

Übungsaufgaben

71 Aufgaben nach Kompetenzliste
Analyse von Algorithmen und Komplexität – CAU Kiel

Contents

1	Anwendung	2
2	Vorlesungsbeweis	13
3	NP-Vollständigkeit	20
4	Approximative Algorithmen	56
5	ETH	70

1 Anwendung

1 Greedy anwenden

Rucksack mit Kapazität $B = 10$; Gegenstände als (p_i, w_i) :

$$A = (10, 6), \quad B = (7, 5), \quad C = (6, 4), \quad D = (3, 3), \quad E = (4, 2).$$

Wenden Sie Greedy an und geben Sie den Lösungswert an.

Hinweis: Greedy sortiert absteigend nach Profitdichte p_i/w_i und packt in dieser Reihenfolge jeden Gegenstand ein, der noch passt.

Lösung.

- Dichten p_i/w_i : A 1,67, B 1,4, C 1,5, D 1,0, E 2,0. Sortiert: E, A, C, B, D .
- E ($w=2$) einpacken – Restkapazität 8.
- A ($w=6$) einpacken – Restkapazität 2.
- C ($w=4$) passt nicht.
- B ($w=5$) passt nicht.
- D ($w=3$) passt nicht.

Auswahl $\{E, A\}$, Lösungswert $GA = 4 + 10 = 14$. (Optimum wäre $\{A, C\}$ mit Wert 16.) □

2 ModifiedGreedy anwenden

Rucksack mit Kapazität $B = 8$; Gegenstände (p_i, w_i) :

$$a = (3, 2), \quad b = (3, 2), \quad c = (9, 7), \quad d = (1, 2).$$

Wenden Sie ModifiedGreedy an. Wie ändert sich die Lösung gegenüber Greedy?

Hinweis: ModifiedGreedy gibt das Bessere aus Greedy-Lösung und profitreichstem Einzel-Gegenstand (der allein passt) aus.

Lösung.

- Dichten: a 1,5, b 1,5, c 1,29, d 0,5. Reihenfolge a, b, c, d .
- Greedy: a rein (Rest 6), b rein (Rest 4), c ($w=7$) passt nicht, d rein (Rest 2). Greedy-Wert $GA = 3 + 3 + 1 = 7$.
- Profitreichster Einzel-Gegenstand: c mit Profit 9 (und $w = 7 \leq 8$).
- $MGA = \max\{7, 9\} = 9$.

Die Lösung wechselt von der Greedy-Auswahl $\{a, b, d\}$ (Wert 7) zum einzelnen Gegenstand c (Wert 9). $\square$

3 Sahni anwenden

Rucksack mit Kapazität $B = 11$; Gegenstände (p_i, w_i) :

$$A = (5, 4), \quad B = (6, 5), \quad C = (7, 6), \quad D = (4, 6).$$

Wenden Sie Sahni mit $k = 2$ an.

Hinweis: Sahni probiert jede Vorauswahl aus höchstens k Gegenständen, füllt sie jeweils mit Greedy auf und gibt die beste gefundene Lösung aus; das liefert Güte $1 + \frac{1}{k}$.

Lösung.

- Reines Greedy (Dichten A 1,25, B 1,2, C 1,17, D 0,67): A rein (Rest 7), B rein (Rest 2), C und D passen nicht – Wert 11.
- Vorauswahl $\{A, C\}$: Gewicht $4 + 6 = 10$, Greedy füllt nichts mehr (Rest 1) – Wert 12.
- Vorauswahl $\{B, C\}$: Gewicht $5 + 6 = 11$, kein Platz mehr – Wert $6 + 7 = 13$.
- Alle übrigen Vorauswahlen bleiben ≤ 12 .

Beste Lösung: $\{B, C\}$ mit Wert 13 (hier zugleich das Optimum).

□

4 ListScheduling anwenden

$m = 3$ Maschinen; Jobs in gegebener Reihenfolge $p = (3, 5, 2, 4, 1)$. Wenden Sie ListScheduling an.

Hinweis: Jeder Job kommt der Reihe nach auf die aktuell am wenigsten belastete Maschine; Lastvektor (M_1, M_2, M_3) .

Lösung.

- Start $(0, 0, 0)$.
- $3 \rightarrow M_1$: $(3, 0, 0)$.
- $5 \rightarrow M_2$: $(3, 5, 0)$.
- $2 \rightarrow M_3$: $(3, 5, 2)$.
- $4 \rightarrow M_3$ (kleinste Last 2): $(3, 5, 6)$.
- $1 \rightarrow M_1$ (kleinste Last 3): $(4, 5, 6)$.

Makespan $C_{\max} = 6$.

□

5 LPT anwenden

$m = 3$ Maschinen; Jobs $p = (4, 2, 6, 3, 5, 1)$. Wenden Sie LPT an.

Hinweis: LPT ist ListScheduling nach absteigender Sortierung der Bearbeitungszeiten.

Lösung.

- Absteigend sortiert: 6, 5, 4, 3, 2, 1.
- $6 \rightarrow M_1$: (6, 0, 0); $5 \rightarrow M_2$: (6, 5, 0); $4 \rightarrow M_3$: (6, 5, 4).
- $3 \rightarrow M_3$ (kleinste Last 4): (6, 5, 7).
- $2 \rightarrow M_2$ (kleinste Last 5): (6, 7, 7).
- $1 \rightarrow M_1$ (kleinste Last 6): (7, 7, 7).

Makespan $C_{\max} = 7$; wegen $\sum_j p_j = 21 = 3 \cdot 7$ ist das zugleich optimal.

□

6 RoundRobin anwenden

$m = 2$ Maschinen; Jobs $p = (3, 6, 1, 4)$. Wenden Sie RoundRobin an.

Hinweis: RoundRobin sortiert die Jobs absteigend und verteilt sie reihum: der j -te sortierte Job kommt auf Maschine $((j - 1) \bmod m) + 1$.

Lösung.

- Absteigend sortiert: 6, 4, 3, 1.
- Reihum bei $m = 2$: $6 \rightarrow M_1, 4 \rightarrow M_2, 3 \rightarrow M_1, 1 \rightarrow M_2$.
- Lasten: $M_1 = 6 + 3 = 9, \quad M_2 = 4 + 1 = 5$.

Makespan $C_{\max} = 9$. (Optimal wäre $\{6, 1\} / \{4, 3\}$ mit $\text{OPT} = 7$.)

□

7 Parallel-Task-Scheduling anwenden

$m = 3$ Maschinen. Ein Job (p_j, q_j) belegt q_j Maschinen gleichzeitig für Zeit p_j . Jobs in Reihenfolge:

$$(2, 1), \quad (2, 2), \quad (1, 3), \quad (3, 1).$$

Wenden Sie Parallel-Task-ListScheduling an.

Hinweis: Jeder Job startet zum frühesten Zeitpunkt, an dem q_j Maschinen frei sind, auf den q_j am wenigsten belasteten Maschinen.

Lösung.

Lastvektor (M_1, M_2, M_3) als Fertigstellungszeiten; Start $(0, 0, 0)$.

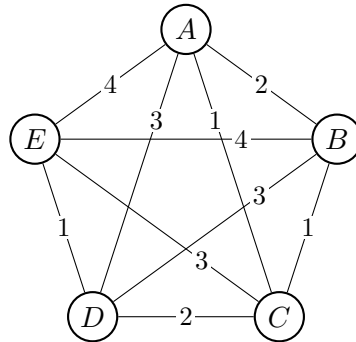
- $(2, 1)$: eine Maschine M_1 (Last 0), Start 0, läuft $[0, 2] - (2, 0, 0)$.
- $(2, 2)$: zwei Maschinen M_2, M_3 (Last 0), Start 0, läuft $[0, 2] - (2, 2, 2)$.
- $(1, 3)$: alle drei Maschinen, frühester Start = $\max = 2$, läuft $[2, 3] - (3, 3, 3)$.
- $(3, 1)$: leerste Maschine M_1 (Last 3), Start 3, läuft $[3, 6] - (6, 3, 3)$.

Makespan $C_{\max} = 6$.

□

8 TSP1 anwenden

Vollständiger, symmetrischer Graph G auf $\{A, B, C, D, E\}$; die Dreiecksungleichung gilt für jede Kante. Wenden Sie $\Delta TSP1$ ab Startknoten A an; geben Sie alle Zwischenschritte und die Tourlänge an.



Hinweis ($\Delta TSP1$): (1) MST T berechnen; (2) alle MST-Kanten verdoppeln; (3) Eulerkreis ab Startknoten; (4) Abkürzen – bereits besuchte Knoten überspringen. Güte 2.

Lösung.

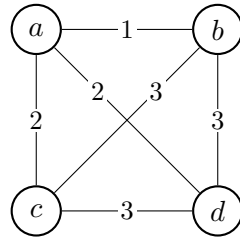
- (1) MST (Kruskal, aufsteigend): $A-C$ (1), $B-C$ (1), $D-E$ (1), $C-D$ (2); Gewicht 5. Knoten C hat Grad 3.
- (2) Verdoppeln: alle Grade werden gerade, Multigraph-Gewicht 10.
- (3) Eulerkreis ab A : $[A, C, B, C, D, E, D, C, A]$.
- (4) Abkürzen (besuchte Knoten überspringen): Tour $[A, C, B, D, E, A]$.

$$\text{Tourlänge} = AC + CB + BD + DE + EA = 1 + 1 + 3 + 1 + 4 = 10.$$

□

9 Christofides anwenden

Vollständiger, symmetrischer Graph G auf $\{a, b, c, d\}$; die Dreiecksungleichung gilt für jede Kante. Wenden Sie Christofides (Δ TSP2) ab Startknoten a an; geben Sie alle Zwischenschritte und die Tourlänge an.



Hinweis (Christofides): (1) MST T berechnen; (2) X = Knoten mit ungeradem Grad in T ; (3) minimales perfektes Matching K auf X ; (4) Eulerkreis in $T + K$; (5) Abkürzen. Güte $\frac{3}{2}$.

Lösung.

- (1) MST (Kruskal): $a-b$ (1), $a-c$ (2), $a-d$ (2); Gewicht 5 (Stern um a).
- (2) Grade im MST: $a: 3$, $b: 1$, $c: 1$, $d: 1$ – alle ungerade, also $X = \{a, b, c, d\}$.
- (3) Minimales perfektes Matching auf X : Kandidaten $\{ab, cd\} = 1+3 = 4$, $\{ac, bd\} = 2+3 = 5$, $\{ad, bc\} = 2+3 = 5$. Wähle $K = \{\{a, b\}, \{c, d\}\}$, Kosten 4.
- (4) Multigraph $T + K$ (Kante $a-b$ doppelt): alle Grade gerade. Eulerkreis ab a : $[a, b, a, c, d, a]$.
- (5) Abkürzen: Tour $[a, b, c, d, a]$.

Tourlänge = $ab + bc + cd + da = 1 + 3 + 3 + 2 = 9$.

□

10 Strip Packing: NFDH anwenden

Streifen der Breite 1. Rechtecke als (Breite, Höhe):

$$r_1 = (0,6, 3), r_2 = (0,5, 2), r_3 = (0,5, 2), r_4 = (0,4, 1), r_5 = (0,3, 1).$$

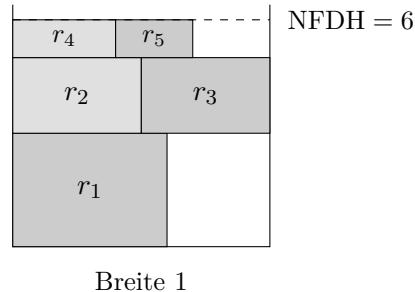
Wenden Sie NFDH an und geben Sie die Gesamthöhe an.

Hinweis: NFDH sortiert nach Höhe absteigend und packt stufenweise von links; passt ein Rechteck nicht mehr, beginnt eine neue Stufe darüber. Die Stufenhöhe ist die Höhe ihres ersten (höchsten) Rechtecks.

Lösung.

- Sortiert nach Höhe: $r_1(3), r_2(2), r_3(2), r_4(1), r_5(1)$.
- Stufe 1: r_1 (Breite 0,6, Höhe 3). r_2 : $0,6 + 0,5 = 1,1 > 1$ – passt nicht.
- Stufe 2: r_2 (Höhe 2), Breite 0,5. r_3 : $0,5 + 0,5 = 1,0 \leq 1$ – passt. r_4 : $1,0 + 0,4 > 1$ – passt nicht.
- Stufe 3: r_4 (Höhe 1), Breite 0,4. r_5 : $0,4 + 0,3 = 0,7 \leq 1$ – passt.

Gesamthöhe NFDH = $3 + 2 + 1 = 6$.



Die Gesamtfläche ist 4,5 (untere Schranke $\text{OPT} \geq 4,5$); eine Packung der Höhe 5 existiert. $\square$

2 Vorlesungsbeweis

11 $P \subseteq \mathbf{NP}$ zeigen

Zeigen Sie: $P \subseteq \mathbf{NP}$.

Lösung.

Sei $L \in P$ beliebig. Dann entscheidet ein Algorithmus A die Sprache L in Zeit $O(n^d)$.

Konstruktion: Fasse A als Verifizierer auf durch $V(x, c) := A(x)$; das Zertifikat c wird ignoriert.

Korrektheit: Ist $x \in L$, so akzeptiert $V(x, c)$ für jedes c , etwa $c = \varepsilon$ – ein akzeptiertes Zertifikat existiert. Ist $x \notin L$, so verwirft $V(x, c)$ für jedes c – kein Zertifikat wird akzeptiert. Also $L = \{x \mid \exists c : V(x, c) = 1\}$.

Laufzeit: V läuft wie A in $O(n^d)$. Damit ist V ein Verifizierer für L , und $L \in \mathbf{NP}$. Da L beliebig war, folgt $P \subseteq \mathbf{NP}$. $\square$

12 NDTM-Definition $\subseteq$ Verifizierer-Definition zeigen

Sei L eine Sprache, die von einer NDTM M in nichtdeterministischer Zeit T (ein Polynom) entschieden wird. Zeigen Sie, dass ein polynomieller Verifizierer für L existiert.

Lösung.

Eingabe: Wort x und als Zertifikat eine Folge c von höchstens $T(|x|)$ nichtdeterministischen Entscheidungen von M .

Der Verifizierer $V(x, c)$ arbeitet wie folgt:

- Simuliere M auf x deterministisch für höchstens $T(|x|)$ Schritte.
- Wähle an jeder Verzweigung den Übergang, den die Folge c vorgibt.
- Akzeptiere, falls die Simulation akzeptierend hält; sonst verwirf.

Korrektheit: Ist $x \in L$, so hat M einen akzeptierenden Rechenweg mit höchstens $T(|x|)$ Schritten; dessen Entscheidungsfolge ist ein Zertifikat, das V akzeptiert. Ist $x \notin L$, so akzeptiert kein Rechenweg von M ; jede Folge c steuert die Simulation einen Rechenweg entlang, also verwirft V jedes Zertifikat.

Laufzeit: Höchstens $T(|x|)$ Simulationsschritte mit konstantem Aufwand – $O(T(|x|))$; das Zertifikat hat Länge höchstens $T(|x|)$. Also existiert ein polynomieller Verifizierer für L . $\square$

13 Verifizierer-Definition $\subseteq$ NDTM-Definition zeigen

Sei L eine Sprache mit polynomielltem Verifizierer M (Laufzeit $O(|x|^d)$): Für $x \in L$ existiert ein Zertifikat c mit $M(x, c) = 1$, für $x \notin L$ akzeptiert M kein Zertifikat. Zeigen Sie, dass eine NDTM N die Sprache L in Polynomialzeit entscheidet.

Lösung.

Sei $p(|x|) = O(|x|^d)$ die Laufzeit von M . In $p(|x|)$ Schritten liest M höchstens $p(|x|)$ Zeichen des Zertifikats.

Die NDTM N arbeitet bei Eingabe x in zwei Phasen:

- *Raten:* Schreibe mit nichtdeterministischen Entscheidungen einen String u mit $|u| \leq p(|x|)$ auf ein Arbeitsband.
- *Verifizieren:* Simuliere M auf (x, u) und übernimm dessen Antwort.

Korrektheit: Ist $x \in L$, so existiert ein Zertifikat c mit $M(x, c) = 1$; M liest davon höchstens die ersten $p(|x|)$ Zeichen. Der Rechenweg, der genau diesen Anfang als u rät, akzeptiert – also akzeptiert N . Ist $x \notin L$, so verwirft M jedes geratene u ; alle Rechenwege verwerfen – also verwirft N .

Laufzeit: Phase 1 braucht $p(|x|)$ Rateschritte, Phase 2 läuft in $O(|x|^d)$ – gesamt $O(|x|^d)$. Also entscheidet N die Sprache L in nichtdeterministischer Polynomialzeit. $\square$

14 Transitivität von Polynomialzeitreduktionen zeigen

Zeigen Sie: Für Entscheidungsprobleme L_1, L_2, L_3 folgt aus $L_1 \leq_p L_2$ und $L_2 \leq_p L_3$ auch $L_1 \leq_p L_3$.

Lösung.

Sei R_1 eine Reduktion $L_1 \leq_p L_2$ mit Laufzeit $O(n^a)$ und R_2 eine Reduktion $L_2 \leq_p L_3$ mit Laufzeit $O(n^b)$. Setze $R := R_2 \circ R_1$.

Korrektheit: Für jede Eingabe x gilt

$$x \in L_1 \iff R_1(x) \in L_2 \iff R_2(R_1(x)) \in L_3,$$

also $x \in L_1 \iff R(x) \in L_3$.

Laufzeit: Sei $|x| = n$.

- $R_1(x)$ braucht $O(n^a)$ Zeit; die Ausgabe hat damit Größe $O(n^a)$.
- R_2 auf dieser Ausgabe braucht $O((n^a)^b) = O(n^{ab})$ Zeit.

Somit ist R eine Reduktion $L_1 \leq_p L_3$ mit Laufzeit $O(n^{ab})$. □

15 Vererbung der NP-Vollständigkeit zeigen

Zeigen Sie: Ist L_0 NP-vollständig, gilt $L_0 \leq_p L_1$ und ist $L_1 \in \text{NP}$, so ist auch L_1 NP-vollständig.
Hinweis: Die Relation $\leq_p$ ist transitiv.

Lösung.

$L_1 \in \text{NP}$ gilt nach Voraussetzung. Zu zeigen bleibt, dass L_1 NP-schwer ist, also $L \leq_p L_1$ für jedes $L \in \text{NP}$.

Sei $L \in \text{NP}$ beliebig.

- Da L_0 NP-vollständig ist, gilt $L \leq_p L_0$.
- Nach Voraussetzung gilt $L_0 \leq_p L_1$.
- Da $\leq_p$ transitiv ist, folgt $L \leq_p L_1$.

L war beliebig, also reduziert jedes NP-Problem auf L_1 – L_1 ist NP-schwer. Zusammen mit $L_1 \in \text{NP}$ ist L_1 NP-vollständig. $\square$

16 L NP-vollständig: $L \in P \Leftrightarrow P = \mathbf{NP}$ zeigen

Sei L eine NP-vollständige Sprache. Beweisen Sie: $L \in P$ genau dann, wenn $P = \mathbf{NP}$.

Lösung.

$\Leftarrow$: Sei $P = \mathbf{NP}$. L ist NP-vollständig, also $L \in \mathbf{NP} = P$.

$\Rightarrow$: Sei $L \in P$. $P \subseteq \mathbf{NP}$ gilt stets. Zu zeigen bleibt $\mathbf{NP} \subseteq P$.

Sei $L' \in \mathbf{NP}$ beliebig. Da L NP-vollständig ist, gibt es eine Reduktion f mit $L' \leq_p L$ und Laufzeit $O(n^a)$. Sei A ein Algorithmus, der L in Zeit $O(n^c)$ entscheidet.

So entscheidet man L' : Berechne $f(x)$ und wende A darauf an. Das ist korrekt, denn $x \in L' \iff f(x) \in L$. Die Ausgabe $f(x)$ hat Größe $O(n^a)$, also braucht A darauf $O((n^a)^c) = O(n^{ac})$ Zeit.

