



# Theoretische Grundlagen der Informatik

Vorlesung am 02.12.2021

Torsten Ueckerdt | 2. Dezember 2021

# Letzte Vorlesung

Entscheidungsproblem  $\Pi$   
 ist  **$\mathcal{NP}$ -vollständig** wenn

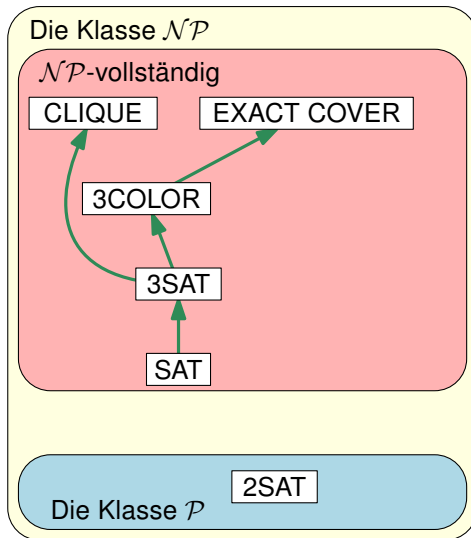
- $\Pi \in \mathcal{NP}$
- $\Pi' \propto \Pi$  für ein  $\mathcal{NP}$ -vollständiges Problem  $\Pi'$

	$\Pi'$	$\propto$	$\Pi$
	bekannt $\mathcal{NP}$ -vollständig (wähle ähnlich zu $\Pi$ )		$\mathcal{NP}$ -vollständig zu zeigen
	beliebige Instanzen	$\xrightarrow{f}$	spezielle Instanzen (größer, aber noch polynomial)
	Ja-Instanzen	$\longleftrightarrow$	Ja-Instanzen

- Die  $\mathcal{NP}$ -vollständigen Probleme sind die “schwierigsten” Probleme in  $\mathcal{NP}$ .
- Solange  $\mathcal{P} \neq \mathcal{NP}$  (wovon wir ausgehen), kann ein  $\mathcal{NP}$ -vollständiges Problem nicht von einer DTM in polynomialer Laufzeit entschieden werden.

# Der Plan

- EXACT COVER ist  $\mathcal{NP}$ -vollständig
- $\rightsquigarrow$  EXACT COVER  $\in \mathcal{NP}$
- $\rightsquigarrow$  3COLOR  $\propto$  EXACT COVER



# Das Problem EXACT COVER

## Problem EXACT COVER

**Gegeben:** Eine endliche Menge  $X$  und eine Familie  $\mathcal{S}$  von Teilmengen von  $X$ .

**Frage:** Existiert eine Menge  $\mathcal{S}' \subseteq \mathcal{S}$ , so dass jedes Element aus  $X$  in genau einer Menge aus  $\mathcal{S}'$  liegt?

# Das Problem EXACT COVER

## Problem EXACT COVER

**Gegeben:** Eine endliche Menge  $X$  und eine Familie  $\mathcal{S}$  von Teilmengen von  $X$ .

**Frage:** Existiert eine Menge  $\mathcal{S}' \subseteq \mathcal{S}$ , so dass jedes Element aus  $X$  in genau einer Menge aus  $\mathcal{S}'$  liegt?

### Beispiel:

$$X = \{1, 2, \dots, 7\}$$

$$\mathcal{S} = \{\{1, 2, 3\}, \{1, 2, 4\}, \{2, 3, 4\}, \{1, 3, 4\}, \{1, 5\}, \{3, 5\}, \{1, 3\}, \\ \{5, 6, 7\}, \{4, 5, 6\}, \{4, 5, 7\}, \{4, 6, 7\}, \{5, 6\}, \{5, 7\}, \{6, 7\}\}$$

Ist  $(X, \mathcal{S})$  eine Ja-Instanz?

# Das Problem EXACT COVER

## Problem EXACT COVER

**Gegeben:** Eine endliche Menge  $X$  und eine Familie  $\mathcal{S}$  von Teilmengen von  $X$ .

**Frage:** Existiert eine Menge  $\mathcal{S}' \subseteq \mathcal{S}$ , so dass jedes Element aus  $X$  in genau einer Menge aus  $\mathcal{S}'$  liegt?

### Beispiel:

$$X = \{1, 2, \dots, 7\}$$

$$\mathcal{S} = \{\{1, 2, 3\}, \{1, 2, 4\}, \{2, 3, 4\}, \{1, 3, 4\}, \{1, 5\}, \{3, 5\}, \{1, 3\}, \\ \{5, 6, 7\}, \{4, 5, 6\}, \{4, 5, 7\}, \{4, 6, 7\}, \{5, 6\}, \{5, 7\}, \{6, 7\}\}$$

$$\mathcal{S}' = \{\{1, 5\}, \{2, 3, 4\}, \{6, 7\}\}$$

Ist  $(X, \mathcal{S})$  eine Ja-Instanz? **Ja.**

# Das Problem EXACT COVER

## Problem EXACT COVER

**Gegeben:** Eine endliche Menge  $X$  und eine Familie  $\mathcal{S}$  von Teilmengen von  $X$ .

**Frage:** Existiert eine Menge  $\mathcal{S}' \subseteq \mathcal{S}$ , so dass jedes Element aus  $X$  in genau einer Menge aus  $\mathcal{S}'$  liegt?

## Satz.

Problem EXACT COVER ist  $\mathcal{NP}$ -vollständig.

# Beweis: $\mathcal{NP}$ -Vollständigkeit von EXACT COVER

EXACT COVER  $\in \mathcal{NP}$

- Es kann in Polynomialzeit überprüft werden, ob eine Teilmenge  $\mathcal{S}' \subseteq \mathcal{S}$  aus disjunkten Mengen besteht und  $X$  überdeckt.



# Beweis: $\mathcal{NP}$ -Vollständigkeit von EXACT COVER

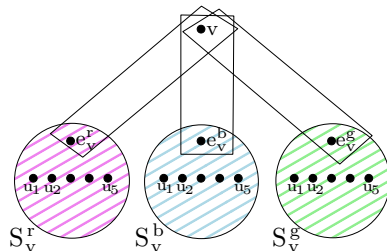
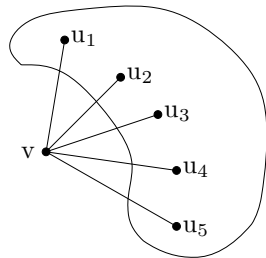
## 3COLOR $\propto$ EXACT COVER

- Sei  $G = (V, E)$  eine 3COLOR-Instanz.
- Wir konstruieren in Polynomialzeit eine EXACT COVER-Instanz  $(X, S)$ .
- Es soll gelten:  $G$  ist 3-färbbar  $\Leftrightarrow (X, S)$  hat exakte Überdeckung

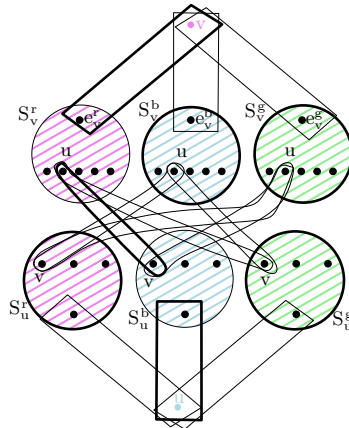
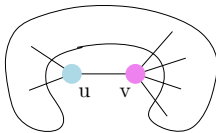
# Konstruktion von $(X, S)$

- Sei  $C = \{\text{rot, blau, grün}\}$ .
- Für jedes  $v \in V$ :
  - ein "Element"  $v$  in  $X$  und  $3 \cdot |N(v)| + 3$  zusätzliche Elemente,
  - drei disjunkte Mengen  $S_v^r, S_v^b, S_v^g$  in  $S$  mit jeweils  $|N(v)| + 1$  Elementen,
  - drei Mengen  $\{v, e_v^r\}, \{v, e_v^b\}$  und  $\{v, e_v^g\}$  mit  $e_v^r \in S_v^r, e_v^b \in S_v^b$  und  $e_v^g \in S_v^g$

**Interpretation:**  $S_v^r$  entspricht der "Farbe" **rot**, enthält für jeden Knoten aus  $N(v)$  eine Kopie und einen zusätzlichen Knoten  $e_v^r$ .



# Konstruktion von $(X, S)$



- Außerdem enthält  $S$  für jede Kante  $\{u, v\} \in E$  und je zwei  $c, c' \in C, c \neq c'$ , die zweielementigen Mengen  $\{u_v^c, v_u^{c'}\}$ ,  $u_v^c \in S_v^c$  "Kopie" von  $u$ ,  $v_u^{c'} \in S_u^{c'}$  "Kopie" von  $v$ .

