5 Komplexitätstheorie

Wir kennen Probleme, die durch keinen Algorithmus gelöst werden können (die nicht entscheidbar bzw. nicht einmal semi-entscheidbar sind).

In diesem Kapitel betrachten wir nur entscheidbare Probleme.

In anderen Vorlesungen haben Sie gelernt, solche Probleme möglichst effizient zu lösen (Stichwort: divide-and-conquer, dynamische Programmierung, etc.) und ihren Resourcenverbrauch abzuschätzen (Stichwort: O-Notation).

Fragen: Kann jedes entscheidbare Problem „effizient“ gelöst werden oder gibt es entscheidbare Probleme, deren Lösung notwendigerweise „ineffizient“ ist? Was heißt hier eigentlich „effizient“?

Definition (deterministische Komplexitätsklassen)**.** Sei $f\colon \mathbb{N} \to \mathbb{N}$ eine monotone Funktion und L eine Sprache. Dann ist L *in Zeit* $O(f(n))$ *entscheidbar*, wenn es eine det. TM M und $d, e \in \mathbb{N}$ gibt mit:

- M akzeptiert L, d.h. $L = L(M)$.
- Für jede Eingabe $w \in \Sigma^*$ hält M nach höchstens $d \cdot f(|w|) + e$ vielen Rechenschritten.

Die Sprache L ist *in Polynomialzeit entscheidbar*, wenn es $c \in \mathbb{N}$ gibt, so daß L in Zeit $O(n^c)$ entschieden werden kann. Die Komplexitätsklasse P umfaßt die Sprachen, die in Polynomialzeit entscheidbar sind.

Die Sprache L ist *in Exponentialzeit entscheidbar*, wenn es $c \in \mathbb{N}$ gibt, so daß L in Zeit $O\big(2^{n^c}\big)$ entschieden werden kann. Die Komplexitätsklasse $\mathsf{EXPTIME}$ umfaßt die Sprachen, die in Exponentialzeit entscheidbar sind.

Die Sprache L ist *in doppelter Exponentialzeit entscheidbar*, wenn es $c \in \mathbb{N}$ gibt, so daß L in Zeit $O\big(2^{2^{n^c}}\big)$ entschieden werden kann. Die Komplexitätsklasse $\mathsf{2EXPTIME}$ umfaßt die Sprachen, die in doppelter Exponentialzeit entscheidbar sind.

Die Sprache L ist *in dreifacher Exponentialzeit entscheidbar*, wenn es $c \in \mathbb{N}$ gibt, so daß L in Zeit $O\Big(2^{2^{2^{n^c}}}\Big)$ entschieden werden kann. Die Komplexitätsklasse $\mathsf{3EXPTIME}$ umfaßt die Sprachen, die in dreifacher Exponentialzeit entscheidbar sind.

$\vdots$

erweiterte Church-Turing-These *Eine Sprache $L \subseteq \Sigma^*$ kann genau dann durch einen effizienten Algorithmus entschieden werden, wenn L in Polynomialzeit entschieden werden kann.*

Bemerkung. Dies ist eine erste Näherung des Begriffs „effizient entscheidbar“, die aber sehr angreifbar ist:

- Ist ein Algorithmus, der Zeit n^{1000} benötigt, „effizient“?
- Er ist doch (für alle sinnvollen Eingabegrößen) langsamer als einer, der Zeit $1,00001^n$ benötigt.

Ihr Vorteil: Verwendet man ein anderes Algorithmenmodell (Mehrband-TM, Registermaschine, Random Access Machine, ...), so erhält man u.U. schnellere Algorithmen (Bsp.: jede Mehrband-TM kann in eine äquivalente det. TM übersetzt werden. Verwendet die Mehrband-TM Zeit $f(n)$, so verwendet die det. TM Zeit $f(n)^2$). Aber die Laufzeitunterschiede sind immer durch ein Polynom beschreibbar. Daher erhält man dieselben Klassen P, EXPTIME, ...

Beispiel 5.1. Wir betrachten aussagenlogische Formeln wie z.B.

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)\,.$$

Solche Formeln lassen sich z.B. durch Wörter über dem Alphabet $\Sigma = \{x, \vee, \wedge, \neg,), (\}$ kodieren (die atomare Formel x_i wird durch das Wort x^{i+1} kodiert).

