

13a

**NP-Vollständigkeit von SET-COVER, SUBSET-SUM,
KNAPSACK, PARTITION und BIN-PACKING**

PD Dr. Jan Johannsen

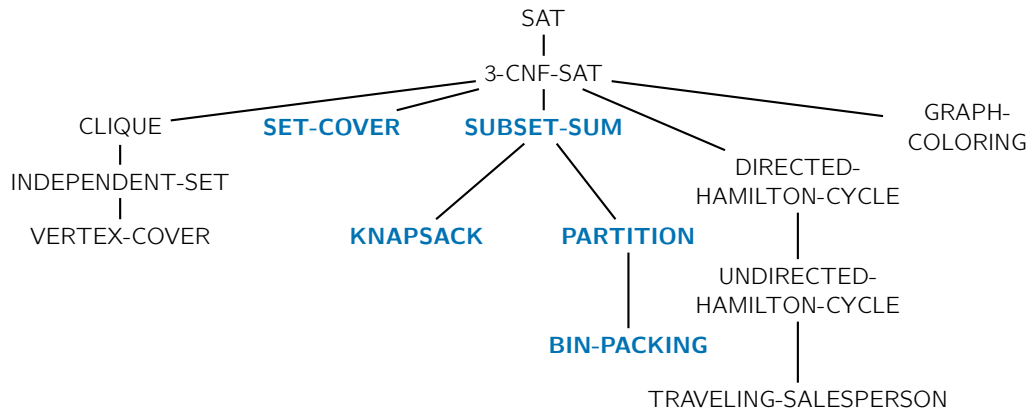
Institut für Informatik

Stand: 14. Juli 2026

Basierend auf Folien von Prof. Dr. David Sabel und Prof. Dr. Jasmin Blanchette



Überblick über NP-Vollständigkeitsbeweise



Das SET-COVER-Problem

Definition

Das **SET-COVER-Problem** lässt sich wie folgt formulieren.

gegeben: Mengen $T_1, \dots, T_k$ mit $T_1, \dots, T_k \subseteq M$,
wobei M eine endliche Grundmenge ist, und
eine Zahl $n \leq k$

gefragt: Gibt es eine Auswahl von n Mengen $T_{i_1}, \dots, T_{i_n}$ ($i_j \in \{1, \dots, k\}$) mit
 $T_{i_1} \cup \dots \cup T_{i_n} = M$?

Beispiel:

Gegeben seien $T_1 = \{1, 2, 3, 5\}$, $T_2 = \{1, 2\}$, $T_3 = \{3, 4\}$, $T_4 = \{3\}$ mit
 $M = \{1, 2, 3, 4, 5\}$ und $n = 2$.

Lösung: T_1, T_3 , da $T_1 \cup T_3 = M$.

NP-Vollständigkeit von SET-COVER

Satz

SET-COVER ist NP-vollständig.

Beweis

1. SET-COVER $\in$ NP:

Ein Zeuge für $(T_1, \dots, T_k, n) \in \text{SET-COVER}$ ist eine Auswahl $T_{i_1}, \dots, T_{i_n}$ von n Mengen.

Die Prüfrelation $R_{\text{SET-COVER}}$ überprüft, ob $T_{i_1} \cup \dots \cup T_{i_n} = M$ gilt.

Dies ist deterministisch in Polynomialzeit überprüfbar.

NP-Vollständigkeit von SET-COVER

Beweis (Fortsetzung)

2. SET-COVER ist NP-schwer, erster Versuch:

Wir zeigen $3\text{-CNF-SAT} \leq_p \text{SET-COVER}$.

Sei $F = K_1 \wedge \dots \wedge K_m$ eine 3-CNF.

Seien $x_1, \dots, x_n$ die in F vorkommenden aussagenlogischen Variablen.

Setze $M := \{1, \dots, m\}$.

Für $i \in \{1, \dots, n\}$ sei

$$P_i := \{j \mid \text{Literal } x_i \text{ kommt in Klausel } K_j \text{ vor}\}$$

$$N_i := \{j \mid \text{Literal } \neg x_i \text{ kommt in Klausel } K_j \text{ vor}\}$$

Das Mengensystem sei $P_1, \dots, P_n, N_1, \dots, N_n \subseteq M$.

Gesucht wird eine Vereinigung von n Mengen.

Grundgedanke: Wenn P_i bzw. N_i gewählt wird, dann werden die Klauseln, die das Literal enthalten, erfüllt. Das Ziel ist, alle m Klauseln zu erfüllen.

