

Theoretische Informatik und Logik

10. Vorlesung: NP-Vollständigkeit, Teil 2

Markus Krötzsch

Professur Wissensbasierte Systeme

TU Dresden, 21. Mai 2026

NP-vollständige Probleme

NP-vollständige Probleme = Probleme, die mindestens so schwer sind, wie alle anderen Probleme in NP

Bisher haben wir gezeigt:

SAT
 \leq_p
CLIQUE
 \leq_p
Unabhängige Menge

Probleme mit Gewichten und Zahlen

Teilmengen-Summe

Teilmengen-Summe (subset sum)

Gegeben: Eine Menge von Gegenständen $S = \{a_1, \dots, a_n\}$, wobei jedem Gegenstand a_i ein Wert $v(a_i)$ zugeordnet ist; eine gewünschte Zahl z

Frage: Gibt es eine Teilmenge $T \subseteq S$ mit $\sum_{a \in T} v(a) = z$?

Anmerkung: Mehrere Gegenstände können gleiche Werte haben.

Satz: Teilmengen-Summe ist NP-vollständig.

Beweis:

- (1) **Teilmengen-Summe** \in NP haben wir bereits festgestellt
- (2) NP-Schwere zeigen wir durch Reduktion **SAT** \leq_p **Teilmengen-Summe**

Beispiel

$$(p_1 \vee p_2 \vee p_3) \wedge (\neg p_1 \vee \neg p_4) \wedge (p_4 \vee p_5 \vee \neg p_2 \vee \neg p_3)$$

	p_1	p_2	p_3	p_4	p_5	C_1	C_2	C_3			
$v(t_1)$	= 1	0	0	0	0	1	0	0			
$v(f_1)$	= 1	0	0	0	0	0	1	0			
$v(t_2)$	=	1	0	0	0	1	0	0			
$v(f_2)$	=		1	0	0	0	0	1			
$v(t_3)$	=			1	0	0	1	0	0		
$v(f_3)$	=				1	0	0	0	1		
$v(t_4)$	=					1	0	0	1		
$v(f_4)$	=						1	0	0		
$v(t_5)$	=							1	0	1	
$v(f_5)$	=								1	0	0
$v(m_{1,1})$	=						1	0	0		
$v(m_{1,2})$	=						1	0	0		
$v(m_{2,1})$	=							1	0		
$v(m_{3,1})$	=									1	
$v(m_{3,2})$	=									1	
$v(m_{3,3})$	=									1	
z	=	1	1	1	1	1	3	2	4		

SAT \leq_p Teilmengen-Summe

Gegeben: Formel $F = (C_1 \wedge \dots \wedge C_k)$ in KNF mit maximal 9 Literalen pro Klausel

Das ist „o.B.d.A.“ **Übung:** Reduzieren Sie **SAT** für beliebige Formeln auf eine KNF mit maximal 3 Literalen pro Klausel.

Vorsicht: Unsachgemäß erstellte KNF können explodieren! (siehe 3. Vorlesung oder auch Wikipedia [Tseitin-Transformation](#))

Seien p_1, \dots, p_n die Variablen in F .

Für jedes p_i definieren wir Gegenstände t_i und f_i :

$$v(t_i) := a_1 \cdots a_n c_1 \cdots c_k \quad \text{wobei} \quad a_j := \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$
$$c_j := \begin{cases} 1 & p_i \text{ kommt in } C_j \text{ vor} \\ 0 & \text{sonst} \end{cases}$$
$$v(f_i) := a_1 \cdots a_n c_1 \cdots c_k \quad \text{wobei} \quad a_j := \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$
$$c_j := \begin{cases} 1 & \neg p_i \text{ kommt in } C_j \text{ vor} \\ 0 & \text{sonst} \end{cases}$$

Beispiel

$$(p_1 \vee p_2 \vee p_3) \wedge (\neg p_1 \vee \neg p_4) \wedge (p_4 \vee p_5 \vee \neg p_2 \vee \neg p_3)$$

	p_1	p_2	p_3	p_4	p_5	C_1	C_2	C_3	
$v(t_1)$	=	1	0	0	0	0	1	0	0
$v(f_1)$	=	1	0	0	0	0	0	1	0
$v(t_2)$	=		1	0	0	0	1	0	0
$v(f_2)$	=		1	0	0	0	0	0	1
$v(t_3)$	=			1	0	0	1	0	0
$v(f_3)$	=			1	0	0	0	0	1
$v(t_4)$	=				1	0	0	0	1
$v(f_4)$	=				1	0	0	1	0
$v(t_5)$	=					1	0	0	1
$v(f_5)$	=					1	0	0	0
$v(m_{1,1})$	=						1	0	0
$v(m_{1,2})$	=						1	0	0
$v(m_{2,1})$	=							1	0
$v(m_{3,1})$	=								1
$v(m_{3,2})$	=								1
$v(m_{3,3})$	=								1
z	=	1	1	1	1	1	3	2	4