Zur Erinnerung: Eine aussagenlogische Formel φ ist erfüllbar, falls es eine Belegung der atomaren Formeln mit $0, 1$ gibt, so daß die gesamte Formel sich zu 1 auswertet.

Das *Erfüllbarkeitsproblem* SAT $\subseteq \Sigma^*$ *der Aussagenlogik* ist die Menge der erfüllbaren aussagenlogischen Formeln.

Beobachtung: Die Sprache SAT ist in EXPTIME.

Begründung: Sei φ eine aussagenlogische Formel, in der höchtstens die atomaren Formeln $x_1, x_2, \ldots, x_n$ vorkommen.

Die Erfüllbarkeit kann getestet werden, indem jede Belegung der n atomaren Formeln mit 0 und 1 überprüft wird.

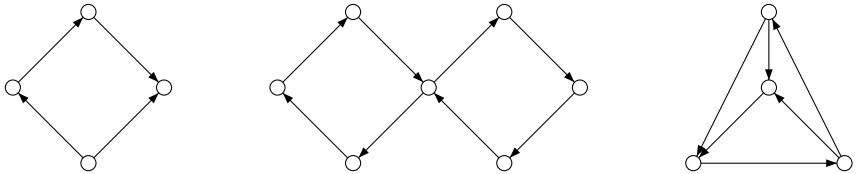
Die Überprüfung einer einzelnen Belegung kann in polynomieller Zeit ausgeführt werden. Da es 2^n viele Belegungen gibt, ist man insgesamt in exponentieller Zeit fertig. □

Beispiel 5.2. Wir betrachten gerichtete endliche Graphen $G = (V, E)$ (d.h. $E \subseteq V \times V$). Diese Graphen können über dem Alphabet $\Sigma = \{0, 1\}$ kodiert werden (ähnlich wie die Codes von Turingmaschinen)

Ein *Hamiltonkreis* ist eine Nummerierung der Knoten von G, so daß

- aufeinanderfolgende Knoten durch eine Kante verbunden sind und
- es eine Kante vom letzten zum ersten Knoten gibt.

Z.B. haben die ersten beiden Graphen keinen Hamiltonkreis, der letzte hingegen hat einen:



Das *Hamiltonizitätsproblem* DHC $\subseteq \Sigma^*$ ist die Menge der Graphen, die einen Hamiltonkreis besitzen.

Beobachtung: Die Sprache DHC ist in EXPTIME.

Begründung: Sei $G = (V, E)$ ein endlicher gerichteter Graph.

Die Existenz eines Hamiltonkreises kann getestet werden, indem jede der Aufzählungen der Knoten überprüft wird.

Die Überprüfung einer einzelnen Aufzählung kann in polynomieller Zeit ausgeführt werden. Da es $n! \leq 2^{n^2}$ viele Aufzählungen gibt, ist man insgesamt in exponentieller Zeit fertig. □

Beispiel 5.3. Wir betrachten ungerichtete endliche Graphen $G = (V, E)$ (d.h. $E \subseteq V \times V$ ist symmetrich). Diese Graphen können über dem Alphabet $\Sigma = \{0, 1\}$ kodiert werden (ähnlich wie die Codes von Turingmaschinen)

Eine *legale 3-Färbung* ist eine Abbildung der Knoten von G in die Menge $\{\text{rot}, \text{gelb}, \text{blau}\}$, so daß benachbarte Knoten unterschiedliche Farben bekommen.

Das *3-Färbbarkeitsproblem* $3\text{C} \subseteq \Sigma^*$ ist die Menge der Graphen, die eine legale 3-Färbung besitzen.

Beobachtung: Die Sprache 3C ist in EXPTIME.

Begründung: Sei $G = (V, E)$ ein endlicher ungerichteter Graph.

Die Existenz einer legalen 3-Färbung kann getestet werden, indem jede Färbung der Knoten überprüft wird.

Die Überprüfung einer einzelnen Färbung kann in polynomieller Zeit ausgeführt werden. Da es 3^n viele Färbungen gibt, ist man insgesamt in exponentieller Zeit fertig. □

Beispiel 5.4. Wir betrachten gerichtete endliche Graphen $G = (V, E)$ (d.h. $E \subseteq V \times V$), zusammen mit einer Abbildung $\ell\colon E \to \mathbb{N}_{>0}$ und einer Zahl $L \in \mathbb{N}$. Diese Graphen können über einem geeigneten Alphabet Σ kodiert werden.