Also gilt $L' \in P$ für jedes $L' \in \mathbf{NP}$, und mit $P \subseteq \mathbf{NP}$ folgt $P = \mathbf{NP}$. $\square$

3 NP-Vollständigkeit

17 NP-Vollständigkeit zeigen für k -Clique

Beweisen Sie, dass k -CLIQUE NP-vollständig ist. *Hinweis:* Reduzieren Sie allgemeines SAT (nicht 3-SAT).

Lösung.

∈ **NP**: Eingabe: Graph $G = (V, E)$, Zahl k und Zertifikat $C \subseteq V$. Der Verifizierer arbeitet wie folgt:

- Falls $|C| < k$, lehne ab.
- Prüfe für jedes Paar $\{u, v\} \subseteq C$, ob $\{u, v\} \in E$; lehne sonst ab.
- Sonst akzeptiere.

Existiert eine k -Clique, so ist sie ein akzeptiertes Zertifikat; sonst verletzt jedes Zertifikat eine Prüfung. Laufzeit: Größe zählen $O(|V|)$, alle Paare testen $O(|V|^2)$ – gesamt $O(|V|^2)$. Also gilt k -CLIQUE ∈ NP.

NP-schwer: Zeige die NP-Schwere durch die Reduktion

$$\text{SAT} \leq_p k\text{-CLIQUE}.$$

SAT ist bereits als NP-vollständig bekannt.

Konstruktion: Sei $F = F_1 \wedge \dots \wedge F_m$ eine KNF-Formel; y_{ij} sei das j -te Literal von Klausel F_i . Wir konstruieren daraus eine k -CLIQUE-Instanz durch:

- $V = \{[i, j] \mid y_{ij} \text{ ist Literal in } F_i\}$
- $E = \{\{[i, j], [i', j']\} \mid i \neq i', y_{ij} \neq \neg y_{i'j'}\}$
- $k = m$

Beweis hin:

- Sei ψ eine erfüllende Belegung von F .
- In jeder Klausel F_i ist ein Literal wahr; wähle je eines, Knoten $[i, j_i]$.
- Sei $C = \{[i, j_i] \mid i \in [m]\}$; die m Knoten stammen aus verschiedenen Klauseln.
- Zwei gewählte Literale sind beide unter ψ wahr, widersprechen sich also nicht.
- Somit sind ihre Knoten durch eine Kante verbunden.
- Also ist C eine m -Clique.

Beweis zurück:

- Sei C eine m -Clique in G .
- Knoten derselben Klausel sind nie verbunden, also enthält C aus jeder Klausel genau einen Knoten.

- Die zugehörigen Literale widersprechen sich paarweise nicht (sonst fehlte eine Kante).
- Setze diese Literale wahr – eine widerspruchsfreie Belegung.
- Sie erfüllt in jeder Klausel ein Literal.
- Also ist F erfüllbar.

Laufzeit: Sei L die Zahl der Literalvorkommen. Knoten anlegen $O(L)$, Kantenpaare prüfen $O(L^2)$ – gesamt $O(L^2)$.

Da k -CLIQUE $\in$ NP und NP-schwer ist, folgt: k -CLIQUE ist NP-vollständig. □

18 NP-Vollständigkeit zeigen für 3-SAT

Beweisen Sie, dass 3-SAT NP-vollständig ist. *Hinweis:* Reduzieren Sie SAT auf 3-SAT.

Lösung.

∈ **NP**: Eingabe: KNF-Formel F mit Klauseln der Länge ≤ 3 und Zertifikat β (eine Belegung der Variablen). Der Verifizierer arbeitet wie folgt:

- Werte jede Klausel unter β aus.
- Lehne ab, falls eine Klausel unerfüllt ist.
- Sonst akzeptiere.

Existiert eine erfüllende Belegung, so ist sie ein akzeptiertes Zertifikat; sonst scheitert jede. Laufzeit: alle Klauseln auswerten $O(|F|)$ – gesamt $O(|F|)$. Also gilt $3\text{-SAT} \in \text{NP}$.

NP-schwer: Zeige die NP-Schwere durch die Reduktion

$$\text{SAT} \leq_p 3\text{-SAT}.$$

SAT ist bereits als NP-vollständig bekannt.

Konstruktion: Ersetze jede Klausel $(y_1 \vee \dots \vee y_\ell)$ mit $\ell > 3$ durch die Kette

$$(y_1 \vee y_2 \vee x_1) \wedge (\neg x_1 \vee y_3 \vee x_2) \wedge \dots \wedge (\neg x_{\ell-3} \vee y_{\ell-1} \vee y_\ell)$$

mit neuen Hilfsvariablen $x_1, \dots, x_{\ell-3}$ (pro Klausel eigene). Klauseln mit $\ell \leq 3$ bleiben unverändert.

Beweis hin:

- Sei die Originalklausel erfüllt, etwa y_i wahr.
- Setze $x_1 = \dots = x_{i-2} = \text{wahr}$ und $x_{i-1} = \dots = x_{\ell-3} = \text{falsch}$.
- Die Kettenklauseln links von y_i sind durch ihr positives x -Literal erfüllt.
- Die Klausel mit y_i ist durch y_i erfüllt.
- Die Kettenklauseln rechts sind durch ihr Literal $\neg x$ erfüllt.
- Also ist die ganze Kette erfüllt.

Beweis zurück:

- Sei die Kette erfüllt, aber angenommen, alle y_i seien falsch.
- Dann erzwingt die erste Klausel $x_1 = \text{wahr}$, die zweite $x_2 = \text{wahr}$, induktiv $x_{\ell-3} = \text{wahr}$.
- Dann ist die letzte Klausel $(\neg x_{\ell-3} \vee y_{\ell-1} \vee y_\ell)$ falsch – Widerspruch.
- Also ist ein y_i wahr und die Originalklausel erfüllt.

Laufzeit: Jede Klausel der Länge ℓ wird zu $\ell - 2$ Klauseln – gesamt $O(|F|)$.

Da $3\text{-SAT} \in \text{NP}$ und NP-schwer ist, folgt: 3-SAT ist NP-vollständig. □

19 NP-Vollständigkeit zeigen für VertexCover

Beweisen Sie, dass VERTEXCOVER NP-vollständig ist. *Hinweis:* Reduzieren Sie CLIQUE.

Lösung.

∈ **NP**: Eingabe: Graph $G = (V, E)$, Zahl k und Zertifikat $C \subseteq V$. Der Verifizierer arbeitet wie folgt:

- Falls $|C| > k$, lehne ab.
- Prüfe für jede Kante $\{u, v\} \in E$, ob $u \in C$ oder $v \in C$; lehne sonst ab.
- Sonst akzeptiere.

Existiert ein Vertex Cover der Größe $\leq k$, so ist es ein akzeptiertes Zertifikat; sonst bleibt eine Kante ungedeckt. Laufzeit: Größe zählen $O(|V|)$, Kanten prüfen $O(|E|)$ – gesamt $O(|V| + |E|)$. Also gilt VERTEXCOVER ∈ NP.

NP-schwer: Zeige die NP-Schwere durch die Reduktion

$$\text{CLIQUE} \leq_p \text{VERTEXCOVER}.$$

CLIQUE ist bereits als NP-vollständig bekannt.

Konstruktion: Sei $(G = (V, E), k)$ eine CLIQUE-Instanz mit $n = |V|$. Wir konstruieren daraus eine VERTEXCOVER-Instanz $(G' = (V, E'), k')$ durch:

- $E' = \binom{V}{2} \setminus E$
- $k' = n - k$

Damit ist G' der Komplementgraph von G .

Beweis hin:

- Sei C eine Clique in G mit $|C| \geq k$.
- Setze $W = V \setminus C$; dann gilt $|W| \leq n - k = k'$.
- Alle Kanten innerhalb C liegen in E , also enthält E' keine Kante mit beiden Endpunkten in C .
- Somit hat jede Kante von G' einen Endpunkt in W .
- Also ist W ein Vertex Cover von G' mit $|W| \leq k'$.

Beweis zurück:

- Sei W ein Vertex Cover von G' mit $|W| \leq k'$.
- Setze $C = V \setminus W$; dann gilt $|C| \geq n - k' = k$.
- Für $\{u, v\} \subseteq C$ gilt $u, v \notin W$, also $\{u, v\} \notin E'$.
- Da E' genau die Nicht-Kanten von G enthält, gilt $\{u, v\} \in E$.
- Also ist C eine Clique in G mit $|C| \geq k$.

Laufzeit: Alle Knotenpaare durchgehen und die Kanten invertieren – $O(|V|^2)$.

Da VERTEXCOVER ∈ NP und NP-schwer ist, folgt: VERTEXCOVER ist NP-vollständig. □

20 NP-Vollständigkeit zeigen für CliqueAndIndependentSet

Problem CliqueAndIndependentSet: Gegeben ein Graph $G = (V, E)$ und k . Enthält G zugleich eine Clique der Größe k und ein Independent Set der Größe k (eine Knotenmenge ohne Kanten untereinander)? Beweisen Sie, dass CLIQUEANDINDEPENDENTSET NP-vollständig ist. *Hinweis:* Reduzieren Sie CLIQUE.

Lösung.

∈ **NP**: Eingabe: Graph $G = (V, E)$, Zahl k und Zertifikat (C, I) mit $C, I \subseteq V$. Der Verifizierer arbeitet wie folgt:

- Falls $|C| < k$ oder $|I| < k$, lehne ab.
- Prüfe für jedes Paar $\{x, y\} \subseteq C$, ob $\{x, y\} \in E$; lehne sonst ab.
- Prüfe für jedes Paar $\{x, y\} \subseteq I$, ob $\{x, y\} \notin E$; lehne sonst ab.
- Sonst akzeptiere.

Existiert beides, so ist (C, I) ein akzeptiertes Zertifikat; sonst scheitert eine Prüfung. Laufzeit: alle Paare in C und I testen – $O(|V|^2)$. Also gilt CLIQUEANDINDEPENDENTSET ∈ NP.

NP-schwer: Zeige die NP-Schwere durch die Reduktion

$$\text{CLIQUE} \leq_p \text{CLIQUEANDINDEPENDENTSET}.$$

CLIQUE ist bereits als NP-vollständig bekannt.

Konstruktion: Sei $(G = (V, E), k)$ eine CLIQUE-Instanz, o.B.d.A. $k \geq 2$. Wir konstruieren daraus eine CLIQUEANDINDEPENDENTSET-Instanz $(G' = (V', E'), k)$ durch:

- $V' = V \cup \{u_1, \dots, u_k\}$ mit k neuen isolierten Knoten
- $E' = E$
- k bleibt unverändert

Beweis hin:

- Sei C eine k -Clique in G .
- Da keine Kanten entfernt wurden, ist C auch eine k -Clique in G' .
- Die Knoten $u_1, \dots, u_k$ sind isoliert, also paarweise nicht adjazent.
- Somit ist $\{u_1, \dots, u_k\}$ ein Independent Set der Größe k .
- Also enthält G' eine k -Clique und ein Independent Set der Größe k .

Beweis zurück:

- Sei (G', k) eine Ja-Instanz; dann enthält G' eine k -Clique C' .
- Wegen $k \geq 2$ hat jeder Knoten von C' einen Nachbarn in C' .

- Die $u_1, \dots, u_k$ sind isoliert, liegen also nicht in C' .
- Somit gilt $C' \subseteq V$.
- Also ist C' eine k -Clique in G .

Laufzeit: k isolierte Knoten anhängen – $O(k) \subseteq O(|V|)$.

Da $\text{CLIQUEANDINDEPENDENTSET} \in \mathbf{NP}$ und NP-schwer ist, folgt: $\text{CLIQUEANDINDEPENDENTSET}$ ist NP-vollständig. $\square$

21 NP-Vollständigkeit zeigen für k -CliqueUniversal

Problem k -CliqueUniversal: Gegeben ein Graph $G = (V, E)$ und k , wobei ein Knoten $u \in V$ mit allen anderen verbunden ist. Enthält G eine Clique der Größe $\geq k$? Beweisen Sie, dass k -CLIQUEUNIVERSAL NP-vollständig ist. *Hinweis:* Reduzieren Sie CLIQUE.

Lösung.

∈ **NP:** Eingabe: Graph $G = (V, E)$, Zahl k und Zertifikat $C \subseteq V$. Der Verifizierer arbeitet wie folgt:

- Falls $|C| < k$, lehne ab.
- Prüfe für jedes Paar $\{x, y\} \subseteq C$, ob $\{x, y\} \in E$; lehne sonst ab.
- Sonst akzeptiere.

Existiert eine k -Clique, so ist sie ein akzeptiertes Zertifikat; sonst verletzt jedes Zertifikat eine Prüfung. Laufzeit: Größe zählen $O(|V|)$, alle Paare testen $O(|V|^2)$ – gesamt $O(|V|^2)$. Also gilt k -CLIQUEUNIVERSAL ∈ NP.

NP-schwer: Zeige die NP-Schwere durch die Reduktion

$$\text{CLIQUE} \leq_p k\text{-CLIQUEUNIVERSAL}.$$

CLIQUE ist bereits als NP-vollständig bekannt.

Konstruktion: Sei $(G = (V, E), k)$ eine CLIQUE-Instanz. Wir konstruieren daraus eine k -CLIQUEUNIVERSAL-Instanz $(\bar{G} = (\bar{V}, \bar{E}), \bar{k})$ durch:

- $\bar{V} = V \cup \{u\}$ mit einem neuen Knoten u
- $\bar{E} = E \cup \{\{u, v\} \mid v \in V\}$
- $\bar{k} = k + 1$

Der Knoten u ist mit allen anderen verbunden, also ist $(\bar{G}, \bar{k})$ eine gültige Instanz.

Beweis hin:

- Sei C eine Clique in G mit $|C| \geq k$.
- u ist mit allen Knoten von C verbunden.
- Somit ist $C \cup \{u\}$ eine Clique in $\bar{G}$.
- Also gilt $|C \cup \{u\}| \geq k + 1 = \bar{k}$.

Beweis zurück:

- Sei $\bar{C}$ eine Clique in $\bar{G}$ mit $|\bar{C}| \geq \bar{k}$.
- Setze $C = \bar{C} \setminus \{u\}$; dann gilt $|C| \geq \bar{k} - 1 = k$.
- Alle Kanten zwischen Knoten von C liegen in E , denn neu sind nur Kanten an u .
- Also ist C eine Clique in G mit $|C| \geq k$.

Laufzeit: Einen Knoten und $|V|$ Kanten anlegen – $O(|V|)$.

Da k -CLIQUEUNIVERSAL ∈ NP und NP-schwer ist, folgt: k -CLIQUEUNIVERSAL ist NP-vollständig. $\square$

22 NP-Vollständigkeit zeigen für Clique-Nomember

Problem Clique-Nomember: Gegeben ein Graph $G = (V, E)$, ein Knoten $v \in V$ und k . Gibt es eine k -Clique, die v *nicht* enthält? Beweisen Sie, dass CLIQUE-NOMEMBER NP-vollständig ist. *Hinweis:* Reduzieren Sie CLIQUE.

Lösung.

∈ **NP:** Eingabe: Graph $G = (V, E)$, Knoten v , Zahl k und Zertifikat $C \subseteq V$. Der Verifizierer arbeitet wie folgt:

- Falls $|C| < k$ oder $v \in C$, lehne ab.
- Prüfe für jedes Paar $\{x, y\} \subseteq C$, ob $\{x, y\} \in E$; lehne sonst ab.
- Sonst akzeptiere.

Existiert eine k -Clique ohne v , so ist sie ein akzeptiertes Zertifikat; sonst scheitert eine Prüfung. Laufzeit: Größe und Ausschluss prüfen $O(|V|)$, alle Paare testen $O(|V|^2)$ – gesamt $O(|V|^2)$. Also gilt CLIQUE-NOMEMBER ∈ NP.

NP-schwer: Zeige die NP-Schwere durch die Reduktion

$$\text{CLIQUE} \leq_p \text{CLIQUE-NOMEMBER}.$$

CLIQUE ist bereits als NP-vollständig bekannt.

Konstruktion: Sei $(G = (V, E), k)$ eine CLIQUE-Instanz. Wir konstruieren daraus eine CLIQUE-NOMEMBER-Instanz $(G' = (V', E'), v, k)$ durch:

- $V' = V \cup \{v\}$ mit einem neuen, isolierten Knoten $v \notin V$
- $E' = E$
- k bleibt unverändert

Beweis hin:

- Sei C eine k -Clique in G .
- Da keine Kanten entfernt wurden, ist C auch eine k -Clique in G' .
- Da $v \notin V$, gilt $v \notin C$.
- Also ist C eine k -Clique in G' , die v nicht enthält.

Beweis zurück:

- Sei C eine k -Clique in G' mit $v \notin C$.
- Dann gilt $C \subseteq V$, denn v ist der einzige neue Knoten.
- Alle Kanten zwischen Knoten von C liegen in E .
- Also ist C eine k -Clique in G .

Laufzeit: Graph kopieren und einen isolierten Knoten anhängen – $O(|V| + |E|)$.

Da CLIQUE-NOMEMBER ∈ NP und NP-schwer ist, folgt: CLIQUE-NOMEMBER ist NP-vollständig. $\square$

23 NP-Vollständigkeit zeigen für k -Color

Problem k -Color: Gegeben ein Graph $G = (V, E)$ und $k \in \mathbb{N}$. Gibt es eine Färbung $f: V \rightarrow \{1, \dots, k\}$ mit $f(u) \neq f(v)$ für alle $\{u, v\} \in E$? Beweisen Sie, dass k -COLOR NP-vollständig ist. *Hinweis:* Reduzieren Sie 3-SAT; o.B.d.A. enthält keine Klausel ein Paar $x, \neg x$.

Lösung.

∈ **NP**: Eingabe: Graph G , Zahl k und Zertifikat $f: V \rightarrow \{1, \dots, k\}$. Der Verifizierer arbeitet wie folgt:

- Prüfe für jede Kante $\{u, v\} \in E$, ob $f(u) \neq f(v)$ gilt; lehne sonst ab.
- Sonst akzeptiere.

Existiert eine k -Färbung, wird ihr Zertifikat akzeptiert; sonst scheitert jedes an einer Kante. Kanten prüfen – $O(|V| + |E|)$. Also gilt k -COLOR ∈ NP.

NP-schwer: Zeige die NP-Schwere durch die Reduktion

$$3\text{-SAT} \leq_p k\text{-COLOR}.$$

3-SAT ist bereits als NP-vollständig bekannt.

Konstruktion: Sei $F = F_1 \wedge \dots \wedge F_m$ über $x_1, \dots, x_n$. Wir konstruieren G und $k = n + 1$ durch:

- Knoten: $x_i, \bar{x}_i, v_i$ für $i \in [n]$; F_j für $j \in [m]$; z
- $\{v_i, v_j\}$ für $i \neq j$ und $\{v_i, z\}$ – die v_i und z bilden eine $(n+1)$ -Clique
- $\{v_i, x_j\}$ und $\{v_i, \bar{x}_j\}$ für $i \neq j$
- $\{x_i, \bar{x}_i\}$ für alle i
- $\{F_j, y\}$ für jedes Literal y , das *nicht* in F_j vorkommt
- $\{F_j, z\}$ für alle j

Vorüberlegung (für jede $(n+1)$ -Färbung f):

- $\{v_1, \dots, v_n, z\}$ ist eine $(n+1)$ -Clique. O.B.d.A. gilt $f(v_i) = i$ und $f(z) = n + 1$.
- x_j und $\bar{x}_j$ sind mit allen v_i , $i \neq j$, verbunden. Somit $f(x_j), f(\bar{x}_j) \in \{j, n + 1\}$.
- Wegen der Kante $\{x_j, \bar{x}_j\}$ trägt genau einer die Farbe j , der andere $n + 1$.

Beweis 3-SAT nach k -Color:

- Sei ψ eine erfüllende Belegung.
- Färbe je Paar das wahre Literal mit i , das falsche mit $n + 1$; dazu $f(v_i) = i$, $f(z) = n + 1$.
- Jede Klausel F_j enthält ein wahres Literal y aus Paar i . y hat Farbe i , und $\{F_j, y\} \notin E$.

- Färbe $f(F_j) = i$. Kein Nachbar von F_j trägt Farbe i : Das andere Literal des Paares i hat Farbe $n + 1$, und v_i ist nicht mit F_j verbunden.
- Also ist G mit $n + 1$ Farben färbbar.