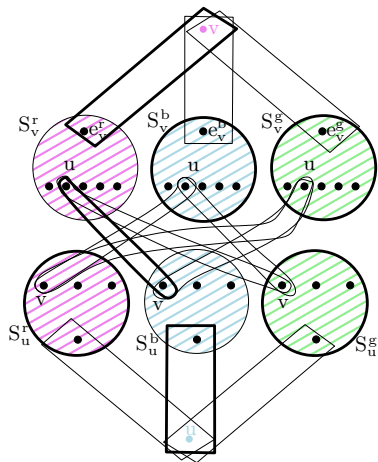
## Konstruktion von $(X, S)$

- Die Konstruktion ist polynomial.

Noch zu zeigen:

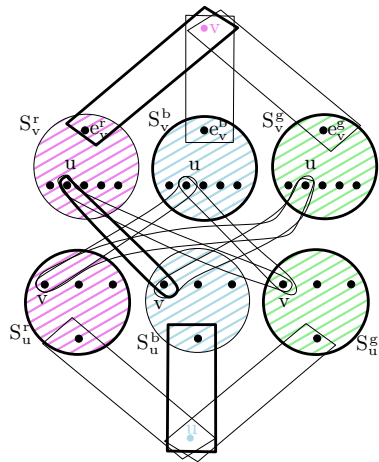
- $G$  ist 3-färbbar  $\Leftrightarrow (X, S)$  hat exakte Überdeckung

## $G$ 3-färbbar $\Rightarrow (X, S)$ hat exakte Überdeckung



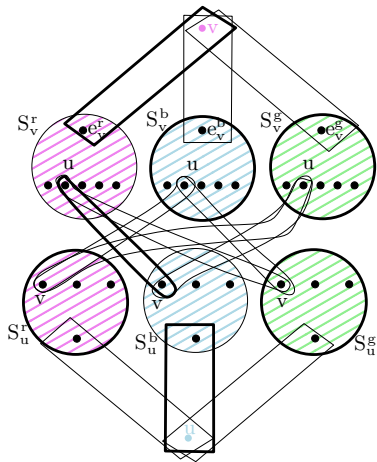
- Sei  $\chi : V \rightarrow C$  eine zulässige Dreifärbung.
- Nehme für jeden Knoten  $v \in V$  die Mengen  $\{v, e_v^{\chi(v)}\}$  und  $S_v^c$  mit  $c \neq \chi(v)$ .
- Nehme für jede Kante  $\{u, v\} \in E$  die Menge  $\{u_v^{\chi(v)}, v_u^{\chi(u)}\}$ . Diese Menge existiert, da  $\chi(u) \neq \chi(v)$ .
- Diese Mengen  $S'$  überdecken jedes Element aus  $X$  genau einmal.

## $G$ 3-färbbar $\Leftrightarrow (X, S)$ hat exakte Überdeckung



- Sei also  $S'$  eine exakte Überdeckung.
- Jedes Element  $v$  muss von genau einer Menge der Form  $\{v, e_v^c\}$  überdeckt sein.
- Dies induziert eine Färbung  $\chi$  von  $G$  mit rot, blau, grün.
- Wir müssen beweisen, dass diese Färbung zulässig ist
- Da für jedes  $v$  bereits  $\{v, e_v^{\chi(v)}\} \in S'$ , kann  $e_v^c$  mit  $c \neq \chi(v)$  nur durch die Menge  $S_v^c$  überdeckt werden.

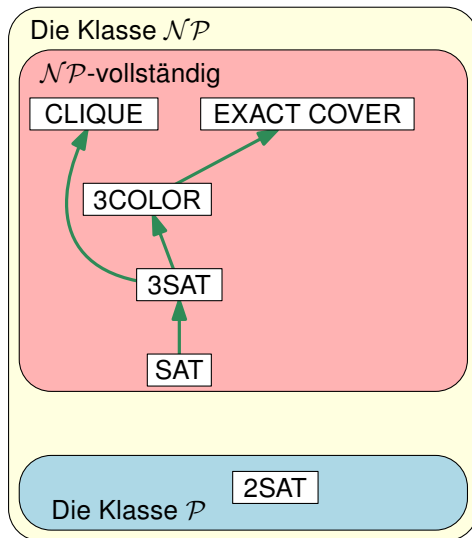
## $G$ 3-färbbar $\Leftrightarrow (X, S)$ hat exakte Überdeckung



- Da für jedes  $v$  bereits  $\{v, e_v^{\chi(v)}\} \in S'$ , kann  $e_v^c$  mit  $c \neq \chi(v)$  nur durch die Menge  $S_v^c$  überdeckt werden.
- Da die Mengen der Form  $\{v, e_v^{\chi(v)}\}$  und  $S_v^c$ ,  $c \neq \chi(v)$ , alle Elemente außer den  $u_v^{\chi(v)}$  mit  $\{u, v\} \in E$  überdecken, müssen auch die Mengen  $\{u_v^{\chi(v)}, v_u^{\chi(u)}\}$  für  $\{u, v\} \in E$  in  $S'$  enthalten sein.
- Für diese gilt per Konstruktion  $\chi(v) \neq \chi(u)$ .

## Der Plan

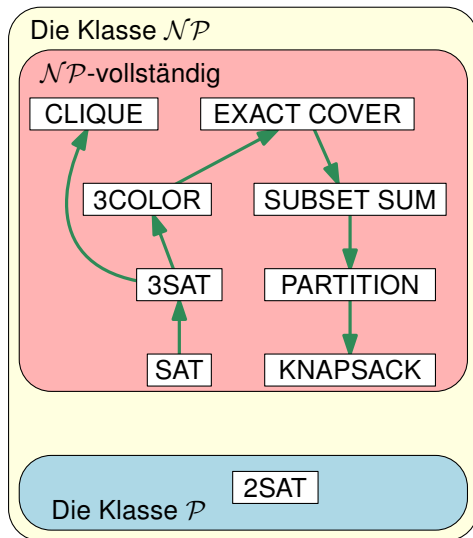
- EXACT COVER ist  $\mathcal{NP}$ -vollständig
- $\rightsquigarrow$  EXACT COVER  $\in \mathcal{NP}$
- $\rightsquigarrow$  3COLOR  $\propto$  EXACT COVER





## Der Plan

- EXACT COVER ist  $\mathcal{NP}$ -vollständig
  - $\rightsquigarrow$  EXACT COVER  $\in \mathcal{NP}$
  - $\rightsquigarrow$  3COLOR  $\propto$  EXACT COVER
- SUBSET SUM ist  $\mathcal{NP}$ -vollständig
  - $\rightsquigarrow$  SUBSET SUM  $\in \mathcal{NP}$
  - $\rightsquigarrow$  EXACT COVER  $\propto$  SUBSET SUM
- PARTITION ist  $\mathcal{NP}$ -vollständig
  - $\rightsquigarrow$  PARTITION  $\in \mathcal{NP}$
  - $\rightsquigarrow$  SUBSET SUM  $\propto$  PARTITION



- KNAPSACK ist  $\mathcal{NP}$ -vollständig
  - $\rightsquigarrow$  KNAPSACK  $\in \mathcal{NP}$
  - $\rightsquigarrow$  PARTITION  $\propto$  KNAPSACK

# Das Problem SUBSET SUM

## Problem SUBSET SUM

**Gegeben:** Eine endliche Menge  $M$ ,  
eine Gewichtsfunktion  $w: M \rightarrow \mathbb{N}_0$ ,  
eine Zahl  $K \in \mathbb{N}_0$

**Frage:** Existiert eine Teilmenge  $M' \subseteq M$  mit  $\sum_{a \in M'} w(a) = K$  ?

Satz.

Problem SUBSET SUM ist  $\mathcal{NP}$ -vollständig.

# Das Problem SUBSET SUM

## Problem SUBSET SUM

**Gegeben:** Eine endliche Menge  $M$ ,  
eine Gewichtsfunktion  $w: M \rightarrow \mathbb{N}_0$ ,  
eine Zahl  $K \in \mathbb{N}_0$

**Frage:** Existiert eine Teilmenge  $M' \subseteq M$  mit  $\sum_{a \in M'} w(a) = K$  ?

## Satz.

Problem SUBSET SUM ist  $\mathcal{NP}$ -vollständig.

