Komplexitätstheorie

WiSe 2009/10 in Trier

Henning Fernau
Universität Trier
fernau@uni-trier.de

Komplexitätstheorie Gesamtübersicht

- Organisatorisches / Einführung
  Motivation / Erinnerung / Fragestellungen

- Diskussion verschiedener Komplexitätsklassen:
  Zeitkomplexität
  Platzkomplexität

- zugehörige Reduktionsbegriffe

- vollständige Probleme

- Anpassung von Klassenbegriffen und Reduktionen

Umfangreiche Liste NP-vollständiger Probleme

- Garey, M.R., Johnson, D.S., Computers and Intractability:
  A Guide to the Theory of **NP**-Completeness, Freeman, San Francisco 1979
  sowie als Fortsetzung davon:

- Johnson, D.S., The **NP**-completeness column: an ongoing guide,
  seit 1981 in der Zeitschrift "Journal of Algorithms",
  später in "ACM Transactions on Algorithms"

Noch einmal als Merksatz:

Solange es nicht gelungen ist, $\mathbf{P} = \mathbf{NP}$ zu beweisen, ist für keines der **NP**-vollständigen Probleme ein praktisch verwendbarer Algorithmus bekannt!

Im Folgenden **NP**-vollständige Probleme aus verschiedenen Problembereichen:

Logik, Graphentheorie, Mengentheorie und Zahlentheorie

Graphentheorie / Mengentheorie: $\rightsquigarrow$ Letzte Vorlesung
Zahlen und weitere Kuriosa: heute

Mengen- und zahlentheoretische Probleme

Satz 1 **NP**-*vollständig (bzgl.* $\leq_{\log}$*) sind:*

- `INTEGER PROGRAMMING`
- `KNAPSACK`

- INTEGER (LINEAR) PROGRAMMING (ILP)

  Gegeben:

  —$m \times n$-Matrix A über $\mathbb{Z}$

  —$b \in \mathbb{Z}^m$

  —$c \in \mathbb{Z}^n$

  —$k \in \mathbb{Z}$

  Frage: Existiert eine ganzzahlige Lösung für $Ax \geq b$ mit $x \geq 0$, die $c^T x$ minimiert (d.h. $c^T x \leq k$) ?

Das **gewichtete Knotenüberdeckungsproblem** (mit Knotengewichten c_i) lässt sich wie folgt als ILP ausdrücken:

minimiere	$\sum_{v_i \in V} c_i x_i$
unter	$x_i + x_j \geq 1$ für $\{v_i, v_j\} \in E$
	$x_i \in \{0, 1\}$ für $v_i \in V$

Ausgegangen wird dabei von einem Graphen $G = (V, E)$, $V = \{v_1, \ldots, v_n\}$ mit Knotenkosten $c_i (c_i \geq 0)$ für v_i. Die Variable x_i „entspricht“ dem Knoten v_i und die Bedingung $x_i + x_j \geq 1$ „entspricht“ der Kante $\{v_i, v_j\}$.

Daher ist eine Knotenmenge $C \subseteq V$, gegeben durch $C = \{v_i \mid x_i = 1\}$, genau dann eine Knotenüberdeckung, wenn für alle Kanten $\{v_i, v_j\}$ die Ungleichung $x_i + x_j \geq 1$ gilt.

Merkwürdigkeiten I

ILP ist ein *generisches Problem*:
Viele Optimierungsprobleme lassen sich durch ILPs ausdrücken.
Versuchen Sie es!
$\leadsto$ Viele Näherungsalgorithmen etc. für Optimierungsprobleme beruhen auf ILP-Techniken.
$\leadsto$ "Schwestervorlesung" Approximationsalgorithmen (jedes 2. WS in TR)

Wenn man die *Ganzzahligkeitsbedingungen* (wie $x_i \in \{0,1\}$ für $v_i \in V$ im VC-Beispiel) fallen lässt, erhält man ein *Lineares Programm*.
Dies lässt sich in Polynomzeit lösen!
Diese sogen. *Relaxation* ist eine wesentliche Technik zum Umgang mit ILPs.
Problem: Gelieferte Lösungen sind nicht notwendig ganzzahlig / zulässig.
(Nimm 40% von Knoten v_1 in die Überdeckung...)