SAT \leq_p Teilmengen-Summe

Außerdem definieren wir für jede Klausel C_i genau $r := |C_i| - 1$ Gegenstände $m_{i,1}, \dots, m_{i,r}$

mit $v(m_{i,j}) := c_i \cdots c_k$ wobei $c_\ell := \begin{cases} 1 & \ell = i \\ 0 & \ell \neq i \end{cases}$

Definition von S : Damit ergibt sich die Menge

$$S := \{t_i, f_i \mid 1 \leq i \leq n\} \cup \{m_{i,j} \mid 1 \leq i \leq k, 1 \leq j \leq |C_i| - 1\}$$

Gesuchte Zahl: Wir bestimmen die gesuchte Zahl z wie folgt:

$$z := a_1 \cdots a_n c_1 \cdots c_k \text{ wobei } a_i := 1 \text{ und } c_i := |C_i|$$

Behauptung: Es gibt $T \subseteq S$ mit $\sum_{a_i \in T} v(a_i) = z$ gdw. F erfüllbar ist.

Beispiel

$$(p_1 \vee p_2 \vee p_3) \wedge (\neg p_1 \vee \neg p_4) \wedge (p_4 \vee p_5 \vee \neg p_2 \vee \neg p_3)$$

	p_1	p_2	p_3	p_4	p_5	C_1	C_2	C_3	
$v(t_1)$	= 1	0	0	0	0	1	0	0	
$v(f_1)$	= 1	0	0	0	0	0	1	0	
$v(t_2)$	=	1	0	0	0	1	0	0	
$v(f_2)$	=		1	0	0	0	0	1	
$v(t_3)$	=			1	0	0	1	0	0
$v(f_3)$	=				1	0	0	0	1
$v(t_4)$	=					1	0	0	1
$v(f_4)$	=						1	0	0
$v(t_5)$	=							1	0
$v(f_5)$	=								1
$v(m_{1,1})$	=						1	0	0
$v(m_{1,2})$	=						1	0	0
$v(m_{2,1})$	=							1	0
$v(m_{3,1})$	=								1
$v(m_{3,2})$	=								1
$v(m_{3,3})$	=								1
z	=	1	1	1	1	1	3	2	4

NP-Vollständigkeit von TEILMENGEN-SUMME

Behauptung:

Wenn F erfüllbar ist, dann gibt es $T \subseteq S$ mit $\sum_{a_i \in T} v(a_i) = z$.

- Sei w eine erfüllende Belegung für F
- Wir definieren:

$$T_1 := \{t_i \mid w(p_i) = \mathbf{1}, 1 \leq i \leq m\} \cup \{f_i \mid w(p_i) = \mathbf{0}, 1 \leq i \leq m\}$$

- Sei r_i jeweils die Zahl der von w erfüllten Literale in C_i .
Wir definieren:

$$T_2 := \{m_{i,j} \mid 1 \leq i \leq k, 1 \leq j \leq |C_i| - r_i\}$$

- Die gesuchte Menge von Gegenständen ist $T := T_1 \cup T_2$.

Damit folgt: $\sum_{s \in T} v(s) = z$

NP-Vollständigkeit von TEILMENGEN-SUMME

Behauptung:

Gibt es $T \subseteq S$ mit $\sum_{a_i \in T} v(a_i) = z$ dann ist F erfüllbar.

Sei $T \subseteq S$ die gesuchte Menge mit $\sum_{s \in T} v(s) = z$

Wir definieren $w(p_i) = \begin{cases} 1 & \text{falls } t_i \in T \\ 0 & \text{falls } f_i \in T \end{cases}$

w ist wohldefiniert, da für alle i gilt:
 $t_i \in T$ oder $f_i \in T$, aber nicht beides

Außerdem muss für jede Klausel ein Literal auf **1** abgebildet werden, da die entsprechenden Hilfszahlen $m_{i,j} \in S$ zusammen nicht die geforderte Zahl an Literalen pro Klausel erreichen. □

Das Rucksack-Problem

Rucksack (Knapsack)

Gegeben: Eine Menge von Gegenständen $G = \{a_1, \dots, a_n\}$, wobei jedem Gegenstand a_i ein Wert $v(a_i)$ und ein Gewicht $g(a_i)$ zugeordnet ist; ein Mindestwert w und ein Gewichtslimit ℓ

Frage: Gibt es eine Teilmenge $T \subseteq G$, so dass

- $\sum_{a \in T} g(a) \leq \ell$ und
- $\sum_{a \in T} v(a) \geq w$?