Eine *Rundreise* ist ein Kreis in G, der alle Knoten enthält und insgesamt die Länge $\leq L$ hat (d.h. die Summe der Längen $\ell(e)$ aller im Kreis enthaltenen Kanten e).

Das *Rundreiseproblem* (neudeutsch: *Traveling Salesman Problem*) TSP $\subseteq \Sigma^*$ ist die Menge der Graphen, Längenabbildungen und maximalen Längen, die eine Rundreise besitzen.

Beobachtung: Die Sprache TSP ist in EXPTIME.

Begründung: Sei $G = (V, E)$ ein endlicher gerichteter Graph, $\ell\colon E \to \mathbb{N}_{>0}$ und $L \in \mathbb{N}$.

Die Existenz einer Rundreise kann getestet werden, indem alle Kreise der Länge $\leq L$ überprüft werden.

Die Überprüfung eines einzelnen Kreises kann in polynomieller Zeit ausgeführt werden. Da es $\leq |E|^L$ viele Kreise der Länge $\leq L$ gibt, ist man insgesamt in exponentieller Zeit fertig. □

Viele „natürliche“ Probleme sind also in EXPTIME. Damit sind sie entscheidbar, aber bisher haben wir keinen „effizienten“ Algorithmus für ihre Entscheidung angegeben.

Vielmehr hatten all unsere Beispielalgorithmen die Form „Überprüfe alle ...“, wobei

- es exponentiell viele zu überprüfende Kandidaten gab und
- jeder einzelne Kandidat in polynomieller Zeit überprüft werden konnte.

nichtdeterministischer „Algorithmus“:

- rate einen Kandidaten
- überprüfe diesen.

Dieser Algorithmus läuft in polynomieller Zeit.

Definition (nichtdeterministische Komplexitätsklassen)**.** Sei $f\colon \mathbb{N} \to \mathbb{N}$ eine monotone Funktion und L eine Sprache. Dann ist L *nichtdeterministisch in Zeit* $O(f(n))$ *entscheidbar*, wenn es eine nichtdet. TM M und $d, e \in \mathbb{N}$ gibt mit:

- M akzeptiert L, d.h. $L = L(M)$.
- Für jede Eingabe $w \in \Sigma^*$ hält M auf jeden Fall nach höchstens $d \cdot f(|w|) + e$ vielen Rechenschritten.
  (dabei kann es passieren, daß $w \in L$ liegt, die NTM M aber „falsch geraten“ hat und damit nicht in einer akzeptierenden Haltekonfiguration ist.)

Die Sprache L ist *nichtdeterministisch in Polynomialzeit entscheidbar*, wenn es $c \in \mathbb{N}$ gibt, so daß L nichtdeterministisch in Zeit $O(n^c)$ entschieden werden kann. Die Komplexitätsklasse NP umfaßt die Sprachen, die nichtdeterministisch in Polynomialzeit entscheidbar sind.

Die Sprache L ist *nichtdeterministisch in Exponentialzeit entscheidbar*, wenn es $c \in \mathbb{N}$ gibt, so daß L nichtdeterministisch in Zeit $O\left(2^{n^c}\right)$ entschieden werden kann. Die Komplexitätsklasse NEXPTIME umfaßt die Sprachen, die nichtdeterministisch in Exponentialzeit entscheidbar sind.

$\vdots$

Auf Seite 133 haben wir gesehen, daß jede NTM durch eine äquivalente det. TM ersetzt werden kann. Man kann zeigen, daß diese höchstens exponentiell mehr Zeit benötigt. Daher erhalten wir

$$\mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{EXPTIME} \subseteq \mathsf{NEXPTIME} \subseteq \mathsf{2EXPTIME} \cdots$$

Beispiel 5.1 (Fortsetzung). Die Sprache SAT gehört sogar zu NP.

Begründung: Sei φ eine aussagenlogische Formel, die höchstens die atomaren Formeln $x_1, x_2, \ldots, x_n$ verwendet.

eine NTM geht wie folgt vor:

- für alle $1 \leq i \leq n$ rate $b_i \in \{0, 1\}$ und ersetze alle Vorkommen der atomaren Formel x_i in φ durch das Bit b_i.
- Danach werte die Formel (die jetzt keine atomaren Formeln mehr enthält) aus.
- Ergibt sich dabei 1, so gehe in einen akzeptierenden Zustand und halte an ergibt sich 0, so gehe in einen nicht-akzeptierenden Zustand und halte an.