- KNAPSACK

  Gegeben:
  —$W \in \mathbb{N}$
  —n Gegenstände S_i mit einem Gewicht w_i und einem Wert v_i
  —dabei $v_i, w_i \in \mathbb{N}$

  Frage: Gesucht ist Maximum von $\sum_{j=1}^{m} v_{i_j}$
  unter allen Auswahlen $\{S_{i_j}\}_{j=1,\ldots,m}$ von Gegenständen,
  die die folgende Bedingung erfüllen:

$$\sum_{j=1}^{m} w_{i_j} \leq W$$

Bei der obigen Fassung von KNAPSACK wird das Maximum *gesucht*.

- *Entscheidungsvariante* von KNAPSACK: Gegeben:
  —$W \in \mathbb{N}$
  —$k \in \mathbb{N}$
  —n Gegenstände S_i mit einem Gewicht w_i und einem Wert v_i
  —dabei $v_i, w_i \in \mathbb{N}$
  Frage: Existiert eine Teilmenge $\{S_{i_j}\}_{j=1,\ldots,m}$ mit

$$\sum_{j=1}^{m} w_{i_j} \leq W \quad , \quad \sum_{j=1}^{m} v_{i_j} \geq k$$

Spezialfall KNAPSACK$*$: $\forall 1 \leq i \leq n : v_i = w_i$ und $k = W$.

Merkwürdigkeiten II

Beobachte: Jede Eingabe von KNAPSACK kann in Zeit $\mathcal{O}(n \cdot W)$ gelöst werden (*pseudo-polynomiell*).
Dies ist nicht polynomial in der Eingabelänge $\approx n \cdot \log W$!,
denn Zahlen sind binär codiert.

Bei Problemen wie KNAPSACK kommt es aber (anders als bei z.B. dem gewichteten Knotenüberdeckungsproblem) darauf an, ob die Zahlen in der Eingabe binär oder unär codiert sind.

KNAPSACK* ist NP-hart

Wir zeigen: EXACT COVER BY 3-SETS $\leq_{\log}$ KNAPSACK*.

Sei $\mathcal{F} = \{S_1, \ldots, S_n \mid |S_i| = 3\}$ eine Instanz von EXACT COVER BY 3-SETS mit Universum $U = \{1, \ldots, 3m\}$.

Gesucht sind m disjunkte Mengen S_{i_j} mit $\bigcup_j S_{i_j} = U$.

Repräsentiere S_i durch Zahl $s_i = \sum_{\ell \in S_i} (3m+1)^{3m-\ell}$.

Setze $k = \sum_{\ell=0}^{3m-1} (3m+1)^{\ell}$.

k wird genau dann durch $\sum s_{i_j}$ erreicht, wenn S_{i_j} das Universum überdeckt.

Wichtig: Die Basis $(3m+1)$ vermeidet Überträge beim Rechnen mit Zahlen (gemeint sind ja eigentlich "Bitvektoren").

Beachte: Es entstehen "sehr große Zahlen".

Zusammenfassung NP; Merkwürdigkeiten III

Für alle $\mathbf{NP}$-vollständigen Probleme A:

$$A \in \mathbf{P} \Leftrightarrow \mathbf{P} = \mathbf{NP}$$

Beispiele für weitere Probleme mit dieser Äquivalenz, wo nicht bekannt ist, ob diese Probleme in $\mathbf{NP}$ liegen:

Viele *Minimierungs- und Maximierungsprobleme* aus der Graphentheorie, z.B.:

CHROMATIC NUMBER:
Gegeben seien ungerichteter Graph G und Zahl $k > 0$.
Frage: Ist k kleinste Zahl, für die G k-färbbar ist?

wird als *Minimierungsproblem* zu: FÄRBBARKEIT:
Gegeben seien ungerichteter Graph G und Zahl $k > 0$.
Frage: Ist G k-färbbar?

Bekanntes und Fragen

—FÄRBBARKEIT $\in$ **NP**

—FÄRBBARKEIT **NP**-vollständig

da trivialerweise 3-FÄRBBARKEIT $\leq_{\log}$ FÄRBBARKEIT

Für CHROMATIC NUMBER ergibt sich daraus:

1. CHROMATIC NUMBER ist **NP**-hart.
2. CHROMATIC NUMBER liegt in **PSPACE**.
3. CHROMATIC NUMBER $\in \mathbf{P} \Leftrightarrow \mathbf{P} = \mathbf{NP}$.
4. CHROMATIC NUMBER $\in \mathbf{NP} \Leftrightarrow \mathrm{co} - \mathbf{NP} = \mathbf{NP}$.