## Notation:

$$w(X) := \sum_{a \in X} w(a)$$

# Das Problem SUBSET SUM

## Problem SUBSET SUM

**Gegeben:** Eine endliche Menge  $M$ ,  
eine Gewichtsfunktion  $w: M \rightarrow \mathbb{N}_0$ ,  
eine Zahl  $K \in \mathbb{N}_0$

**Frage:** Existiert eine Teilmenge  $M' \subseteq M$  mit  $w(M') = K$  ?

Satz.

Problem SUBSET SUM ist  $\mathcal{NP}$ -vollständig.

**Notation:**

$$w(X) := \sum_{a \in X} w(a)$$

# $\mathcal{NP}$ -Vollständigkeit von SUBSET SUM

SUBSET SUM  $\in \mathcal{NP}$ .

- Es kann für eine gegebene Teilmenge  $M' \subseteq M$  in Polynomialzeit der Wert  $w(M') = \sum_{a \in M'} w(a)$  ausgerechnet und mit  $K$  verglichen werden.

## Beweis: $\mathcal{NP}$ -Vollständigkeit von SUBSET SUM

EXACT COVER  $\propto$  SUBSET SUM

- Sei  $(X = \{0, 1, \dots, m-1\}, \mathcal{S})$  eine beliebige EXACT COVER-Instanz.
- Konstruiere SUBSET SUM Instanz  $(M, w, K)$

$$M := \mathcal{S}$$

$$\#x := \#\{Y \in \mathcal{S} \mid x \in Y\}$$

$$p := \max\{\#x + 1 \mid x \in X\}$$

$$w(x) := p^x$$

$$K := w(X) = \sum_{x=0}^{m-1} p^x$$

- Die Konstruktion benötigt nur Polynomialzeit.

## Beweis: $\mathcal{NP}$ -Vollständigkeit von SUBSET SUM

$$M := S$$

$$\#x := \#\{Y \in S \mid x \in Y\}$$

$$p := \max\{\#x + 1 \mid x \in X\}$$

$$w(x) := p^x$$

$$K := w(X) = \sum_{x=0}^{m-1} p^x$$

Beispiel:

$$X = \{0, \dots, 6\}$$

$$Y = \{0, 2, 5\} \in S$$

$$\hookrightarrow w(Y) = p^0 + p^2 + p^5$$

$$\hookrightarrow w(Y) = 0100101_p$$

$$K = 1111111_p$$

### Veranschaulichung:

- Wir stellen die Mengenzugehörigkeiten als Zahlen zur Basis  $p$  dar.
- Kodiere  $w(Y)$  für  $Y \in S$  als 01-String der Länge  $m$ , wobei an  $(m - i)$ -ter Stelle eine 1 steht genau dann, wenn  $i \in Y$ ;
- entsprechend ist  $K$  ein String der Länge  $m$  aus Einsen.

## Beweis: $\mathcal{NP}$ -Vollständigkeit von SUBSET SUM

$$M := S$$

$$\#x := \#\{Y \in S \mid x \in Y\}$$

$$p := \max\{\#x + 1 \mid x \in X\}$$

$$w(x) := p^x$$

$$K := w(X) = \sum_{x=0}^{m-1} p^x$$

Beispiel:

$$X = \{0, \dots, 6\}$$

$$Y = \{0, 2, 5\} \in S$$

$$\hookrightarrow w(Y) = p^0 + p^2 + p^5$$

$$\hookrightarrow w(Y) = 0100101_p$$

$$K = 1111111_p$$

### Veranschaulichung:

- Komponentenweise Addition der Strings  $w(Y_1), \dots, w(Y_n)$  ergibt einen String der Länge  $m$ , an dessen  $(m - i)$ -ter Stelle steht in wievielen der  $Y_j, j = 1, \dots, n$ , das Element  $i$  vorkommt.
- $\sum_{Y \in S'} w(Y) = K$  bedeutet also, dass jedes  $x \in X$  in genau einem  $Y \in S'$  vorkommt.



## Beweis: $\mathcal{NP}$ -Vollständigkeit von SUBSET SUM

$$M := S$$

$$\#x := \#\{Y \in S \mid x \in Y\}$$

$$p := \max\{\#x + 1 \mid x \in X\}$$

$$w(x) := p^x$$

$$K := w(X) = \sum_{x=0}^{m-1} p^x$$

Beispiel:

$$X = \{0, \dots, 6\}$$

$$Y = \{0, 2, 5\} \in S$$

$$\hookrightarrow w(Y) = p^0 + p^2 + p^5$$

$$\hookrightarrow w(Y) = 0100101_p$$

$$K = 1111111_p$$

Beispiel:

$$S = \{Y_1 = \{0, 2, 5\}, Y_2 = \{0, 1, 2\}, Y_3 = \{2, 4, 5\}, Y_4 = \{3, 4, 6\}\}$$

$$\#0 = 2, \#1 = 1, \#2 = 3, \#3 = 1, \#4 = 2, \#5 = 1, \#6 = 1, p = 4$$

$$w(Y_1) = 0100101_4, w(Y_2) = 0000111_4$$

$$w(Y_3) = 0110100_4, w(Y_4) = 1011000_4.$$

## Beweis: $\mathcal{NP}$ -Vollständigkeit von SUBSET SUM

$$M := S$$

$$\#x := \#\{Y \in S \mid x \in Y\}$$

$$p := \max\{\#x + 1 \mid x \in X\}$$

$$w(x) := p^x$$

$$K := w(X) = \sum_{x=0}^{m-1} p^x$$

Beispiel:

$$X = \{0, \dots, 6\}$$

$$Y = \{0, 2, 5\} \in S$$

$$\hookrightarrow w(Y) = p^0 + p^2 + p^5$$

$$\hookrightarrow w(Y) = 0100101_p$$

$$K = 1111111_p$$

- $(X, S)$  lösbar  $\Rightarrow (M, w, K)$  lösbar.

Sei  $S' \subseteq S$  exakte Überdeckung von  $(X, S)$ . Dann gilt

$$\sum_{Y \in S'} w(Y) = \sum_{Y \in S'} \sum_{x \in Y} p^x = \sum_{x=0}^{m-1} p^x = K,$$

da jedes  $x \in X$  genau einmal überdeckt wird.

$\rightsquigarrow S'$  erfüllt die Bedingung für SUBSET SUM.

## Beweis: $\mathcal{NP}$ -Vollständigkeit von SUBSET SUM

$$M := S$$

$$\#x := \#\{Y \in S \mid x \in Y\}$$

$$p := \max\{\#x + 1 \mid x \in X\}$$

$$w(x) := p^x$$

$$K := w(X) = \sum_{x=0}^{m-1} p^x$$

Beispiel:

$$X = \{0, \dots, 6\}$$

$$Y = \{0, 2, 5\} \in S$$

$$\hookrightarrow w(Y) = p^0 + p^2 + p^5$$

$$\hookrightarrow w(Y) = 0100101_p$$

$$K = 1111111_p$$

- $(X, S)$  lösbar  $\Leftrightarrow (M, w, K)$  lösbar.

Ist  $S' \subseteq M = S$  eine geeignete Menge für SUBSET SUM, so gilt

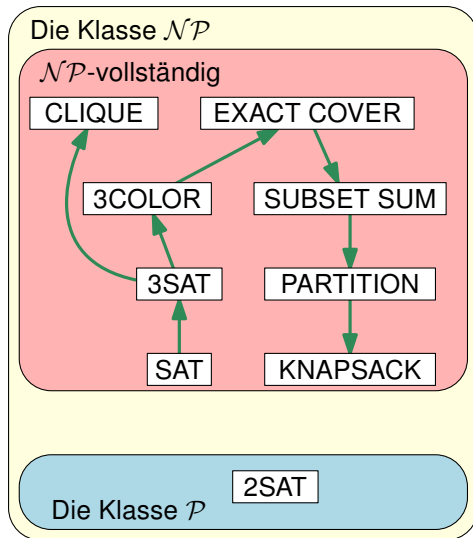
$$\sum_{Y \in S'} w(Y) = K = \sum_{x=0}^{m-1} p^x.$$

Nach Wahl von  $p$  kommt jedes  $x \in X$  in genau einem  $Y \in S'$  vor.

$\rightsquigarrow S'$  ist damit eine exakte Überdeckung von  $(X, S)$ .