Es gilt: diese nichtdet. TM kann genau dann in einem akzeptierenden Zustand anhalten, wenn Formel φ erfüllbar ist. Sie erkennt also SAT.

Gleichzeitig wird sie garantiert in polynomieller Zeit anhalten. □

Beispiel 5.2 (Fortsetzung). Die Sprache DHC gehört sogar zu NP.

Begründung: Sei $G = (V, E)$ ein endlicher gerichteter Graph.

eine NTM geht wie folgt vor:

- Rate einen Nummerierung der Knoten.
- Überprüfe, ob aufeinanderfolgende Knoten durch eine Kante verbunden sind (und ebenso der letzte und der erste).
- Ist dies der Fall, so gehe in einen akzeptierenden Zustand und halte an
  ist es nicht der Fall, so gehe in einen nicht-akzeptierenden Zustand und halte an.

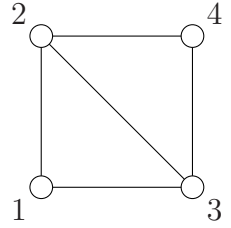
Es gilt: diese nichtdet. TM kann genau dann in einem akzeptierenden Zustand anhalten, wenn der Graph G einen Hamiltonkreis enthält. Sie erkennt also DHC.

Gleichzeitig wird sie garantiert in polynomieller Zeit anhalten. □

Beispiele 5.3 und 5.4 (Fortsetzung). Analog kann gezeigt werden, daß die Sprachen 3C und TSP zu NP gehören: Eine NTM rät eine „Lösung“ und prüft dann, daß es tatsächlich eine ist. Dies kann in polynomieller Zeit geschehen. □

Für den Fakt 3C $\in$ NP geben wir jetzt noch einen anderen Beweis an.

Sei $G = (V, E)$ ein endlicher ungerichteter Graph, z.B. der folgende:



Hieraus konstruieren wir eine aussagenlogische Formel φ_G, die genau dann erfüllbar ist, wenn G 3-färbbar ist. Dabei verwenden wir für jeden Knoten i die atomaren Formeln r_i, g_i und b_i (die bedeuten sollen, daß der Knoten i rot, gelb bzw. blau gefärbt wird). Die Formel φ_G ist dann die Konjunktion der folgenden Teilformeln:

- $r_i \vee g_i \vee b_i$ für alle $1 \leq i \leq 4$ („der Knoten i hat eine Farbe“)
- $\neg(r_i \wedge g_i) \wedge \neg(r_i \wedge b_i) \wedge \neg(g_i \wedge b_i)$ für alle $1 \leq i \leq 4$
  („der Knoten i hat nicht zwei Farben“)
- $\neg(r_i \wedge r_j) \wedge \neg(g_i \wedge g_j) \wedge \neg(b_i \wedge b_j)$ für alle Kanten (i, j)
  („benachbarte Knoten haben verschiedene Farben“)

Es gelten:

- Der Graph G ist genau dann 3-färbbar, wenn die Formel φ_G erfüllbar ist.
- Die Formel φ_G kann in polynomieller Zeit aus dem Graphen G berechnet werden.

Eine NTM, die in polynomieller Zeit 3C akzeptiert, geht also wie folgt vor:

1. Berechne aus dem Graphen G die Formel φ_G.
2. Starte eine NTM, die die Erfüllbarkeit von φ_G testet, d.h. die testet, ob $\varphi_G \in \text{SAT}$ gilt.

Damit haben wir tatsächlich $\text{3C} \in \mathsf{NP}$. □

Dieses Vorgehen kann verallgemeinert werden:

Definition. Seien $A, B \subseteq \Sigma^*$. Eine Funktion $f\colon \Sigma^* \to \Sigma^*$ heißt *Polynomialzeitreduktion von A auf B*, wenn gilt:

- Es gibt eine TM, die aus $w \in \Sigma^*$ in polynomieller Zeit $f(w)$ berechnet.
- Für alle $w \in \Sigma^*$ gilt $w \in A \iff f(w) \in B$.