Eigenschaften von CHROMATIC NUMBER I

1. CHROMATIC NUMBER ist **NP**-hart.

Sei $G = (V, E)$ ein ungerichteter Graph.
Betrachte $G' = (V', E')$ mit $V' = V \cup \{a, b, c\}$ und $E' = E \cup \{\{a, b\}, \{b, c\}, \{a, c\}\}$.
a, b, c sind "neue" Knoten $\leadsto$ G' enthält (zusätzlich) das Dreieck (a, b, c).
$\leadsto$ G 3-färbbar gdw. G' 3-färbbar.
Zum Färben von G' sind mindestens 3 Farben notwendig.
$\leadsto$ G 3-färbbar gdw. $(G', 3) \in$ CHROMATIC NUMBER.

$\leadsto$ 3-FÄRBBARKEIT $\leq_{\log}$ CHROMATIC NUMBER

Eigenschaften von CHROMATIC NUMBER II

2. CHROMATIC NUMBER liegt in **PSPACE**.

FÄRBBARKEIT $\in$ **NP** $\subseteq$ **PSPACE**.

Es gilt:

$(G, k) \in$ CHROMATIC NUMBER gdw.
$(G, k) \in$ FÄRBBARKEIT $\wedge\, (G, k-1) \notin$ FÄRBBARKEIT.

$\rightsquigarrow$ **PSPACE**-Algorithmus für CHROMATIC NUMBER

Achtung: Dies ist kein **NP**-Algorithmus, da dort keine Nein-Antwort erhältlich ist.

3. CHROMATIC NUMBER $\in$ **P** $\Leftrightarrow$ **P** $=$ **NP**.

Wegen 1. (CHROMATIC NUMBER ist **NP**-hart) gilt "$\Rightarrow$".
Wäre **P** $=$ **NP**, so läge FÄRBBARKEIT in **P**, und der Algorithmus aus 2. wäre ein **P**-Algorithmus für CHROMATIC NUMBER.

Eigenschaften von CHROMATIC NUMBER III

4. CHROMATIC NUMBER $\in \mathbf{NP} \Leftrightarrow co - \mathbf{NP} = \mathbf{NP}$.

"$\Leftarrow$": Wäre $co - \mathbf{NP} = \mathbf{NP}$, so gölte: $co -$ FÄRBBARKEIT $\in \mathbf{NP}$.
Der Algorithmus aus 2. wäre dann ein **NP**-Algorithmus für CHROMATIC NUMBER.

Klar: $G \in co{-}3 -$ FÄRBBARKEIT $\iff \exists k \in (3, n] : (G, k) \in$ CHROMATIC NUMBER

"$\Rightarrow$": Wäre CHROMATIC NUMBER $\in \mathbf{NP}$, so läge $co - 3 -$ FÄRBBARKEIT in **NP**:
Betrachte **NP**-Algorithmus:
(a) Rate $k \in (3, n]$;
(b) Gib JA aus, falls $(G, k) \in$ CHROMATIC NUMBER.

Da 3-FÄRBBARKEIT **NP**-vollständig, folgt $co - \mathbf{NP} = \mathbf{NP}$.

Merkwürdigkeiten IV Es gibt Probleme in $\mathbf{NP}$, für die man weder einen guten deterministischen Algorithmus kennt/kannte noch bisher ihre $\mathbf{NP}$-Vollständigkeit zeigen konnte.

- Menge PRIMES aller Primzahlen in binärer Darstellung.
  —Beweis von $\texttt{PRIMES} \in \mathbf{NP}$ bereits nichttrivial
  —neueres Resultat:

  $$\texttt{PRIMES} \in \mathbf{P}$$

  M. Agarwal, N Saxena, N. Kayal (Kanpur/Indien, 2002)
- GRAPH ISOMORPHISM: Menge aller Paare (G_1, G_2) von Graphen, die isomorph sind.
  Für spezielle Teilprobleme gibt es polynomiale Algorithmen:
  z.B. für Graphen, bei denen jeder Knoten höchstens mit k (konstant!) vielen anderen Knoten verbunden ist.
  Andererseits: SUBGRAPH ISOMORPHISM ist $\mathbf{NP}$-vollständig...