Beweis k -Color nach 3-SAT:

- Sei f eine $(n+1)$ -Färbung, o.B.d.A. wie in der Vorüberlegung.
- Setze $\psi(x_j) = \text{wahr}$ genau dann, wenn $f(x_j) = j$.
- Angenommen, eine Klausel F_j wäre unter ψ falsch. Dann tragen alle ihre Literale die Farbe $n + 1$.
- Für jedes i liegt dann der Farbe- i -Knoten des Paares i nicht in F_j – also ist er mit F_j verbunden.
- F_j sieht so alle Farben $1, \dots, n$ und über z die Farbe $n + 1$. Keine Farbe bleibt frei – Widerspruch.
- Also erfüllt ψ jede Klausel.

Laufzeit: Knoten und Kanten anlegen – $O(n^2 + nm)$.

Da k -COLOR $\in \text{NP}$ und NP-schwer ist, folgt: k -COLOR ist NP-vollständig. □

24 NP-Vollständigkeit zeigen für k -COLOR-PRECOLORING

Problem k -COLOR-PRECOLORING: Gegeben ein Graph $G = (V, E)$, eine Zahl k und paarweise verschiedene Knoten $v_1, \dots, v_k \in V$. Gibt es eine Färbung $f : V \rightarrow [k]$ mit $f(u) \neq f(w)$ für alle $\{u, w\} \in E$ und $f(v_i) = i$ für alle i ? Beweisen Sie, dass k -COLOR-PRECOLORING NP-vollständig ist. *Hinweis:* Reduzieren Sie k -COLOR.

Lösung.

∈ **NP**: Eingabe: Graph $G = (V, E)$, Zahl k , Knoten $v_1, \dots, v_k$ und Zertifikat $f : V \rightarrow [k]$. Der Verifizierer arbeitet wie folgt:

- Prüfe für jedes $i \in [k]$, ob $f(v_i) = i$; lehne sonst ab.
- Prüfe für jede Kante $\{u, w\} \in E$, ob $f(u) \neq f(w)$; lehne sonst ab.
- Sonst akzeptiere.

Existiert eine gültige Färbung mit Vorgabe, so ist sie ein akzeptiertes Zertifikat; sonst scheitert eine Prüfung. Laufzeit: Vorgaben prüfen $O(k)$, Kanten prüfen $O(|E|)$ – gesamt $O(k + |E|)$. Also gilt k -COLOR-PRECOLORING ∈ NP.

NP-schwer: Zeige die NP-Schwere durch die Reduktion

$$k\text{-COLOR} \leq_p k\text{-COLOR-PRECOLORING}.$$

k -COLOR ist bereits als NP-vollständig bekannt.

Konstruktion: Sei $(G = (V, E), k)$ eine k -COLOR-Instanz. Wir konstruieren daraus eine k -COLOR-PRECOLORING-Instanz $(G' = (V', E'), k, v_1, \dots, v_k)$ durch:

- $V' = V \cup \{v_1, \dots, v_k\}$ mit k neuen Knoten
- $E' = E \cup \{\{v_i, v_j\} \mid i \neq j\}$; die v_i bilden ein K_k , ohne Kante nach V
- Vorgabe $f(v_i) = i$

Beweis hin:

- Sei f eine k -Färbung von G .
- Erweitere f durch $f(v_i) = i$ für alle $i \in [k]$.
- Die v_i sind nur untereinander verbunden und tragen paarweise verschiedene Farben.
- Somit ist f eine gültige Färbung von G' und erfüllt die Vorgabe.

Beweis zurück:

- Sei f eine gültige Färbung von G' mit $f(v_i) = i$.
- Alle Kanten von G liegen in G' .
- Also ist die Einschränkung von f auf V eine k -Färbung von G .

Laufzeit: Das K_k anlegen – $O(k^2)$.

Da k -COLOR-PRECOLORING ∈ NP und NP-schwer ist, folgt: Das Problem ist NP-vollständig. $\square$

25 NP-Vollständigkeit zeigen für HamiltonianCycle

Beweisen Sie, dass HAMILTONIANCYCLE NP-vollständig ist. *Hinweis:* Reduzieren Sie HAMILTONIANPATH.

Lösung.

∈ **NP**: Eingabe: Graph $G = (V, E)$ mit $n = |V|$ und Zertifikat: eine Folge $(w_1, \dots, w_n)$ von Knoten. Der Verifizierer arbeitet wie folgt:

- Prüfe, ob $(w_1, \dots, w_n)$ jeden Knoten aus V genau einmal enthält; lehne sonst ab.
- Prüfe für $i = 1, \dots, n - 1$, ob $\{w_i, w_{i+1}\} \in E$, und ob $\{w_n, w_1\} \in E$; lehne sonst ab.
- Sonst akzeptiere.

Existiert ein Hamiltonkreis, so ist seine Knotenfolge ein akzeptiertes Zertifikat; sonst scheitert eine Prüfung. Laufzeit: Permutation prüfen $O(|V|)$, die $|V|$ Kreiskanten nachschlagen $O(|V|)$ – gesamt $O(|V|)$. Also gilt HAMILTONIANCYCLE ∈ **NP**.

NP-schwer: Zeige die NP-Schwere durch die Reduktion

$$\text{HAMILTONIANPATH} \leq_p \text{HAMILTONIANCYCLE}.$$

HAMILTONIANPATH ist bereits als NP-vollständig bekannt.

Konstruktion: Sei $G = (V, E)$ eine HAMILTONIANPATH-Instanz. Wir konstruieren daraus eine HAMILTONIANCYCLE-Instanz $G' = (V', E')$ durch:

- $V' = V \cup \{u\}$ mit einem neuen Knoten u
- $E' = E \cup \{\{u, v\} \mid v \in V\}$

So wird u ein universeller Knoten.

Beweis hin:

- Sei $v_1, \dots, v_n$ ein Hamiltonpfad in G .
- Da u universell ist, existieren $\{v_n, u\}$ und $\{u, v_1\}$ in G' .
- Also ist $v_1, \dots, v_n, u, v_1$ ein Hamiltonkreis in G' .

Beweis zurück:

- Sei K ein Hamiltonkreis in G' .
- K besucht u genau einmal, zwischen zwei Nachbarn $v_i, v_j \in V$.
- Entferne u aus K ; es bleibt ein Pfad von v_i nach v_j durch alle Knoten von V .
- Dieser Pfad benutzt nur Kanten aus E , denn neu sind nur die Kanten an u .
- Also ist er ein Hamiltonpfad in G .

Laufzeit: Einen Knoten und $|V|$ Kanten anlegen – $O(|V|)$.

Da HAMILTONIANCYCLE ∈ **NP** und NP-schwer ist, folgt: HAMILTONIANCYCLE ist NP-vollständig. $\square$

26 NP-Vollständigkeit zeigen für FeedbackVertexSet

Beweisen Sie, dass `FEEDBACKVERTEXSET` NP-vollständig ist: Gegeben ein *gerichteter* Graph $G = (V, E)$ und k , gibt es $X \subseteq V$ mit $|X| \leq k$, sodass $G \setminus X$ kreisfrei ist? *Hinweis:* Reduzieren Sie `VERTEXCOVER`.

Lösung.

∈ **NP**: Eingabe: gerichteter Graph $G = (V, E)$, Zahl k und Zertifikat $X \subseteq V$. Der Verifizierer arbeitet wie folgt:

- Falls $|X| > k$, lehne ab.
- Prüfe per topologischer Sortierung, ob $G \setminus X$ kreisfrei ist; lehne bei einem Kreis ab.
- Sonst akzeptiere.

Existiert ein Feedback Vertex Set der Größe $\leq k$, so ist es ein akzeptiertes Zertifikat; sonst scheitert eine Prüfung. Laufzeit: Größe zählen und topologisch sortieren – $O(|V| + |E|)$. Also gilt `FEEDBACKVERTEXSET` ∈ **NP**.

NP-schwer: Zeige die NP-Schwere durch die Reduktion

$$\text{VERTEXCOVER} \leq_p \text{FEEDBACKVERTEXSET}.$$

`VERTEXCOVER` ist bereits als NP-vollständig bekannt.

Konstruktion: Sei $(G = (V, E), k)$ eine `VERTEXCOVER`-Instanz. Wir konstruieren daraus eine `FEEDBACKVERTEXSET`-Instanz $(G' = (V', E'), k')$ durch:

- $V' = V$
- $E' = \{(u, v), (v, u) \mid \{u, v\} \in E\}$
- $k' = k$

So wird jede Kante zu zwei antiparallelen Bögen, also einem 2-Kreis.

Beweis hin:

- Sei C ein Vertex Cover von G mit $|C| \leq k$.
- Dann ist $V \setminus C$ unabhängig in G .
- Also gibt es in G' keinen Bogen zwischen zwei Knoten aus $V \setminus C$.
- Somit ist $G' \setminus C$ bogenlos, insbesondere kreisfrei.
- Also ist C ein Feedback Vertex Set mit $|C| \leq k' = k$.

Beweis zurück:

- Sei X ein Feedback Vertex Set von G' mit $|X| \leq k'$.
- Für jede Kante $\{u, v\} \in E$ bilden $(u, v), (v, u)$ einen 2-Kreis in G' .

- X zerstört jeden Kreis, trifft also diesen 2-Kreis: $u \in X$ oder $v \in X$.
- Somit deckt X jede Kante von G .
- Also ist X ein Vertex Cover von G mit $|X| \leq k$.

Laufzeit: Pro Kante zwei Bögen anlegen – $O(|E|)$.

Da `FEEDBACKVERTEXSET` $\in$ NP und NP-schwer ist, folgt: `FEEDBACKVERTEXSET` ist NP-vollständig. $\square$

27 NP-Vollständigkeit zeigen für Δ -Cover

Problem Δ -Cover: Gegeben ein Graph $G = (V, E)$ und k . Gibt es $C_\Delta \subseteq V$ mit $|C_\Delta| \leq k$, sodass jedes Dreieck (3-Clique) von G mindestens einen Knoten in C_Δ hat? Beweisen Sie, dass Δ -COVER NP-vollständig ist. *Hinweis:* Reduzieren Sie VERTEXCOVER.

Lösung.

∈ **NP:** Eingabe: Graph $G = (V, E)$, Zahl k und Zertifikat $C_\Delta \subseteq V$. Der Verifizierer arbeitet wie folgt:

- Falls $|C_\Delta| > k$, lehne ab.
- Prüfe für jedes Tripel $\{a, b, c\} \subseteq V$ mit $\{a, b\}, \{b, c\}, \{a, c\} \in E$, ob es einen Knoten in C_Δ hat; lehne sonst ab.
- Sonst akzeptiere.

Existiert eine Dreiecksüberdeckung der Größe $\leq k$, so ist sie ein akzeptiertes Zertifikat; sonst bleibt ein Dreieck ungedeckt. Laufzeit: Größe zählen $O(|V|)$, alle Tripel prüfen $O(|V|^3)$ – gesamt $O(|V|^3)$. Also gilt Δ -COVER ∈ NP.

NP-schwer: Zeige die NP-Schwere durch die Reduktion

$$\text{VERTEXCOVER} \leq_p \Delta\text{-COVER}.$$

VERTEXCOVER ist bereits als NP-vollständig bekannt.

Konstruktion: Sei $(G = (V, E), k)$ eine VERTEXCOVER-Instanz. Wir konstruieren daraus eine Δ -COVER-Instanz $(G' = (V', E'), k')$ durch:

- $V' = V \cup \{w_e \mid e \in E\}$ mit je einem neuen Knoten w_e pro Kante
- $E' = E \cup \{\{w_e, u\}, \{w_e, v\} \mid e = \{u, v\} \in E\}$
- $k' = k$

So wird jede Kante $e = \{u, v\}$ zum Dreieck $\{u, v, w_e\}$.

Beweis hin:

- Sei C ein Vertex Cover von G mit $|C| \leq k$.
- Jeder neue Knoten w_e hat genau die Nachbarn u, v ; jedes Dreieck von G' ist daher ein Gadget-Dreieck $\{u, v, w_e\}$ oder ein Dreieck $\{a, b, c\} \subseteq V$ aus G .
- Beim Gadget-Dreieck deckt C die Kante $\{u, v\}$, enthält also u oder v .
- Beim geerbten Dreieck deckt C die Kante $\{a, b\}$, enthält also einen Eckknoten.
- Somit trifft C jedes Dreieck von G' .
- Also ist C eine Dreiecksüberdeckung mit $|C| \leq k'$.

Beweis zurück:

- Sei C_Δ eine Dreiecksüberdeckung von G' mit $|C_\Delta| \leq k'$.
- Enthält C_Δ ein w_e mit $e = \{u, v\}$, ersetze es durch u ; da w_e nur im Dreieck $\{u, v, w_e\}$ liegt und dieses auch u enthält, bleibt es eine Überdeckung und wird nicht größer.
- So entsteht $C \subseteq V$ mit $|C| \leq k$, das jedes Dreieck von G' trifft.
- Jede Kante $\{u, v\} \in E$ bildet mit w_e das Dreieck $\{u, v, w_e\}$; C trifft es, und da $w_e \notin C$, gilt $u \in C$ oder $v \in C$.
- Also ist C ein Vertex Cover von G mit $|C| \leq k$.

Laufzeit: Pro Kante einen Knoten und zwei Kanten anlegen – $O(|E|)$.

Da Δ -COVER $\in$ NP und NP-schwer ist, folgt: Δ -COVER ist NP-vollständig. □

28 NP-Vollständigkeit zeigen für 3-COLOR mit Minimalgrad 3

Problem 3-COLOR-Grad-3: 3-COLOR, eingeschränkt auf Graphen $G = (V, E)$ mit $\deg(v) \geq 3$ für alle $v \in V$. Ist G mit 3 Farben färbbar? Beweisen Sie, dass diese Variante NP-vollständig ist. *Hinweis:* Reduzieren Sie 3-COLOR.

Lösung.

∈ **NP**: Eingabe: Graph $G = (V, E)$ und Zertifikat $f : V \rightarrow \{1, 2, 3\}$. Der Verifizierer arbeitet wie folgt:

- Prüfe für jeden Knoten v , ob $\deg(v) \geq 3$; lehne sonst ab.
- Prüfe für jede Kante $\{u, v\} \in E$, ob $f(u) \neq f(v)$; lehne sonst ab.
- Sonst akzeptiere.

Existiert eine 3-Färbung der gültigen Instanz, so ist sie ein akzeptiertes Zertifikat; sonst hat eine Kante gleich gefärbte Endpunkte. Laufzeit: Gradprüfung $O(|V| + |E|)$, Kantenprüfung $O(|E|)$ – gesamt $O(|V| + |E|)$. Also gilt die Grad-3-Variante ∈ **NP**.

NP-schwer: Zeige die NP-Schwere durch die Reduktion

$$3\text{-COLOR} \leq_p 3\text{-COLOR-GRAD-3}.$$

3-COLOR ist bereits als NP-vollständig bekannt.

Konstruktion: Benutze das Diamant-Gadget D auf Knoten p, q, r, s mit den Kanten pq, pr, qr, qs, rs (ein K_4 ohne die Kante $\{p, s\}$; insbesondere sind p, s nicht benachbart). Sei $G = (V, E)$ eine 3-COLOR-Instanz. Wir konstruieren G' durch:

- Hänge an jeden Knoten $v \in V$ mit $\deg(v) < 3$ zwei eigene Kopien von D .
- Verbinde v mit jeder Kopie über die Kanten $\{v, p\}$ und $\{v, s\}$.

Dann steigt $\deg(v)$ um 4, und alle Gadget-Knoten haben Grad 3 (q, r über drei Gadget-Kanten, p, s über zwei plus die Kante zu v); somit hat G' Minimalgrad 3.

Beweis hin:

- Sei f eine 3-Färbung von G .
- Erweitere f auf jedes an v angehängte Gadget: Wähle eine Farbe $c \neq f(v)$ und setze $p = s = c$ (erlaubt, da p, s nicht benachbart).
- q und r sind untereinander sowie zu p und s benachbart; da p, s dieselbe Farbe c tragen, bleiben für die Kante qr die beiden anderen Farben – färbe q, r damit.
- Die Kanten $\{v, p\}, \{v, s\}$ sind gültig, da $c \neq f(v)$.
- Also ist G' mit 3 Farben färbbar.

Beweis zurück:

- Sei f' eine 3-Färbung von G' .

- Die Gadgets lassen die Originalkanten unverändert, also $E \subseteq E'$.
- Somit ist die Einschränkung von f' auf V eine 3-Färbung von G .

Laufzeit: Pro untergradigem Knoten zwei Gadgets konstanter Größe anlegen – $O(|V|)$.

Da die Grad-3-Variante $\in \text{NP}$ und NP-schwer ist, folgt: 3-COLOR mit Minimalgrad 3 ist NP-vollständig. $\square$

29 NP-Vollständigkeit zeigen für HamiltonianPath

Problem HamiltonianPath: Gegeben ein ungerichteter Graph $G = (V, E)$. Entscheide: Gibt es einen Pfad, der jeden Knoten genau einmal besucht? Zeigen Sie, dass HAMILTONIANPATH NP-vollständig ist. *Hinweis: Reduzieren Sie HAMILTONIANCYCLE.*

Lösung.

∈ **NP**: Eingabe: Graph $G = (V, E)$ und Zertifikat: eine Knotenfolge $(v_1, \dots, v_n)$. Der Verifizierer arbeitet wie folgt:

- Prüfe, ob $(v_1, \dots, v_n)$ jeden Knoten aus V genau einmal enthält; lehne sonst ab.
- Prüfe für jedes $i < n$, ob $\{v_i, v_{i+1}\} \in E$ gilt; lehne sonst ab.
- Sonst akzeptiere.

Korrektheit: Hat G einen Hamiltonpfad, dann ist seine Knotenfolge ein akzeptiertes Zertifikat. Hat G keinen, dann scheitert jedes Zertifikat an einer Prüfung – abgelehnt.

Laufzeit: Knoten auf Vollständigkeit zählen $O(|V|)$, Kantenprüfungen $O(|V|)$ – gesamt $O(|V|)$. Also gilt HAMILTONIANPATH ∈ NP.

NP-schwer: Zeige die NP-Schwere durch die Reduktion

$$\text{HAMILTONIANCYCLE} \leq_p \text{HAMILTONIANPATH}.$$

HAMILTONIANCYCLE ist bereits als NP-vollständig bekannt.

Konstruktion: Sei $G = (V, E)$ gegeben, o.B.d.A. $|V| \geq 3$. Wähle einen festen Knoten $v \in V$. Konstruiere $G' = (V', E')$ durch:

- $V' = V \cup \{v^*, s, t\}$
- $E' = E \cup \{\{v^*, w\} \mid \{v, w\} \in E\} \cup \{\{s, v\}, \{t, v^*\}\}$

Der Zwilling v^* erbt die Nachbarn von v ; die Pendants s, t haben Grad 1.

Beweis HamiltonianCycle nach HamiltonianPath:

- Sei $v, w_1, \dots, w_{n-1}, v$ ein Hamiltonkreis in G .
- Da v^* dieselben Nachbarn wie v hat, existiert die Kante $\{w_{n-1}, v^*\}$.
- Also ist $s, v, w_1, \dots, w_{n-1}, v^*, t$ ein Hamiltonpfad in G' .

Beweis HamiltonianPath nach HamiltonianCycle:

- Sei P ein Hamiltonpfad in G' .
- s und t haben Grad 1, sind also die beiden Endpunkte von P .
- Da s nur an v und t nur an v^* hängt, hat P die Form $s, v, x_1, \dots, x_{n-1}, v^*, t$.
- Dann sind $x_1, \dots, x_{n-1}$ alle Knoten aus $V \setminus \{v\}$.
- Da $\{x_{n-1}, v^*\} \in E'$ und v^* dieselben Nachbarn wie v hat, ist x_{n-1} auch Nachbar von v .

- Ersetze v^* durch v : Es entsteht der Kreis $v, x_1, \dots, x_{n-1}, v$, der jeden Knoten von G genau einmal besucht.
- Also ist er ein Hamiltonkreis in G .

Laufzeit: Zwillung mit bis zu $|V|$ Nachbarn und zwei Pendants anlegen – $O(|V|)$.