Satz: **Rucksack** ist NP-vollständig.

Beweis:

(1) **Rucksack** \in NP: Das Zertifikat ist T

(2) **Rucksack** ist NP-schwer:

durch Reduktion **Teilmengen-Summe** \leq_p **Rucksack**

Teilmengen-Summe \leq_p Rucksack

Gegeben: Eine Instanz von **Teilmengen-Summe** (Menge von Gegenständen $S = \{a_1, \dots, a_n\}$ mit Wert $v(a_i)$; gewünschte Zahl z)

Daraus konstruieren wir:

- $G := \{1, \dots, n\}$: Menge der Gegenstände
- Für alle $i \in G$ setzen wir: $v(i) = g(i) = v(a_i)$
- Zielwert $w := z$ und Gewichtslimit $\ell := z$

Offensichtlich ist diese Übersetzung **polynomiell**.

Damit gilt für jede Teilmenge $T \subseteq G$

$$\sum_{i \in T} v(a_i) = z \quad \text{gdw.} \quad \begin{array}{l} \sum_{i \in T} v(i) \geq w = z \\ \sum_{i \in T} g(i) \leq \ell = z \end{array}$$

\leadsto Die Reduktion ist **korrekt**.

□

Pseudopolynomielle Probleme

Eine polynomielle Lösung für **Rucksack**

Mittels **dynamischer Programmierung** kann man **Rucksack** in der Zeit $O(n\ell)$ lösen

Initialisierung:

Erzeuge eine $(\ell + 1) \times (n + 1)$ -Matrix M

Setze $M(g, 0) = M(0, i) = 0$ für alle $1 \leq g \leq \ell$ $1 \leq i \leq n$

Berechnung: Für $i = 0, 1, \dots, n - 1$ berechne $M(g, i + 1)$ als

$$M(g, i + 1) := \max(M(g, i), M(g - g(i + 1), i) + v(i + 1))$$

(Falls $g - g(i + 1) < 0$, dann verwenden wir immer $M(g, i)$.)

$M(g, i)$ = Größter Gesamtwert unter den ersten i Gegenständen
bei Einhaltung des Gewichtslimits g

Akzeptanz: Akzeptiere falls M einen Eintrag $\geq w$ hat.

Beispiel

Eingabe: $G := \{1, 2, 3, 4\}$ mit

Werten: $v(1) := 1$ $v(2) := 3$ $v(3) := 4$ $v(4) := 2$

Gewichten: $g(1) := 1$ $g(2) := 1$ $g(3) := 3$ $g(4) := 2$

Gewichtslimit: $\ell := 5$

Mindestwert: $w := 7$

Gewichts- limit g	max. Wert für $\leq i$ Gegenstände				
	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
0	0	0	0	0	0
1	0	1	3	3	3
2	0	1	4	4	4
3	0	1	4	4	5
4	0	1	4	7	7
5	0	1	4	8	8

Initialisiere $M(g, 0) = M(0, i) = 0$ für alle $1 \leq g \leq \ell$ und $1 \leq i \leq n$

$M(g, i + 1) := \max(M(g, i), M(g - g(i + 1), i) + v(i + 1))$

Eine polynomielle Lösung für **Rucksack**

Mittels **dynamischer Programmierung** kann man **Rucksack** in der Zeit $O(n\ell)$ lösen

Initialisierung:

Erzeuge eine $(\ell + 1) \times (n + 1)$ -Matrix M

Setze $M(g, 0) = M(0, i) = 0$ für alle $1 \leq g \leq \ell$ $1 \leq i \leq n$

Berechnung: Für $i = 0, 1, \dots, n - 1$ berechne $M(g, i + 1)$ als

$$M(g, i + 1) := \max(M(g, i), M(g - g(i + 1), i) + v(i + 1))$$

(Falls $g - g(i + 1) < 0$, dann verwenden wir immer $M(g, i)$.)

$M(g, i)$ = Größter Gesamtwert unter den ersten i Gegenständen
bei Einhaltung des Gewichtslimits g

Akzeptanz: Akzeptiere falls M einen Eintrag $\geq w$ hat.