Beispiel für den ersten Versuch

3-CNF:

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

SET-COVER-Instanz dazu:

$$P_1 = \{j \mid x_1 \text{ kommt in Klausel } K_j \text{ vor}\} = \{1, 2\}$$

$$N_1 = \{j \mid \neg x_1 \text{ kommt in Klausel } K_j \text{ vor}\} = \{3\}$$

$$P_2 = \{j \mid x_2 \text{ kommt in Klausel } K_j \text{ vor}\} = \{1\}$$

$$N_2 = \{j \mid \neg x_2 \text{ kommt in Klausel } K_j \text{ vor}\} = \{2, 3\}$$

$$P_3 = \{j \mid x_3 \text{ kommt in Klausel } K_j \text{ vor}\} = \{1, 3\}$$

$$N_3 = \{j \mid \neg x_3 \text{ kommt in Klausel } K_j \text{ vor}\} = \emptyset$$

Gesucht wird eine Vereinigung von $n = 3$ Mengen U_1, U_2, U_3 mit $U_1 \cup U_2 \cup U_3 = \{1, 2, 3\}$.

Lösung: z.B. P_1, N_2, N_3 .

Belegung dazu: $I(x_1) = 1, I(x_2) = 0, I(x_3) = 0$.

Weiteres Beispiel für den ersten Versuch

3-CNF:

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2)$$

SET-COVER-Instanz dazu:

$$P_1 = \{j \mid x_1 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+1\} = \{1, 3\}$$

$$N_1 = \{j \mid \neg x_1 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+1\} = \{2, 4\}$$

$$P_2 = \{j \mid x_2 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+2\} = \{1, 4\}$$

$$N_2 = \{j \mid \neg x_2 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+2\} = \{2, 3\}$$

Gesucht wird eine Vereinigung von $n = 2$ Mengen U_1, U_2 mit $U_1 \cup U_2 = \{1, 2, 3, 4\}$.

Lösung: z.B. P_1, N_1 .

Aber **diese Lösung** entspricht **keiner gültigen Belegung**.

NP-Vollständigkeit von SET-COVER

Beweis (Fortsetzung)

2. SET-COVER ist NP-schwer, zweiter Versuch:

Wir zeigen $3\text{-CNF-SAT} \leq_p \text{SET-COVER}$.

Sei $F = K_1 \wedge \dots \wedge K_m$ eine 3-CNF.

Seien $x_1, \dots, x_n$ die in F vorkommenden aussagenlogischen Variablen.

Setze $M := \{1, \dots, m + n\}$.

Für $i \in \{1, \dots, n\}$ sei

$$P_i := \{j \mid \text{Literal } x_i \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m + i\}$$

$$N_i := \{j \mid \text{Literal } \neg x_i \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m + i\}$$

Das Mengensystem sei $P_1, \dots, P_n, N_1, \dots, N_n \subseteq M$.

Gesucht wird eine Vereinigung von n Mengen.

Grundgedanke: Für jedes i muss mindestens (und daher höchstens) eins von P_i, N_i gewählt werden.

Beispiel für den zweiten Versuch

3-CNF:

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

SET-COVER-Instanz dazu:

$$P_1 = \{j \mid x_1 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+1\} = \{1, 2, 4\}$$

$$N_1 = \{j \mid \neg x_1 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+1\} = \{3, 4\}$$

$$P_2 = \{j \mid x_2 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+2\} = \{1, 5\}$$

$$N_2 = \{j \mid \neg x_2 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+2\} = \{2, 3, 5\}$$

$$P_3 = \{j \mid x_3 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+3\} = \{1, 3, 6\}$$

$$N_3 = \{j \mid \neg x_3 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+3\} = \{6\}$$

Gesucht wird eine Vereinigung von $n = 3$ Mengen U_1, U_2, U_3 mit $U_1 \cup U_2 \cup U_3 = \{1, 2, 3, 4, 5, 6\}$.

Lösung: z.B. P_1, N_2, N_3 .

Belegung dazu: $I(x_1) = 1, I(x_2) = 0, I(x_3) = 0$.

Weiteres Beispiel für den zweiten Versuch

3-CNF:

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2)$$

SET-COVER-Instanz dazu:

$$P_1 = \{j \mid x_1 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+1\} = \{1, 3, 5\}$$

$$N_1 = \{j \mid \neg x_1 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+1\} = \{2, 4, 5\}$$

$$P_2 = \{j \mid x_2 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+2\} = \{1, 4, 6\}$$

$$N_2 = \{j \mid \neg x_2 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+2\} = \{2, 3, 6\}$$

Gesucht wird eine Vereinigung von $n = 2$ Mengen U_1, U_2 mit $U_1 \cup U_2 = \{1, 2, 3, 4, 5, 6\}$.

