize)

P vs NP

Konsequenzen eines Beweises von

- P $=$ NP: dramatisch. Tausende bisher als sehr schwierig eingestufter Probleme wären dann wohl effizient lösbar (z.B. CLIQUE, SEQ, IPROG)
- P $\neq$ NP: weniger dramatisch, aber wichtig zu wissen möglicherweise interessante Konsequenzen in der Kryptographie
  (dort: schwierige Probleme nützlich statt ärgerlich)

Kapitel 3

NP und nicht-deterministische TMs

Die Klasse NP

Die Klasse NP hat viele äquivalente Charakterisierungen:

- polynomielle Beweissysteme
- nicht-deterministische Turingmaschinen
- existentielle Logik zweiter Stufe (Fagins Theorem)
- randomisierte Beweissysteme (PCP Theorem)
- etc.

Nicht-deterministische TMs

Nicht-deterministische Turingmaschinen (kurz: NTMs) generalisieren DTMs:

- Statt Übergangs*funktion* δ haben sie Übergangs*relation*

$$\Delta \subseteq (Q \setminus \{q_{\text{acc}}, q_{\text{rej}}\}) \times \Gamma \times Q \times \Gamma \times \{L, R, N\}$$

- (q, a, q', a', L) heißt: wenn M in q ist und a liest, so kann sie nach q' gehen, a' schreiben und sich nach links bewegen
- *Relation*: NTM hat u.U. mehrere Möglichkeiten für nächsten Schritt
- Konfigurationen / Berechnungen definiert wie für DTMs
- Es gibt jetzt mehrere Berechnungen auf derselben Eingabe
- NTM *akzeptiert* Eingabe w wenn mind. eine Berechnung auf w akzeptierend

Nicht-deterministische TMs

Definition NTime

Für NTM M und $w \in \Sigma^*$ schreiben wir $\text{time}_M(w) = n$ wenn alle Berechnungen von M auf w höchstens n Schritte umfassen.

Sei $t : \mathbb{N} \to \mathbb{N}$ monoton wachsende Funktion mit $t(n) \geq n$.
M ist t-*zeitbeschränkt* wenn $\text{time}_M(w) \leq t(n)$ für alle w der Länge n

Zeitkomplexitätsklasse $\text{NTime}_i(t)$ ist definiert als Menge der Probleme

$$\{L \subseteq \Sigma^* \mid \exists \mathcal{O}(t)\text{-zeitbeschr. } i\text{-Band NTM } M \text{ mit } L(M) = L\}$$

Wir setzen $\text{NTime}(t) := \bigcup_{i \geq 1} \text{NTime}_i(t)$

Theorem (NTM Charakterisierung von NP)

$$\text{NP} = \bigcup_{i \geq 1} \text{NTime}(n^i)$$

Nicht-deterministische TMs

Klassischerweise ist diese Charakterisierung sogar die eigentliche Definition von NP

Die P vs NP Frage kann also auch wie folgt verstanden werden:

Kann eine nicht-deterministische Maschine in polynomieller Zeit mehr erreichen als eine deterministische Maschine?

Kapitel 3

NP-Härte, NP-Vollständigkeit

NP-Härte

Bisher ist es von sehr vielen “typischen” NP-Problemen nicht gelungen, zu zeigen, ob sie in P (effizient lösbar) sind oder nicht.

Der Begriff der NP-Härte liefert ein Mittel, solche Probleme trotzdem als schwierig klassifizieren zu können.

Starke Indikation dafür, dass Problem L nicht in P ist:

1. L ist "mindestens so schwierig" wie alle anderen Probleme in NP
2. daraus folgt, dass L nur in P ist wenn P=NP (also unwahrscheinlich!)

Reduktionen

Reduktionen sind zentral in der Komplexitätstheorie:

- Werkzeug zum Vergleich verschiedener Probleme, nicht nur im Kontext von NP
- existieren in zahllosen Variationen (für verschiedene Zwecke)

Definition (Polynomielle) Reduktion

Sei $L \subseteq \Sigma^*$, $L' \subseteq \Gamma^*$. Eine *Reduktion* von L auf L' ist berechenbare Funktion $f : \Sigma^* \to \Gamma^*$ so dass

$$w \in L \text{ gdw. } f(w) \in L' \text{ für alle } w \in \Sigma^*.$$

f heisst *polynomiell* wenn f in polynomieller Zeit berechenbar.

Wir schreiben $L \leq_p L'$ wenn polynomielle Reduktion von L auf L' existiert.

Reduktionen

Intuitiv: $L \leq_p L'$ formalisiert "L' ist mindestens so schwer wie L"

Denn: Algorithmus für L' kann mit nur polynomiellem Mehraufwand zum Lösen von L verwendet werden!

Theorem

1. P ist abgeschlossen unter polynomiellen Reduktionen:
   $L' \in$ P und $L \leq_p L'$ impliziert $L \in$ P;
2. NP ist abgeschlossen unter polynomiellen Reduktionen:
   $L' \in$ NP und $L \leq_p L'$ impliziert $L \in$ NP;
3. "Polynomiell reduzierbar" ist transitive Relation:
   $L_1 \leq_p L_2$ und $L_2 \leq_p L_3$ impliziert $L_1 \leq_p L_3$.

NP-Härte

Definition NP-Härte, NP-Vollständigkeit

Problem L ist

- *NP-hart* wenn $L' \leq_p L$ für alle $L' \in$ NP;
- *NP-vollständig* wenn L sowohl NP-hart als auch in NP.

Ein NP-hartes Problem ist also mindestens so schwer wie jedes andere Problem in NP.

Beobachtung

Wenn L NP-hart und $L \in P$, dann P $=$ NP.

Solange das P = NP Problem ungelöst ist, wird NP-härte deshalb als "nicht effizient lösbar" gewertet.