Beispiel

Eingabe: $G := \{1, 2, 3, 4\}$ mit

Werten: $v(1) := 1$ $v(2) := 3$ $v(3) := 4$ $v(4) := 2$

Gewichten: $g(1) := 1$ $g(2) := 1$ $g(3) := 3$ $g(4) := 2$

Gewichtslimit: $\ell := 5$

Mindestwert: $w := 7$

Gewichts- limit g	max. Wert für $\leq i$ Gegenstände				
	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
0	0	0	0	0	0
1	0	1	3	3	3
2	0	1	4	4	4
3	0	1	4	4	5
4	0	1	4	7	7
5	0	1	4	8	8

Initialisiere $M(g, 0) = M(0, i) = 0$ für alle $1 \leq g \leq \ell$ und $1 \leq i \leq n$

$M(g, i + 1) := \max(M(g, i), M(g - g(i + 1), i) + v(i + 1))$

P = NP?

Zusammenfassung:

- **Rucksack** ist NP-vollständig
- **Rucksack** ist mittels Dynamischer Programmierung in Zeit $O(n\ell)$ lösbar

Rucksack (Knapsack)

Gegeben: Eine Menge von Gegenständen $G = \{a_1, \dots, a_n\}$, wobei jedem Gegenstand a_i ein Wert $v(a_i)$ und ein Gewicht $g(a_i)$ zugeordnet ist; ein Mindestwert w und ein Gewichtslimit ℓ

Frage: Gibt es eine Teilmenge $T \subseteq G$, so dass

- $\sum_{a \in T} g(a) \leq \ell$ und
- $\sum_{a \in T} v(a) \geq w$?

Haben wir also gezeigt, dass P = NP?

Pseudopolynomielle Probleme

Unser Algorithmus zeigt nicht **Rucksack** $\in P$!

- Die Eingabe von **Rucksack** hat die Länge $O(n + \log w + \log \ell)$
- Die Zeit $O(n\ell)$ ist daher **nicht** polynomiell bzgl. der Eingabelänge

Ein Problem ist in **pseudopolynomieller Zeit**, wenn es durch eine DTM gelöst wird, die polynomiell zeitbeschränkt ist bzgl. der Eingabelänge **und** des Betrages aller Zahlen in der Eingabe.

Äquivalent: ein Problem ist pseudopolynomiell, wenn es in P_{TIME} liegt, wenn man alle Zahlen **unär** kodiert.

Beispiel: Rucksack ist pseudopolynomiell: beschränkt man das Problem auf Instanzen mit $\ell \leq p(n)$ für ein Polynom p , so erhält man ein Problem in P (wobei n die Zahl der Gegenstände ist).

Starke NP-Vollständigkeit

Probleme, welche selbst dann noch NP-vollständig sind, wenn man alle Zahlen unär kodiert, heißen **stark NP-vollständig**.

Beispiele:

- **SAT** ist stark NP-vollständig (keine Zahlen in der Eingabe)
- **CLIQUE** ist stark NP-vollständig
- ...

Beispiele:

- **Rucksack** ist pseudopolynomiell
- **Teilmengen-Summe** ist pseudopolynomiell

Anmerkung: Die Reduktion **SAT** \leq_p **Teilmengen-Summe** erzeugt eine polynomielle Instanz, bei der die Beträge der Zahlen exponentiell groß sind.

Fake-News erkennen (1)

Immer wieder gibt es Berichte darüber, dass irgendein neuartiges Berechnungsmodell oder ausgefallene physikalische Anordnung NP-vollständige Probleme in polynomieller Zeit lösen könne ...

↪ Meist kann man leicht sehen, wo der Fehler liegt

Typische Kontrollfragen:

Ist das Problem pseudopolynomiell?
Wie sind Zahlenbeträge kodiert?

Beispiel: Molekulare Anordnungen zum Lösen von **Teilmengen-Summe**, wobei Zahlen durch eine entsprechende Zahl an Molekülen kodiert werden.

Fake-News erkennen (2)

Immer wieder gibt es Berichte darüber, dass irgendein neuartiges Berechnungsmodell oder ausgefallene physikalische Anordnung NP-vollständige Probleme in polynomieller Zeit lösen könne . . .

↪ Meist kann man leicht sehen, wo der Fehler liegt

Typische Kontrollfragen:

Wächst der neuartige Computer mit dem Problem?
Ist dieses Wachstum exponentiell?