## Der Plan

- EXACT COVER ist  $\mathcal{NP}$ -vollständig
  - $\rightsquigarrow$  EXACT COVER  $\in \mathcal{NP}$
  - $\rightsquigarrow$  3COLOR  $\propto$  EXACT COVER
- SUBSET SUM ist  $\mathcal{NP}$ -vollständig
  - $\rightsquigarrow$  SUBSET SUM  $\in \mathcal{NP}$
  - $\rightsquigarrow$  EXACT COVER  $\propto$  SUBSET SUM
- PARTITION ist  $\mathcal{NP}$ -vollständig
  - $\rightsquigarrow$  PARTITION  $\in \mathcal{NP}$
  - $\rightsquigarrow$  SUBSET SUM  $\propto$  PARTITION



- KNAPSACK ist  $\mathcal{NP}$ -vollständig
  - $\rightsquigarrow$  KNAPSACK  $\in \mathcal{NP}$
  - $\rightsquigarrow$  PARTITION  $\propto$  KNAPSACK

# Das Problem PARTITION

## Problem PARTITION

**Gegeben:** Eine endliche Menge  $M$ ,  
eine Gewichtsfunktion  $w: M \rightarrow \mathbb{N}_0$

**Frage:** Existiert eine Teilmenge  $M' \subseteq M$  mit  
 $\sum_{a \in M'} w(a) = \sum_{a \in M \setminus M'} w(a)$ , d.h.  $w(M') = w(M \setminus M')$ ?

## Satz.

Problem PARTITION ist  $\mathcal{NP}$ -vollständig.

## Beweis: $\mathcal{NP}$ -Vollständigkeit von PARTITION

PARTITION  $\in \mathcal{NP}$ .

- Für eine Menge  $M'$  können in Polynomialzeit die Werte  $w(M') = \sum_{a \in M'} w(a)$  und  $w(M \setminus M') = \sum_{a \in M \setminus M'} w(a)$  ausgerechnet und verglichen werden.

## Beweis: $\mathcal{NP}$ -Vollständigkeit von PARTITION

SUBSET SUM  $\propto$  PARTITION.

- Sei  $(M, w, K)$  eine beliebige SUBSET SUM-Instanz.
- Konstruiere PARTITION-Instanz  $(M^*, w^*)$

$$N := w(M) + 1$$

$$M^* := M \cup \{b, c\}$$

$$w^*(a) := w(a) \quad \text{für } a \in M$$

$$w^*(b) := N - K$$

$$w^*(c) := K + 1$$

- Die Konstruktion benötigt nur Polynomialzeit.

## Beweis: $\mathcal{NP}$ -Vollständigkeit von PARTITION

$$\begin{aligned}
 N &:= w(M) + 1 \\
 M^* &:= M \cup \{b, c\} \\
 w^*(a) &:= w(a) \quad \text{für } a \in M \\
 w^*(b) &:= N - K \\
 w^*(c) &:= K + 1
 \end{aligned}$$

Beispiel:

$$\begin{aligned}
 M &= \{i, j, k, \ell\} \\
 w(j) &= 13 \\
 &\downarrow \\
 M^* &= \{i, j, k, \ell, b, c\} \\
 w^*(j) &= 13
 \end{aligned}$$

- $(M^*, w^*)$  Ja-Instanz genau dann, wenn  $(M, w, K)$  Ja-Instanz:

$$\boxed{\exists M' \subseteq M^* : w^*(M') = w^*(M^* \setminus M')} \iff \boxed{\exists M'' \subseteq M : w(M'') = K}$$

$$w^*(b) + w^*(c) = (N - K) + (K + 1) = N + 1$$

$$w^*(M^*) = (N - 1) + (N - K) + (K + 1) = 2N$$

- Also:  $b, c$  nicht beide in  $M'$  bzw.  $M^* \setminus M'$  enthalten
- o.B.d.A. sei  $b \in M'$  und  $c \in M^* \setminus M'$



## Beweis: $\mathcal{NP}$ -Vollständigkeit von PARTITION

$$\begin{aligned}
 N &:= w(M) + 1 \\
 M^* &:= M \cup \{b, c\} \\
 w^*(a) &:= w(a) \quad \text{für } a \in M \\
 w^*(b) &:= N - K \\
 w^*(c) &:= K + 1
 \end{aligned}$$

Beispiel:

$$\begin{aligned}
 M &= \{i, j, k, \ell\} \\
 w(j) &= 13 \\
 &\downarrow \\
 M^* &= \{i, j, k, \ell, b, c\} \\
 w^*(j) &= 13
 \end{aligned}$$

- $(M^*, w^*)$  Ja-Instanz genau dann, wenn  $(M, w, K)$  Ja-Instanz:

$$\boxed{\exists M' \subseteq M^* : w^*(M') = w^*(M^* \setminus M')} \iff \boxed{\exists M'' \subseteq M : w(M'') = K}$$

- “ $\implies$ ”
- Sei  $M' \subseteq M^*$ , so dass  $w^*(M') = w^*(M^* \setminus M')$  mit  $b \in M'$ .
  - Dann gilt  $w^*(M') = N$ , da  $w^*(M^*) = 2N$ .
  - Damit erfüllt  $M'' := M' \setminus \{b\}$  die Bedingung für SUBSET SUM, denn  $w(M'') = w^*(M') - w^*(b) = N - (N - K) = K$ .

## Beweis: $\mathcal{NP}$ -Vollständigkeit von PARTITION

$$\begin{aligned}
 N &:= w(M) + 1 \\
 M^* &:= M \cup \{b, c\} \\
 w^*(a) &:= w(a) \quad \text{für } a \in M \\
 w^*(b) &:= N - K \\
 w^*(c) &:= K + 1
 \end{aligned}$$

Beispiel:

$$\begin{aligned}
 M &= \{i, j, k, \ell\} \\
 w(j) &= 13 \\
 &\downarrow \\
 M^* &= \{i, j, k, \ell, b, c\} \\
 w^*(j) &= 13
 \end{aligned}$$

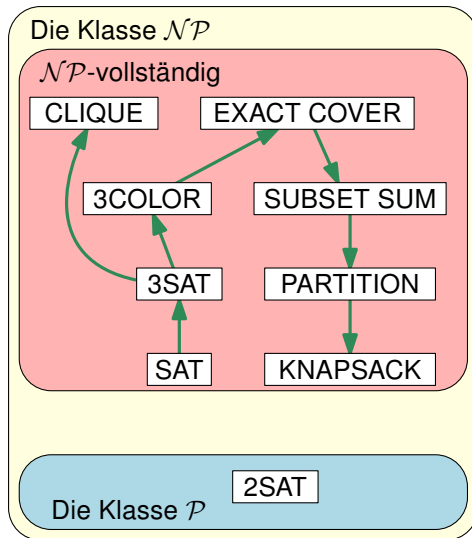
- $(M^*, w^*)$  Ja-Instanz genau dann, wenn  $(M, w, K)$  Ja-Instanz:

$$\boxed{\exists M' \subseteq M^* : w^*(M') = w^*(M^* \setminus M')} \iff \boxed{\exists M'' \subseteq M : w(M'') = K}$$

- “ $\Leftarrow$ ”
- Sei  $M''$ , so dass  $w(M'') = K$ .
  - Dann erfüllt  $M' := M'' \cup \{b\}$  die Bedingung für PARTITION, denn  $w^*(M') = w(M') + w^*(b) = K + (N - K) = N$ .

## Der Plan

- EXACT COVER ist  $\mathcal{NP}$ -vollständig
  - $\rightsquigarrow$  EXACT COVER  $\in \mathcal{NP}$
  - $\rightsquigarrow$  3COLOR  $\propto$  EXACT COVER
- SUBSET SUM ist  $\mathcal{NP}$ -vollständig
  - $\rightsquigarrow$  SUBSET SUM  $\in \mathcal{NP}$
  - $\rightsquigarrow$  EXACT COVER  $\propto$  SUBSET SUM
- PARTITION ist  $\mathcal{NP}$ -vollständig
  - $\rightsquigarrow$  PARTITION  $\in \mathcal{NP}$
  - $\rightsquigarrow$  SUBSET SUM  $\propto$  PARTITION



- KNAPSACK ist  $\mathcal{NP}$ -vollständig
  - $\rightsquigarrow$  KNAPSACK  $\in \mathcal{NP}$
  - $\rightsquigarrow$  PARTITION  $\propto$  KNAPSACK

# Das Problem KNAPSACK

## Problem KNAPSACK

**Gegeben:** Eine endliche Menge  $M$ ,  
eine Gewichtsfunktion  $w: M \rightarrow \mathbb{N}_0$ ,  
eine Kostenfunktion  $c: M \rightarrow \mathbb{N}_0$   
Zahlen  $W, C \in \mathbb{N}_0$

**Frage:** Existiert eine Teilmenge  $M' \subseteq M$  mit  $w(M') \leq W$  und  $c(M') \geq C$ ?

Satz.