Konstruktion von Lösungen ist meist interessanter als reine Entscheidungsprobleme (Lösbarkeit).
Klar: Gibt es guten Algorithmus zur Konstruktion von Lösungen, so ist Entscheidungsproblem ebenfalls gut lösbar.

In vielen Fällen gilt auch die Umkehrung.

Beispiel: `SAT` hat Eigenschaft der *Selbstreduzierbarkeit*:*

Lemma 2 *Wenn es eine Polynomzeit-Lösung für* `SAT` *gibt, dann gibt es auch eine Polynomzeit-berechenbare Funktion, die zu einer erfüllbaren Formel eine erfüllende Belegung bestimmt.*

Anm.: Die meisten natürlichen Optimierungsprobleme sind selbstreduzierbar!

*Auf eine formale Definition dieses sehr intuitiven Begriffs sei hier verzichtet.

Selbstreduzierbarkeit von SAT; Merkwürdigkeiten V

Sei w Formel mit Variablen $x_1, \ldots, x_n$.
Bezeichne $w[x_\ell = 0]$ bzw. $w[x_\ell = 1]$ die (vereinfachte) Formel, die sich aus w ergibt, wenn man überall x_ℓ auf 0 bzw. auf 1 setzt.

Wir benutzen den angenommenen $\mathbf{P}$-Algorithmus für SAT zunächst um festzustellen, ob w erfüllbar ist.

Wenn ja, wird eine erfüllende Belegung wie folgt konstruiert:
(a) Setze $w' \leftarrow w$; $\ell \leftarrow n$.
(b) Ist $w'[x_\ell = 1]$ erfüllbar (SAT-Alg.!), setze $v_\ell \leftarrow 1$; andernfalls setze $v_\ell \leftarrow 0$.
(c) Falls $\ell = 1$, ist $\phi(x_i) = v_i$, $1 \leq i \leq n$, eine erfüllende Belegung für w.
(d) Andernfalls setze: $w' \leftarrow w'[x_\ell = v_\ell]$, $\ell \leftarrow \ell - 1$. Fahre fort bei (b).

Die Korrektheit des Verfahrens folgt bei Punkt (b), da wir mit einer erfüllbaren Formel zu tun haben.
$\rightsquigarrow$ Mit SAT hat auch das Konstruktionsproblem für SAT einen $\mathbf{P}$-Algorithmus.

Polynomial entscheidbare Relationen

$R \subseteq \Sigma^* \times \Sigma^*$ sei eine binäre Relation.

R heißt *polynomial entscheidbar* genau dann, wenn eine deterministische TM existiert, die die Sprache

$$\{x;y \mid (x,y) \in R\}$$

in polynomialer Zeit entscheidet.

R heißt *polynomial balanciert* genau dann, wenn ein $k \in \mathbb{N}$ existiert mit

$$(x,y) \in R \Rightarrow |y| \leq |x|^k$$

Satz 3 *Es gilt* $L \in \mathbf{NP}$ *genau dann, wenn eine polynomial entscheidbare und polynomial balancierte Relation* R *existiert mit* $L = \{x \mid \exists y \quad (x, y) \in R\}$.

Für jedes $x \in L$ gibt es damit einen *Beweis* oder *Zeugen* y (engl.: witness, succinct certificate) polynomialer Länge für die Eigenschaft "$x \in L$".

Beispiele von Zeugen:

—erfüllende Belegungen bei `SAT`
—Auflistung der Markierungsreihenfolge bei `AGAPvP`.

Beweis "$\Leftarrow$":

R sei polynomial entscheidbare und polynomial balancierte Relation mit $L = \{x \mid \exists y \quad (x, y) \in R\}$.

Sei k Konstante mit $(x, y) \in R \rightsquigarrow |y| \leq |x|^k$.
Sei M Turingmaschine für R.

Betrachte nichtdet. TM M', die wie folgt auf Eingabe x arbeitet:
(1) Rate y mit $|y| \leq |x|^k$.
(2) Simuliere M auf Eingabe $(x; y)$.
(3) M' akzeptiert gdw. M akzeptiert.