Kapitel 3

Cook's Theorem

NP-Härte

Zwei Ansätze, NP-Härte von Problem L zu zeigen:

1. Zeigen, dass $L' \leq_p L$ für alle $L' \in$ NP
2. Zeigen, dass $L' \leq_p L$ für ein NP-hartes Problem L'

Lemma

Wenn L NP-hart ist und $L \leq_p L'$, dann ist L' NP-hart.

SAT

Definition Formel, Wertzuweisung, Erfüllbar

Sei AV unendlich abzählbare Menge von *Aussagenvariablen*.
Menge der *aussagenlogischen Formeln* (kurz: AL-Formeln) ist kleinste Menge so dass:

- jedes $p \in$ AV ist AL-Formel
- wenn φ, ψ AL-Formeln, so auch $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$

Wertzuweisung (kurz: WZ) ist Abbildung π : AV $\to \{0, 1\}$.
WZ π wird wie folgt auf zusammengesetzte AL-Formeln erweitert:

- $\pi(\neg\varphi) = 1 - \pi(\varphi)$;
- $\pi(\varphi \wedge \psi) = \min\{\pi(\varphi), \pi(\psi)\}$;
- $\pi(\varphi \vee \psi) = \max\{\pi(\varphi), \pi(\psi)\}$.

AL-Formel φ ist erfüllbar, wenn es WZ π gibt mit $\pi(\varphi) = 1$.

Cook's Theorem

Problem SAT ist Menge aller erfüllbaren AL-Formeln.

Theorem (Cook/Levin aka "Cooks Theorem")

SAT ist NP-vollständig.

In NP: $\{(\varphi, \pi) \mid \pi$ WZ für Variablen in φ, die φ erfüllt $\}$

ist polynomielles Beweissystem

NP-Härte:

Zeigen, dass Wortproblem jeder poly-zeitbeschränkten NTM polynomiell reduziert werden kann.

Cook's Theorem

Sei $L \in$ NP und M eine $p(n)$-zeitbeschränkte NTM mit $L(M) = L$.

Ziel: gegeben w, finden von AL-Formel φ_w so dass

1. M akzeptiert w gdw. φ_w erfüllbar und
2. φ_w in Zeit polynomiell in $|w|$ konstruiert werden kann

Akzeptierende Berechnung von M auf $w = a_1 \cdots a_n$ ist Matrix:

$\triangleright$	q_0, a_1	a_2	$\cdots$	a_n	$\bot$	$\cdots$	$\bot$
$\triangleright$	b	q, a_2	$\cdots$	a_n	$\bot$	$\cdots$	$\bot$
$\triangleright$	b	q', b'	$\cdots$	a_n	$\bot$	$\cdots$	$\bot$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

$0..p(n)$

$0..p(n)+1$

Cook's Theorem

Repräsentation der Matrix mittels Variablen:

- $B_{a,i,t}$: zum Zeitpunkt t ist Zelle i mit a beschriftet;
- $K_{i,t}$: zum Zeitpunkt t ist der Kopf über Zelle i;
- $Z_{q,t}$: zum Zeitpunkt t ist q der aktuelle Zustand

für alle $t \leq p(n) + 1$, $i \leq p(n)$, $a \in \Gamma$, $q \in Q$.

Gesuchte Formel φ_w ist Konjunktion folgender Formeln:

Berechnung beginnt mit Startkonfiguration für $w = a_1 \cdots a_n$:

$$\psi_{\mathsf{ini}} := Z_{q_0,0} \wedge K_{1,0} \wedge B_{\triangleright,0,0} \wedge \bigwedge_{1 \leq i \leq n} B_{a_i,i,0} \wedge \bigwedge_{n < i \leq p(n)} B_{\perp,i,0}.$$

Cook's Theorem

Sei $R(i) = i + 1, \quad N(i) = i, \quad L(i) = i - 1$ wenn $i > 1, \quad L(0) = 0$.

Schritte folgen der Übergangsrelation:

$$\psi_{\mathsf{move}} := \bigwedge_{t<p(n),q\in Q\setminus\{q_{\mathsf{acc}},q_{\mathsf{rej}}\},a\in\Gamma,i\leq p(n)+1} \Big((Z_{q,t} \wedge K_{i,t} \wedge B_{a,i,t}) \rightarrow \bigvee_{(q,a,q',a',M)\in\Delta,M(i)\leq p(n)+1} (Z_{q',t+1} \wedge K_{M(i),t+1} \wedge B_{a',i,t+1}) \Big)$$

Zellen ohne Kopf ändern sich nicht:

$$\psi_{\mathsf{keep}} := \bigwedge_{t<p(n),i\leq p(n)+1,a\in\Gamma} \big(\neg K_{i,t} \wedge B_{a,i,t} \rightarrow B_{a,i,t+1} \big)$$

Eingabe wird akzeptiert:

$$\psi_{\mathsf{acc}} := \bigvee_{t\leq p(n)} \big(Z_{q_{\mathsf{acc}},t} \wedge \bigwedge_{t'<t} \neg Z_{q_{\mathsf{rej}},t'} \big)$$

Cook's Theorem

Bandbeschriftung, Kopfposition, Zustand sind eindeutig und definiert:

$$\psi_{\mathsf{aux}} := \bigwedge_{t,q,q',q\neq q'} \neg(Z_{q,t} \wedge Z_{q',t}) \wedge \bigwedge_{t,i,a,a',a\neq a'} \neg(B_{a,i,t} \wedge B_{a',i,t}) \wedge \bigwedge_{t,i,j,i\neq j} \neg(K_{i,t} \wedge K_{j,t})$$

$$\bigwedge_{t\leq p(n)} \bigvee_{q\in Q} Z_{q,t} \wedge \bigwedge_{t\leq p(n)} \bigvee_{i\leq p(n)+1} K_{i,t} \wedge \bigwedge_{t\leq p(n),i\leq p(n)+1} \bigvee_{a\in\Gamma} B_{a,i,t}$$

Wir setzen nun

$$\varphi_w := \psi_{\mathsf{ini}} \wedge \psi_{\mathsf{move}} \wedge \psi_{\mathsf{keep}} \wedge \psi_{\mathsf{acc}} \wedge \psi_{\mathsf{aux}}.$$

Leicht zu sehen:

φ hat Länge $\mathcal{O}(p(n)^2)$, kann in Zeit $\mathcal{O}(p(n)^2)$ generiert werden.