Problem KNAPSACK ist  $\mathcal{NP}$ -vollständig.

# Beweis: $\mathcal{NP}$ -Vollständigkeit von KNAPSACK

KNAPSACK  $\in \mathcal{NP}$ .

- Für eine Menge  $M'$  kann in Polynomialzeit überprüft werden, ob
    - $w(M') = \sum_{a \in M'} w(a) \leq W$  und
    - $c(M') = \sum_{a \in M'} c(a) \geq C$
- gilt.

# Beweis: $\mathcal{NP}$ -Vollständigkeit von KNAPSACK

PARTITION  $\propto$  KNAPSACK.

- Sei  $(M, w)$  eine PARTITION-Instanz.
- Konstruiere KNAPSACK-Instanz  $(M, w', c, W, C)$

$$w' := 2w$$

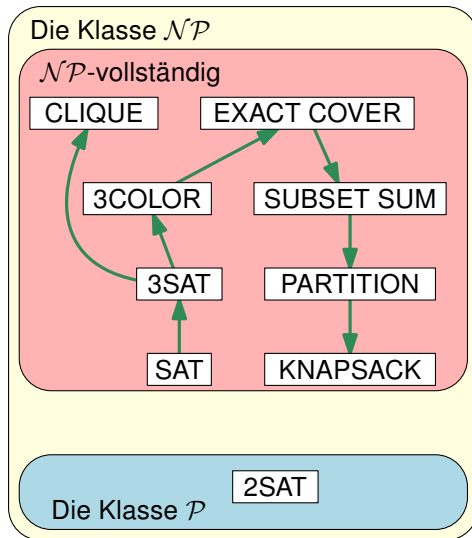
$$c := 2w$$

$$W = C := w(M) = \sum_{a \in M} w(a)$$

- Die Konstruktion benötigt nur Polynomialzeit.
- Es ist  $(M, w)$  genau dann eine Ja-Instanz, wenn  $(M, w', c, W, C)$  eine Ja-Instanz ist (ohne Beweis).

## Der Plan

- EXACT COVER ist  $\mathcal{NP}$ -vollständig
  - $\rightsquigarrow$  EXACT COVER  $\in \mathcal{NP}$
  - $\rightsquigarrow$  3COLOR  $\propto$  EXACT COVER
- SUBSET SUM ist  $\mathcal{NP}$ -vollständig
  - $\rightsquigarrow$  SUBSET SUM  $\in \mathcal{NP}$
  - $\rightsquigarrow$  EXACT COVER  $\propto$  SUBSET SUM
- PARTITION ist  $\mathcal{NP}$ -vollständig
  - $\rightsquigarrow$  PARTITION  $\in \mathcal{NP}$
  - $\rightsquigarrow$  SUBSET SUM  $\propto$  PARTITION



- KNAPSACK ist  $\mathcal{NP}$ -vollständig
  - $\rightsquigarrow$  KNAPSACK  $\in \mathcal{NP}$
  - $\rightsquigarrow$  PARTITION  $\propto$  KNAPSACK

## Auswirkung auf die Frage $\mathcal{P} = \mathcal{NP}$

- Wir haben gesehen, dass es für je zwei  $\mathcal{NP}$ -vollständige Probleme eine polynomiale Transformation von einem zum anderen Problem gibt.
- Deshalb sind alle  $\mathcal{NP}$ -vollständigen Probleme im wesentlichen gleich schwer.
- Dies hat Auswirkungen auf die Frage, ob  $\mathcal{P} = \mathcal{NP}$  ist.

### Satz.

Sei  $L$  eine  $\mathcal{NP}$ -vollständige Sprache. Dann gilt:

- $L \in \mathcal{P} \implies \mathcal{P} = \mathcal{NP}$
- $L \notin \mathcal{P} \implies$  für jede  $\mathcal{NP}$ -vollständige Sprache  $L'$  gilt  $L' \notin \mathcal{P}$



## Auswirkung auf die Frage $\mathcal{P} = \mathcal{NP}$

### Satz.

Sei  $L$  eine  $\mathcal{NP}$ -vollständige Sprache. Dann gilt:

- $L \in \mathcal{P} \implies \mathcal{P} = \mathcal{NP}$
- $L \notin \mathcal{P} \implies$  für jede  $\mathcal{NP}$ -vollständige Sprache  $L'$  gilt  $L' \notin \mathcal{P}$

### Beweis Teil 1:

- Da  $L \in \mathcal{P}$ , existiert eine polynomiale DTM  $\mathcal{M}$  für  $L$ .
- Da  $L$   $\mathcal{NP}$ -vollständig, gibt es für jede Sprache  $L' \in \mathcal{NP}$  eine polynomiale Transformation  $L' \propto L$ .
- Hintereinanderausführung von  $L' \propto L$  und  $\mathcal{M}$  liefert eine polynomiale DTM für  $L'$ .
- Damit ist jede Sprache  $L' \in \mathcal{NP}$  auch in  $\mathcal{P}$ .

## Auswirkung auf die Frage $\mathcal{P} = \mathcal{NP}$

### Satz.

Sei  $L$  eine  $\mathcal{NP}$ -vollständige Sprache. Dann gilt:

- $L \in \mathcal{P} \implies \mathcal{P} = \mathcal{NP}$
- $L \notin \mathcal{P} \implies$  für jede  $\mathcal{NP}$ -vollständige Sprache  $L'$  gilt  $L' \notin \mathcal{P}$

### Beweis Teil 2:

- Sei  $L \notin \mathcal{P}$  und  $L$   $\mathcal{NP}$ -vollständig.
- Angenommen für eine  $\mathcal{NP}$ -vollständige Sprache  $L'$  gilt:  $L' \in \mathcal{P}$
- Dann folgt aus Teil 1 des Satzes  $\mathcal{P} = \mathcal{NP}$ .
- Dies ist aber ein Widerspruch zur Voraussetzung  $L \notin \mathcal{P}$ .

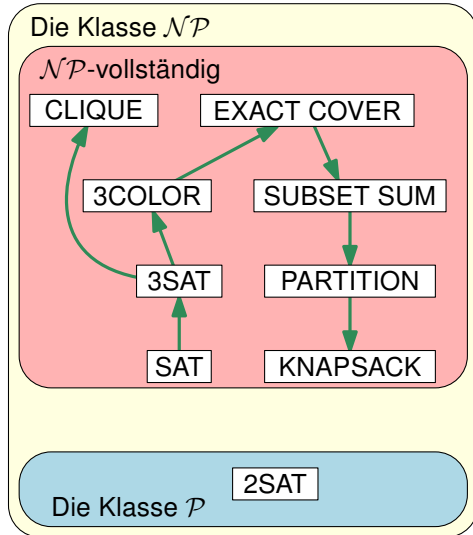
# Zusammenfassung

- Die Klasse  $\mathcal{P}$  ist die Klasse aller Entscheidungsprobleme/Sprachen die mit einer **deterministischen Turingmaschine** in **polynomieller Zeit** gelöst werden können.
- Die Klasse  $\mathcal{NP}$  ist die Klasse aller Entscheidungsprobleme/Sprachen die mit einer **nicht-deterministischen Turingmaschine** in **polynomieller Zeit** gelöst werden können.
- Informell ausgedrückt gehört  $\Pi$  zu  $\mathcal{NP}$ , falls  $\Pi$  folgende Eigenschaft hat:  
Ist die Antwort bei Eingabe eines Instanz  $I$  von  $\Pi$  Ja, dann kann die Korrektheit eines Beweises (Zeugen) dafür in polynomialer Zeit überprüft werden.

# Zusammenfassung

- Eine **polynomiale Transformation** einer Sprache  $L_1 \subseteq \Sigma_1^*$  in eine Sprache  $L_2 \subseteq \Sigma_2^*$  ist eine Funktion  $f: \Sigma_1^* \rightarrow \Sigma_2^*$  mit den Eigenschaften:
  - es existiert eine polynomiale deterministische Turing-Maschine, die  $f$  berechnet;
  - für alle  $x \in \Sigma_1^*$  gilt:  $x \in L_1 \Leftrightarrow f(x) \in L_2$ .
  