Es gibt **keine Lösung** aber auch **keine Belegung** für die 3-CNF.

NP-Vollständigkeit von SET-COVER

Beweis (Fortsetzung)

Zur Erinnerung:

$$P_i := \{j \mid \text{Literal } x_i \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m + i\}$$

$$N_i := \{j \mid \text{Literal } \neg x_i \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m + i\}$$

Die Reduktionsfunktion ist total und kann in Polynomialzeit berechnet werden.

Zu zeigen: F ist erfüllbar g.d.w. es eine Vereinigung von n Mengen

$U_1, \dots, U_n \in \{P_1, N_1, \dots, P_n, N_n\}$ gibt, sodass $U_1 \cup \dots \cup U_n = M$.

⇒ Wenn F erfüllbar ist, gibt es eine Belegung I mit $I(F) = 1$.

Wenn $I(x_i) = 1$, dann wähle P_i , sonst wähle N_i . Dies ergibt n gewählte Mengen.

Jede Zahl aus M kommt vor:

- ▶ Da jede Klausel durch I erfüllt ist, kommen alle $1, \dots, m$ vor.
- ▶ Da jede Variable x_i mit 0 oder 1 belegt wird, kommen alle $m + 1, \dots, m + n$ vor.

Die Vereinigung der n Mengen ergibt daher M .

NP-Vollständigkeit von SET-COVER

Beweis (Fortsetzung)

Zur Erinnerung:

$$P_i := \{j \mid \text{Literal } x_i \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m + i\}$$

$$N_i := \{j \mid \text{Literal } \neg x_i \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m + i\}$$

Zu zeigen: F ist erfüllbar g.d.w. es eine Vereinigung von n Mengen $U_1, \dots, U_n \in \{P_1, N_1, \dots, P_n, N_n\}$ gibt, sodass $U_1 \cup \dots \cup U_n = M$.

$\Leftarrow$ Seien $U_1, \dots, U_n \in \{P_1, N_1, \dots, P_n, N_n\}$ mit $U_1 \cup \dots \cup U_n = M$.

Da $m + 1, \dots, m + n$ in der Vereinigung sind, muss für jedes $i \in \{1, \dots, n\}$ genau entweder P_i oder N_i in der Vereinigung sein. O.B.d.A. $U_i \in \{P_i, N_i\}$.

Setze $I(x_i) = 1$ wenn $U_i = P_i$ und $I(x_i) = 0$ wenn $U_i = N_i$.

Da $1, \dots, m$ in der Vereinigung sind, wird in jeder Klausel ein Literal durch I wahr gemacht. Daher ist F erfüllbar. $\square$

Das SUBSET-SUM-Problem

Definition

Das **SUBSET-SUM-Problem** lässt sich wie folgt formulieren.

gegeben: natürliche Zahlen $a_1, \dots, a_k \in \mathbb{N}$ und $s \in \mathbb{N}$

gefragt: Gibt es eine Teilmenge $I \subseteq \{1, \dots, k\}$, sodass $\sum_{i \in I} a_i = s$?

Beispiel:

Gegeben seien $a_1, \dots, a_6 = 1, 21, 5, 16, 12, 19$ und $s = 49$.

Lösung: 2, 4, 5, da $21 + 16 + 12 = 49$.

Satz

Das SUBSET-SUM-Problem ist NP-vollständig.

Beweis Siehe Skript.



Definition

Das **KNAPSACK-Problem** lässt sich wie folgt formulieren.

gegeben: k Gegenstände mit Gewichten $w_1, \dots, w_k \in \mathbb{N}$ und
Nutzenwerten $n_1, \dots, n_k \in \mathbb{N}$,
sowie zwei Schwellenwerte $s_w, s_n \in \mathbb{N}$

gefragt: Gibt es eine Teilmenge $I \subseteq \{1, \dots, k\}$, sodass $\sum_{i \in I} w_i \leq s_w$ und $\sum_{i \in I} n_i \geq s_n$?

Beachte: Für $w_i = n_i$ und $s_n = s_w$ ergibt sich genau das SUBSET-SUM-Problem.

NP-Vollständigkeit von KNAPSACK

Satz

KNAPSACK ist NP-vollständig.

Beweis

1. KNAPSACK $\in$ NP:

Ein Zeuge ist eine Menge $I \subseteq \{1, \dots, k\}$.