Lemma

φ_w erfüllbar gdw. M akzeptiert w

Kapitel 3

Weitere NP-vollständige Probleme

3SAT

Oft ist es jedoch bequemer, nur Formeln in bestimmter Form zu betrachten.

Definition 3SAT

3-Formel hat die Form

$$\bigwedge_i (\ell_{1,i} \vee \ell_{2,i} \vee \ell_{3,i})$$

wobei die $\ell_{j,i}$ *Literale* sind, also Variable oder deren Negation.

Jede Disjunktion $(\ell_{1,i} \vee \ell_{2,i} \vee \ell_{3,i})$ heisst *Klausel*.

3SAT ist die Menge aller erfüllbaren 3-Formeln.

Bekannt aus Theoretische Informatik 2:

3SAT ist NP-vollständig.

Hier nur kurze Erinnerung.

3SAT

Theorem

3SAT ist NP-vollständig.

Beweis Härte: polynomielle Reduktion von SAT

Gegeben AL-Formel φ, konstruiere in polynomieller Zeit 3-Formel ψ so dass φ erfüllbar gdw. ψ erfüllbar.

Idee:

- führe Aussagenvariable für jede Teilformel von φ ein
- beschreibe das Verhalten von Teilformeln in AL
- Konvertiere entstehende Formel in 3-Formel
- Resultierende Formel ist nicht äquivalent, aber äqui-erfüllbar

CLIQUE

Ebenfalls bekannt aus Theoretische Informatik 2:

CLIQUE ist NP-vollständig.

Beweis Härte: polynomielle Reduktion von 3SAT

3-Färbbarkeit

Wie CLIQUE sind viele interessante Probleme auf Graphen NP-vollständig.

Wir betrachten ein weiteres Beispiel

Definition 3-Färbbarkeit

Ungerichteter Graph $G = (V, E)$ ist *3-färbbar* gdw. es Abbildung

$$f : V \to \{R, G, B\}$$

gibt, so dass $\{v, v'\} \in E$ impliziert $f(v) \neq f(v')$ (*3-Färbung*).
3F ist die Menge aller Graphen G, die 3-färbbar sind.

Leicht zu sehen: 3F $\in$ NP

3-Färbbarkeit

Theorem

3F ist NP-vollständig.

Beweis Härte: polynomielle Reduktion von 3SAT

Gegeben 3-Formel φ, konstruiere in polynomieller Zeit Graph G so dass φ erfüllbar gdw. G 3-färbbar

Idee:

- führe einen Knoten für jedes Literal ein
- verwende die Farben “wahr”, “falsch” und “hilf”
- verbinde komplementäre Literale mit Kanten, um konsistente WZ zu erzwingen
- verwende Teilgraphen, um Erfülltsein jeder Klausel zu erzwingen

NP-Vollständigkeit

Nicht nur in der Logik und Graphentheorie, sondern in vielen Bereichen der Informatik gibt es NP-vollständige Probleme.

1972 veröffentlichte Richard Karp in einem berühmten Aufsatz 21 solche (und sehr verschiedene) Probleme

Im Buch "Computers and Intractability" von Garey und Johnson finden sich ca 300 NP-vollständige Probeme, heute kennt man tausende!

Wir betrachten zwei weitere Beispiele:

- Integer Programming
- Rucksackproblem

IPROG

Zur Erinnerung:

Definition Integer Programming

Sei V eine Menge von Variablen und G eine Menge von linearen Gleichungen

$$c_1 \cdot x_1 + \cdots + c_n \cdot x_n = \alpha$$

mit $x_1, \ldots, x_n \in V$, $c_1, \ldots, c_n \in \mathbb{N}$ und $\alpha \in V \cup \mathbb{N}$.

Lösung für G ist Abbildung $\tau : V \to \mathbb{N}$, so dass alle Gleichungen in G erfüllt sind.

IPROG ist Menge aller Gleichungssysteme G, für die Lösung existiert

Integer Programming

Theorem

IPROG ist NP-vollständig.

Beweis Härte: polynomielle Reduktion von 3SAT

Gegeben 3-Formel φ, konstruiere in polynomieller Zeit Gleichungssystem G so dass φ erfüllbar gdw. G Lösung hat.

Idee:

- Verwende für jedes Literal eine numerische Variable
- Gleichung stellt sicher, dass diese Variablen konsistente Werte aus dem Bereich $\{0, 1\}$ erhalten
- Zusätzliche Gleichungen stellen sicher, dass jede Klausel mindestens ein wahres Literal enthält

Rucksackproblem

- es soll Rucksack mit gegebener Kapazität gepackt werden
- es gibt eine Menge von zu packenden Gegenständen mit unterschiedlichem Wert und Gewicht
- man möchte Gegenstände von maximalem Gesamtwert mitnehmen

Definition Rucksackproblem

Sei $M = \{a_1, \ldots, a_k\}$ eine Menge von *Gegenständen*, wobei Gegenstand a_i Gewicht g_i hat und Nutzen n_i.

Sei $G \geq 0$ eine Gewichtsgrenze und N ein intendierter Nutzen.

Lösung für Rucksackproblem (M, G, N) ist Teilmenge $R \subseteq M$ so dass

1. $\sum_{a_i \in R} g_i \leq G$ und
2. $\sum_{a_i \in R} n_i \geq N$.

