

# Die $P = NP$ Frage

Ideen der Informatik

Kurt Mehlhorn

Adrian Neumann



**mp** max planck institut  
informatik

- Ziele von Theorie
- Die  $P = NP$  Frage
  - $P$  = Menge der Probleme, bei denen eine Lösung in Polynomzeit gefunden werden kann.
  - $NP$  = Menge aller Probleme, bei denen die Korrektheit eines Lösungsvorschlags in Problemzeit überprüft werden kann.
- Das Erfüllbarkeitsproblem der Aussagenlogik (SATisfiability Problem): das prototypische Problem für  $NP$ .
- Satz von Cook-Levin:  $P = NP$  genau wenn  $SAT \in P$  .
- Was wäre, wenn  $P \neq NP$ ?
- Was wäre, wenn  $P = NP$ ?

## Ziel einer jeden Theorie: schaffe Einsicht und Ordnung

Hier:

- Erfüllbarkeitsproblem der Aussagenlogik, Rucksackproblem, Problem des Handlungsreisenden, Graphenfärbung und tausend andere Probleme sind alle gleich schwer: falls es für eines diese Probleme einen effizienten Algorithmus gibt, dann für alle.
- es gibt einen Polynomzeitalg für eines diese Probleme, wenn zwischen Beweisen und Überprüfen eines Beweises kein wesentlicher Unterschied besteht. Das ist die  $P = NP$  Frage.

# P = die Klasse der effizient lösbaren Probleme

---

P = Menge, der in Polynomzeit lösbaren Probleme

Ein Problem ist in P, wenn es eine Programm  $M$  und eine natürliche Zahl  $k$  gibt, so dass

- $M$  für jede Problemstellung  $x$  die richtige Antwort liefert, und
- seine Laufzeit an Eingabe  $x$  beschränkt ist durch  $c \cdot |x|^k$ ; dabei ist  $|x|$  die Länge von  $x$  (Anzahl der Zeichen) und  $c$  eine Konstante (die nicht von  $x$  abhängt).



# P = die Klasse der effizient lösbaren Probleme

---

P = Menge, der in Polynomzeit lösbaren Probleme

Ein Problem ist in P, wenn es eine Programm  $M$  und eine natürliche Zahl  $k$  gibt, so dass

- $M$  für jede Problemstellung  $x$  die richtige Antwort liefert, und
- seine Laufzeit an Eingabe  $x$  beschränkt ist durch  $c \cdot |x|^k$ ; dabei ist  $|x|$  die Länge von  $x$  (Anzahl der Zeichen) und  $c$  eine Konstante (die nicht von  $x$  abhängt).

$k = 1$ , lineare Laufzeit,  $k = 2$ , quadratische Laufzeit,...

Beispiele: kürzeste Wege, Sortieren, Lösen von linearen Gleichungen, ...



# P = die Klasse der effizient lösbaren Probleme

P = Menge, der in Polynomzeit lösbaren Probleme

Ein Problem ist in P, wenn es eine Programm  $M$  und eine natürliche Zahl  $k$  gibt, so dass

- $M$  für jede Problemstellung  $x$  die richtige Antwort liefert, und
- seine Laufzeit an Eingabe  $x$  beschränkt ist durch  $c \cdot |x|^k$ ; dabei ist  $|x|$  die Länge von  $x$  (Anzahl der Zeichen) und  $c$  eine Konstante (die nicht von  $x$  abhängt).

Glaubenssatz der Informatik: nur Algorithmen mit polynomieller Laufzeit sind gut.

Verdoppelung der Problemgröße vergrößert Laufzeitschranke von

$$c \cdot |x|^k \quad \text{auf} \quad c \cdot (2|x|)^k = 2^k \cdot c \cdot |x|^k,$$

also nur um den Faktor  $2^k$ .  $k = 1$ , Faktor 2,  $k = 2$ , Faktor 4, ... Bei linearer Laufzeit verdoppelt sich die in 1h lösbare Problemgröße alle zwei Jahre.



# P = die Klasse der effizient lösbaren Probleme

## P = Menge, der in Polynomzeit lösbaren Probleme

Ein Problem ist in P, wenn es eine Programm  $M$  und eine natürliche Zahl  $k$  gibt, so dass

- $M$  für jede Problemstellung  $x$  die richtige Antwort liefert, und
- seine Laufzeit an Eingabe  $x$  beschränkt ist durch  $c \cdot |x|^k$ ; dabei ist  $|x|$  die Länge von  $x$  (Anzahl der Zeichen) und  $c$  eine Konstante (die nicht von  $x$  abhängt).