Da HAMILTONIANCYCLE NP-schwer ist, $\text{HAMILTONIANCYCLE} \leq_p \text{HAMILTONIANPATH}$ gilt und $\text{HAMILTONIANPATH} \in \text{NP}$, ist HAMILTONIANPATH NP-vollständig. $\square$

30 NP-Vollständigkeit zeigen für Hitchhiker's-HamiltonianCycle

Problem Hitchhiker's-HamiltonianCycle: Gegeben ein ungerichteter Graph $G = (V, E)$ mit $\deg(v) \geq 42$ für alle $v \in V$. Entscheide: Gibt es einen Hamiltonkreis? Zeigen Sie, dass das Problem NP-vollständig ist. *Hinweis: Reduzieren Sie HAMILTONIANCYCLE.*

Lösung.

∈ **NP**: Eingabe: Graph $G = (V, E)$ und Zertifikat: eine Knotenfolge $(v_1, \dots, v_n)$. Der Verifizierer arbeitet wie folgt:

- Prüfe für jeden Knoten $v \in V$, ob $\deg(v) \geq 42$ gilt; lehne sonst ab.
- Prüfe, ob $(v_1, \dots, v_n)$ jeden Knoten aus V genau einmal enthält; lehne sonst ab.
- Prüfe für jedes $i < n$, ob $\{v_i, v_{i+1}\} \in E$, und ob $\{v_n, v_1\} \in E$ gilt; lehne sonst ab.
- Sonst akzeptiere.

Korrektheit: Ist die Instanz gültig und hat G einen Hamiltonkreis, dann ist dessen Knotenfolge ein akzeptiertes Zertifikat; sonst scheitert jedes Zertifikat an einer Prüfung – abgelehnt.

Laufzeit: Gradprüfung $O(|V| + |E|)$, Folge zählen $O(|V|)$, Kantenprüfungen $O(|V|)$ – gesamt $O(|V| + |E|)$. Also gilt HITCHHIKER'S-HAMILTONIANCYCLE ∈ NP.

NP-schwer: Zeige die NP-Schwere durch die Reduktion

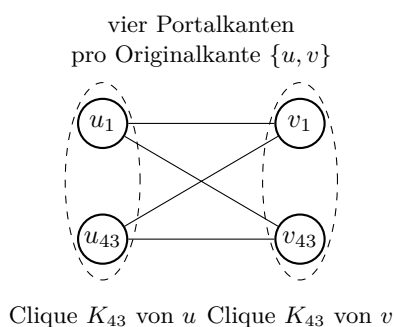
$$\text{HAMILTONIANCYCLE} \leq_p \text{HITCHHIKER'S-HAMILTONIANCYCLE}.$$

HAMILTONIANCYCLE ist bereits als NP-vollständig bekannt.

Konstruktion: Sei $G = (V, E)$ gegeben. Ersetze jeden Knoten v durch eine Clique K_{43} auf $\{v_1, \dots, v_{43}\}$ (Portale sind v_1 und v_{43}). Konstruiere $G' = (V', E')$ durch:

- $V' = \{v_1, \dots, v_{43} \mid v \in V\}$
- $E' = \{\{v_i, v_j\} \mid v \in V, i \neq j\} \cup \{\{u_i, v_j\} \mid \{u, v\} \in E, i, j \in \{1, 43\}\}$

Innere Clique-Knoten haben Grad 42, die Portale Grad ≥ 42 – die Instanz ist gültig.



Beweis HamiltonianCycle nach Hitchhiker's-HamiltonianCycle:

- Sei $v^{(1)}, \dots, v^{(n)}, v^{(1)}$ ein Hamiltonkreis in G .

- Durchlaufe die Clique jedes $v^{(j)}$ auf dem Pfad $v_1^{(j)} \rightarrow \dots \rightarrow v_{43}^{(j)}$; dieser Pfad existiert in jeder Clique.
- Wechsle über die Portalkante $\{v_{43}^{(j)}, v_1^{(j+1)}\}$ zur nächsten Clique.
- Diese Kante existiert, denn $\{v^{(j)}, v^{(j+1)}\} \in E$.
- Also wird jeder Knoten von G' genau einmal besucht – ein Hamiltonkreis in G' .

Beweis Hitchhiker's-HamiltonianCycle nach HamiltonianCycle:

- Sei C' ein Hamiltonkreis in G' .
- Innere Clique-Knoten haben nur Nachbarn in der eigenen Clique; nach außen führen nur die Portalkanten.
- Somit durchläuft C' jede Clique als ein Segment von Portal zu Portal.
- Ziehe jede Clique auf ihren Originalknoten zusammen.
- Jede benutzte Portalkante stammt von einer Originalkante $\{u, v\} \in E$.
- Also entsteht ein Kreis in G , der jeden Knoten genau einmal besucht – ein Hamiltonkreis in G .

Laufzeit: Pro Knoten eine K_{43} mit 43^2 Kanten, pro Kante vier Portalkanten – $O(43^2 \cdot |V| + |E|)$.

Da das Problem in NP liegt und NP-schwer ist, folgt: HITCHHIKER'S-HAMILTONIANCYCLE ist NP-vollständig. $\square$

31 NP-Vollständigkeit zeigen für TSP-Entscheidung

Problem TSP-Entscheidung: Gegeben n Städte mit Distanzen $d(u, v) > 0$ und eine Schranke L . Gibt es eine Rundtour der Länge $\leq L$ durch alle Städte? Beweisen Sie, dass das Problem NP-vollständig ist. *Hinweis:* Reduzieren Sie HAMILTONIANCYCLE.

Lösung.

∈ **NP**: Eingabe: Distanzen, Schranke L und Zertifikat: eine Rundtour π . Der Verifizierer arbeitet wie folgt:

- Prüfe, ob π jede Stadt genau einmal besucht; lehne sonst ab.
- Summiere die Distanzen entlang π ; lehne ab, falls $> L$.
- Sonst akzeptiere.

Existiert eine Tour der Länge $\leq L$, wird sie akzeptiert; sonst scheitert jedes Zertifikat. Besuchsprüfung und Summieren – $O(n^2)$. Also liegt TSP-ENTSCHEIDUNG in NP.

NP-schwer: Zeige die NP-Schwere durch die Reduktion

$$\text{HAMILTONIANCYCLE} \leq_p \text{TSP-ENTSCHEIDUNG}.$$

HAMILTONIANCYCLE ist bereits als NP-vollständig bekannt.

Konstruktion: Sei $G = (V, E)$ eine HAMILTONIANCYCLE-Instanz mit $n = |V|$. Wir konstruieren daraus:

- Städte = V
- $d(u, v) = 1$ falls $\{u, v\} \in E$, sonst $d(u, v) = n + 1$
- $L = n$

Beweis HamiltonianCycle nach TSP:

- Sei C ein Hamiltonkreis in G .
- C ist eine Rundtour aus n Kanten mit Distanz je 1.
- Also hat die Tour Länge $n \leq L$.

Beweis TSP nach HamiltonianCycle:

- Sei π eine Rundtour der Länge $\leq n$.
- π hat n Schritte, jeder kostet mindestens 1.
- Enthielte π eine Nicht-Kante, wäre die Länge $\geq (n - 1) + (n + 1) = 2n > n$.
- Also nutzt π nur Kanten aus E – π ist ein Hamiltonkreis.

Laufzeit: Alle Paare belegen – $O(n^2)$.

Da TSP-ENTSCHEIDUNG ∈ NP und NP-schwer ist, folgt: TSP-ENTSCHEIDUNG ist NP-vollständig. $\square$

32 NP-Vollständigkeit zeigen für k -CLIQUE-DEG-3

Problem k -CLIQUE-DEG-3: Gegeben $G = (V, E)$ mit $\deg(v) \geq 3$ für alle $v \in V$ und $k \in \mathbb{N}_{>0}$. Entscheide: Gibt es eine Clique C mit $|C| \geq k$? Zeigen Sie NP-Vollständigkeit. *Hinweis: Reduzieren Sie k -CLIQUE.*

Lösung.

∈ **NP**: Eingabe: Graph $G = (V, E)$, Zahl k und Zertifikat $C \subseteq V$. Der Verifizierer arbeitet wie folgt:

- Prüfe für jeden Knoten $v \in V$, ob $\deg(v) \geq 3$ gilt; lehne sonst ab.
- Falls $|C| < k$, lehne ab.
- Prüfe für jedes Paar $\{x, y\} \subseteq C$, ob $\{x, y\} \in E$ gilt; lehne sonst ab.
- Sonst akzeptiere.

Korrektheit: Ist die Instanz gültig und hat G eine Clique mit $|C| \geq k$, dann ist diese ein akzeptiertes Zertifikat; sonst verletzt jedes Zertifikat eine Prüfung – abgelehnt.

Laufzeit: Gradprüfung $O(|V| + |E|)$, Größe zählen $O(|V|)$, alle Paare testen $O(|V|^2)$ – gesamt $O(|V|^2)$. Also gilt k -CLIQUE-DEG-3 ∈ NP.

Fall $k \leq 4$: Teste jede Knotenmenge $C \subseteq V$ mit $|C| = k$ und prüfe alle Paare per Adjazenzmatrix. Es gibt $O(|V|^k) \subseteq O(|V|^4)$ Mengen mit je $O(1)$ Prüfaufwand – gesamt $O(|V|^4)$. Für $k \leq 4$ ist das Problem also direkt lösbar; die Konstruktion nimmt daher o.B.d.A. $k \geq 5$ an.

NP-schwer: Zeige die NP-Schwere durch die Reduktion

$$k\text{-CLIQUE} \leq_p k\text{-CLIQUE-DEG-3}.$$

k -CLIQUE ist bereits als NP-vollständig bekannt.

Konstruktion: Sei $(G = (V, E), k)$ eine k -CLIQUE-Instanz, o.B.d.A. $k \geq 5$. Hänge an jeden Knoten v ein privates K_4 . Konstruiere $(G' = (V', E'), k')$ durch:

- $V' = V \cup \{a_v, b_v, c_v \mid v \in V\}$
- $E' = E \cup \{\{x, y\} \mid v \in V, x, y \in \{v, a_v, b_v, c_v\}, x \neq y\}$
- $k' = k$

Jedes $v \in V$ hat die drei Nachbarn a_v, b_v, c_v , Gadget-Knoten haben Grad genau 3 – die Instanz ist gültig.

Beweis k -Clique nach k -CLIQUE-DEG-3 (Fall $k \geq 5$):

- Sei C eine k -Clique in G .
- Es wurde keine Kante entfernt.
- Also ist C auch eine k -Clique in G' .

Beweis k -CLIQUE-DEG-3 nach k -Clique (Fall $k \geq 5$):

- Sei C' eine k -Clique in G' .
- Jeder Knoten in C' hat mindestens $k - 1 \geq 4$ Nachbarn innerhalb C' .
- Gadget-Knoten haben Grad $3 < 4$, können also keine vier Clique-Nachbarn haben.
- Somit liegt kein Gadget-Knoten in C' , also $C' \subseteq V$.
- Zwischen Knoten aus V kam keine Kante hinzu.
- Also ist C' eine k -Clique in G .

Im Fall $k \leq 4$ wird (G, k) direkt gelöst und eine triviale Instanz mit gleicher Antwort ausgegeben.

Laufzeit: Pro Knoten ein K_4 -Gadget konstanter Größe – $O(|V|)$; im Fall $k \leq 4$ alle Mengen testen – $O(|V|^4)$.

Da k -CLIQUE-DEG-3 $\in$ NP und NP-schwer ist, ist k -CLIQUE-DEG-3 NP-vollständig. $\square$

33 NP-Vollständigkeit zeigen für Knapsack

Problem Knapsack: Gegeben n Gegenstände mit Gewichten w_i und Profiten p_i , Kapazität B , Zielprofit P . Entscheide: Gibt es $S \subseteq [n]$ mit $\sum_{i \in S} w_i \leq B$ und $\sum_{i \in S} p_i \geq P$? Zeigen Sie NP-Vollständigkeit. *Hinweis: Reduzieren Sie SUBSETSUM.*

Lösung.

∈ **NP**: Eingabe: Gewichte w_i , Profite p_i , Schranken B, P und Zertifikat $S \subseteq [n]$. Der Verifizierer arbeitet wie folgt:

- Prüfe $\sum_{i \in S} w_i \leq B$; lehne sonst ab.
- Prüfe $\sum_{i \in S} p_i \geq P$; lehne sonst ab.
- Sonst akzeptiere.

Korrektheit: Gibt es eine gültige Auswahl, dann ist sie ein akzeptiertes Zertifikat; sonst wird jedes Zertifikat abgelehnt.

Laufzeit: Zwei Summen bilden und vergleichen – $O(n)$. Also gilt **KNAPSACK** ∈ **NP**.

NP-schwer: Zeige die NP-Schwere durch die Reduktion

$$\text{SUBSETSUM} \leq_p \text{KNAPSACK}.$$

SUBSETSUM ist bereits als NP-vollständig bekannt.

Konstruktion: Sei $(c_1, \dots, c_n, K)$ eine SUBSETSUM-Instanz. Setze

$$w_i := c_i, \quad p_i := c_i \quad (i \in [n]), \quad B := K, \quad P := K.$$

Beweis SubsetSum nach Knapsack:

- Sei S mit $\sum_{i \in S} c_i = K$.
- Dann gilt $\sum_{i \in S} w_i = K \leq B$.
- Und $\sum_{i \in S} p_i = K \geq P$.
- Also ist S eine Knapsack-Lösung.

Beweis Knapsack nach SubsetSum:

- Sei S eine Knapsack-Lösung.
- Gewicht und Profit von S sind dieselbe Zahl $\sum_{i \in S} c_i$.
- Aus der Kapazität folgt $\sum_{i \in S} c_i \leq B = K$.
- Aus dem Zielprofit folgt $\sum_{i \in S} c_i \geq P = K$.
- Also gilt $\sum_{i \in S} c_i = K$ – eine SubsetSum-Lösung.

Laufzeit: Werte kopieren – $O(n)$.

Da SUBSETSUM NP-vollständig ist, $\text{SUBSETSUM} \leq_p \text{KNAPSACK}$ gilt und **KNAPSACK** ∈ **NP**, ist **KNAPSACK** NP-vollständig. $\square$

34 NP-Vollständigkeit zeigen für Partition

Problem Partition: Gegeben Zahlen $c_1, \dots, c_n$. Entscheide: Gibt es S mit $\sum_{j \in S} c_j = \frac{1}{2} \sum_{j=1}^n c_j$? Zeigen Sie NP-Vollständigkeit. *Hinweis: Reduzieren Sie SUBSETSUM.*

Lösung.

∈ **NP:** Eingabe: Zahlen $c_1, \dots, c_n$ und Zertifikat $S \subseteq [n]$. Der Verifizierer arbeitet wie folgt:

- Berechne $\Sigma = \sum_{j=1}^n c_j$.
- Prüfe $\sum_{j \in S} c_j = \Sigma/2$; lehne sonst ab.
- Sonst akzeptiere.

Korrektheit: Gibt es eine Partition, dann ist eine ihrer Seiten ein akzeptiertes Zertifikat; sonst wird jedes Zertifikat abgelehnt.

Laufzeit: Zwei Summen bilden und vergleichen – $O(n)$. Also gilt **PARTITION** ∈ **NP**.

NP-schwer: Zeige die NP-Schwere durch die Reduktion

$$\text{SUBSETSUM} \leq_p \text{PARTITION}.$$

SUBSETSUM ist bereits als NP-vollständig bekannt.

Konstruktion: Sei $(c_1, \dots, c_n, K)$ eine SUBSETSUM-Instanz. Setze $N := \sum_{j=1}^n c_j + 1$ und hänge zwei Ausgleichszahlen an:

$$c_{n+1} := N - K, \quad c_{n+2} := K + 1.$$

Die neue Gesamtsumme ist $(N - 1) + (N - K) + (K + 1) = 2N$, die Hälfte also N .

Schlüsselbeobachtung: $c_{n+1} + c_{n+2} = N + 1 > N$ – in jeder Partition liegen die beiden Ausgleichszahlen auf verschiedenen Seiten.

Beweis SubsetSum nach Partition:

- Sei $S \subseteq [n]$ mit $\sum_{j \in S} c_j = K$.
- Setze $S' := S \cup \{n+1\}$.
- Dann gilt $\sum_{j \in S'} c_j = K + (N - K) = N$.
- Also ist S' eine Seite halber Gesamtsumme – eine Partition.

Beweis Partition nach SubsetSum:

- Sei S' eine Seite mit $\sum_{j \in S'} c_j = N$.
- Nach der Schlüsselbeobachtung liegen $n+1$ und $n+2$ auf verschiedenen Seiten.
- O.B.d.A. $n+1 \in S'$ und $n+2 \notin S'$.
- Setze $S := S' \setminus \{n+1\} \subseteq [n]$.
- Dann gilt $\sum_{j \in S} c_j = N - (N - K) = K$ – eine SubsetSum-Lösung.

Laufzeit: Gesamtsumme berechnen und zwei Zahlen anhängen – $O(n)$.

Da SUBSETSUM NP-vollständig ist, $\text{SUBSETSUM} \leq_p \text{PARTITION}$ gilt und **PARTITION** ∈ **NP**, ist **PARTITION** NP-vollständig. $\square$

35 NP-Vollständigkeit zeigen für $P2 || C_{\max}$

Problem $P2 || C_{\max}$ (Entscheidung): Gegeben n Jobs mit Zeiten $p_1, \dots, p_n$, zwei identische Maschinen und eine Schranke T . Gibt es einen Schedule mit Makespan $\leq T$? Beweisen Sie, dass das Problem NP-vollständig ist. *Hinweis:* Reduzieren Sie PARTITION.

Lösung.

∈ **NP**: Eingabe: Zeiten, Schranke T und Zertifikat $S \subseteq [n]$ – die Jobs auf Maschine 1. Der Verifizierer arbeitet wie folgt:

- Prüfe $\sum_{i \in S} p_i \leq T$; lehne sonst ab.
- Prüfe $\sum_{i \notin S} p_i \leq T$; lehne sonst ab.
- Sonst akzeptiere.

Existiert ein Schedule mit Makespan $\leq T$, wird seine Maschinen-1-Menge akzeptiert; sonst scheitert jedes Zertifikat. Zwei Summen bilden – $O(n)$. Also liegt $P2 || C_{\max}$ in NP.

NP-schwer: Zeige die NP-Schwere durch die Reduktion

$$\text{PARTITION} \leq_p P2 || C_{\max}.$$

PARTITION ist bereits als NP-vollständig bekannt.

Konstruktion: Sei $(c_1, \dots, c_n)$ eine PARTITION-Instanz mit Summe Σ . Ist Σ ungerade, gib eine feste Nein-Instanz aus. Sonst konstruiere:

- $p_i = c_i$ für $i \in [n]$
- $m = 2$ Maschinen, $T = \Sigma/2$

Beweis Partition nach $P2 || C_{\max}$:

- Sei S mit $\sum_{i \in S} c_i = \Sigma/2$.
- Lege S auf Maschine 1, den Rest auf Maschine 2.
- Beide Maschinen haben Last $\Sigma/2 = T$.

Beweis $P2 || C_{\max}$ nach Partition:

- Sei ein Schedule mit Makespan $\leq \Sigma/2$ gegeben.
- Beide Lasten summieren zu Σ , und jede ist $\leq \Sigma/2$.
- Also sind beide Lasten genau $\Sigma/2$.
- Die Jobs auf Maschine 1 bilden eine Menge S mit $\sum_{i \in S} c_i = \Sigma/2$.

Laufzeit: Zeiten übernehmen und Summe halbieren – $O(n)$.

Da $P2 || C_{\max} \in \text{NP}$ und NP-schwer ist, folgt: $P2 || C_{\max}$ ist NP-vollständig. □

36 NP-Vollständigkeit zeigen für SubsetSumCardinality

Problem SubsetSumCardinality: Gegeben $c_1, \dots, c_n$ (n gerade) und K . Entscheide: Gibt es S mit $|S| = n/2$ und $\sum_{i \in S} c_i = K$? Zeigen Sie NP-Vollständigkeit. *Hinweis: Reduzieren Sie SUBSETSUM.*

Lösung.