RP ist die Menge aller Instanzen (M, G, N), für die Lösung existiert.

Rucksackproblem

Theorem

RP ist NP-Vollständig

Beweis Härte: polynomielle Reduktion von 3SAT

Gegeben 3-Formel φ, konstruiere in polynomieller Zeit Rucksackproblem (M, G, N) so dass φ erfüllbar gdw. (M, G, N) Lösung hat.

Idee:

- Verwende für jede Variable p_i zwei Gegenstände a_i und $\overline{a}_i$
- Ist p_i wahr ist, ist a_i im Rucksack aber $\overline{a}_i$ nicht
- Ist p_i falsch, so ist es genau andersherum
- Für jede Aufgabe a_i ist Gewicht g_i gleich Nutzen n_i und $G = N$
- Zusätzlich werden Gegenstände b_i und c_i für jede Klausel benötigt

NP Vollständigkeit

NP-vollständige Probleme finden sich auch im täglichen Leben:

- Minesweeper. Gegeben den momentanen Stand eines Minesweeper Spieles (mit Brett beliebiger Größe), ist an einer bestimmten (verdeckten) Stelle mit Sicherheit eine Mine versteckt?
- Sudoku. Gegeben ein partiell gelöstes Sudoku, kann es zu einem vollständig gelösten erweitert werden?
- Bundesliga. Gegeben den momentanen Punktestand der Bundesliga (mit beliebig vielen Manschaften), kann eine bestimmte Manschaft noch Meister werden?
- etc.

Kapitel 3

Einige weitere Themen rund um P und NP:

- Komplemente und co-NP
- Isomorphie von NP-Problemen und Mahaney's Theorem
- Zwischen P und NP / Constraint Satisfaction

Kapitel 3

Komplemente und co-NP

Motivation

Sei $\overline{3F}$ die Menge aller nicht 3-färbbaren Graphen.

In welcher Komplexitätsklasse ist $\overline{3F}$?

Um in NP zu sein, müsste es polynomielles Beweissystem geben.

Aber was ist polynomieller Beweis dafür, dass Graph nicht 3-färbbar?

Z.B. Menge aller Färbungen mit 3 Farben; exponentiell viele!

Offensichtlich ist $\overline{3F}$ ein Problem, bei dem es kurze und einfach zu verifizierende Beweise für nein-Instanten gibt statt für ja-Instanzen

Komplement

Definition Komplement (Problem)

Sei $L \subseteq \Sigma^*$ Problem. Das *Komplement* von L ist $\overline{L} := \Sigma^* \setminus L$.

Z.B. $\overline{\text{3F}}$

$\overline{\text{SAT}}$ ist die Menge aller unerfüllbaren AL-Formeln

$\overline{\text{CLIQUE}}$ ist die Menge aller Paare (G, k) so dass G keine k-Clique hat

Lemma

Deterministische Zeitkomplexitätsklassen $\mathcal{C}$ wie $\text{DTime}(t)$ und P sind abgeschlossen unter Komplement, also $L \in \mathcal{C}$ impliziert $\overline{L} \in \mathcal{C}$.

Also z.B. $\overline{\text{GAP}} \in \text{P}$

Komplement

Dieses Argument schlägt fehl für nicht-deterministische TMs,

funktioniert also nicht für die alternative Charakterisierung von NP!

Definition Komplement (Komplexitätsklasse)

Wenn $\mathcal{C}$ Komplexitätsklasse, dann co-$\mathcal{C} := \{L \mid \overline{L} \in \mathcal{C}\}$.

Wir interessieren uns hier insbesondere für die Klasse co-NP

Z.B. sind $\overline{\text{3F}}, \overline{\text{SAT}}, \overline{\text{CLIQUE}}$ per Definition in co-NP.

Definition Gültigkeit in Aussagenlogik

AL-Formel φ ist *gültig* gdw. jede WZ φ erfüllt

Lemma

Gültigkeit ist in co-NP.

Vollständigkeit

Härte und Vollständigkeit sind definiert wie für NP:

Definition co-NP-Härte, co-NP-Vollständigkeit

Problem L ist

- *co-NP-hart* wenn $L' \leq_p L$ für alle $L' \in$ co-NP;
- *co-NP-vollständig* wenn L co-NP-hart und in co-NP.

Lemma

L ist NP-hart gdw. $\overline{L}$ co-NP-hart.

$\overline{\text{3F}}, \overline{\text{SAT}}, \overline{\text{CLIQUE}}$, etc sind also co-NP-vollständig.