- Eine Sprache  $L$  heißt **NP-vollständig**, falls gilt:
  - $L \in \mathcal{NP}$  und
  - für alle  $L' \in \mathcal{NP}$  gilt  $L' \leq L$  (NP-Schwere).
  
- **Bedeutung:**  
Unter der Annahme  $\mathcal{P} \neq \mathcal{NP}$  gibt es kein polynomielles Lösungsverfahren für ein NP-vollständiges Problem.

# Zusammenfassung



- Mit dem Satz von Cook haben wir direkt gezeigt, dass das Problem SAT  $\mathcal{NP}$ -schwer ist.
- Bei allen anderen Problemen haben wir polynomielle Transformationen (Reduktionen) benutzt um die  $\mathcal{NP}$ -Schwere nachzuweisen:  

$$\text{SAT} \propto \text{3SAT} \propto \text{3COLOR} \propto \text{EXACT COVER}$$

$$\propto \text{SUBSET SUM} \propto \text{PARTITION} \propto \text{KNAPSACK}$$

# Ein Blick über den Tellerrand

- Entscheidungsprobleme außerhalb von  $\mathcal{P}$  und  $\mathcal{NP}$
- Probleme die nicht Entscheidungsprobleme sind

## Die Klassen $NP_C$ und $NP_I$

- Die Klasse  $NP_C$  ( $NP$ -complete) sei die Klasse der  $NP$ -vollständigen Sprachen/Probleme.
- Die Klasse  $NP_I$  ( $NP$ -intermediate) ist definiert durch  $NP_I := NP \setminus (P \cup NP_C)$ .

## Die Klassen $\mathcal{NPC}$ und $\mathcal{NPI}$

- Die Klasse  $\mathcal{NPC}$  ( $\mathcal{NP}$ -complete) sei die Klasse der  $\mathcal{NP}$ -vollständigen Sprachen/Probleme.
- Die Klasse  $\mathcal{NPI}$  ( $\mathcal{NP}$ -intermediate) ist definiert durch  $\mathcal{NPI} := \mathcal{NP} \setminus (\mathcal{P} \cup \mathcal{NPC})$ .

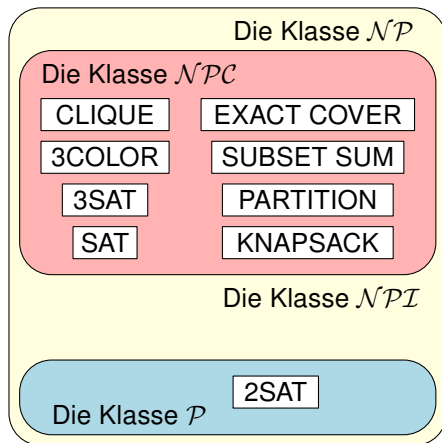
### Satz von Ladner (1975).

Falls  $\mathcal{P} \neq \mathcal{NP}$ , so folgt  $\mathcal{NPI} \neq \emptyset$ .

- **Entweder:**  $\mathcal{P} = \mathcal{NP} = \mathcal{NPC}$  (wovon wir nicht ausgehen)
- **Oder:**  $\mathcal{P} \neq \mathcal{NP}$  und es gibt Entscheidungsprobleme  $\Pi$  in  $\mathcal{NP}$ , die weder in  $\mathcal{P}$  sind, noch  $\mathcal{NP}$ -vollständig sind.



# Vermutete Situation



# Die Klassen $\text{co-}\mathcal{P}$ und $\text{co-NP}$

## Klasse der Komplementsprachen

- Die Klasse  $\text{co-}\mathcal{P}$  ist die Klasse aller Sprachen

$$L' = \Sigma^* \setminus L \text{ mit } L \subseteq \Sigma^* \text{ und } L \in \mathcal{P}.$$

- Die Klasse  $\text{co-NP}$  ist die Klasse aller Sprachen

$$L' = \Sigma^* \setminus L \text{ mit } L \subseteq \Sigma^* \text{ und } L \in \text{NP}.$$

# Die Klassen $\text{co-}\mathcal{P}$ und $\text{co-NP}$

## Klasse der Komplementsprachen

- Die Klasse  $\text{co-}\mathcal{P}$  ist die Klasse aller Sprachen

$$L' = \Sigma^* \setminus L \text{ mit } L \subseteq \Sigma^* \text{ und } L \in \mathcal{P}.$$

- Die Klasse  $\text{co-NP}$  ist die Klasse aller Sprachen

$$L' = \Sigma^* \setminus L \text{ mit } L \subseteq \Sigma^* \text{ und } L \in \text{NP}.$$

**Wir wissen:**  $\mathcal{P} = \text{co-}\mathcal{P} \subseteq \text{co-NP}$ .

# Die Klassen $\text{co-}\mathcal{P}$ und $\text{co-NP}$

## Klasse der Komplementsprachen

- Die Klasse  $\text{co-}\mathcal{P}$  ist die Klasse aller Sprachen

$$L' = \Sigma^* \setminus L \text{ mit } L \subseteq \Sigma^* \text{ und } L \in \mathcal{P}.$$

- Die Klasse  $\text{co-NP}$  ist die Klasse aller Sprachen

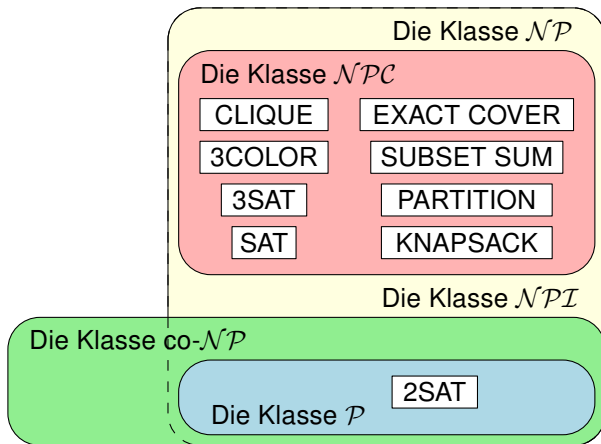
$$L' = \Sigma^* \setminus L \text{ mit } L \subseteq \Sigma^* \text{ und } L \in \text{NP}.$$

**Wir wissen:**  $\mathcal{P} = \text{co-}\mathcal{P} \subseteq \text{co-NP}$ .

**Frage:** Gilt auch  $\text{NP} = \text{co-NP}$ ?

- Sollte  $\text{NP} \neq \text{co-NP}$  sein, dann würde auch  $\mathcal{P} \neq \text{NP}$  gelten.
- Vermutlich ist  $\text{NP} \neq \text{co-NP}$   
 $\rightsquigarrow$  Verschärfung der  $\mathcal{P} \neq \text{NP}$ -Vermutung.
- Aber auch  $\text{NP} = \text{co-NP}$  und  $\mathcal{P} \neq \text{NP}$  ist theoretisch möglich.

# Vermutete Situation



# Das TSP-Komplement-Problem

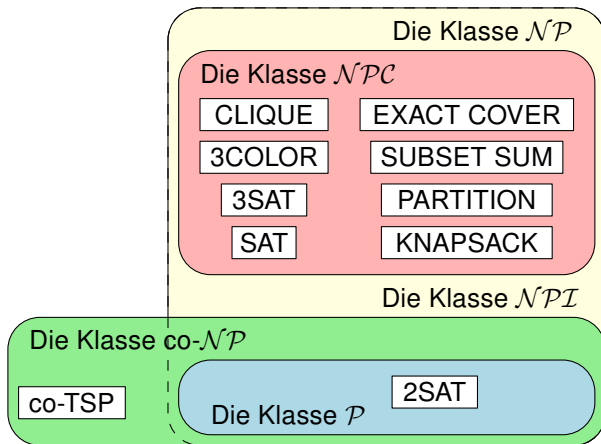
## Problem co-TSP

**Gegeben:** Graph  $G = (V, E)$ ,  
Kantengewichtung  $c: E \rightarrow \mathbb{Z}^+$ ,  
Parameter  $K \in \mathbb{Z}$ .

**Frage:** Gibt es **keine** Tour der Länge  $\leq K$ ?

- co-TSP ist in  $\text{co-NP}$  (denn TSP ist in  $\text{NP}$ )  
 $\rightsquigarrow$  Eine **Nein**-Instanz kann durch einen geeigneten Zeugen einfach verifiziert werden.
- Frage: Ist co-TSP in  $\text{NP}$ ?  
 $\rightsquigarrow$  Kann eine **Ja**-Instanz durch einen geeigneten Zeugen einfach verifiziert werden?  
Vermutung: Nein.