**exponentielle Laufzeit  $c \cdot a^{|x|}$ :** Vergrößerung der Problemgröße um Eins vergrößert die Laufzeitschranke von

$$c \cdot a^{|x|} \text{ auf } c \cdot a^{|x|+1} = a \cdot (c \cdot a^{|x|}),$$

d.h. um den Faktor  $a$ .

$a = 2$ : Verdoppelung der Laufzeit



- Wir haben einen Algorithmus, dessen Laufzeit an der Eingabe  $x$  durch  $|x|^3$  beschränkt ist.  
Verdoppelung der Eingabegröße erhöht die Laufzeitschranke um nicht mehr als den Faktor 3, 8, oder 9?
- Wir haben einen Algorithmus, dessen Laufzeit an der Eingabe  $x$  linear in  $|x|$  wächst. Nehmen wir ferner an, dass sich die Geschwindigkeit von Rechnern jedes Jahr verdoppelt.  
Wenn wir heute Eingaben mit  $|x| \leq 10^4$  in 1h bearbeiten können, was geht dann in 4 Jahren?
- Wir haben einen Algorithmus mit der Laufzeit  $c \cdot 2^{|x|}$  für eine Konstante  $c$ . Wenn wir heute Eingaben mit  $|x| \leq 10^4$  in 1h bearbeiten können, was geht dann in 4 Jahren?



NP = Menge aller Probleme, bei denen die Korrektheit eines Lösungsvorschlags in Problemzeit überprüft werden kann.

### Rucksackproblem

Gegeben sind  $n$  Objekte und zwei Zahlen  $G$  und  $W$ . Das  $i$ -te Object hat Gewicht  $g_i$  und Wert  $w_i$ .

Gibt es eine Teilmenge der Objekte, die Gesamtgewicht höchstens  $G$  und Wert mindestens  $W$  hat?

Lösungsvorschlag:

Überprüfung.

## Problem des Handlungsreisenden

Gibt es eine Tour durch die alle Orte Deutschlands mit mehr als 5 Tausend Einwohnern, die höchstens 4000 Kilometer lang ist?

Allgemein: Gegeben ist ein Graph, für jede Kante ihre Länge, und eine Zahl  $L$ .

Gibt es eine Rundreise, die alle Knoten besucht und Länge höchstens  $L$  hat?

Lösungsvorschlag:

Überprüfung.

Zeigen sie, dass das **Cliquenproblem** in NP ist:

Eingabe: Graph  $G$  und eine Zahl  $k$

Frage: gibt es Teilmenge von  $k$  Knoten, in der jeder Knoten mit jedem anderen verbunden ist?

Was ist ein Lösungsvorschlag?

Wie überprüfen sie einen Lösungsvorschlag?

## Das Erfüllbarkeitsproblem (SATisfiability-Problem)

**Eingabe:** Eine Formel der Aussagenlogik

**Frage:** Ist die Formel erfüllbar, d.h., gibt es eine Belegung der Variablen mit Wahrheitswerten Wahr (W) und Falsch (F), die die Formel erfüllt?

Formeln der Aussagenlogik: Wahrheitswerte und Variablen verknüpft mit und ( $\wedge$ ), oder ( $\vee$ ) und Negation ( $\neg$ ). Details auf nächster Folie.

## Beispiel

Formel:  $(x \vee y) \wedge \neg x$

Belegung 1:  $x \rightarrow W, y \rightarrow F$ , dann  $(W \vee F) \wedge \neg W = W \wedge F = F$

Belegung 2:  $x \rightarrow F, y \rightarrow W$ , dann  $(F \vee W) \wedge \neg F = W \wedge W = W$

## Formeln der Aussagenlogik

- (1)  $W$  (true, wahr),  $F$  (falsch, false) und Variablen sind Formeln.
- (2) Wenn  $F$  und  $G$  Formeln sind, dann auch  $(F \wedge G)$ ,  $(F \vee G)$ , und  $\neg F$ .

## Belegung, Wert einer Formel, erfüllbar

Eine Belegung weist jeder Variablen einen Wahrheitswert zu.  
Der Wert der Formel ergibt sich nach folgenden Regeln:

$x$	$y$	$x \vee y$	$x \wedge y$	$\neg x$
$F$	$F$	$F$	$F$	$W$
$F$	$W$	$W$	$F$	$W$
$W$	$F$	$W$	$F$	$F$
$W$	$W$	$W$	$W$	$F$

Eine Formel ist erfüllbar, wenn es eine Belegung gibt, unter der sie den Wert wahr erhält.

# Algorithmen für SAT, Rucksack, Handlungsreisender,

...