∈ **NP**: Eingabe: Zahlen $c_1, \dots, c_n$ (n gerade), Zielwert K und Zertifikat $S \subseteq [n]$. Der Verifizierer arbeitet wie folgt:

- Prüfe $|S| = n/2$; lehne sonst ab.
- Prüfe $\sum_{i \in S} c_i = K$; lehne sonst ab.
- Sonst akzeptiere.

Korrektheit: Gibt es eine Lösung passender Größe und Summe, dann ist sie ein akzeptiertes Zertifikat; sonst wird jedes Zertifikat abgelehnt.

Laufzeit: $|S|$ zählen und Summe bilden – $O(n)$. Also gilt SUBSETSUMCARDINALITY ∈ NP.

NP-schwer: Zeige die NP-Schwere durch die Reduktion

$$\text{SUBSETSUM} \leq_p \text{SUBSETSUMCARDINALITY}.$$

SUBSETSUM ist bereits als NP-vollständig bekannt.

Konstruktion: Sei $(c_1, \dots, c_n, K)$ eine SUBSETSUM-Instanz. Shifte alle Größen um 1 und füge n Padding-Items der Größe 1 hinzu:

$$c'_i := c_i + 1 \quad (i \leq n), \quad c'_i := 1 \quad (n < i \leq 2n), \quad K' := K + n.$$

Die Ausgabeinstanz hat $2n$ (gerade) Items und verlangt $|S'| = n$.

Beweis SubsetSum nach SubsetSumCardinality:

- Sei $S \subseteq [n]$ mit $\sum_{i \in S} c_i = K$.
- Dann gilt $\sum_{i \in S} c'_i = \sum_{i \in S} (c_i + 1) = K + |S|$.
- Setze $S' := S \cup \{n+1, \dots, 2n - |S|\}$, also $n - |S|$ Padding-Items dazu.
- Dann gilt $|S'| = |S| + (n - |S|) = n$.
- Und $\sum_{i \in S'} c'_i = (K + |S|) + (n - |S|) = K + n = K'$.
- Also ist S' eine Lösung der neuen Instanz.

Beweis SubsetSumCardinality nach SubsetSum:

- Sei S' mit $|S'| = n$ und $\sum_{i \in S'} c'_i = K + n$.
- Setze $S := S' \cap [n]$; dann enthält S' genau $n - |S|$ Padding-Items.
- Es folgt $\sum_{i \in S} (c_i + 1) = (K + n) - (n - |S|) = K + |S|$.
- Also $\sum_{i \in S} c_i = K$ – eine SubsetSum-Lösung.

Laufzeit: n Zahlen shiften und n Padding-Items anhängen – $O(n)$.

Da SUBSETSUM NP-vollständig ist, SUBSETSUM $\leq_p$ SUBSETSUMCARDINALITY gilt und SUBSETSUMCARDINALITY ∈ NP, ist SUBSETSUMCARDINALITY NP-vollständig. $\square$

37 NP-Vollständigkeit zeigen für $(a_1=1)$ -SubsetSum

Problem $(a_1=1)$ -SubsetSum: Gegeben n Items mit Größen $a_i \in \mathbb{N}_{>0}$, wobei $a_1 = 1$, und Zielwert T . Entscheide: Gibt es $I \subseteq [n]$ mit $\sum_{i \in I} a_i = T$? Zeigen Sie NP-Vollständigkeit.
Hinweis: Reduzieren Sie SUBSETSUM.

Lösung.

∈ **NP**: Eingabe: Größen $a_1, \dots, a_n$, Zielwert T und Zertifikat $I \subseteq [n]$. Der Verifizierer arbeitet wie folgt:

- Prüfe $a_1 = 1$; lehne sonst ab.
- Prüfe $\sum_{i \in I} a_i = T$; lehne sonst ab.
- Sonst akzeptiere.

Korrektheit: Ist die Instanz gültig und hat sie eine Lösung, dann ist diese ein akzeptiertes Zertifikat; sonst wird jedes Zertifikat abgelehnt.

Laufzeit: a_1 prüfen und Summe bilden – $O(n)$. Also gilt $(a_1=1)$ -SUBSETSUM ∈ NP.

NP-schwer: Zeige die NP-Schwere durch die Reduktion

$$\text{SUBSETSUM} \leq_p (a_1=1)\text{-SUBSETSUM}.$$

SUBSETSUM ist bereits als NP-vollständig bekannt.

Konstruktion: Sei $(c_1, \dots, c_n, K)$ eine SUBSETSUM-Instanz. Verdopple alle Größen und stelle ein Ballast-Item voran:

$$a_1 := 1, \quad a_{i+1} := 2c_i \quad (i \in [n]), \quad T := 2K.$$

Wegen $a_1 = 1$ ist die Instanz gültig.

Beweis SubsetSum nach $(a_1=1)$ -SubsetSum:

- Sei S mit $\sum_{i \in S} c_i = K$.
- Setze $I := \{i + 1 \mid i \in S\}$.
- Dann gilt $\sum_{j \in I} a_j = \sum_{i \in S} 2c_i = 2K = T$.
- Also ist I eine Lösung der neuen Instanz.

Beweis $(a_1=1)$ -SubsetSum nach SubsetSum:

- Sei I mit $\sum_{i \in I} a_i = T = 2K$.
- Alle Items außer a_1 sind gerade, und $T = 2K$ ist gerade.
- Angenommen $1 \in I$: Dann wäre die Summe ungerade, da $a_1 = 1$ das einzige ungerade Item ist – Widerspruch.
- Also $1 \notin I$; setze $S := \{i - 1 \mid i \in I\}$.
- Dann gilt $\sum_{i \in S} 2c_i = 2K$, also $\sum_{i \in S} c_i = K$ – eine SubsetSum-Lösung.

Laufzeit: n Zahlen verdoppeln und das Ballast-Item voranstellen – $O(n)$.

Da SUBSETSUM NP-vollständig ist, die Reduktion gilt und $(a_1=1)$ -SUBSETSUM ∈ NP, ist $(a_1=1)$ -SUBSETSUM NP-vollständig. $\square$

38 NP-Vollständigkeit zeigen für SubsetSum mit Teilbarkeit

Sei L die Menge der SUBSETSUM-Instanzen, bei denen jede Itemgröße durch 3 oder durch 7 teilbar ist. Entscheide: Gibt es S mit $\sum_{i \in S} c_i = K$? Zeigen Sie, dass L NP-vollständig ist. *Hinweis: Reduzieren Sie SUBSETSUM.*

Lösung.

∈ **NP**: Eingabe: Zahlen $c_1, \dots, c_n$, Zielwert K und Zertifikat $S \subseteq [n]$. Der Verifizierer arbeitet wie folgt:

- Prüfe für jede Größe c_i , ob $3 \mid c_i$ oder $7 \mid c_i$ gilt; lehne sonst ab.
- Prüfe $\sum_{i \in S} c_i = K$; lehne sonst ab.
- Sonst akzeptiere.

Korrektheit: Ist die Instanz eine gültige Variante mit Lösung, dann ist diese ein akzeptiertes Zertifikat; sonst wird jedes Zertifikat abgelehnt.

Laufzeit: Teilbarkeitstests und Summe bilden – $O(n)$. Also gilt $L \in \text{NP}$.

NP-schwer: Zeige die NP-Schwere durch die Reduktion

$$\text{SUBSETSUM} \leq_p L.$$

SUBSETSUM ist bereits als NP-vollständig bekannt.

Konstruktion: Sei $(c_1, \dots, c_n, K)$ eine SUBSETSUM-Instanz. Skaliere mit 3:

$$c'_i := 3c_i \quad (i \in [n]), \quad K' := 3K.$$

Jede Größe c'_i ist durch 3 teilbar – die Instanz liegt in L .

Beweis SubsetSum nach Variante:

- Sei S mit $\sum_{i \in S} c_i = K$.
- Dann gilt $\sum_{i \in S} c'_i = \sum_{i \in S} 3c_i = 3 \sum_{i \in S} c_i = 3K = K'$.
- Also ist S eine Lösung der neuen Instanz.

Beweis Variante nach SubsetSum:

- Sei S mit $\sum_{i \in S} c'_i = K'$.
- Wegen $c'_i = 3c_i$ heißt das $3 \sum_{i \in S} c_i = 3K$.
- Division durch 3 liefert $\sum_{i \in S} c_i = K$.
- Also ist S eine SubsetSum-Lösung.

Laufzeit: Jede der n Zahlen und K mit 3 multiplizieren – $O(n)$.

Da SUBSETSUM NP-vollständig ist, $\text{SUBSETSUM} \leq_p L$ gilt und $L \in \text{NP}$, ist L NP-vollständig.

□

39 NP-Vollständigkeit zeigen für SubsetSum ohne Zweierpotenzen

Sei L die Menge der SUBSETSUM-Instanzen, bei denen keine Itemgröße eine Zweierpotenz ist. Entscheide: Gibt es S mit $\sum_{i \in S} c_i = K$? Zeigen Sie, dass L NP-vollständig ist. *Hinweis: Reduzieren Sie SUBSETSUM.*

Lösung.

∈ **NP**: Eingabe: Zahlen $c_1, \dots, c_n$, Zielwert K und Zertifikat $S \subseteq [n]$. Der Verifizierer arbeitet wie folgt:

- Prüfe für jede Größe c_i , dass sie keine Zweierpotenz ist (c_i ist Zweierpotenz genau dann, wenn $c_i \wedge (c_i - 1) = 0$); lehne sonst ab.
- Prüfe $\sum_{i \in S} c_i = K$; lehne sonst ab.
- Sonst akzeptiere.

Korrektheit: Ist die Instanz eine gültige Variante mit Lösung, dann ist diese ein akzeptiertes Zertifikat; sonst wird jedes Zertifikat abgelehnt.

Laufzeit: Bit-Test je Größe und Summe bilden – $O(n)$. Also gilt $L \in \text{NP}$.

NP-schwer: Zeige die NP-Schwere durch die Reduktion

$$\text{SUBSETSUM} \leq_p L.$$

SUBSETSUM ist bereits als NP-vollständig bekannt.

Konstruktion: Sei $(c_1, \dots, c_n, K)$ eine SUBSETSUM-Instanz. Skaliere mit 3:

$$c'_i := 3c_i \quad (i \in [n]), \quad K' := 3K.$$

Jede Größe $c'_i \geq 3$ hat den Primfaktor 3; Zweierpotenzen haben nur den Primfaktor 2. Also ist kein c'_i eine Zweierpotenz – die Instanz liegt in L .

Beweis SubsetSum nach Variante:

- Sei S mit $\sum_{i \in S} c_i = K$.
- Dann gilt $\sum_{i \in S} c'_i = 3 \sum_{i \in S} c_i = 3K = K'$.
- Also ist S eine Lösung der neuen Instanz.

Beweis Variante nach SubsetSum:

- Sei S mit $\sum_{i \in S} c'_i = K'$.
- Das heißt $3 \sum_{i \in S} c_i = 3K$.
- Division durch 3 liefert $\sum_{i \in S} c_i = K$.
- Also ist S eine SubsetSum-Lösung.

Laufzeit: Jede der n Zahlen und K mit 3 multiplizieren – $O(n)$.

Da SUBSETSUM NP-vollständig ist, $\text{SUBSETSUM} \leq_p L$ gilt und $L \in \text{NP}$, ist L NP-vollständig. $\square$

40 NP-Vollständigkeit zeigen für AtMostTwoPerSize-SubsetSum

Problem AtMostTwoPerSize-SubsetSum: Gegeben ein Zielwert $T > 0$ und n Items mit Größen $a_i \in \mathbb{N}_{>0}$, wobei jede Größe höchstens zweimal auftritt. Entscheide: Gibt es S mit $\sum_{i \in S} a_i = T$? Zeigen Sie NP-Vollständigkeit. *Hinweis: Bei 3-EXACTCOVER sind ein Universum U mit $|U| = 3m$ und Dreiermengen $S_1, \dots, S_n \subseteq U$ gegeben; gesucht ist eine Auswahl, die jedes Element genau einmal überdeckt. Reduzieren Sie 3-EXACTCOVER.*

Lösung.

∈ **NP**: Eingabe: Größen $a_1, \dots, a_n$, Zielwert T und Zertifikat $S \subseteq [n]$. Der Verifizierer arbeitet wie folgt:

- Prüfe für jede Größe, ob sie unter $a_1, \dots, a_n$ höchstens zweimal vorkommt; lehne sonst ab.
- Prüfe $\sum_{i \in S} a_i = T$; lehne sonst ab.
- Sonst akzeptiere.

Korrektheit: Ist die Instanz eine gültige Variante mit Lösung, dann ist diese ein akzeptiertes Zertifikat; sonst wird jedes Zertifikat abgelehnt.

Laufzeit: Häufigkeiten per Paarvergleich $O(n^2)$ und Summe bilden $O(n)$ – gesamt $O(n^2)$. Also gilt **ATMOSTTWOPEXSIZE-SUBSETSUM** ∈ **NP**.

NP-schwer: Zeige die NP-Schwere durch die Reduktion

$$3\text{-EXACTCOVER} \leq_p \text{ATMOSTTWOPEXSIZE-SUBSETSUM}.$$

3-EXACTCOVER ist bereits als NP-schwer bekannt.

Konstruktion: Sei $(U, S_1, \dots, S_n)$ eine 3-EXACTCOVER-Instanz mit $U = \{u_1, \dots, u_{3m}\}$, o.B.d.A. die S_j paarweise verschieden (Duplikate vorab löschen). Kodiere jede Menge als Ziffernzahl in Basis $n + 1$:

- $c_j := \sum_{u_i \in S_j} (n + 1)^{i-1}$ für $j = 1, \dots, n$ – Ziffer 1 an den drei Stellen der Elemente von S_j
- $T := \sum_{j=0}^{3m-1} (n + 1)^j$ – an jeder der $3m$ Stellen eine 1

Verschiedene Mengen setzen verschiedene Potenzen, also sind $c_1, \dots, c_n$ paarweise verschieden – jede Größe tritt nur einmal auf, die Instanz ist gültig.

Beweis 3-ExactCover nach AtMostTwoPerSize-SubsetSum:

- Sei $\mathcal{S}$ eine exakte Überdeckung.
- Jedes Element u_i wird von genau einer Menge in $\mathcal{S}$ überdeckt.
- Also kommt in $\sum_{S_j \in \mathcal{S}} c_j$ jede Potenz $(n + 1)^{i-1}$ genau einmal vor.
- Damit gilt $\sum_{S_j \in \mathcal{S}} c_j = \sum_{j=0}^{3m-1} (n + 1)^j = T$.
- Also ist $\mathcal{S}$ eine Lösung der SubsetSum-Instanz.

Beweis AtMostTwoPerSize-SubsetSum nach 3-ExactCover:

- Sei S eine Auswahl mit $\sum_{j \in S} c_j = T$.
- An jeder Ziffernstelle addieren sich höchstens n Einsen, da es nur n Mengen gibt.
- Da die Basis $n + 1$ ist, entsteht beim Addieren kein Übertrag.
- In T hat jede der $3m$ Stellen den Wert 1.
- Also trägt zu jeder Stelle genau eine gewählte Menge bei.
- Damit überdecken die gewählten Mengen jedes Element genau einmal – eine exakte Überdeckung.

Laufzeit: Duplikate per Paarvergleich der n Mengen entfernen – $O(n^2)$; pro Menge drei Potenzen von $n + 1$ und T aus $3m$ Potenzen aufaddieren – $O(n + m)$ Additionen.

Da das Problem in NP liegt und NP-schwer ist, folgt: ATMOSTTWOPEPERSIZE-SUBSETSUM ist NP-vollständig. $\square$

41 NP-Schwere zeigen für HALT_{TM}

$\text{HALT}_{\text{TM}} := \{\langle M, w \rangle \mid M \text{ ist eine DTM und hält auf } w\}$. Zeigen Sie, dass HALT_{TM} NP-schwer ist, und begründen Sie, warum $\text{HALT}_{\text{TM}} \notin \text{NP}$. *Hinweis: Reduzieren Sie SAT.*

Lösung.

NP-schwer: Zeige die NP-Schwere durch die Reduktion

$$\text{SAT} \leq_p \text{HALT}_{\text{TM}}.$$

SAT ist bereits als NP-schwer bekannt.

Konstruktion: Sei φ eine SAT-Instanz über den Variablen $x_1, \dots, x_n$. Gib die Instanz $\langle M, w \rangle$ aus:

- $w := \langle \varphi \rangle$ – die Kodierung von φ .
- M ist eine feste DTM, die auf Eingabe $\langle \varphi \rangle$ so arbeitet: probiere nacheinander alle 2^n Belegungen der Variablen; halte, sobald eine Belegung φ erfüllt; gehe in eine Endlosschleife, falls keine der 2^n Belegungen erfüllt.

Beweis SAT nach HALT:

- Sei $\varphi \in \text{SAT}$.
- Dann existiert eine erfüllende Belegung.
- M probiert alle Belegungen, findet diese und hält auf w .
- Also $\langle M, w \rangle \in \text{HALT}_{\text{TM}}$.

Beweis HALT nach SAT:

- Sei $\langle M, w \rangle \in \text{HALT}_{\text{TM}}$, also hält M auf w .
- M hält nur, wenn es eine erfüllende Belegung findet.
- Denn ohne erfüllende Belegung geht M in die Endlosschleife.
- Also existiert eine erfüllende Belegung für φ .
- Damit $\varphi \in \text{SAT}$.

Laufzeit: Nur φ kodieren und die feste Maschine M ausgeben – $O(|\varphi|)$. Ob M auf w hält, spielt für die Reduktion keine Rolle.

Da SAT NP-schwer ist und $\text{SAT} \leq_p \text{HALT}_{\text{TM}}$ gilt, ist HALT_{TM} NP-schwer.

Warum $\text{HALT}_{\text{TM}} \notin \text{NP}$: HALT_{TM} ist das Halteproblem und damit unentscheidbar. Jede Sprache in NP ist jedoch entscheidbar: Ein Verifizierer liefert durch Absuchen aller polynomiell langen Zertifikate ein Entscheidungsverfahren. Eine unentscheidbare Sprache kann also nicht in NP liegen. Somit ist HALT_{TM} NP-schwer, aber nicht NP-vollständig. $\square$

4 Approximative Algorithmen

42 Güte zeigen für ListScheduling

$P \parallel C_{\max}$: n Jobs mit Zeiten $p_1, \dots, p_n$ auf m Maschinen, minimiere den Makespan. *Hinweis:* ListScheduling legt jeden Job der Reihe nach auf die aktuell am wenigsten belastete Maschine. Zeigen Sie die Güte $2 - \frac{1}{m}$.

Lösung.

Zu zeigen:

$$\text{LS}(I) \leq \left(2 - \frac{1}{m}\right) \text{OPT}(I).$$

O.B.d.A. trage Maschine M_1 am Ende die höchste Last $L = \text{LS}(I)$, und sei J_k der *letzte* Job, der auf M_1 gelegt wurde.

Kritischer Moment. Als J_k zugeteilt wurde, war M_1 die leerste Maschine mit Last $L - p_k$. Also trugen zu diesem Zeitpunkt alle m Maschinen mindestens Last $L - p_k$, woraus für die Gesamtarbeit folgt

$$\sum_{i=1}^n p_i \geq m(L - p_k) + p_k.$$

Untere Schranken für das Optimum. Die Gesamtarbeit verteilt sich auf m Maschinen, und jeder Job läuft irgendwo, also

$$\text{OPT}(I) \geq \frac{1}{m} \sum_{i=1}^n p_i \quad \text{und} \quad \text{OPT}(I) \geq p_k.$$

Kombination. Erst die Durchschnittsschranke mit der Kernbeobachtung verbinden:

$$\text{OPT}(I) \geq \frac{1}{m} \sum_{i=1}^n p_i \geq \frac{m(L - p_k) + p_k}{m} = L - p_k + \frac{p_k}{m} = L - \left(1 - \frac{1}{m}\right)p_k.$$

Nach L aufgelöst:

$$\text{LS}(I) = L \leq \text{OPT}(I) + \left(1 - \frac{1}{m}\right)p_k.$$

Nun $p_k \leq \text{OPT}(I)$ einsetzen:

$$\text{LS}(I) \leq \text{OPT}(I) + \left(1 - \frac{1}{m}\right)\text{OPT}(I) = \left(2 - \frac{1}{m}\right)\text{OPT}(I).$$

Somit hat ListScheduling die Güte $2 - \frac{1}{m}$.