Die Prüfrelation R_{KNAPSACK} überprüft, ob $\sum_{i \in I} w_i \leq s_w$ und $\sum_{i \in I} n_i \geq s_n$ gilt.

Dies ist deterministisch in Polynomialzeit überprüfbar.

NP-Vollständigkeit von KNAPSACK

Beweis (Fortsetzung)

2. KNAPSACK ist NP-schwer:

Wir zeigen $\text{SUBSET-SUM} \leq_p \text{KNAPSACK}$.

Sei $((a_1, \dots, a_k), s)$ eine SUBSET-SUM-Instanz.

Sei $f((a_1, \dots, a_k), s) = ((w_1, \dots, w_k), (n_1, \dots, n_k), s_w, s_m)$ mit $w_i = a_i$, $n_i = a_i$ für $i \in \{1, \dots, k\}$ und $s_w = s$ und $s_m = s$.

f ist in polynomieller Zeit von einer DTM berechenbar.

$((a_1, \dots, a_k), s)$ ist SUBSET-SUM-lösbar g.d.w.

$f((a_1, \dots, a_k), s)$ KNAPSACK-lösbar ist.



Das PARTITION-Problem

Definition

Das **PARTITION-Problem** lässt sich wie folgt formulieren.

gegeben: natürliche Zahlen $a_1, \dots, a_k \in \mathbb{N}$

gefragt: Gibt es eine Teilmenge $I \subseteq \{1, \dots, k\}$, sodass $\sum_{i \in I} a_i = \sum_{i \in \{1, \dots, k\} \setminus I} a_i$?

Beispiel:

Gegeben seien $k = 4$ und $a_1, \dots, a_4 = 7, 2, 8, 13$.

Lösung: z.B. $I = \{1, 3\}$, da $\sum_{i \in \{1,3\}} a_i = 15 = \sum_{i \in \{2,4\}} a_i$.

NP-Vollständigkeit von PARTITION

Satz

PARTITION ist NP-vollständig.

Beweis

1. PARTITION $\in$ NP:

Ein Zeuge ist $I \subseteq \{1, \dots, k\}$.

Die Prüfrelation testet, ob $\sum_{i \in I} a_i = \sum_{i \in \{1, \dots, k\} \setminus I} a_i$.

Dies kann in Polynomialzeit auf einer DTM entschieden werden.

Beweis (Fortsetzung)

2. PARTITION ist NP-schwer:

Wir zeigen $\text{SUBSET-SUM} \leq_p \text{PARTITION}$.

Sei $f((a_1, \dots, a_k), s) = (a_1, \dots, a_k, a_{k+1}, a_{k+2})$ mit $a_{k+1} = A + s$ und $a_{k+2} = 2A - s$, wobei $A = \sum_{i=1}^k a_i$.

Beachte: $\sum_{i=1}^{k+2} a_i = 4A$.

Die Funktion f ist total und kann in polynomieller Zeit berechnet werden.

Beispiel für die SUBSET-SUM-Instanzkonstruktion

SUBSET-SUM-Instanz:

$$((1, 2, 3, 4, 5, 6), 14)$$

Wir haben $s = 14$, $A = 21$, $A + s = 35$, $2A = 42$ und $2A - s = 28$.

PARTITION-Instanz zur SUBSET-SUM-Instanz:

$$(1, 2, 3, 4, 5, 6, 35, 28)$$

Lösung der PARTITION-Instanz: z.B. $I = \{1, 3, 4, 6, 8\}$,
da $1 + 3 + 4 + 6 + 28 = 42 = 2 + 5 + 35$.

Lösung der SUBSET-SUM-Instanz: $\{1, 3, 4, 6\}$.
Tatsächlich $1 + 3 + 4 + 6 = 14$.

NP-Vollständigkeit von PARTITION

Beweis (Fortsetzung)

Wir zeigen: $((a_1, \dots, a_k), s)$ ist SUBSET-SUM-lösbar g.d.w.

$(a_1, \dots, a_k, a_{k+1}, a_{k+2})$ PARTITION-lösbar ist.

$\Rightarrow$ Wenn $I \subseteq \{1, \dots, k\}$ eine Lösung für $((a_1, \dots, a_k), s)$ ist, dann ist $I \cup \{k+2\}$ eine Lösung für $(a_1, \dots, a_k, a_{k+1}, a_{k+2})$, weil $\sum_{i \in I \cup \{k+2\}} a_i = s + (2A - s) = 2A$, was die Hälfte von $\sum_{i=1}^{k+2} a_i = 4A$ ausmacht.