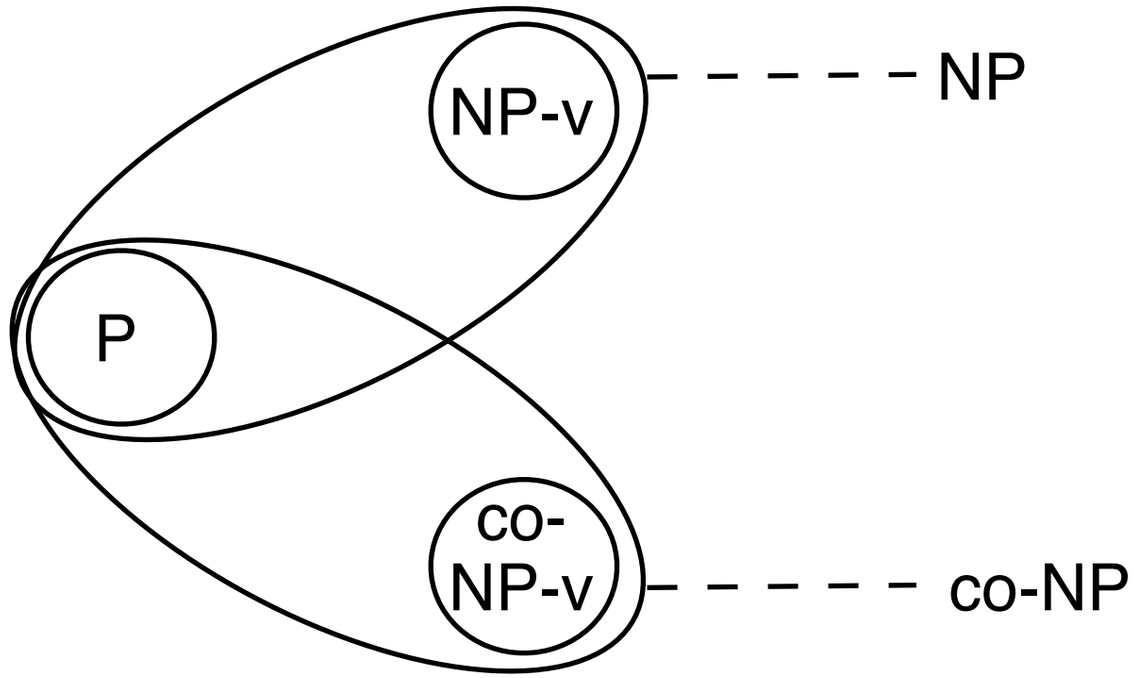
Lemma

Gültigkeit ist co-NP-vollständig.

P vs NP vs co-NP

Leicht zu sehen:

P $\subseteq$ co-NP



Es wird vermutet, dass

1. NP $\neq$ co-NP
2. NP $\cap$ co-NP $\neq$ P

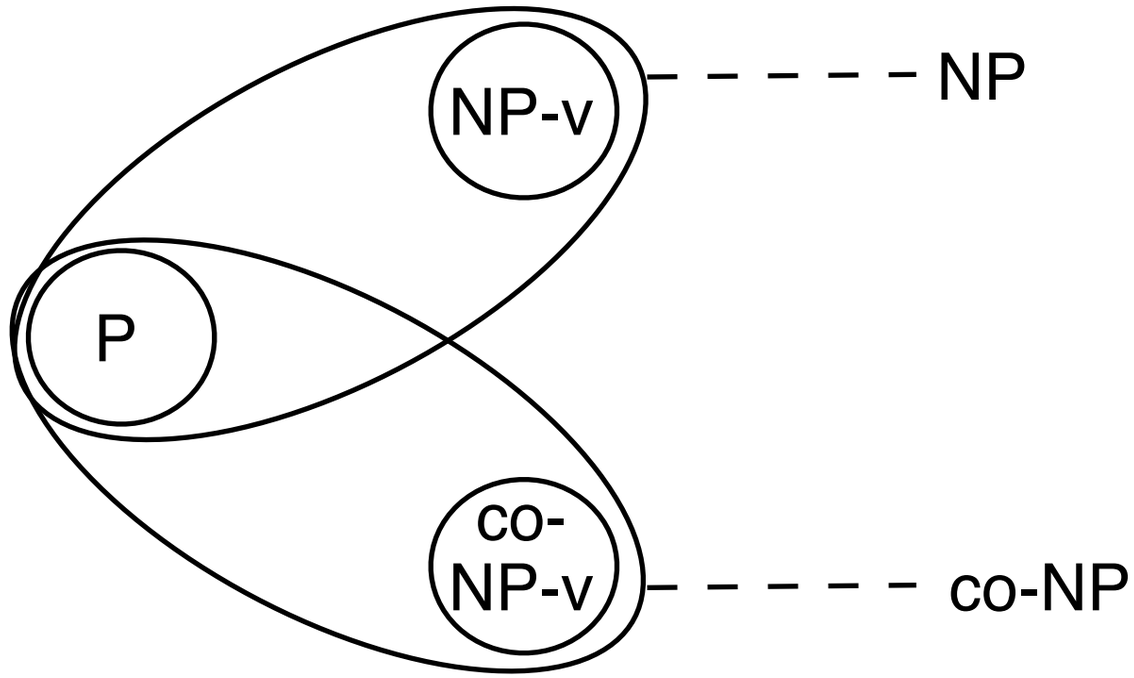
Beide Vermutungen implizieren P $\neq$ NP, umgekehrt ist das nicht bekannt.

(wenn P $=$ NP, dann P $=$ NP $=$ co-NP, Abschluss von P unter Komplement)

NP ∩ co-NP

Leicht zu sehen:

P ⊆ co-NP



Damit ergibt sich NP ∩ co-NP als weitere interessante Klasse.

NP: kurze / einfach zu verifizierende Beweise für ja-Instanzen

coNP: kurze / einfach zu verifizierende Beweise für nein-Instanzen

NP ∩ co-NP: beides ist der Fall

NP ∩ co-NP

Probleme in NP ∩ co-NP sind wahrscheinlich nicht NP-vollständig:

Lemma

Wenn es ein NP-vollständiges $L \in$ NP $\neq$ gibt, dann NP $=$ co-NP.

Ein bekanntes Problem in NP ∩ co-NP ist Integer Faktorisierung (IF):

Gegeben $n, k \in \mathbb{N}$, entscheide ob n einen Faktor $\leq k$ hat.

RSA Public Key Kryptographie ist nicht sicher wenn IF in P

(mit binärer Suche ließen sich dann effizient Faktoren finden)

Kapitel 3

Isomorphie von NP-Problemen und Mahaney's Theorem

Isomorphievermutung

NP-vollständige Probleme scheinen sich alle ähnlich zu sein.

Beobachtung

Wenn L, L' NP-vollständig, dann $L \leq_p L'$ und $L' \leq_p L$.

Die Reduktionen sind aber nicht unbedingt strukturerhaltend:

- Injektivität nicht gewährleistet
  z.B. 3SAT $\leq_p$ 3F: verdoppeln von Klauseln in 3-Formel
  ändert konstruierten Graph nicht.
- Surjektivität nicht gewährleistet
  z.B. 3SAT $\leq_p$ 3F: nur Graphen mit bestimmter Struktur konstruiert
- Reduktionen $L \leq_p L'$ und $L' \leq_p L$ sind unabhängig voneinander

Wie ähnlich sind NP-vollständige Probleme genau?