43 Güte zeigen für LPT

$P \parallel C_{\max}$ wie zuvor. *Hinweis:* LPT sortiert die Jobs zuerst *absteigend* nach Laufzeit und wendet darauf ListScheduling an. Zeigen Sie die Güte $\frac{4}{3} - \frac{1}{3m}$.

Lösung.

Zu zeigen:

$$\text{LPT}(I) \leq \left(\frac{4}{3} - \frac{1}{3m}\right) \text{OPT}(I).$$

Nummeriere absteigend, $p_1 \geq \dots \geq p_n$. Sei J_n der Job, der den Makespan setzt (der kleinste Job auf der vollsten Maschine).

Kritischer Moment. Als J_n platziert wurde, hatte seine Maschine die kleinste Last s . Wie bei ListScheduling folgt $s \leq \text{OPT}(I) - p_n/m$, also

$$\text{LPT}(I) = s + p_n \leq \text{OPT}(I) + \left(1 - \frac{1}{m}\right)p_n.$$

Fallunterscheidung nach p_n .

- Ist $p_n \leq \text{OPT}(I)/3$, so liefert Einsetzen direkt $\text{LPT}(I) \leq \left(\frac{4}{3} - \frac{1}{3m}\right) \text{OPT}(I)$.
- Ist $p_n > \text{OPT}(I)/3$, dann sind *alle* Jobs $> \text{OPT}(I)/3$. Im Optimum trägt jede Maschine höchstens zwei Jobs. Für diesen Fall ist bekannt, dass LPT sogar optimal ist ($\text{LPT}(I) = \text{OPT}(I)$).

In beiden Fällen gilt die Schranke, also hat LPT die Güte $\frac{4}{3} - \frac{1}{3m}$.

44 Güte zeigen für RoundRobin

$P \parallel C_{\max}$ wie zuvor. *Hinweis:* RoundRobin sortiert die Jobs absteigend und verteilt sie reihum, d. h. J_j kommt auf Maschine $((j - 1) \bmod m) + 1$. Zeigen Sie die Güte 2 (die Rate wird im Worst Case bis $2 - \frac{1}{m}$ ausgereizt).

Lösung.

Zu zeigen:

$$\text{RR}(I) \leq 2 \text{OPT}(I).$$

Angenommen, es gälte $\text{RR}(I) > 2 \text{OPT}(I)$. Sei M_i die Maschine mit maximaler Last; sie trägt die Jobs $J_i, J_{i+m}, J_{i+2m}, \dots$

Blockschränke. Wegen der absteigenden Sortierung ist $p_{i+lm} \leq p_j$ für jedes $j \leq lm$; insbesondere ist p_{i+lm} höchstens so groß wie der Durchschnitt des vorangehenden m -er-Blocks:

$$p_{i+lm} \leq \frac{1}{m} \sum_{j=(l-1)m+1}^{lm} p_j \quad (l \geq 1).$$

Aufsummieren. Summation über $l \geq 1$ ergibt

$$\text{RR}(I) - p_i = \sum_{l \geq 1} p_{i+lm} \leq \frac{1}{m} \sum_j p_j \leq \text{OPT}(I).$$

Kombination. Mit $p_i \leq p_1 \leq \text{OPT}(I)$ folgt $\text{RR}(I) \leq 2 \text{OPT}(I)$ – im Widerspruch zur Annahme.

Somit hat RoundRobin die Güte 2; die Familie aus einem Job der Größe m und $m(m - 1)$ Einser-Jobs treibt die Rate gegen $2 - \frac{1}{m}$.

45 Güte zeigen für Parallel-Task-ListScheduling

Beim Parallel-Task-Scheduling belegt Job j während seiner Laufzeit p_j *gleichzeitig* $q_j \in [m]$ Maschinen. *Hinweis:* ListScheduling platziert jeden Job zum frühestmöglichen Zeitpunkt auf den q_j am wenigsten belasteten Maschinen. Zeigen Sie für alle Instanzen mit $\frac{m}{3} < q_j \leq \frac{m}{2}$: $LS(I) \leq OPT(I) + \frac{1}{2} p_{\max}$.

Lösung.

Zu zeigen:

$$LS(I) \leq OPT(I) + \frac{1}{2} p_{\max}.$$

Sei j ein Job, der den Makespan bestimmt, mit Startzeit $s = LS(I) - p_j$.

In $[0, s)$ laufen stets genau zwei Jobs. Zu jedem Zeitpunkt $t < s$ sind weniger als $q_j \leq \frac{m}{2}$ Maschinen frei – sonst wäre j früher gestartet worden –, also mehr als $\frac{m}{2}$ belegt. Da jeder Job höchstens $\frac{m}{2}$ Maschinen belegt, laufen mindestens zwei Jobs; da jeder Job mehr als $\frac{m}{3}$ belegt, laufen höchstens zwei (drei bräuchten $> m$ Maschinen). Also laufen genau zwei, und

$$2s = \int_0^s \#\{\text{laufende Jobs}\} dt \leq \sum_i p_i - p_j.$$

Schranke für das Optimum. Auch im Optimum laufen nie drei Jobs gleichzeitig (sie bräuchten $> 3 \cdot \frac{m}{3} = m$ Maschinen), also $\sum_i p_i \leq 2 OPT(I)$. Einsetzen:

$$s \leq \frac{1}{2} \left(\sum_i p_i - p_j \right) \leq OPT(I) - \frac{1}{2} p_j.$$

Kombination. Mit $LS(I) = s + p_j$ folgt

$$LS(I) \leq OPT(I) + \frac{1}{2} p_j \leq OPT(I) + \frac{1}{2} p_{\max}.$$

Somit gilt die behauptete Schranke.

46 Güte zeigen für ModifiedGreedy

KNAPSACK: Gegenstände mit Gewichten w_i , Profiten p_i , Kapazität B ; maximiere den Profit.
Hinweis: Greedy sortiert absteigend nach Profitdichte p_i/w_i und packt in dieser Reihenfolge jeden Gegenstand ein, der noch passt; ModifiedGreedy (MGA) gibt das Bessere aus Greedy-Lösung und profitreichstem Einzel-Gegenstand aus. Zeigen Sie die Güte 2.

Lösung.

Zu zeigen:

$$\text{OPT}(I) \leq 2 \text{MGA}(I).$$

Nummeriere nach Dichte, $p_1/w_1 \geq \dots \geq p_n/w_n$, und sei $k+1$ der erste Gegenstand, den Greedy nicht mehr einpacken kann.

Schritt 1 (Relaxierung). Erlaubt man, Gegenstände *anteilig* einzupacken, wächst der Lösungsraum; das fraktionale Optimum OPT_f erfüllt daher $\text{OPT}(I) \leq \text{OPT}_f(I)$.

Schritt 2 (fraktionales Optimum). Die beste fraktionale Lösung füllt den Rucksack in Dichte-Reihenfolge: die Gegenstände $1, \dots, k$ ganz, vom Gegenstand $k+1$ einen Bruchteil ≤ 1 . Also $\text{OPT}_f(I) \leq p_1 + \dots + p_k + p_{k+1}$. Da Greedy die Gegenstände $1, \dots, k$ ebenfalls einpackt, ist $p_1 + \dots + p_k \leq \text{GA}(I)$, mithin

$$\text{OPT}_f(I) \leq \text{GA}(I) + p_{k+1}.$$

Schritt 3 (Kombination). Mit $p_{k+1} \leq p_{\max}$ und $\text{MGA}(I) = \max\{\text{GA}(I), p_{\max}\}$ folgt

$$\text{OPT}(I) \leq \text{GA}(I) + p_{\max} \leq 2 \max\{\text{GA}(I), p_{\max}\} = 2 \text{MGA}(I).$$

Somit hat ModifiedGreedy die Güte 2.

47 Güte zeigen für Sahni

KNAPSACK wie zuvor. *Hinweis:* Sahni (k -Enumeration) probiert jede Vorauswahl von höchstens k Gegenständen als festen Grundstock, füllt den Rest mit Greedy (Dichte-Reihenfolge) auf und gibt die beste gefundene Packung aus. Zeigen Sie die Güte $1 + \frac{1}{k}$.

Lösung.

Zu zeigen:

$$\text{OPT}(I) \leq \left(1 + \frac{1}{k}\right) \text{Sahni}(I).$$

Sei O eine optimale Auswahl mit Profit $\text{OPT}(I) = p(O)$.

Fall $|O| \leq k$. Sahni probiert O selbst als Vorauswahl. Also $\text{Sahni}(I) \geq p(O) = \text{OPT}(I)$.

Fall $|O| > k$. Sei $H \subseteq O$ die Menge der k profitreichsten Gegenstände aus O . Betrachte die Iteration mit Vorauswahl H .

- Greedy füllt mit den restlichen Gegenständen auf. Es verliert höchstens den ersten übersprungenen Gegenstand $o^* \in O$: $\text{Sahni}(I) \geq \text{OPT}(I) - p_{o^*}$.
- o^* und die k Gegenstände aus H sind $k + 1$ Gegenstände aus O , jeder mit Profit $\geq p_{o^*}$. Also $(k + 1)p_{o^*} \leq \text{OPT}(I)$, d. h. $p_{o^*} \leq \text{OPT}(I)/(k + 1)$.

Kombiniert:

$$\text{Sahni}(I) \geq \text{OPT}(I) - \frac{\text{OPT}(I)}{k + 1} = \frac{k}{k + 1} \text{OPT}(I),$$

also $\text{OPT}(I) \leq \left(1 + \frac{1}{k}\right) \text{Sahni}(I)$. Somit hat Sahni die Güte $1 + \frac{1}{k}$.

48 Güte widerlegen für Greedy

KNAPSACK wie zuvor. *Hinweis:* Greedy sortiert absteigend nach Profitudichte p_i/w_i und packt jeden noch passenden Gegenstand ein. Zeigen Sie, dass Greedy keine konstante Güte hat: Für kein c gilt $\text{OPT}(I) \leq c \cdot \text{GA}(I)$ für alle Instanzen.

Lösung.

Zu zeigen: Die Rate $\text{OPT}(I)/\text{GA}(I)$ ist unbeschränkt.

Instanzfamilie. Für einen Parameter $B \geq 2$ betrachte zwei Gegenstände mit Kapazität B :

$$(w_1, p_1) = (1, 1), \quad (w_2, p_2) = (B, B - 1).$$

Die Dichten sind $\frac{p_1}{w_1} = 1$ und $\frac{p_2}{w_2} = \frac{B-1}{B} < 1$.

Greedy. Nach Dichte kommt der erste Gegenstand zuerst; er passt und wird eingepackt. Danach bleibt Restkapazität $B - 1 < B$, sodass der zweite nicht mehr passt. Also $\text{GA}(I) = 1$.

Optimum. Der zweite Gegenstand allein hat Gewicht $B \leq B$ und Profit $B - 1$, also $\text{OPT}(I) \geq B - 1$.

Rate. Damit

$$\frac{\text{OPT}(I)}{\text{GA}(I)} \geq \frac{B - 1}{1} = B - 1 \xrightarrow{B \rightarrow \infty} \infty.$$

Zu jedem c liefert also schon $B = c + 2$ eine Instanz mit $\text{OPT} > c \cdot \text{GA}$. Somit besitzt Greedy keine konstante Güte.

49 Güte zeigen für ΔTSP1

Metrisches TSP: vollständiger Graph mit symmetrischen Distanzen, die die Dreiecksungleichung $d(u, v) \leq d(u, w) + d(w, v)$ erfüllen; gesucht eine kürzeste Rundreise. *Hinweis (ΔTSP1):* (1) MST T berechnen; (2) alle MST-Kanten verdoppeln; (3) Eulerkreis; (4) Abkürzen (schon besuchte Knoten überspringen). Zeigen Sie die Güte 2.

Lösung.

Zu zeigen:

$$d(R) \leq 2 \text{OPT}(I)$$

für die berechnete Tour R , in drei Schritten.

Schritt 1 ($w(T) \leq \text{OPT}$). Entfernt man aus einer optimalen Rundreise eine Kante, bleibt ein aufspannender Baum. Sein Gewicht ist $\leq \text{OPT}(I)$, und da T ein *minimaler* Spannbaum ist, gilt $w(T) \leq \text{OPT}(I)$.

Schritt 2 (Eulerkreis). Durch das Verdoppeln haben alle Knoten geraden Grad, es existiert ein Eulerkreis. Er benutzt jede MST-Kante genau zweimal, hat also Länge

$$2w(T) \leq 2\text{OPT}(I).$$

Schritt 3 (Abkürzen). Beim Überspringen bereits besuchter Knoten wird eine Teilfolge $u \rightarrow w \rightarrow v$ durch die direkte Kante $u \rightarrow v$ ersetzt. Nach der Dreiecksungleichung ist $d(u, v) \leq d(u, w) + d(w, v)$, das Abkürzen verlängert die Tour also nicht:

$$d(R) \leq 2w(T) \leq 2\text{OPT}(I).$$

Somit hat ΔTSP1 die Güte 2.

50 Güte zeigen für Christofides

Metrisches TSP wie zuvor. *Hinweis* ($\Delta TSP2$, Christofides): (1) MST T ; (2) X = Knoten mit ungeradem Grad in T ; (3) minimales perfektes Matching M auf X ; (4) Eulerkreis im Multigraphen $T + M$; (5) Abkürzen. Zeigen Sie die Güte $\frac{3}{2}$.

Lösung.

Zu zeigen:

$$d(R) \leq \frac{3}{2} \text{OPT}(I).$$

Schritt 1 ($w(T) \leq \text{OPT}$). Wie bei $\Delta TSP1$: Entfernt man aus einer optimalen Rundreise eine Kante, entsteht ein Spannbaum; sein Gewicht ist $\leq \text{OPT}(I)$, und da T ein minimaler Spannbaum ist, gilt $w(T) \leq \text{OPT}(I)$.

Schritt 2 ($w(M) \leq \text{OPT}/2$). Die Zahl $|X|$ der ungerad-gradigen Knoten ist gerade. Kürzt man eine optimale Rundreise auf die Knoten aus X ab (alle übrigen überspringen), so entsteht nach der Dreiecksungleichung ein Kreis C auf X mit $d(C) \leq \text{OPT}(I)$. Ein Kreis über die gerade Knotenzahl $|X|$ zerfällt in zwei disjunkte perfekte Matchings M_1, M_2 (die abwechselnden Kanten). Wegen $w(M_1) + w(M_2) = d(C) \leq \text{OPT}(I)$ ist das billigere höchstens $\text{OPT}(I)/2$, und das minimale perfekte Matching erfüllt

$$w(M) \leq \min\{w(M_1), w(M_2)\} \leq \frac{1}{2} \text{OPT}(I).$$

Schritt 3 (Eulerkreis und Abkürzen). In $T + M$ hat jeder Knoten geraden Grad, es existiert ein Eulerkreis der Länge $w(T) + w(M)$. Das Abkürzen verlängert wegen der Dreiecksungleichung nicht, also

$$d(R) \leq w(T) + w(M) \leq \text{OPT}(I) + \frac{1}{2} \text{OPT}(I) = \frac{3}{2} \text{OPT}(I).$$

Somit hat Christofides die Güte $\frac{3}{2}$.

51 Güte zeigen für 2ApproxVC

VERTEXCOVER: kleinste Knotenmenge C , die jede Kante abdeckt. *Hinweis:* 2ApproxVC durchläuft alle Kanten; sind beide Endpunkte noch nicht in C , werden *beide* zu C hinzugefügt. Zeigen Sie die Güte 2.

Lösung.

Zu zeigen:

$$|C| \leq 2|C^*|$$

für ein minimales Vertex Cover C^* . Sei A die Menge der Kanten, bei denen beide Endpunkte aufgenommen wurden; dann ist $|C| = 2|A|$.

A ist ein Matching. Hätten zwei Kanten aus A einen gemeinsamen Endpunkt, so wäre bei der später betrachteten dieser Endpunkt bereits in C gewesen – sie wäre nicht aufgenommen worden. Also sind die Kanten aus A paarweise knotendisjunkt.

Schranke für das Optimum. C^* deckt jede Kante aus A mit mindestens einem Endpunkt ab; da die Kanten aus A knotendisjunkt sind, braucht jede einen *eigenen* Knoten, also $|C^*| \geq |A|$.

Kombination.

$$|C| = 2|A| \leq 2|C^*|.$$

Somit hat 2ApproxVC die Güte 2.

52 Güte zeigen für MAX-3-SAT

MAX-3-SAT: Formel φ in KNF mit drei Literalen pro Klausel; gesucht eine Belegung, die die Anzahl $v(\cdot)$ der erfüllten Klauseln maximiert. *Hinweis:* Algorithmus A wertet β_0 (alle falsch) und β_1 (alle wahr) aus und gibt die bessere Belegung zurück. Zeigen Sie die Güte 2.

Lösung.

Zu zeigen:

$$v(A(\varphi)) \geq \frac{1}{2} v(\text{OPT}(\varphi)).$$

Sei φ eine Formel mit m Klauseln.

Jede Klausel zählt einmal. Unter β_0 ist jedes negative Literal wahr, unter β_1 jedes positive. Da jede Klausel mindestens ein Literal enthält und jedes Literal positiv oder negativ ist, wird jede Klausel von β_0 oder von β_1 erfüllt. Somit

$$v(\beta_0) + v(\beta_1) \geq m.$$

Obere Schranke für das Optimum. Es können höchstens alle m Klauseln erfüllt sein, also $v(\text{OPT}(\varphi)) \leq m$.

Kombination. A wählt die bessere der beiden Belegungen, daher

$$v(A(\varphi)) = \max\{v(\beta_0), v(\beta_1)\} \geq \frac{v(\beta_0) + v(\beta_1)}{2} \geq \frac{m}{2} \geq \frac{v(\text{OPT}(\varphi))}{2}.$$

Somit hat A die Güte 2.

53 Güte zeigen für ApproximateSubsetSum

APPROXIMATESUBSETSUM: Zahlen $a_1, \dots, a_n \leq T$ mit $\sum_i a_i > T$; maximiere $\sum_{j \in S} a_j$ unter $\sum_{j \in S} a_j \leq T$. *Hinweis:* GA sortiert absteigend, nimmt der Reihe nach jedes noch passende Element und *stoppt* beim ersten nicht mehr passenden. Zeigen Sie die Güte 2.

Lösung.

Zu zeigen:

$$\text{GA}(I) \geq \frac{1}{2} \text{OPT}(I).$$

Sei a_ℓ das erste Element, das nicht mehr passte; es existiert wegen $\sum_i a_i > T$.

Überlauf-Schranke. Zum Stoppzeitpunkt hätte a_ℓ die Kapazität überschritten:

$$\text{GA}(I) + a_\ell > T \geq \text{OPT}(I).$$

Größe des gestoppten Elements. Wegen der absteigenden Sortierung ist jedes bereits gewählte Element $\geq a_\ell$, also enthält die GA-Auswahl ein Element $a_j \geq a_\ell$ und damit $\text{GA}(I) \geq a_\ell$.

Kombination. Einsetzen liefert

$$2 \text{GA}(I) \geq \text{GA}(I) + a_\ell > T \geq \text{OPT}(I),$$

also $\text{GA}(I) \geq \text{OPT}(I)/2$. Somit hat GA die Güte 2.

54 Güte zeigen für Strip Packing: NFDH

STRIPPACKING: Rechtecke mit Höhen $h(r_i) \leq 1$, absteigend nach Höhe sortiert, in einen Streifen der Breite 1 packen und die Höhe minimieren. *Hinweis:* NFDH packt stufenweise von links; passt ein Rechteck nicht mehr, beginnt eine neue Stufe (deren Höhe die ihres ersten, also höchsten Rechtecks ist). Zeigen Sie $\text{NFDH}(L) \leq 2 \text{OPT}(L) + h_{\max}$.

Lösung.

Zu zeigen:

$$\text{NFDH}(L) \leq 2 \text{OPT}(L) + h_{\max}.$$

Sei H_i die Höhe der Stufe i (Höhe ihres ersten Rechtecks), W_i ihre belegte Gesamtbreite und A_i ihre Rechteck-Gesamtfläche.