# Vermutete Situation



## $\mathcal{NP}$ -vollständig vs. $\text{co-}\mathcal{NP}$

Lemma.

Falls  $L$   $\mathcal{NP}$ -vollständig ist und  $L \in \text{co-}\mathcal{NP}$ , so ist  $\mathcal{NP} = \text{co-}\mathcal{NP}$ .



## $\mathcal{NP}$ -vollständig vs. $\text{co-}\mathcal{NP}$

### Lemma.

Falls  $L$   $\mathcal{NP}$ -vollständig ist und  $L \in \text{co-}\mathcal{NP}$ , so ist  $\mathcal{NP} = \text{co-}\mathcal{NP}$ .

### Beweis:

- Da  $L \in \text{co-}\mathcal{NP}$ , existiert eine polynomiale NTM  $\mathcal{M}$  für  $L^c$ .
- Da  $L$   $\mathcal{NP}$ -vollständig ist, existiert für alle  $L' \in \mathcal{NP}$  eine deterministische polynomiale Transformation  $f_{L'} : L' \rightarrow L$ .
- Eine leichte Anpassung liefert eine deterministische polynomiale Transformation  $f_{L'^c} : L'^c \rightarrow L^c$ .
- Hintereinanderausführung von  $f_{L'^c}$  und  $\mathcal{M}$  ergibt eine polynomiale NTM für  $L'^c$ .
- Also  $L' \in \text{co-}\mathcal{NP}$ .

## $NP$ -vollständig vs. $co-NP$

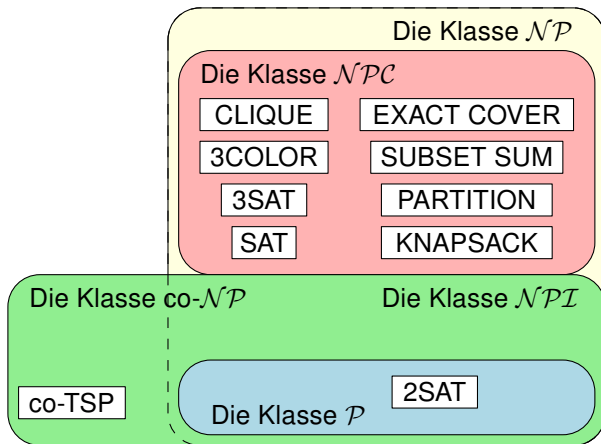
### Lemma.

Falls  $L$   $NP$ -vollständig ist und  $L \in co-NP$ , so ist  $NP = co-NP$ .

### Bemerkung:

- Mit der Vermutung  $NP \neq co-NP$  folgt also  $NP \cap co-NP = \emptyset$ .
- Unter dieser Annahme:  
Wenn ein Problem in  $NP$  und  $co-NP$  ist, aber nicht in  $P$ , so ist es in  $NP \setminus P$ .

# Vermutete Situation



# Das Problem PRIMTEILER

## Problem PRIMTEILER

**Gegeben:** Zahl  $N \in \mathbb{N}$ , Zahl  $K \in \mathbb{N}$

**Frage:** Hat  $N$  einen Primteiler  $P \leq K$ ?

- Ein Primteiler ist eine Primzahl  $P$  mit  $N/P \in \mathbb{N}$ .

# Das Problem PRIMTEILER

## Problem PRIMTEILER

**Gegeben:** Zahl  $N \in \mathbb{N}$ , Zahl  $K \in \mathbb{N}$

**Frage:** Hat  $N$  einen Primteiler  $P \leq K$ ?

- Ein Primteiler ist eine Primzahl  $P$  mit  $N/P \in \mathbb{N}$ .

## Satz.

PRIMTEILER ist in  $\mathcal{NP}$  und in  $\text{co-}\mathcal{NP}$ .

## Beweis:

- Für eine gegebene Zahl  $P$  kann in polynomialer Zeit getestet werden, ob  $P$  Primzahl,  $N/P \in \mathbb{N}$  und  $P \leq K$ .
- Also ist PRIMTEILER  $\in \mathcal{NP}$ .

# Das Problem PRIMTEILER

## Problem PRIMTEILER

**Gegeben:** Zahl  $N \in \mathbb{N}$ , Zahl  $K \in \mathbb{N}$

**Frage:** Hat  $N$  einen Primteiler  $P \leq K$ ?

- Ein Primteiler ist eine Primzahl  $P$  mit  $N/P \in \mathbb{N}$ .

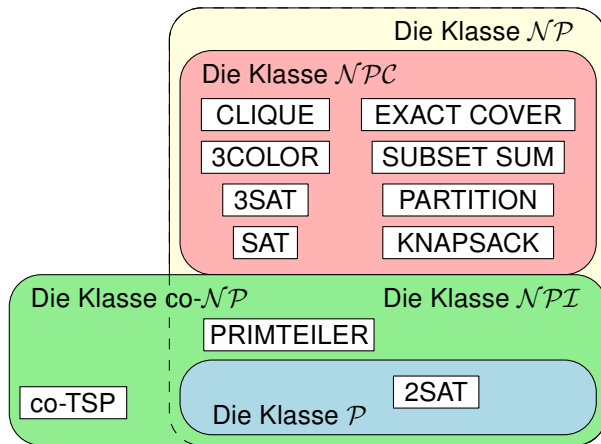
## Satz.

PRIMTEILER ist in  $\mathcal{NP}$  und in  $\text{co-}\mathcal{NP}$ .

## Beweis:

- Für gegebene Zahlen  $P_1, \dots, P_m$  (mit Vielfachheit) kann in polynomialer Zeit getestet werden, ob  $P_1, \dots, P_m$  Primzahlen,  $N = P_1 \times \dots \times P_m$  und  $K < P_1, \dots, P_m$ .
- Also ist  $\text{PRIMTEILER} \in \text{co-}\mathcal{NP}$ .

# Vermutete Situation



# Die Klasse PSPACE

## Die Klasse PSPACE

**PSPACE** ist die Klasse aller Sprachen / Entscheidungsprobleme die von einer **deterministischen** Turingmaschine mit **polynomiell**em Platz entschieden werden können.



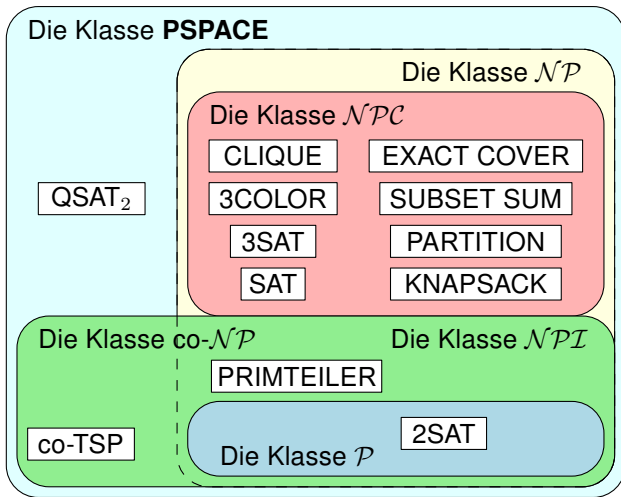
# Die Klasse PSPACE

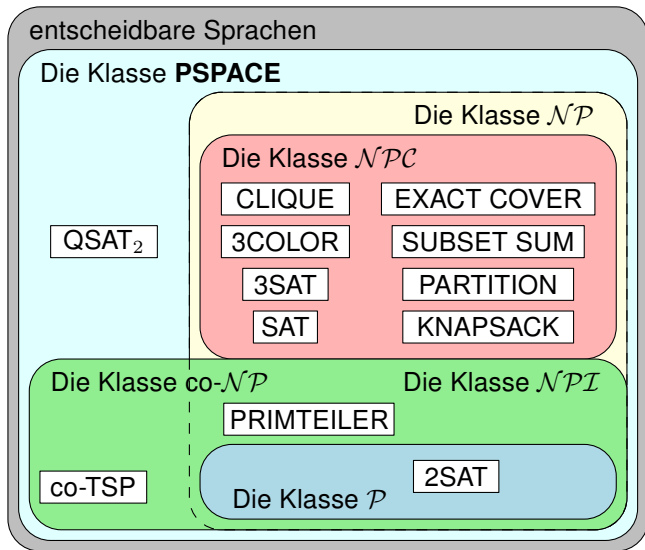
## Die Klasse PSPACE

**PSPACE** ist die Klasse aller Sprachen / Entscheidungsprobleme die von einer **deterministischen** Turingmaschine mit **polynomiell**em Platz entschieden werden können.