Isomorphievermutung

Wir

- fordern Injektivität und Surjektivität
- verschmelzen die zwei unabhängigen Reduktionen zu einer bidirektionalen

Definition p-Isomorphismus

Polynomielle Reduktion $f : \Sigma^* \to \Gamma^*$ von L' auf L ist *p-Isomorphismus* wenn

- f *Bijektion* ist (also injektiv und surjektiv)
- nicht nur f, sondern auch f^{-1} in Polyzeit berechenbar ist.

Existiert so ein f sind L' und L *p-isomorph*.

Leicht zu sehen: f^{-1} ist Polyzeit-Reduktion von L auf L'

Isomorphievermutung

Vermutung (Berman/Hartmanis 1977)

Alle NP-vollständigen Probleme sind paarweise p-isomorph.

Intuitiv sagt diese Vermutung: alle NP-vollständigen Probleme sind dasselbe Problem, in unterschiedlicher "Verkleidung"

Sollte die Vermutung wahr sein, so ist sie nicht leicht zu beweisen:

Theorem

Wenn alle NP-vollständigen Probleme paarweise p-isomorph sind, dann $P \neq NP$.

Es ist recht unklar, ob die Isomorphievermutung wahr oder falsch ist.

Alle bekannten NP-vollständigen Probleme sind aber p-isomorph

Mahaney's Theorem

Im Zshg. mit der Isomorphie-Vermutung steht folgendes Theorem

Definition

Eine Menge $L \subseteq \Sigma^*$ heißt *spärlich* wenn es ein Polynom p gibt so dass L für jede Wortlänge n höchstens $p(n)$ Wörter der Länge n enthält.

Beachte: wenn $|\Sigma| > 1$, dann erhält Σ^n exponentiell viele Wörter

Spärliche Mengen sind also Probleme mit sehr wenigen "ja"-Instanzen

Mahaney's Theorem

Wenn es eine spärliche Menge gibt, die NP-vollständig ist, dann P=NP.

Man lernt also etwas über die Stuktur von NP-vollständigen Problemen

Mahaney's Theorem

Für einen Beweis betrachten wir eine geeignete Variante von SAT

Sei "$<$" die lexikographische Ordnung aussagenlogischer Belegungen

Definition

LeftSAT ist die Menge aller Paare (φ, π) so dass

- φ eine aussagenlogische Formel ist und
- π eine Belegung (für die Variablen in φ)

so dass es Belegung $\tau \leq \pi$ gibt, die φ erfüllt.

Man überlegt sich sehr leicht: LeftSAT ist NP-vollständig

Beweisidee Mahaney's Theorem:

Wir nehmen an, dass es eine spärliche NP-vollständige Menge S gibt und zeigen, dass LeftSAT dann in P ist. Also gilt P=NP.

Kapitel 3

Zwischen P und NP / Constraint Satisfaction

NP Intermediate

Man mag sich fragen, ob jedes NP-Problem in P oder NP-vollständig ist.

Das ist nicht der Fall, wie sich bereits bei NP $\cap$ coNP andeutete

Definiere Komplexitätsklasse NPI (NP-Intermediate):

Menge aller Probleme $L \in \text{NP} \setminus \text{P}$, die nicht NP-vollständig sind.

Ladner's Theorem

Wenn P $\neq$ NP, dann NPI $\neq \emptyset$.

Zeigt, dass NP-Vollständigkeit noch wichtig ist wenn P $\neq$ NP bewiesen!

NP Intermediate

Ladner's Theorem

Wenn P $\neq$ NP, dann NPI $\neq \emptyset$.

Überblick Beweis:

- subtiles Diagonalisierungsargument
- starte mit SAT, entferne auf systematische Weise ”ja”-Instanzen man erhält “SAT mit Löchern”
- erreiche damit, dass jede mögliche Reduktion von SAT auf SAT mit Löchern fehlschlägt, also letzteres nicht NP-hart
- halte die Löcher so klein wie möglich, so dass das Problem trotz Löchern nicht in Polyzeit gelöst werden kann

NP Intermediate

Bisher konnte kein natürliches Problem in NPI identifiziert werden

Kandidaten:

- Graph Isomorphismus. Gegeben ungerichtete Graphen G_1, G_2, kann man die Knoten in G_2 umbenennen so dass $G_1 = G_2$?

- Log-Clique: enthält ein gegebener Graph mit n Knoten eine Clique der Größe $\log(n)$?

- Alle Probleme in NP $\cap$ co-NP wie z.B. Integer Faktorisierung.