Erstens (Fläche einer Stufe). Jedes Rechteck der Stufe i hat wegen der absteigenden Sortierung Höhe $\geq H_{i+1}$; die Rechtecke der Stufe i haben zusammen Breite W_i , also

$$A_i \geq W_i H_{i+1}.$$

Zweitens (das überlaufende Rechteck). Das erste Rechteck der Stufe $i+1$ passte nicht mehr auf Stufe i , hat also Breite $> 1 - W_i$ und Höhe H_{i+1} , mithin Fläche $> (1 - W_i) H_{i+1}$.

Kombination. Zerlege H_{i+1} und setze beide Schranken ein:

$$H_{i+1} = W_i H_{i+1} + (1 - W_i) H_{i+1} < A_i + (\text{Fläche des ersten Rechtecks von Stufe } i+1).$$

Summierung. Summation über $i \geq 1$; dabei zählt jede Stufe höchstens zweimal (einmal als A_i , einmal über ihr erstes Rechteck), und die Gesamtfläche passt in einen Streifen der Breite 1, also Gesamtfläche $\leq \text{OPT}(L)$:

$$\sum_{i \geq 2} H_i < 2 \cdot \text{Gesamtfläche} \leq 2 \text{OPT}(L).$$

Fazit. Die erste Stufe hat Höhe $H_1 = h_{\max}$. Damit

$$\text{NFDH}(L) = \sum_i H_i \leq 2 \text{OPT}(L) + h_{\max}.$$

Somit gilt die behauptete Schranke.

5 ETH

55 ETH-Schranke zeigen für VertexCover

Problem VertexCover:

Gegeben: Ein ungerichteter Graph $G = (V, E)$ und eine Zahl k .

Entscheide: Gibt es $S \subseteq V$ mit $|S| \leq k$, sodass $v \in S$ oder $u \in S$ für jede Kante $\{v, u\} \in E$?

Betrachten Sie die Reduktion $\text{CLIQUE} \leq_p \text{VERTEXCOVER}$. Aus $(G = (V, E), k)$ wird die Instanz $(G' = (V, \bar{E}), k')$ mit:

- $\bar{E} = \{\{v, u\} \mid v \neq u, \{v, u\} \notin E\}$
- $k' = |V| - k$

- Geben Sie $|V'|$, $|E'|$ und k' der VC-Instanz in Abhängigkeit der Clique-Instanz an.
- Beweisen Sie die ETH-Schranken für VERTEXCOVER bzgl. $|V'|$, $|E'|$ und k' basierend auf der Reduktion.

Hinweis: Unter der ETH ist CLIQUE nicht in $2^{o(|V|)} \cdot |I|^{O(1)}$ lösbar.

Lösung.

a) $|V'| = |V|$, $|E'| = \frac{1}{2}|V|(|V| - 1) - |E| \leq |V|^2$, $k' = |V| - k \leq |V|$.

b)

Bzgl. $|V'|$:

- Angenommen, VERTEXCOVER wäre in $2^{o(|V'|)} \cdot |I|^{O(1)}$ lösbar.
- Wegen $|V'| = |V|$ wird daraus ein $2^{o(|V|)} \cdot |I|^{O(1)}$ -Algorithmus für CLIQUE.
- Widerspruch zur ETH.

Bzgl. $|E'|$:

- Angenommen, VERTEXCOVER wäre in $2^{o(\sqrt{|E'|})} \cdot |I|^{O(1)}$ lösbar.
- Wegen $|E'| \leq |V|^2$ ist $\sqrt{|E'|} \leq |V|$, also $\sqrt{|E'|} = O(|V|)$.
- Daraus wird ein $2^{o(|V|)}$ -Algorithmus für CLIQUE.
- Widerspruch zur ETH.

Bzgl. k' :

- Angenommen, VERTEXCOVER wäre in $2^{o(k')} \cdot |I|^{O(1)}$ lösbar.
- Wegen $k' \leq |V|$ wird daraus ein $2^{o(|V|)}$ -Algorithmus für CLIQUE.
- Widerspruch zur ETH.

Bounds: kein $2^{o(|V'|)}$, kein $2^{o(\sqrt{|E'|})}$, kein $2^{o(k')}$. □

56 ETH-Schranke zeigen für HittingSet

Problem HittingSet:

Gegeben: Eine Menge U , Teilmengen $F_1, \dots, F_r \subseteq U$ und eine Zahl k .

Entscheide: Gibt es $S \subseteq U$ mit $|S| \leq k$ und $S \cap F_i \neq \emptyset$ für alle $i \in [r]$?

Betrachten Sie die Reduktion $3\text{-SAT} \leq_p \text{HITTINGSET}$. Aus einer 3-SAT-Instanz mit Variablen $x_1, \dots, x_n$ und Klauseln $C_1, \dots, C_m$ wird:

- $U = \{x_i, \bar{x}_i \mid i \in [n]\}$
- $F_i = \{x_i, \bar{x}_i\}$ für jede Variable $i \in [n]$
- $F_{n+j} = C_j$ für jede Klausel $j \in [m]$
- $k = n$

- a) Geben Sie $|U|$, die Mengenzahl r und k in Abhängigkeit von n und m an.
- b) Beweisen Sie die ETH-Schranken für HITTINGSET bzgl. $|U|$, r und k basierend auf der Reduktion.

Hinweis: Unter der ETH ist 3-SAT nicht in $2^{o(n)} \cdot |I|^{O(1)}$ lösbar, mit dem Sparsification-Lemma auch nicht in $2^{o(m)} \cdot |I|^{O(1)}$.

Lösung.

a) $|U| = 2n$, $r = n + m$, $k = n$.

b)

Bzgl. $|U|$:

- Angenommen, HITTINGSET wäre in $2^{o(|U|)} \cdot |I|^{O(1)}$ lösbar.
- Wegen $|U| = 2n$ wird daraus ein $2^{o(n)}$ -Algorithmus für 3-SAT.
- Direkter Widerspruch zur ETH.

Bzgl. r :

- Angenommen, HITTINGSET wäre in $2^{o(r)} \cdot |I|^{O(1)}$ lösbar.
- Wegen $r = n + m$ und $n \leq 3m$ ist $r = O(m)$.
- Daraus wird ein $2^{o(m)}$ -Algorithmus für 3-SAT.
- Widerspruch zur ETH nach dem Sparsification-Lemma.

Bzgl. k :

- Angenommen, HITTINGSET wäre in $2^{o(k)} \cdot |I|^{O(1)}$ lösbar.
- Wegen $k = n$ wird daraus ein $2^{o(n)}$ -Algorithmus für 3-SAT.
- Direkter Widerspruch zur ETH.

Bounds: kein $2^{o(|U|)}$, kein $2^{o(r)}$, kein $2^{o(k)}$.

□

57 ETH-Schranke zeigen für k -Clique

Problem k -Clique:

Gegeben: Ein ungerichteter Graph $G = (V, E)$ und eine Zahl $k \geq 1$.

Entscheide: Gibt es eine Clique $C \subseteq V$ mit $|C| \geq k$, d.h. $\{v, u\} \in E$ für alle $v, u \in C$ mit $v \neq u$?

Betrachten Sie die Reduktion $3\text{-SAT} \leq_p k\text{-CLIQUE}$. Aus einer 3-SAT-Instanz mit Klauseln $F_1, \dots, F_m$ wird (y_{ij} sei das j -te Literal der Klausel i):

- $V = \{[i, j] \mid j\text{-tes Literal in Klausel } i\}$
- $E = \{\{[i, j], [i', j']\} \mid i \neq i', y_{ij} \neq \neg y_{i'j'}\}$
- $k = m$

- Geben Sie $|V|$, $|E|$ und k der Clique-Instanz in Abhängigkeit von n und m an.
- Beweisen Sie die ETH-Schranken für $k\text{-CLIQUE}$ bzgl. $|V|$, $|E|$ und k basierend auf der Reduktion.

Hinweis: Mit dem Sparsification-Lemma ist 3-SAT unter der ETH nicht in $2^{o(m)} \cdot |I|^{O(1)}$ lösbar.

Lösung.

a) $|V| = O(m)$ (je Klausel ≤ 3 Literale), $|E| = O(m^2)$, $k = m$.

b)

Bzgl. $|V|$:

- Angenommen, $k\text{-CLIQUE}$ wäre in $2^{o(|V|)} \cdot |I|^{O(1)}$ lösbar.
- Wegen $|V| = O(m)$ wird daraus ein $2^{o(m)}$ -Algorithmus für 3-SAT.
- Widerspruch zur ETH nach dem Sparsification-Lemma.

Bzgl. $|E|$:

- Angenommen, $k\text{-CLIQUE}$ wäre in $2^{o(\sqrt{|E|})} \cdot |I|^{O(1)}$ lösbar.
- Wegen $|E| = O(m^2)$ ist $\sqrt{|E|} = O(m)$.
- Daraus wird ein $2^{o(m)}$ -Algorithmus für 3-SAT.
- Widerspruch zur ETH nach dem Sparsification-Lemma.

Bzgl. k :

- Angenommen, $k\text{-CLIQUE}$ wäre in $2^{o(k)} \cdot |I|^{O(1)}$ lösbar.
- Wegen $k = m$ wird daraus ein $2^{o(m)}$ -Algorithmus für 3-SAT.
- Widerspruch zur ETH nach dem Sparsification-Lemma.

Bounds: kein $2^{o(|V|)}$, kein $2^{o(\sqrt{|E|})}$, kein $2^{o(k)}$. □

58 ETH-Schranke zeigen für k -IndependentSet

Problem k -IS:

Gegeben: Ein ungerichteter Graph $G = (V, E)$ und eine Zahl k .

Entscheide: Gibt es $S \subseteq V$ mit $|S| \geq k$, sodass $\{v, u\} \notin E$ für alle $v, u \in S$ mit $v \neq u$?

Betrachten Sie die Reduktion $k\text{-CLIQUE} \leq_p k\text{-IS}$. Aus $(G = (V, E), k)$ wird die Instanz $(G' = (V, \bar{E}), k')$ mit:

- $\bar{E} = \{\{v, u\} \mid v \neq u, \{v, u\} \notin E\}$
- $k' = k$

- a) Geben Sie $|V'|$, $|E'|$ und k' der IS-Instanz in Abhängigkeit der Clique-Instanz an.
- b) Beweisen Sie die ETH-Schranken für k -IS bzgl. $|V'|$, $|E'|$ und k' basierend auf der Reduktion.

Hinweis: Unter der ETH ist $k\text{-CLIQUE}$ weder in $2^{o(|V|)} \cdot |I|^{O(1)}$ noch in $2^{o(\sqrt{|E|})} \cdot |I|^{O(1)}$ noch in $2^{o(k)} \cdot |I|^{O(1)}$ lösbar.

Lösung.

a) $|V'| = |V|$, $|E'| = \frac{1}{2}|V|(|V| - 1) - |E| \leq |V|^2$, $k' = k$.

b)

Bzgl. $|V'|$:

- Angenommen, k -IS wäre in $2^{o(|V'|)} \cdot |I|^{O(1)}$ lösbar.
- Ein unabhängiges Set in $\bar{G}$ ist eine Clique in G ; via Komplementbildung löst der Algorithmus CLIQUE .
- Wegen $|V'| = |V|$ wird daraus ein $2^{o(|V|)}$ -Algorithmus für $k\text{-CLIQUE}$.
- Widerspruch zum Hinweis.

Bzgl. $|E'|$:

- Angenommen, k -IS wäre in $2^{o(\sqrt{|E'|})} \cdot |I|^{O(1)}$ lösbar.
- Wegen $|E'| \leq |V|^2$ ist $\sqrt{|E'|} = O(|V|)$.
- Via Komplementbildung wird daraus ein $2^{o(|V|)}$ -Algorithmus für $k\text{-CLIQUE}$.
- Widerspruch zum Hinweis.

Bzgl. k' :

- Angenommen, k -IS wäre in $2^{o(k')} \cdot |I|^{O(1)}$ lösbar.
- Wegen $k' = k$ wird daraus ein $2^{o(k)}$ -Algorithmus für $k\text{-CLIQUE}$.
- Widerspruch zum Hinweis.

Bounds: kein $2^{o(|V'|)}$, kein $2^{o(\sqrt{|E'|})}$, kein $2^{o(k')}$. □

59 ETH-Schranke zeigen für 3-DM

Problem 3-DM:

Gegeben: Mengen U, V, W mit $|U| = |V| = |W|$ und eine Tripelmengende $T \subseteq U \times V \times W$.

Entscheide: Gibt es $M \subseteq T$ mit $|M| = |U|$, sodass je zwei verschiedene Tripel aus M in allen drei Koordinaten verschieden sind?

Betrachten Sie die Reduktion $\text{SAT} \leq_p \text{3-DM}$ (Vorlesung), angewandt auf 3-SAT-Instanzen: Sie erzeugt drei gleich große Grundmengen mit $|U| = |V| = |W| = O(m^2)$ und eine Tripelmengende mit $|T| = O(m^4)$. Beweisen Sie die ETH-Schranken für 3-DM bzgl. $|U|$ und $|T|$ basierend auf der Reduktion.

Hinweis: Mit dem Sparsification-Lemma ist 3-SAT unter der ETH nicht in $2^{o(m)} \cdot |I|^{O(1)}$ lösbar.

Lösung.

Bzgl. $|U|$:

- Angenommen, 3-DM wäre in $2^{o(\sqrt{|U|})} \cdot |I|^{O(1)}$ lösbar.
- Wegen $|U| = O(m^2)$ ist $\sqrt{|U|} = O(m)$.
- Daraus wird ein $2^{o(m)}$ -Algorithmus für 3-SAT.
- Widerspruch zur ETH nach dem Sparsification-Lemma.

Bzgl. $|T|$:

- Angenommen, 3-DM wäre in $2^{o(\sqrt[4]{|T|})} \cdot |I|^{O(1)}$ lösbar.
- Wegen $|T| = O(m^4)$ ist $\sqrt[4]{|T|} = O(m)$.
- Daraus wird ein $2^{o(m)}$ -Algorithmus für 3-SAT.
- Widerspruch zur ETH nach dem Sparsification-Lemma.

Bounds: kein $2^{o(\sqrt{|U|})}$, kein $2^{o(\sqrt[4]{|T|})}$. □

60 ETH-Schranke zeigen für 3-ExactCover

Problem 3-EC:

Gegeben: Eine Grundmenge U und eine Familie $F = \{S_1, \dots, S_n\}$ von Teilmengen mit $|S_j| = 3$ für alle j .

Entscheide: Gibt es eine Teilfamilie $F' \subseteq F$, deren Mengen paarweise disjunkt sind und U überdecken ($\bigcup_{S \in F'} S = U$)?

Betrachten Sie die Reduktion $3\text{-DM} \leq_p 3\text{-EC}$: 3-DM ist ein Spezialfall von 3-EC. Eine 3-DM-Instanz mit Tripelmenge T wird direkt zur 3-EC-Instanz mit:

- Grundmenge $U =$ Vereinigung der drei 3-DM-Mengen
 - $F = \{\{v, w, u\} \mid (v, w, u) \in T\}$
- a) Geben Sie $|U|$ und die Zahl der Mengen $|F|$ der 3-EC-Instanz in Abhängigkeit der 3-DM-Instanz an.
- b) Beweisen Sie die ETH-Schranken für 3-EC bzgl. $|U|$ und $|F|$ basierend auf der Reduktion.

Hinweis: Unter der ETH ist 3-DM weder in $2^{o(\sqrt{|V|})} \cdot |I|^{O(1)}$ noch in $2^{o(\sqrt[4]{|T|})} \cdot |I|^{O(1)}$ lösbar.

Lösung.

a) $|U| = O(|V|)$, $|F| = |T|$.

b)

Bzgl. $|U|$:

- Angenommen, 3-EC wäre in $2^{o(\sqrt{|U|})} \cdot |I|^{O(1)}$ lösbar.
- Jede 3-DM-Instanz ist eine 3-EC-Instanz mit $|U| = O(|V|)$, also $\sqrt{|U|} = O(\sqrt{|V|})$.
- Daraus wird ein $2^{o(\sqrt{|V|})}$ -Algorithmus für 3-DM.
- Widerspruch zum Hinweis.

Bzgl. $|F|$:

- Angenommen, 3-EC wäre in $2^{o(\sqrt[4]{|F|})} \cdot |I|^{O(1)}$ lösbar.
- Wegen $|F| = |T|$ löst der Algorithmus jede 3-DM-Instanz.
- Daraus wird ein $2^{o(\sqrt[4]{|T|})}$ -Algorithmus für 3-DM.
- Widerspruch zum Hinweis.

Bounds: kein $2^{o(\sqrt{|U|})}$, kein $2^{o(\sqrt[4]{|F|})}$.

□

61 ETH-Schranke zeigen für SubsetSum

Problem SubsetSum:

Gegeben: Zahlen $c_1, \dots, c_n$ und ein Zielwert K .

Entscheide: Gibt es $S \subseteq \{1, \dots, n\}$ mit $\sum_{j \in S} c_j = K$?

Betrachten Sie die Reduktion $3\text{-EC} \leq_p \text{SUBSETSUM}$: Jede Menge $S_j \in F$ der 3-EC-Instanz liefert ein Item c_j (Bitvektor über U), also $n = |F|$ Items.

- a) Geben Sie die Itemzahl n der SubsetSum-Instanz in Abhängigkeit der 3-EC-Instanz an.
- b) Beweisen Sie die ETH-Schranke für SUBSETSUM bzgl. n basierend auf der Reduktion.

Hinweis: Unter der ETH ist 3-EC nicht in $2^{o(\sqrt[4]{|F|})} \cdot |I|^{O(1)}$ lösbar.

Lösung.

a) $n = |F|$.

b)

Bzgl. n :

- Angenommen, SUBSETSUM wäre in $2^{o(\sqrt[4]{n})} \cdot |I|^{O(1)}$ lösbar.
- Wegen $n = |F|$ wird daraus ein $2^{o(\sqrt[4]{|F|})}$ -Algorithmus für 3-EC.
- Widerspruch zum Hinweis.

Bound: kein $2^{o(\sqrt[4]{n})}$.

□

62 ETH-Schranke zeigen für k -Color

Problem k -Color:

Gegeben: Ein ungerichteter Graph $G = (V, E)$ und eine Zahl $k \geq 1$.

Entscheide: Gibt es eine Färbung $f : V \rightarrow [k]$ mit $f(v) \neq f(u)$ für alle $\{v, u\} \in E$?

Betrachten Sie die Reduktion $3\text{-SAT} \leq_p k\text{-COLOR}$ mit:

- $V = \{x_i, \bar{x}_i, v_i \mid i \in [n]\} \cup \{C_j \mid j \in [m]\} \cup \{z\}$
- E : die Kanten des Vorlesungs-Gadgets

- a) Geben Sie $|V|$ und $|E|$ der Color-Instanz in Abhängigkeit von n und m an.
- b) Beweisen Sie die ETH-Schranken für k -COLOR bzgl. $|V|$ und $|E|$ basierend auf der Reduktion.

Hinweis: Mit dem Sparsification-Lemma ist 3-SAT unter der ETH nicht in $2^{o(m)} \cdot |I|^{O(1)}$ lösbar.

Lösung.

a) $|V| = 3n + m + 1$; wegen $n \leq 3m$ also $|V| = O(m)$. $|E| = O(n^2 + nm + m) = O(m^2)$.

b)

Bzgl. $|V|$:

- Angenommen, k -COLOR wäre in $2^{o(|V|)} \cdot |I|^{O(1)}$ lösbar.
- Wegen $|V| = O(m)$ wird daraus ein $2^{o(m)}$ -Algorithmus für 3-SAT.
- Widerspruch zur ETH nach dem Sparsification-Lemma.

Bzgl. $|E|$:

- Angenommen, k -COLOR wäre in $2^{o(\sqrt{|E|})} \cdot |I|^{O(1)}$ lösbar.
- Wegen $|E| = O(m^2)$ ist $\sqrt{|E|} = O(m)$.
- Daraus wird ein $2^{o(m)}$ -Algorithmus für 3-SAT.
- Widerspruch zur ETH nach dem Sparsification-Lemma.

Bounds: kein $2^{o(|V|)}$, kein $2^{o(\sqrt{|E|})}$.