Wir schreiben $A \leq_P B$ für „es gibt eine Polynomialzeitreduktion von A auf B“.

Wir haben gesehen, daß 3C $\leq_P$ SAT gilt. Aus der Polynomialzeitreduktion und einer NTM, die SAT in nichtdet. Polynomialzeit entscheidet, haben wir eine NTM konstruiert, die 3C in nichtdet. Polynomialzeit entscheidet. Allgemeiner erhält man analog zum Satz 4.4:

Satz 5.5. *Seien $A, B \subseteq \Sigma^*$ mit $A \leq_P B$.*

- *Ist $B \in \mathsf{P}$, so gilt auch $A \in \mathsf{P}$.*
- *Ist $B \in \mathsf{NP}$, so gilt auch $A \in \mathsf{NP}$.*

Zwischenzusammenfassung

- Wir hatten uns die Frage gestellt, ob alle entscheidbaren Probleme „effiziente“ Algorithmen besitzen. Im Angesicht der Komplexitätsklassen EXPTIME usw. schränken wir uns auf natürliche Probleme ein.
- Sehr viele natürliche Probleme liegen in der Komplexitätsklasse NP.
- Die Frage, ob es für diese „effiziente“ Algorithmen gibt (d.h. ob sie vielleicht in P liegen), haben wir nicht beantwortet.

Wir werden zeigen: Hat man für eines der genannten Probleme SAT etc. eine „effiziente“ Lösung, so hat man „effiziente“ Lösungen für alle Probleme in NP.

Im Umkehrschluß gilt dann: Kann man zeigen, daß irgendein Problem aus NP *keine* „effiziente“ Lösung hat, so hat auch SAT keine „effiziente“ Lösung (analog für die anderen genannten Probleme).

Satz 5.6. *Sei* $A \in \mathsf{NP}$. *Dann gilt* $A \leq_P \mathrm{SAT}$.

BewIdee:
Da $A \in \mathsf{NP}$, existiert eine NTM $M = (Q, \Sigma, \Gamma, \iota, \delta, \Box, F)$, die A in nichtdet. Polynomialzeit entscheidet.

- Also existieren $c, d, e \in \mathbb{N}$, so daß M bei Eingabe von w nach $\leq d \cdot |w|^c + e$ vielen Schritten anhält. Setze $p(n) = d \cdot n^c + e + n$.
- außerdem kann M bei Eingabe von w genau dann in einem akzeptierenden Zustand anhalten, wenn $w \in A$ gilt.

Wir drücken die Existenz einer akzeptierenden Berechnung bei Eingabe von w durch eine Formel φ_w aus. Hierzu benötigen wir sicher Aussagen der Forem „zum Zeitpunkt t befindet sich die NTM M im Zustand q“ oder „zum Zeitpunkt t steht an der Stelle i des Bandes der Buchstabe a“.

Diese Formel φ verwendet daher die folgenden atomaren Formeln:

Atomformel	Indizes	intendierte Bedeutung
$\text{zust}_{t,z}$	$0 \leq t \leq p(n)$ $z \in Z$	nach t Schritten befindet sich TM im Zustand z
$\text{pos}_{t,i}$	$0 \leq t \leq p(n)$ $-p(n) \leq i \leq p(n)$	nach t Schritten befindet sich Kopf auf Position i
$\text{band}_{t,i,a}$	$0 \leq t \leq p(n)$ $-p(n) \leq i \leq p(n)$ $a \in \Gamma$	nach t Schritten steht in Zelle i der Buchstabe a

Die Konstruktion erfolgt so, daß

- sie in Zeit polynomiell in $|w|$ ausgeführt werden kann und
- die erfüllenden Belegungen von φ_w den akzeptierenden Berechnungen bei Eingabe von w entsprechen. Also ist φ genau dann erfüllbar, wenn w von M akzeptiert wird.

Damit ist die Abbildung $w \mapsto \varphi_w$ eine Polynomialzeitreduktion von A auf SAT. $\square$

Aus den Sätzen 5.5 und 5.6 folgt unmittelbar:

Folgerung. *Wenn* SAT $\in$ P *gilt, so gilt* $\mathsf{NP} \subseteq \mathsf{P}$ *(und damit* $\mathsf{P} = \mathsf{NP}$*).*

Mit anderen Worten: Wenn es einen „effizienten“ Algorithmus für das Erfüllbarkeitsproblem SAT gibt, so gibt es „effiziente“ Algorithmen für alle Probleme aus NP.