$\rightsquigarrow L \in \mathbf{NP}$.

Beweis "$\Rightarrow$":

Betrachte $L \in \mathbf{NP}$; M sei TM, die L in Zeit $\mathcal{O}(n^k)$ akzeptiert.
Verstärkt gelte (bis auf endlich viele Ausnahmen $A \subseteq L$): Zeitbedarf $\leq (|x|+1)^k$.

Definiere R wie folgt:
$(x, 0y) \in R$ gdw. $y \in \{0,1\}^*$ ist Folge von nichtdet. Entscheidungen, die zur Akzeptanz von x führen, falls $x \notin A$.
$(x, \lambda) \in R$ gdw. $x \in A$.
Damit ist R polynomial balanciert.
R ist polynomial entscheidbar durch Simulation von M unter Berücksichtigung von y.

Satz 4 L **NP**-*vollständig* $\Rightarrow$ *co-*L *co-***NP**-*vollständig.*

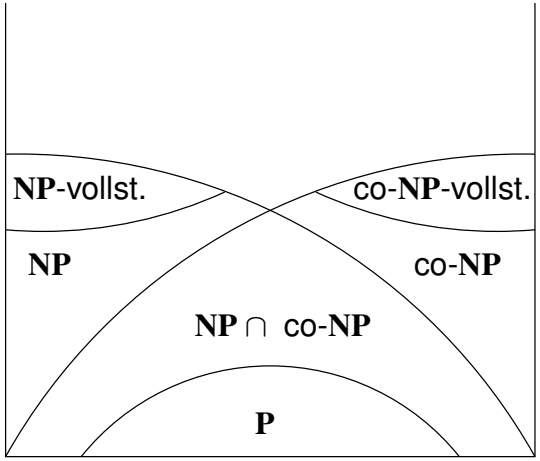
*Gibt es ein co-***NP**-*vollständiges Problem* L *in* NP*, so gilt*

$$\mathbf{NP} = \textit{co-}\mathbf{NP}$$

Es gilt

$$\mathbf{P} \subseteq \mathbf{NP} \cap \textit{co-}\mathbf{NP}$$

- Vermutlich: **NP** $\neq$co-**NP**.
- Vermutlich: **P** echte Teilmenge von **NP**$\cap$ co-**NP**.



Verallgemeinerung der Vorgehensweise bei CHROMATIC NUMBER:

Entscheidungsalgorithmus für FÄRBBARKEIT
$\Longrightarrow$ CHROMATIC NUMBER leicht lösbar.

Eine *Orakel-Turingmaschine* ist eine Turingmaschine $\mathcal{M}$ mit einem speziellen *Anfrage-Band* (engl.: query tape) und drei Zuständen

$$q_? \quad q_{yes} \quad q_{no}$$

Berechnung von $\mathcal{M}$: nutzt Eingabe und zudem eine *Orakel-Menge* A.

Meist: $\mathcal{M}$ arbeitet wie eine normale (det. / n.det.) Turingmaschine.

Orakel-Turingmaschinen

Meist: M arbeitet wie eine normale (det. / n.det.) Turingmaschine.
Ausnahme: M erreicht den Zustand $\mathrm{q}_?$ bei Orakel A.

—Orakel sei A
—a sei Inhalt des Anfrage-Bandes

$\Rightarrow$ in einem Schritt:
—bei $\mathrm{a} \in A$ geht M in q_{yes} über
—bei $\mathrm{a} \notin A$ geht M in q_{no} über

Akzeptanz von Worten: wie bisher ...
$\mathrm{L}(\mathrm{M}^A)$:= von der Orakel-Turingmaschine M bei Orakel A akzeptierte Sprache.

Orakel-Turingmaschinen

Sei $\mathcal{K}$ Komplexitätsklasse, A ein Orakel:
$\mathcal{K}^A$:= Klasse aller Sprachen, die beim Orakel A in der Komplexitätsklasse $\mathcal{K}$ liegen.

Beispiele:

$$\mathcal{K}^{\emptyset} = \mathcal{K}$$

$$\texttt{CHROMATIC NUMBER} \in \mathbf{P}^{\texttt{FÄRBBARKEIT}}$$