- Betrachte SAT Instanz  $(U, C)$  und eine Aufteilung der Variablen in disjunkte Mengen  $U_1$  und  $U_2$ .
- Die Instanz heiÙe *lösbar* wenn
  - es eine **Wahrheitsbelegung  $t_1$  von  $U_1$  gibt**,
  - so dass **für alle Wahrheitsbelegungen  $t_2$  von  $U_2$**
  - alle Klauseln in  $C$  erfüllt sind.
- $\rightsquigarrow$  bezeichnet als  $QSAT_2$  oder  $\exists VSAT$

$QSAT_2$  liegt in **PSPACE** da eine DTM alle Wahrheitsbelegungen  $t_1$  und für jede solche, alle Wahrheitsbelegungen  $t_2$  überprüfen kann.





formale Sprachen

semi-entscheidbare Sprachen

entscheidbare Sprachen

Die Klasse **PSPACE**

Die Klasse  $\mathcal{NP}$

Die Klasse  $\mathcal{NPC}$

CLIQUE

EXACT COVER

QSAT<sub>2</sub>

3COLOR

SUBSET SUM

3SAT

PARTITION

SAT

KNAPSACK

Die Klasse  $\text{co-}\mathcal{NP}$

Die Klasse  $\mathcal{NPI}$

PRIMTEILER

co-TSP

Die Klasse  $\mathcal{P}$

2SAT

formale Sprachen

semi-entscheidbare Sprachen

entscheidbare Sprachen

Die Klasse **PSPACE**

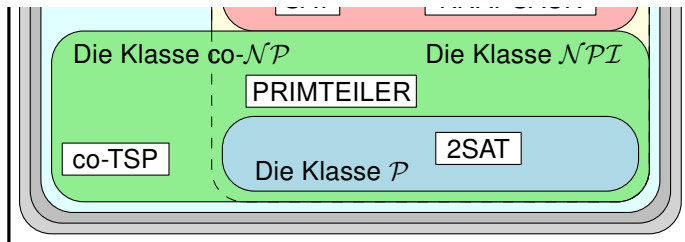
**Testen Sie sich:**

Lesen Sie den Artikel

**A Short Guide to Hard Problems**

auf Quanta Magazine

<https://www.quantamagazine.org/a-short-guide-to-hard-problems-20180716/>



# Ein offenes Problem

Spiel **Leben und Tod**:

- 2 Spieler haben je einen Stapel von Karten.
- Beide spielen ihre oberste Karte.
- Der Spieler mit der höheren Karte bekommt beide Karten unter seinen / ihrer Stapel.
- Spieler verliert wenn er / sie keine Karten mehr hat.

Alice

4

2

Bob

1

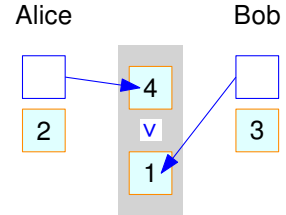
3



# Ein offenes Problem

## Spiel **Leben und Tod**:

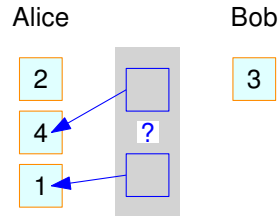
- 2 Spieler haben je einen Stapel von Karten.
- Beide spielen ihre oberste Karte.
- Der Spieler mit der höheren Karte bekommt beide Karten unter seinen / ihrer Stapel.
- Spieler verliert wenn er / sie keine Karten mehr hat.



# Ein offenes Problem

## Spiel **Leben und Tod**:

- 2 Spieler haben je einen Stapel von Karten.
- Beide spielen ihre oberste Karte.
- Der Spieler mit der höheren Karte bekommt beide Karten unter seinen / ihrer Stapel.
- Spieler verliert wenn er / sie keine Karten mehr hat.

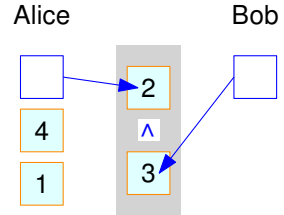




# Ein offenes Problem

## Spiel **Leben und Tod**:

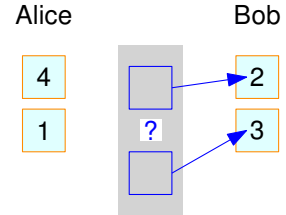
- 2 Spieler haben je einen Stapel von Karten.
- Beide spielen ihre oberste Karte.
- Der Spieler mit der höheren Karte bekommt beide Karten unter seinen / ihrer Stapel.
- Spieler verliert wenn er / sie keine Karten mehr hat.



# Ein offenes Problem

## Spiel **Leben und Tod**:

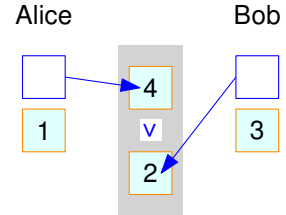
- 2 Spieler haben je einen Stapel von Karten.
- Beide spielen ihre oberste Karte.
- Der Spieler mit der höheren Karte bekommt beide Karten unter seinen / ihrer Stapel.
- Spieler verliert wenn er / sie keine Karten mehr hat.



# Ein offenes Problem

## Spiel **Leben und Tod**:

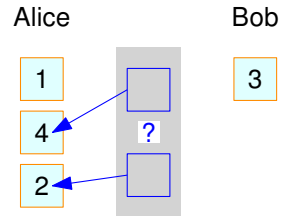
- 2 Spieler haben je einen Stapel von Karten.
- Beide spielen ihre oberste Karte.
- Der Spieler mit der höheren Karte bekommt beide Karten unter seinen / ihrer Stapel.
- Spieler verliert wenn er / sie keine Karten mehr hat.



# Ein offenes Problem

## Spiel **Leben und Tod**:

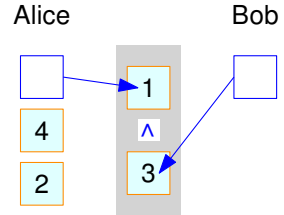
- 2 Spieler haben je einen Stapel von Karten.
- Beide spielen ihre oberste Karte.
- Der Spieler mit der höheren Karte bekommt beide Karten unter seinen / ihrer Stapel.
- Spieler verliert wenn er / sie keine Karten mehr hat.



# Ein offenes Problem

## Spiel **Leben und Tod**:

- 2 Spieler haben je einen Stapel von Karten.
- Beide spielen ihre oberste Karte.
- Der Spieler mit der höheren Karte bekommt beide Karten unter seinen / ihrer Stapel.
- Spieler verliert wenn er / sie keine Karten mehr hat.



# Ein offenes Problem

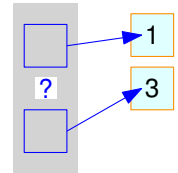
Spiel **Leben und Tod**:

- 2 Spieler haben je einen Stapel von Karten.
- Beide spielen ihre oberste Karte.
- Der Spieler mit der höheren Karte bekommt beide Karten unter seinen / ihrer Stapel.
- Spieler verliert wenn er / sie keine Karten mehr hat.

Alice



Bob



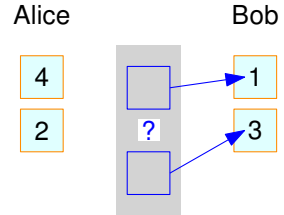
# Ein offenes Problem

## Spiel **Leben und Tod**:

- 2 Spieler haben je einen Stapel von Karten.
- Beide spielen ihre oberste Karte.
- Der Spieler mit der höheren Karte bekommt beide Karten unter seinen / ihrer Stapel.
- Spieler verliert wenn er / sie keine Karten mehr hat.

**Entweder:** Spieler mit höchster Karte gewinnt

**Oder:** Das Spiel geht unendlich lange



# Ein offenes Problem

## Spiel **Leben und Tod**:

- 2 Spieler haben je einen Stapel von Karten.
- Beide spielen ihre oberste Karte.
- Der Spieler mit der höheren Karte bekommt beide Karten unter seinen / ihrer Stapel.
- Spieler verliert wenn er / sie keine Karten mehr hat.

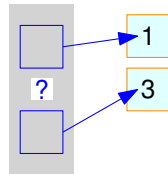
**Entweder:** Spieler mit höchster Karte gewinnt

**Oder:** Das Spiel geht unendlich lange

Alice



Bob



## Offenes Problem

Kann man in Polynomialzeit entscheiden, ob das Spiel unendlich geht?

Wir wissen bisher nur, dass es in **PSPACE** ist.