Oder auch: Wenn es irgendein Problem in NP gibt, das keinen „effizienten“ Algorithmus besitzt, so gibt es auch keinen „effizienten“ Algorithmus für SAT.

Um also unsere Frage „Haben alle natürlichen Probleme einen effiziente Algorithmus?“ zu beantworten, braucht man nur das Erfüllbarkeitsproblem SAT zu untersuchen.

Gibt es vielleicht Alternativen (für diejenigen, die die Aussagenlogik nicht so sehr mögen)?

Zunächst eine Abstraktion:

Definition. Sei $B \subseteq \Sigma^*$.

- B ist NP-*schwer*, wenn $A \leq_P B$ für alle $A \in \mathsf{NP}$ gilt.
- B ist NP-*vollständig*, wenn B NP-schwer ist und $B \in \mathsf{NP}$ gilt.

Bemerkung.
- Nach Beispiel ... und Satz 5.6 ist das Erfüllbarkeitsproblem SAT NP-vollständig.
- Ist A NP-vollständig und sogar in P, so gilt $\mathsf{NP} \subseteq \mathsf{P}$.

Um Alternativen für das Erüllbarkeitsproblem SAT zu haben, suchen wir also andere NP-vollständige Probleme. Um zu zeigen, daß ein gegebenes Problem NP-vollständig ist, verwendet man oft die folgende Aussage:

Satz 5.7. *Sei A NP-vollständig und $A \leq_P B \in \mathsf{NP}$. Dann ist auch B NP-vollständig.*

Wir wissen bereits, daß die folgenden Problem in NP liegen:

- das Hamilton-Problem DHC
- das Färbbarkeitsproblem 3C
- das Rundreiseproblem TSP

Man kann zeigen:

$$\text{SAT} \leq_P \text{3C} \text{ und } \text{SAT} \leq_P \text{DHC} \leq_P \text{TSP}$$

Also sind auch diese Probleme nach obigem Satz NP-vollständig.

Lemma. DHC $\leq_P$ TSP

BewIdee:
Wir wollen eine Polynomialzeitreduktion von DHC auf TSP angeben. Dazu sei $G = (V, E)$ ein endlicher gerichteter Graph.
Sei $G' = (V, E')$ der vollständige gerichtete Graph auf den Knoten V von G. Für $(v, w) \in E'$ setze

$$\ell(v, w) = \begin{cases} 1 & \text{falls } (v, w) \in E \\ 2 & \text{sonst.} \end{cases}$$

Weiterhin sei $L = |V|$.

Die Konstruktion von (G', ℓ, L) aus G erfolgt so, daß

- sie in Zeit polynomiell in $|G|$ ausgeführt werden kann und
- die Hamiltonkreise von G genau den Rundreisen der Länge L in (G', ℓ) entsprechen. Da (G', ℓ) keine Rundreisen der Länge $< L$ erlaubt, gilt also $G \in \text{DHC} \iff (G', \ell, L) \in \text{TSP}$.

Damit ist die Abbildung $G \mapsto (G', \ell, L)$ eine Polynomialzeitreduktion von DHC auf TSP. □

Da DHC NP-vollständig ist und TSP in NP liegt, ist TSP also NP-vollständig.

Folgerung. *Wenn* TSP $\in$ P *gilt, so gilt* NP $\subseteq$ P *(und damit* P $=$ NP*).*

Zusammenfassung

- Die erweiterte Church-Turing These erlaubt es zu untersuchen, inwiefern Probleme durch Algorithmen *effizient* lösbar oder nicht *effizient* lösbar sind.
- viele natürliche Entscheidungsprobleme sind durch einen Algorithmus der Form „rate eine Lösung und teste in polynomieller Zeit, ob es tatsächlich eine ist“ lösbar, daher sind sie in NP
- Polynomialzeit-Reduktionen erlauben es, die Komplexität von Problemen miteinander in Beziehung zu setzen.
- Viele naürliche Probleme sind nicht nur in NP, sondern sogar NP-vollständig
- Ist eines davon in P, so hat jedes Problem aus NP einen deterministischen Polynomialzeitalgorithmus.

  Ob dies der Fall ist, ist nicht bekannt.