SAT: probiere alle Belegungen durch und finde heraus, ob es eine erfüllende gibt.

Bei  $n$  Variablen gibt es  $2^n$  mögliche Belegungen.

Rucksack: probiere alle Teilmengen der Objekte durch und ....

Bei  $n$  Variablen gibt es  $2^n$  mögliche Belegungen.

Handlungsreisender: probiere alle möglichen Touren aus und ....

Bei  $n$  Städten gibt es  $n!$  mögliche Rundreisen.

Alle bekannten Algorithmen haben exponentielle Laufzeit im schlechtesten Fall.



## Aufgaben

---

Gib eine Formel in den Variablen  $x$ ,  $y$  und  $z$  an, die genau dann wahr ist, wenn mindestens eine der Variablen wahr ist.

Gib eine Formel in den Variablen  $x$ ,  $y$  und  $z$  an, die genau dann wahr ist, wenn mindestens zwei der Variablen wahr sind.

Gib eine Formel in den Variablen  $x$ ,  $y$  und  $z$  an, die genau dann wahr ist, wenn genau eine der Variablen wahr ist.

Zeigen Sie:  $\text{SAT} \in \text{NP}$ .

Lösungsvorschlag:

Überprüfung:



# Der Satz von Cook/Levin

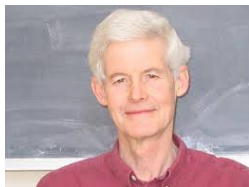
NP = Menge aller Probleme, bei denen die Korrektheit eines Lösungsvorschlags in Problemzeit überprüft werden kann.

Satz (Stephen Cook und Leonid Levin, 71)

$P = NP$  genau wenn  $SAT \in P$

Ich zeige:  $SAT \in P$  impliziert Graphenfärbung  $\in P$ .

Theorie schafft Einsicht.



Cook: Turing Award  
Levin: Knuth Prize.



## Graphenfärbung (mit drei Farben)

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

**Frage:** Gibt es eine Färbung der Knoten von  $G$  mit den Farben rot, blau, und grün, so dass die Endpunkte einer jeden Kante verschieden gefärbt sind?

## Graphenfärbung (mit drei Farben)

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

**Frage:** Gibt es eine Färbung der Knoten von  $G$  mit den Farben rot, blau, und grün, so dass die Endpunkte einer jeden Kante verschieden gefärbt sind?

**Satz:** Falls SAT effizient lösbar ist, dann ist auch Graphenfärbung effizient lösbar (Färbung  $\leq$  SAT).

Gegeben ein Graph  $G$ , konstruiere eine Formel  $F$  mit:

- $G$  ist dreifärbbar genau wenn  $F$  erfüllbar ist.
- die Konstruktion von  $F$  aus  $G$  ist in Polynomzeit durchführbar.

Satz: Falls SAT effizient lösbar ist, dann ist auch Graphenfärbung effizient lösbar (Färbung  $\leq$  SAT).

Gegeben ein Graph  $G$ , konstruiere eine Formel  $F$  mit:

- $G$  ist dreifärbbar genau wenn  $F$  erfüllbar ist.
- die Konstruktion von  $F$  aus  $G$  ist in Polynomzeit durchführbar.

Färbungsalgorithmus: an Eingabe  $G$  tue:

- Konstruiere  $F$ .
- Entscheide Erfüllbarkeit von  $F$  mit Hilfe des SAT-Algorithmus.

Alg ist korrekt und effizient.

Variable  $u_c$  für Knoten  $u \in V$  und Farbe  $c \in \{R, B, G\}$ .

Intendierte Bedeutung:  $u_c = W$  bedeutet  $u$  hat die Farbe  $c$ .

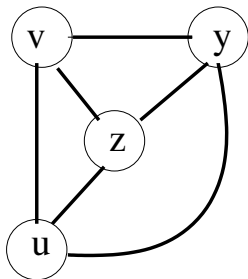
Was müssen wir ausdrücken?

- Jeder Knoten hat eine eindeutige Farbe:  $\bigwedge_{u \in V} GE(u_R, u_B, u_G)$ .  
Dabei ist  $GE(x, y, z) = (x \vee y \vee z) \wedge \neg(xy \vee xz \vee yz)$  GenauEine
- Nicht, es gibt eine Kante  $\{u, v\}$ , so dass  $u$  und  $v$  die gleiche Farbe haben.  $\neg \left( \bigvee_{\{u,v\} \in E} u \text{ und } v \text{ haben die gleiche Farbe} \right)$ .
- $u$  und  $v$  haben die gleiche Farbe:  $u_R v_R \vee u_B v_B \vee u_G v_G$ .