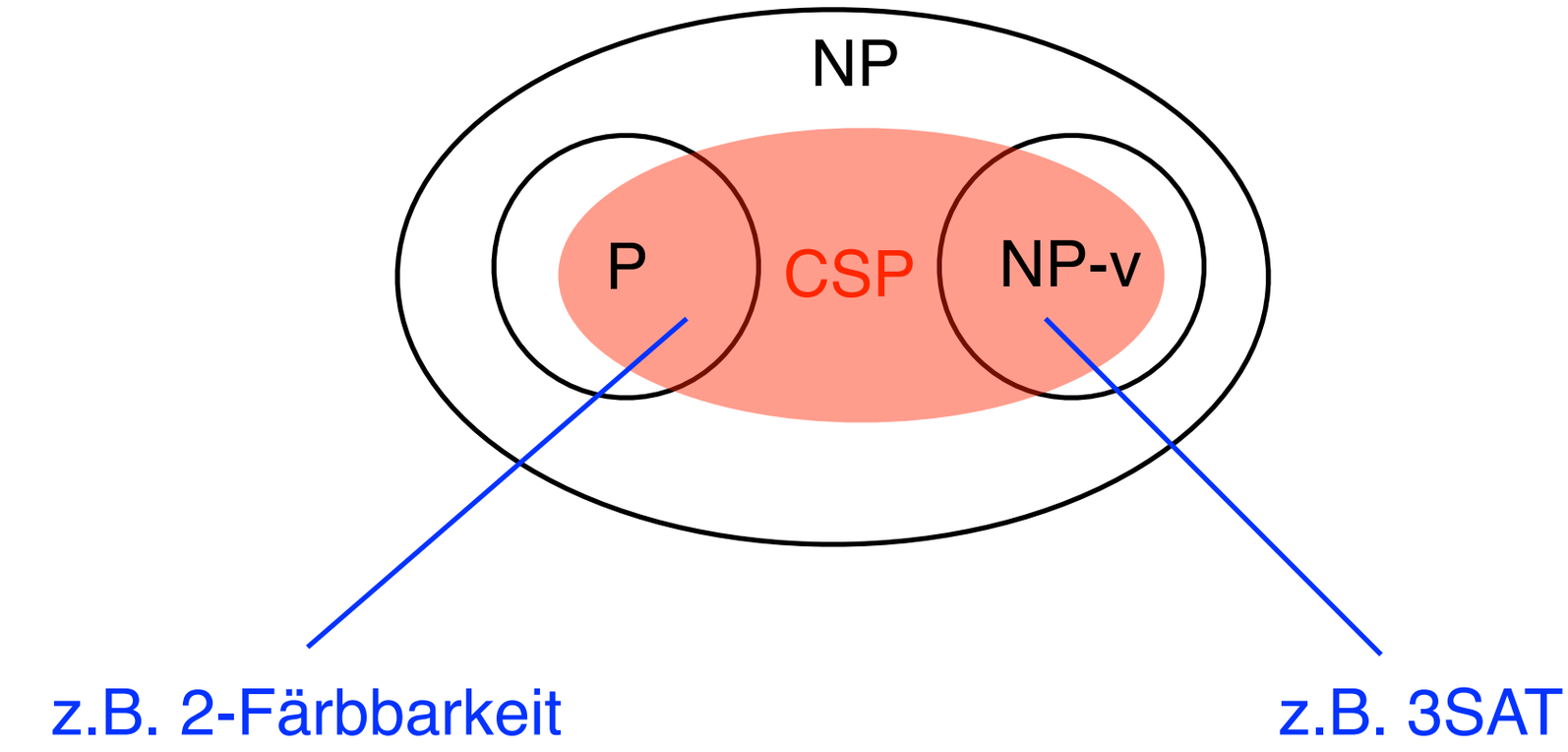
Man kann Ladner's Theorem noch signifikant stärker machen:

Ladner's Theorem, reloaded

Wenn P $\neq$ NP, dann gibt es unendliche Folge von Sprachen $L_1, L_2, \ldots$ in NPI so dass für alle $i \geq 1$ gilt: $L_{i+1} \leq_p L_i$ und $L_i \not\leq_p L_{i+1}$.

Constraint Satisfaction

Constraint-Satisfaction-Probleme (CSPs) bilden eine natürliche Teilklasse von NP, in der viele wichtige Probleme enthalten sind:



Es besteht Grund zur Annahme, dass CSP leichter zu analysieren ist als NP

Insbesondere wird vermutet, dass es kein CSP gibt, das NP-Intermediate ist.

Constraint Satisfaction

CSPs kommen ursprünglich aus der künstlichen Intelligenz

Zum Beispiel: Lösen von Kryptogrammen:

$$\begin{array}{r} \,S\;E\;N\;D \\ +\;\;M\;O\;R\;E \\ \hline =\;M\;O\;N\;E\;Y \end{array}$$

Kann man den Buchstaben Ziffern zuweisen, so dass die oben dargestellte Rechnung aufgeht?

Weitere Beispiele:
Konfigurationsprobleme, Bildinterpretation, Schedulingprobleme, usw.

CSPs haben stark kombinatorischen Charakter.

Constraint Satisfaction

Ein CSP ist definiert durch:

- endlichen *Wertebereich* wie z.B. $\{0, 1, \ldots, 9\}$
- die erlaubten *Arten von Constraints*, z.B.

  einstellige Constraints der Form "$x = 0$" (x, x_i Variablen)

  fünfstellige Constraints der Form "$x_1 + x_2 + x_3 = 10x_4 + x_5$"

Eingabe besteht aus endlicher Menge von Constraints, z.B.:

$$z = 0$$

$$x_D + x_E + z = 10c_1 + x_Y$$

$$x_N + x_R + c_1 = 10c_2 + x_E$$

usw.

$$\begin{array}{r} \texttt{S E N D} \\ + \quad \texttt{M O R E} \\ \hline = \quad \texttt{M O N E Y} \end{array}$$

und ist "ja"-Instanz, wenn man den Variablen Werte zuweisen kann, so dass alle Constraints erfüllt sind.

Constraint Satisfaction

Definition Constraint-Satisfaction-Probleme

Eine *relationale Struktur* $\mathfrak{A}$ besteht aus

- einem nicht-leeren *Wertebereich* A und
- einer Menge $\mathcal{R}$ von *Relationssymbolen* R mit zugeordneter *Stelligkeit* n_R und zugeordneter *Relation* $R^{\mathfrak{A}} \subseteq A^{n_R}$

Jede endliche relationale Struktur $\mathfrak{A}$ definiert Problem CSP$(\mathfrak{A})$ wie folgt:

Eingabe: Menge M von *Constraints* der Form $R(x_1, \ldots, x_n)$ wobei $R \in \mathcal{R}$ n-stellig und $x_1, \ldots, x_n$ Variablen

Ausgabe: ”ja” wenn es *Lösung* für M gibt, d.h.