$\Leftarrow$ Wenn $(a_1, \dots, a_k, a_{k+1}, a_{k+2})$ eine Lösung $I \subseteq \{1, \dots, k+2\}$ hat, dann $\sum_{i \in I} a_i = 2A = \sum_{i \in \{1, \dots, k+2\} \setminus I} a_i$.

$k+1$ und $k+2$ können nicht gleichzeitig in I sein, da sonst die Summe zu groß ist.

► Wenn $k+2 \in I$, dann $I' = I \setminus \{k+2\}$.

► Wenn $k+1 \in I$, dann $I' = (\{1, \dots, k+2\} \setminus I) \setminus \{k+2\}$.

In beiden Fällen ist $\sum_{i \in I'} a_i = 2A - (2A - s) = s$ und

die Indexmenge I' ist daher eine Lösung von $((a_1, \dots, a_k), s)$. □

Das BIN-PACKING-Problem

Definition

Das **BIN-PACKING-Problem** lässt sich wie folgt formulieren.

gegeben: natürliche Zahlen $a_1, \dots, a_k \in \mathbb{N}$,
die Behältergröße $b \in \mathbb{N}$ und
die Anzahl der Behälter m

gefragt: Kann man alle gegebenen Zahlen so auf die Behälter aufteilen, sodass keiner der Behälter überläuft?

Formal: Gibt es eine totale Funktion $assign : \{1, \dots, k\} \rightarrow \{1, \dots, m\}$,
sodass für alle Behälter $j \in \{1, \dots, m\}$ gilt: $(\sum_{i: assign(i)=j} a_i) \leq b$?

Satz

BIN-PACKING ist NP-vollständig.

Beweis

1. BIN-PACKING $\in$ NP:

Ein Zeuge ist eine Zuordnung, die für jede Zahl a_i angibt, in welchen Behälter sie gehört.

Die Prüfrelation testet, dass diese Zuordnung keinen Behälter überlaufen lässt.

Dies kann in Polynomialzeit auf einer DTM getestet werden.

Beweis (Fortsetzung)

2. BIN-PACKING ist NP-schwer:

Wir zeigen $\text{PARTITION} \leq_p \text{BIN-PACKING}$.

Sei $(a_1, \dots, a_k)$ eine PARTITION-Instanz. Sei $f(a_1, \dots, a_k)$ die BIN-PACKING-Instanz mit Zahlen $a_1, \dots, a_k$,

Behältergröße $b = \left\lfloor \frac{\sum_{i=1}^k a_i}{2} \right\rfloor$ und $m = 2$ Behältern.

Die Funktion f ist total und kann in polynomieller Zeit berechnet werden.

NP-Vollständigkeit von BIN-PACKING

Beweis (Fortsetzung)

Wir zeigen: $(a_1, \dots, a_k)$ ist PARTITION-lösbar g.d.w.

$((a_1, \dots, a_k), b, 2)$ BIN-PACKING-lösbar ist.

Wenn $\sum_{i=1}^k a_i$ ungerade ist, dann ist die PARTITION-Instanz unlösbar und die BIN-PACKING-Instanz ebenso. Sonst:

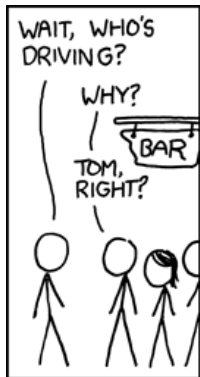
$\Rightarrow$ Wenn $I \subseteq \{1, \dots, k\}$ eine Lösung für PARTITION ist, dann ist

$$\text{assign}(i) = \begin{cases} 1 & \text{wenn } i \in I \\ 2 & \text{wenn } i \notin I \end{cases}$$

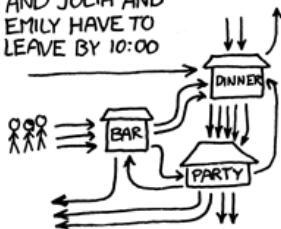
eine Lösung für BIN-PACKING.

$\Leftarrow$ Sei assign eine Lösung für BIN-PACKING.

Mit $I = \{i \mid 1 \leq i \leq k, \text{assign}(i) = 1\}$ kann eine Lösung für PARTITION erstellt werden. □



SOMEONE HAS TO GET PAUL, AND JULIA AND EMILY HAVE TO LEAVE BY 10:00



THE LOGISTICS OF WHO CAN GET DRUNK ARE NONTRIVIAL.



xkcd.com/589/