Insgesamt

$$\bigwedge_{u \in V} GE(u_R, u_B, u_G) \wedge \neg \bigvee_{\{u,v\} \in E} (u_R v_R \vee u_B v_B \vee u_G v_G)$$

## Aufgabe



- Ist der Graph dreifärbbar? JA oder NEIN?
- Gib die Formel für die Existenz einer Dreifärbung an.
- Streiche eine beliebige Kante und ändere die Formel entsprechend ab.
- Finde eine erfüllende Belegung und leite daraus die Dreifärbung ab.

## Definition

Ein Problem  $L$  in NP ist NP-vollständig, wenn aus  $L \in P$  folgt  $P = NP$ .

Cook-Levine bewiesen, dass das Erfüllbarkeitsproblem NP-vollständig ist.

## Satz (Karp, 1972)

*Das Graphenfärbungsproblem, das Hamiltonsche Kreisproblem, Knapsack, und 20 andere Probleme sind NP-vollständig.*

Die Liste ist inzwischen auf mehrere Tausend angewachsen: Theorie schafft Ordnung.

Richard Karp, Turing-Award in 1985.



# Die P = NP Frage (Geschichte)

- P = Menge aller Probleme, die in Polynomzeit lösbar sind
- NP = Menge aller Probleme, bei denen die Korrektheit eines Lösungsvorschlags in Problemzeit überprüft werden kann.
- 1970: man hatte effiziente Algorithmen für kürzeste Wege, Paarungsprobleme, Flussprobleme, aber es gab auch viele Probleme, die man nicht effizient lösen konnte.
- 71,72: Cook, Levin und Karp haben Ordnung in dieses Chaos gebracht. Viele dieser Probleme sind zu SAT äquivalent. SAT  $\in$  P impliziert P = NP.
- Clay Foundation gibt 1 Mio \$ Preisgeld für Lösung eines der 6 großen offenen Probleme der Mathematik
- Frage hat grundlegende philosophische/mathematische Bedeutung (ist Beweisen schwerer als Prüfen?)  
Gleichheit hätte enorme algorithmische Konsequenzen



- es würde sich nicht viel ändern.
- da wir keinen Polynomzeitalgorithmus für das Erfüllbarkeitsproblem kennen, leben wir faktisch in einer Welt, in der  $P$  ungleich  $NP$  ist.
- die meisten Fachleute glauben, dass  $P \neq NP$ ?
- aber: im Augenblick gibt es keinen Ansatz, wie man  $P \neq NP$  beweisen könnte. Man weiß nur, dass einige natürliche Ansätze NICHT funktionieren können.

Wenn man einen Beweis findet, muss dieser eine neue Methode einführen. Diese Methode könnte weitere Anwendungen haben.

- alle paar Jahre wird ein (falscher) Beweis angekündigt.



# War wäre, wenn $P = NP$ ?

- das wäre eine Revolution
- wir hätten Polynomzeitalgorithmen für Erfüllbarkeit, . . . ,
- **Mathematiker würden arbeitslos:**  
**Input:** ein mathematischer Satz  $S$ , eine Anzahl  $n$  unbeschriebener Blätter  
**Frage:** gibt es einen Beweis für  $S$  (in einem formalen System), der auf die  $n$  Blätter passt?  
Dieses Problem ist in NP. Falls  $P = NP$ , dann ist dieses Problem in P.
- Philosophen müssten neu über den Begriff Kreativität nachdenken.
- alle paar Monate wird ein (falscher) Beweis angekündigt.



# Wie geht man mit NP-Vollständigkeit um?

---

Nur weil ein Problem schwer ist, verschwindet es nicht.

NP-Vollständigkeit bedeutet: man kennt keinen Algorithmus, der **jede** Problemstellung in Polynomzeit löst. Es kann durchaus Algorithmen geben, die viele (interessante) Instanzen in Polynomzeit lösen.

Folgende Ansätze gibt es:

- Heuristiken
- Exakte Algorithmen für kleine  $n$
- Spezialfälle
- Approximationsalgorithmen



- $P$  = Menge der Probleme, bei denen eine Lösung in Polynomzeit gefunden werden kann
- $NP$  = Menge aller Probleme, bei denen die Korrektheit eines Lösungsvorschlags in Problemzeit überprüft werden kann.
- $P = NP$  genau wenn es einen polynomiellen Algorithmus für das Erfüllbarkeitsproblem der Aussagenlogik (SATisfiability Problem) gibt.
- $P = NP$ , eines der großen offenen Probleme der Informatik/Mathematik (Clay Prize)
- Falls  $P \neq NP$ , dann ...
- Falls  $P = NP$ , dann ...