Abbildung $\delta : \text{VAR}(M) \to A$ so dass
für alle $R(x_1, \ldots, x_n) \in M$ gilt: $(\delta(x_1), \ldots, \delta(x_n)) \in R^{\mathfrak{A}}$

und ”nein” sonst.

Constraint Satisfaction

Sei CSP die Klasse aller Constraint-Satisfaction-Probleme.

Lemma

CSP $\subseteq$ NP.

Viele bekannte NP-vollständige Probleme sind als CSPs darstellbar:

3F, k-Färbbarkeit für jedes feste $k > 2$, 3SAT,
IPROG über jedem festen Bereich $\{0, \ldots, n\}$ usw.

Es gibt aber auch NP-Probleme, die nicht als CSPs darstellbar sind

Das sieht man mittels Homomorphismen zwischen relationalen Strukturen

Homomorphismen

Definition Homomorphismus

Seien $\mathfrak{A}, \mathfrak{B}$ relationale Strukturen über derselben Menge von Relationssymbolen. *Homomorphismus* von $\mathfrak{B}$ nach $\mathfrak{A}$ ist Abbildung $h : B \to A$ so dass:

aus $(b_1, \ldots, b_n) \in R^{\mathfrak{B}}$ folgt $(h(b_1), \ldots, h(b_n)) \in R^{\mathfrak{A}}$.

Gibt es Homomorphismus von $\mathfrak{B}$ nach $\mathfrak{A}$, so schreiben wir $\mathfrak{B} \to \mathfrak{A}$.

CSPs und Homomorphismen

Auch jede *Eingabe* für ein CSP($\mathfrak{A}$) (also jede Constraintmenge M) kann als relationale Stuktur $\mathfrak{B}_M$ aufgefasst werden:

- der Wertebereich von $\mathfrak{B}_M$ besteht aus den Variablen in M
- die Relationssymbole und Stelligkeiten sind dieselben wie in $\mathfrak{A}$
- $R^{\mathfrak{B}_M} = \{(v_1, \ldots, v_n) \mid R(v_1, \ldots, v_n) \in M\}$.

Äquivalente Definition Constraint-Satisfaction-Probleme

Jede relationale Struktur $\mathfrak{A}$ definiert Problem CSP($\mathfrak{A}$) wie folgt:

$$\text{CSP}(\mathfrak{A}) = \{\mathfrak{B} \mid \mathfrak{B} \to \mathfrak{A}\}$$

CSP vs NP

Wir können nun den Zusammenhang zwischen CSP und NP untersuchen:

Lemma

Jedes CSP($\mathfrak{A}$) ist abgeschlossen unter homomorphen Urbildern, d.h.:

wenn $\mathfrak{B}$ "ja"-Instanz und $\mathfrak{B}' \to \mathfrak{B}$, dann ist $\mathfrak{B}'$ "ja"-Instanz

Probleme in NP und auch Probleme in P sind im Allgemeinen nicht unter homomorphen Urbildern abgeschlossen. Daher gilt:

Theorem

CSP $\subsetneq$ NP.

CSP und NP-Intermediate

Die Klasse CSP hat in vielerlei Hinsicht bessere Eigenschaften als NP

Ihre Analyse könnte ein erster (keineswegs trivialer) Schritt auf dem Weg zu einem besseren Verständnis von NP sein.

Insbesondere gilt die folgende Vermutung:

Feder-Vardi-Vermutung

Jedes CSP ist entweder in P oder NP-vollständig, es gibt also keine CSPs, die NP-Intermediate sind (*Dichotomie* zwischen P und NP).

Bewiesen werden konnte sie bisher nicht, aber es gibt ernsthafte Fortschritte und partielle Resultate

Partielle Resultate

Schaefers Theorem (1978)

Sei $\mathfrak{A}$ eine relationale Struktur mit $A = \{0, 1\}$. Dann ist CSP$(\mathfrak{A})$ in P, wenn eine der folgenden Bedingungen gilt:

1. alle Relationen in $\mathfrak{A}$ enthalten das Tupel $(1, \ldots, 1)$
2. alle Relationen in $\mathfrak{A}$ enthalten das Tupel $(0, \ldots, 0)$
3. alle Relationen in $\mathfrak{A}$ sind durch Horn-Formeln definierbar
4. alle Relationen in $\mathfrak{A}$ sind durch Dual-Horn-Formeln definierbar
5. alle Relationen in $\mathfrak{A}$ sind durch 2-Formeln definierbar
6. alle Relationen in $\mathfrak{A}$ sind durch affine Formeln definierbar

Andernfalls ist CSP$(\mathfrak{A})$ NP-vollständig.

Partielle Resultate

Bulatov hat Schaefers Theorem auf Stukturen der Größe 3 generalisiert.

Anstatt die *Größe* von Strukturen zu beschränken, kann man sie auch anderweitig einschränken.

Relationale Struktur ist *ungerichteter Graph*, wenn sie nur ein einziges Relationssymbol E hat, das zudem binär ist und eine symmetrische Relation bezeichnet.

Theorem (Hell und Nesetril 1990)

Sei $\mathfrak{A}$ ein *ungerichteter* Graph beliebiger Größe. Dann ist CSP($\mathfrak{A}$) in P, wenn $\mathfrak{A}$ 2-färbbar ist und NP-vollständig sonst.

Feder und Vardi zeigen aber, dass eine Klassifikation von *gerichteten* Graphen (Symmetrie fällt weg) ebenso schwer ist wie der allgemeine Fall.

Übersicht Vorlesung

- Kapitel 1: Einführung
- Kapitel 2: Turingmaschinen
- Kapitel 3: P vs. NP
- Kapitel 4: Mehr Ressourcen, mehr Möglichkeiten?
- Kapitel 5: Platzkomplexität
- Kapitel 6: Schaltkreise
- Kapitel 7: Orakel