Beispiel: DNS-Rechner lösen Optimierungsaufgaben, aber müssen alle (exponentiell viele) mögliche Lösungen als DNS-Sequenz kodieren.

Fake-News erkennen (3)

Immer wieder gibt es Berichte darüber, dass irgendein neuartiges Berechnungsmodell oder ausgefallene physikalische Anordnung NP-vollständige Probleme in polynomieller Zeit lösen könne . . .

↪ Meist kann man leicht sehen, wo der Fehler liegt

Typische Kontrollfragen:

Wird das Problem wirklich gelöst oder handelt es sich eher um ein Approximationsverfahren? Ist klar, was ein „Berechnungsschritt“ ist und ist deren Zahl wirklich polynomiell?

Beispiel: Ein Schaltkreis mit Rückkopplungen, welcher „in der Regel“ zu einer „guten“ Lösung eines NP-vollständigen Problems konvergiert, und das „nach kurzer Zeit.“

NP und andere Klassen

P vs. NP

Bis heute ist nicht bekannt, ob $P \neq NP$ gilt oder nicht.

- Intuitiv gefragt: „Wenn es einfach ist, eine mögliche Lösung für ein Problem zu prüfen, ist es dann auch einfach, eine zu finden?“
- Übertrieben: „Can creativity be automated?“ (Wigderson, 2006)
- Seit über 40 Jahren ungelöst
- Eines der größten offenen Probleme der Informatik und Mathematik unserer Zeit
- 1.000.000 USD Preisgeld für die Lösung („Millenium Problem“)

Mögliche Konsequenzen

Falls $P = NP$, dann gilt

- $NP = coNP$ (warum?)
- und jedes nichttriviale Problem in P ist NP -vollständig (warum?)

Falls $P \neq NP$, dann gilt

- $P \neq coNP$ (warum?)
- $L \neq NP$ (warum?)
- $P \neq PSPACE$ (warum?)
- kein NP -vollständiges Problem ist in P (warum?)

Noch schwerere Fragen

Selbst wenn $P \neq NP$ bewiesen werden sollte, gäbe es noch viele offene Fragen ...

$NP = coNP$? Falls es für die lösbaren Instanzen eines Problems kurze Zertifikate gibt, gibt es dann immer auch kurze Zertifikate für die Unlösbarkeit von Instanzen?

- Die meisten Expert:innen glauben das nicht
- Falls die Antwort „nein“ lautet, dann folgt $P \neq NP$

$P = NP \cap coNP$? Falls es für die Lösbarkeit und die Unlösbarkeit eines Problems kurze Zertifikate gibt, ist das Problem dann effizient lösbar?

- Die meisten Expert:innen glauben das nicht
- Falls die Antwort „nein“ lautet, dann folgt $P \neq NP$
- Ist die Antwort „ja“ und zudem $P \neq NP$, so folgt $NP \neq coNP$

Der Satz von Ladner

Drei mögliche Welten:

- (1) $P = NP$: jedes NP-vollständige Problem ist in P
- (2) $P \neq NP$: kein NP-vollständiges Problem ist in P
 - (2.a) Jedes NP-Problem, welches nicht NP-vollständig ist, liegt in P
 - (2.b) Es gibt Probleme, die nicht NP-vollständig sind und dennoch nicht in P liegen

Richard E. Ladner hat gezeigt, dass Fall (2.a) unmöglich ist:

Satz (Ladner): Falls $P \neq NP$, dann gibt es Probleme in NP, die weder NP-vollständig sind noch in P liegen.

Diese Probleme heißen **NP-intermediate** – wenn $P \neq NP$, dann sollte es viele davon geben ... aber wir kennen kaum Kandidaten:

- Faktorisierung (**Faktor-7** und ähnliche)
- Ermitteln, ob zwei Graphen isomorph (gleich bis auf Umbenennung) sind

Zusammenfassung und Ausblick

SAT \leq_p Teilmengen-Summe \leq_p Rucksack

Pseudopolynomielle Probleme werden einfacher, wenn man den Betrag der Zahlen in der Eingabe polynomiell beschränkt

Offene Fragen (geordnet von „leicht“ nach schwer): „L \neq NP?“, „P \neq PSPACE?“, „P \neq NP?“, „NP \neq coNP?“ „P \neq NP \cap coNP?“

Es sollte eigentlich eine Menge Probleme geben, die weder in P liegen, noch NP-vollständig sind

Was erwartet uns als nächstes?

- NL
- Komplexität jenseits von NP
- Spiele