□

63 ETH-Schranke zeigen für CoverClique

Problem CoverClique:

Gegeben: Ein ungerichteter Graph $G = (V, E)$ und eine Zahl $k \in \mathbb{N}_{\geq 1}$.

Entscheide: Existieren k paarweise disjunkte Cliques $C_1, \dots, C_k \subseteq V$, die alle Knoten überdecken?

Betrachten Sie die Reduktion $k\text{-COLOR} \leq_p \text{COVERCLIQUE}$. Aus $(G = (V, E), k)$ wird die Instanz $(G' = (V, \bar{E}), k)$ mit $\bar{E} = \{\{v, u\} \mid v \neq u, \{v, u\} \notin E\}$.

- Geben Sie $|V'|$ und $|\bar{E}|$ der CoverClique-Instanz in Abhängigkeit der Color-Instanz an.
- Beweisen Sie die ETH-Schranken für COVERCLIQUE bzgl. $|V'|$ und $|\bar{E}|$ basierend auf der Reduktion.

Hinweis: Unter der ETH ist $k\text{-COLOR}$ nicht in $2^{o(|V|)} \cdot |I|^{O(1)}$ lösbar.

Lösung.

a) $|V'| = |V|$, $|\bar{E}| = \frac{1}{2}|V|(|V| - 1) - |E| \leq |V|^2$.

b)

Bzgl. $|V'|$:

- Angenommen, COVERCLIQUE wäre in $2^{o(|V'|)} \cdot |I|^{O(1)}$ lösbar.
- Wegen $|V'| = |V|$ wird daraus ein $2^{o(|V|)}$ -Algorithmus für $k\text{-COLOR}$.
- Widerspruch zum Hinweis.

Bzgl. $|\bar{E}|$:

- Angenommen, COVERCLIQUE wäre in $2^{o(\sqrt{|\bar{E}|})} \cdot |I|^{O(1)}$ lösbar.
- Wegen $|\bar{E}| \leq |V|^2$ ist $\sqrt{|\bar{E}|} = O(|V|)$.
- Daraus wird ein $2^{o(|V|)}$ -Algorithmus für $k\text{-COLOR}$.
- Widerspruch zum Hinweis.

Bounds: kein $2^{o(|V'|)}$, kein $2^{o(\sqrt{|\bar{E}|})}$.

□

64 ETH-Schranke zeigen für ΔCover

Problem ΔCover :

Gegeben: Ein ungerichteter Graph $G = (V, E)$ und $k \in \mathbb{N}_0$.

Entscheide: Gibt es $C_\Delta \subseteq V$ mit $|C_\Delta| \leq k$, sodass $\{v, u, w\} \cap C_\Delta \neq \emptyset$ für jedes Dreieck $\{v, u, w\}$ (also $\{v, u\}, \{v, w\}, \{u, w\} \in E$) gilt?

Reduktion $\text{VERTEXCOVER} \leq_p \Delta\text{COVER}$: Aus $(G = (V, E), k)$ wird $(G' = (V', E'), k)$ mit

- $V' = V \cup \{v_e \mid e \in E\}$,
- $E' = E \cup \{\{v_e, e_1\}, \{v_e, e_2\} \mid e = \{e_1, e_2\} \in E\}$ (je Kante ein aufgesetztes Dreieck),
- k unverändert.

- a) Geben Sie $|V'|$ und $|E'|$ der ΔCover -Instanz in Abhängigkeit der VC-Instanz an.
- b) Beweisen Sie die ETH-Schranken für ΔCOVER bzgl. $|V'|$ und $|E'|$ basierend auf der Reduktion.

Hinweis: Unter der ETH ist VERTEXCOVER weder in $2^{o(|V|)} \cdot |I|^{O(1)}$ noch in $2^{o(\sqrt{|E|})} \cdot |I|^{O(1)}$ lösbar.

Lösung.

a) $|V'| = |V| + |E|, \quad |E'| = |E| + 2|E| = 3|E|.$

b)

Bzgl. $|V'|$:

- Angenommen, ΔCOVER wäre in $2^{o(\sqrt{|V'|})} \cdot |I|^{O(1)}$ lösbar.
- Wegen $|V'| = |V| + |E| \leq |V| + |V|^2 = O(|V|^2)$ ist $\sqrt{|V'|} = O(|V|)$.
- Daraus wird ein $2^{o(|V|)}$ -Algorithmus für VERTEXCOVER .
- Widerspruch zum Hinweis.

Bzgl. $|E'|$:

- Angenommen, ΔCOVER wäre in $2^{o(\sqrt{|E'|})} \cdot |I|^{O(1)}$ lösbar.
- Wegen $|E'| = 3|E| = O(|E|)$ ist $\sqrt{|E'|} = O(\sqrt{|E|})$.
- Daraus wird ein $2^{o(\sqrt{|E|})}$ -Algorithmus für VERTEXCOVER .
- Widerspruch zum Hinweis.

Bounds: kein $2^{o(\sqrt{|V'|})}$, kein $2^{o(\sqrt{|E'|})}$.

□

65 ETH-Schranke zeigen für NAE-4-SAT

Problem NAE-4-SAT:

Gegeben: Variablen $x_1, \dots, x_n$ und Klauseln $C_1, \dots, C_m$ mit $|C_i| \leq 4$ für alle $i \in [m]$.

Entscheide: Gibt es eine Belegung φ , sodass jede Klausel C_i zwei Literale $y, z \in C_i$ mit $\varphi(y) \neq \varphi(z)$ enthält?

Reduktion $3\text{-SAT} \leq_p \text{NAE-4-SAT}$: Aus Variablen $x_1, \dots, x_n$ und Klauseln $C_1, \dots, C_m$ wird

- die Variablenmenge $\{x_1, \dots, x_n, w\}$ mit frischer Variable w ,
 - pro Klausel C_i die NAE-Klausel $C_i \cup \{w\}$.
- a) Geben Sie die Variablenzahl n' und die Klauselzahl m' der NAE-4-SAT-Instanz in Abhängigkeit von n und m an.
- b) Beweisen Sie die ETH-Schranken für NAE-4-SAT bzgl. n' und m' basierend auf der Reduktion.

Hinweis: Unter der ETH ist 3-SAT nicht in $2^{o(n)} \cdot |I|^{O(1)}$ lösbar, mit dem Sparsification-Lemma auch nicht in $2^{o(m)} \cdot |I|^{O(1)}$.

Lösung.

a) $n' = n + 1, \quad m' = m.$

b)

Bzgl. n' :

- Angenommen, NAE-4-SAT wäre in $2^{o(n')} \cdot |I|^{O(1)}$ lösbar.
- Wegen $n' = n + 1$ wird daraus ein $2^{o(n)}$ -Algorithmus für 3-SAT.
- Direkter Widerspruch zur ETH.

Bzgl. m' :

- Angenommen, NAE-4-SAT wäre in $2^{o(m')} \cdot |I|^{O(1)}$ lösbar.
- Wegen $m' = m$ wird daraus ein $2^{o(m)}$ -Algorithmus für 3-SAT.
- Widerspruch zur ETH nach dem Sparsification-Lemma.

Bounds: kein $2^{o(n')}$, kein $2^{o(m')}$.

□

66 ETH-Schranke zeigen für SetSplitting

Problem SetSplitting:

Gegeben: Eine Grundmenge S und Teilmengen $F_1, \dots, F_r \subseteq S$.

Entscheide: Gibt es eine Partition $S = S_1 \dot{\cup} S_2$ mit $F_i \cap S_1 \neq \emptyset \neq F_i \cap S_2$ für alle $i \in [r]$?

Reduktion $\text{NAE-4-SAT} \leq_p \text{SETSPLITTING}$: Aus n Variablen $x_1, \dots, x_n$ und m Klauseln $C_1, \dots, C_m$ wird

- die Grundmenge $S = \{x_i, \bar{x}_i \mid i \in [n]\}$,
- pro Variable $i \in [n]$ die Teilmenge $\{x_i, \bar{x}_i\} \subseteq S$,
- pro Klausel C_i die Teilmenge $C_i \subseteq S$.

- a) Geben Sie $|S|$ und die Mengenzahl r der SetSplitting-Instanz in Abhängigkeit von n und m an.
- b) Beweisen Sie die ETH-Schranken für SETSPLITTING bzgl. $|S|$ und r basierend auf der Reduktion.

Hinweis: NAE-4-SAT ist unter der ETH weder in $2^{o(n)} \cdot |I|^{O(1)}$ noch in $2^{o(m)} \cdot |I|^{O(1)}$ lösbar.

Lösung.

a) $|S| = 2n, \quad r = n + m.$

b)

Bzgl. $|S|$:

- Angenommen, SETSPLITTING wäre in $2^{o(|S|)} \cdot |I|^{O(1)}$ lösbar.
- Wegen $|S| = 2n$ wird daraus ein $2^{o(n)}$ -Algorithmus für NAE-4-SAT.
- Widerspruch zum Hinweis.

Bzgl. r :

- Angenommen, SETSPLITTING wäre in $2^{o(r)} \cdot |I|^{O(1)}$ lösbar.
- Wegen ≤ 4 Literalen je Klausel ist $n \leq 4m + 1$, also $r = n + m = O(m)$.
- Daraus wird ein $2^{o(m)}$ -Algorithmus für NAE-4-SAT.
- Widerspruch zum Hinweis.

Bounds: kein $2^{o(|S|)}$, kein $2^{o(r)}$. □

67 ETH-Schranke zeigen für DominatingSet

Problem DominatingSet:

Gegeben: Ein ungerichteter Graph $G = (V, E)$ und $k \in \mathbb{N}_{\geq 1}$.

Entscheide: Gibt es eine Menge $M \subseteq V$ mit $|M| \leq k$, sodass jeder Knoten $v \in V$ in M liegt oder zu einem Knoten in M benachbart ist?

Reduktion $3\text{-SAT} \leq_p \text{DOMINATINGSET}$: Aus Variablen $x_1, \dots, x_n$ und Klauseln $C_1, \dots, C_m$ wird

- pro Variable x_i die Knoten $x_i, \bar{x}_i, d_i$ mit den Dreieckskanten $\{x_i, \bar{x}_i\}, \{x_i, d_i\}, \{d_i, \bar{x}_i\}$,
- pro Klausel C_j ein Knoten C_j mit Kanten $\{C_j, \ell\}$ zu jedem Literal $\ell \in C_j$.

- a) Geben Sie $|V|$ und $|E|$ der DominatingSet-Instanz in Abhängigkeit von n und m an.
- b) Beweisen Sie die ETH-Schranken für DOMINATINGSET bzgl. $|V|$ und $|E|$ basierend auf der Reduktion.

Hinweis: Mit dem Sparsification-Lemma ist 3-SAT unter der ETH nicht in $2^{o(m)} \cdot |I|^{O(1)}$ lösbar.

Lösung.

a) $|V| = 3n + m$; wegen $n \leq 3m$ also $|V| = O(m)$. $|E| = 3n + 3m = O(m)$ (linear in m).

b)

Bzgl. $|V|$:

- Angenommen, DOMINATINGSET wäre in $2^{o(|V|)} \cdot |I|^{O(1)}$ lösbar.
- Wegen $|V| = O(m)$ wird daraus ein $2^{o(m)}$ -Algorithmus für 3-SAT.
- Widerspruch zur ETH nach dem Sparsification-Lemma.

Bzgl. $|E|$:

- Angenommen, DOMINATINGSET wäre in $2^{o(|E|)} \cdot |I|^{O(1)}$ lösbar (ohne Wurzel, da $|E|$ nur linear wächst).
- Wegen $|E| = O(m)$ wird daraus ein $2^{o(m)}$ -Algorithmus für 3-SAT.
- Widerspruch zur ETH nach dem Sparsification-Lemma.

Bounds: kein $2^{o(|V|)}$, kein $2^{o(|E|)}$ (ohne Wurzel). □

68 ETH-Schranke zeigen für GridTiling

Problem GridTiling:

Gegeben: Zahlen $k, N \in \mathbb{N}$ und je $(i, j) \in [k]^2$ eine Menge $S_{i,j} \subseteq [N]^2$.

Entscheide: Gibt es Tupel $(e_{i,j})_{(i,j) \in [k]^2}$ mit $e_{i,j} \in S_{i,j}$, sodass für $e_{i,j} = (a, b)$, $e_{i,j+1} = (a', b')$ stets $a = a'$ und für $e_{i,j} = (a, b)$, $e_{i+1,j} = (a', b')$ stets $b = b'$ gilt?

Reduktion $\text{CLIQUE} \leq_p \text{GRIDTILING}$ mit $N := |V|$ und $c \cdot N \leq k \leq N$: Aus $(G = (V, E), k)$ mit $V = \{1, \dots, N\}$ ohne isolierte Knoten wird

- $S_{i,i} = \{(a, a) \mid a \in V\}$ (Diagonale),
- $S_{i,j} = \{(a, b) \mid a \neq b, \{a, b\} \in E\}$ für $i \neq j$,
- k unverändert.

- a) Geben Sie $X = \sum_{(i,j)} |S_{i,j}|$ und $Y = \max_{(i,j)} |S_{i,j}|$ in Abhängigkeit von N an.
- b) Beweisen Sie die ETH-Schranken für GRIDTILING bzgl. X und Y basierend auf der Reduktion.

Hinweis: Unter der ETH ist CLIQUE nicht in $2^{o(N)} \cdot |I|^{O(1)}$ lösbar.

Lösung.

a) $|S_{i,i}| = N$, $|S_{i,j}| = 2|E| \leq N^2$ und $k = \Theta(N)$, also $X \leq k \cdot N + k^2 \cdot N^2 = O(N^4)$ und $Y \leq \max(N, N^2) = O(N^2)$.

b)

Bzgl. X :

- Angenommen, GRIDTILING wäre in $2^{o(\sqrt[4]{X})} \cdot |I|^{O(1)}$ lösbar.
- Wegen $X = O(N^4)$ ist $\sqrt[4]{X} = O(N)$.
- Daraus wird ein $2^{o(N)}$ -Algorithmus für CLIQUE.
- Widerspruch zum Hinweis.

Bzgl. Y :

- Angenommen, GRIDTILING wäre in $2^{o(\sqrt{Y})} \cdot |I|^{O(1)}$ lösbar.
- Wegen $Y = O(N^2)$ ist $\sqrt{Y} = O(N)$.
- Daraus wird ein $2^{o(N)}$ -Algorithmus für CLIQUE.
- Widerspruch zum Hinweis.

Bounds: kein $2^{o(\sqrt[4]{X})}$, kein $2^{o(\sqrt{Y})}$.

□

69 ETH-Schranke zeigen für SetCover

Problem SetCover:

Gegeben: Eine Menge U , Teilmengen $F_1, \dots, F_r \subseteq U$ und $k \in \mathbb{N}$.

Entscheide: Gibt es $S \subseteq \{1, \dots, r\}$ mit $|S| \leq k$ und $\bigcup_{i \in S} F_i = U$?

Reduktion $\text{SETCOVER} \leq_p \text{HITTINGSET}$ (Rollentausch, involutorisch): Aus $(U, F_1, \dots, F_r, k)$ wird

- die Grundmenge $U' = \{1, \dots, r\}$ (ein Element je Menge F_i),
 - pro $w \in U$ die Menge $F'_w = \{v \mid w \in F_v\}$,
 - $k' = k$.
- a) Geben Sie die Mengenzahl r' und die Elementzahl $|U'|$ der erzeugten HittingSet-Instanz in Abhängigkeit der SetCover-Instanz an.
- b) Beweisen Sie die ETH-Schranken für SETCOVER bzgl. $|U|$ und r basierend auf der Reduktion.

Hinweis: Unter der ETH ist HITTINGSET weder in $2^{o(r')} \cdot |I|^{O(1)}$ noch in $2^{o(|U'|)} \cdot |I|^{O(1)}$ lösbar.

Lösung.

a) $r' = |U|$, $|U'| = r$ (der Rollentausch ist involutorisch).

b)

Bzgl. $|U|$:

- Angenommen, SETCOVER wäre in $2^{o(|U|)} \cdot |I|^{O(1)}$ lösbar.
- Wegen $r' = |U|$ wird daraus über den Rollentausch ein $2^{o(r')}$ -Algorithmus für HITTINGSET.
- Widerspruch zum Hinweis.

Bzgl. r :

- Angenommen, SETCOVER wäre in $2^{o(r)} \cdot |I|^{O(1)}$ lösbar.
- Wegen $|U'| = r$ wird daraus ein $2^{o(|U'|)}$ -Algorithmus für HITTINGSET.
- Widerspruch zum Hinweis.

Bounds: kein $2^{o(|U|)}$, kein $2^{o(r)}$. □

70 ETH-Schranke zeigen für Scheduling $2 \mid \text{prec}, p_i \in \{1, 2\} \mid C_{\max}$

Problem $2 \mid \text{prec}, p_i \in \{1, 2\} \mid C_{\max}$:

Gegeben: Jobs $\mathcal{J}$ (je Job $p_J \in \{1, 2\}$) auf 2 Maschinen, Präzedenzen als DAG $G = (\mathcal{J}, A)$ und $T \in \mathbb{N}_0$.

Entscheide: Gibt es einen Schedule (Start- und Maschinenzuweisung je Job), sodass Jobs einer Maschine sich nicht überlappen, $(j, k) \in A$ den Start von k erst nach dem Ende von j erlaubt und der Makespan $\leq T$ ist?

Reduktion $k\text{-CLIQUE} \leq_p \text{Scheduling}$ (zusammenhängender Clique-Graph): Aus $(G = (V, E), k)$ wird

- ein Knoten-Job J_i mit $p_i = 1$ je $i \in V$,
 - ein Kanten-Job $J_{\{i,j\}}$ mit $p_{\{i,j\}} = 2$ je $\{i, j\} \in E$,
 - $3|V| + 2|E|$ Dummy-Jobs mit $p = 1$, also $n = 4|V| + 3|E|$ Jobs,
 - Präzedenzen $(J_i, J_{\{i,j\}})$ für alle $i \in V$, $\{i, j\} \in E$ (sowie innerhalb der Dummy-Jobs),
 - Makespan $T = 2|V| + 2|E|$.
- a) Geben Sie n und T der Scheduling-Instanz in Abhängigkeit der Clique-Instanz an.
- b) Beweisen Sie die ETH-Schranken für das Scheduling-Problem bzgl. n und T basierend auf der Reduktion.

Hinweis: Unter der ETH ist CLIQUE (zusammenhängend) weder in $2^{o(|V|)} \cdot |I|^{O(1)}$ noch in $2^{o(\sqrt{|E|})} \cdot |I|^{O(1)}$ lösbar.

Lösung.

a) Wegen $|V| \leq |E| + 1$: $n = 4|V| + 3|E| = O(|E|)$, $T = 2|V| + 2|E| = O(|E|)$.

b)

Bzgl. n :

- Angenommen, das Scheduling-Problem wäre in $2^{o(\sqrt{n})} \cdot |I|^{O(1)}$ lösbar.
- Wegen $n = O(|E|)$ ist $\sqrt{n} = O(\sqrt{|E|})$.
- Daraus wird ein $2^{o(\sqrt{|E|})}$ -Algorithmus für CLIQUE .
- Widerspruch zum Hinweis.

Bzgl. T :

- Angenommen, das Scheduling-Problem wäre in $2^{o(\sqrt{T})} \cdot |I|^{O(1)}$ lösbar.
- Wegen $T = O(|E|)$ ist $\sqrt{T} = O(\sqrt{|E|})$.
- Daraus wird ein $2^{o(\sqrt{|E|})}$ -Algorithmus für CLIQUE .
- Widerspruch zum Hinweis.

Bounds: kein $2^{o(\sqrt{n})}$, kein $2^{o(\sqrt{T})}$.

□

71 ETH-Schranke zeigen für ILP-Feasibility

Problem ILP-Feasibility:

Gegeben: Eine Matrix $A \in \mathbb{Z}^{M \times N}$ und eine rechte Seite $b \in \mathbb{Z}^M$.

Entscheide: Gibt es einen Vektor $x \in \mathbb{Z}_{\geq 0}^N$ mit $Ax \leq b$?

Reduktion 3-SAT $\leq_p$ ILP-FEASIBILITY: Aus Variablen $x_1, \dots, x_n$ und Klauseln $C_1, \dots, C_m$ wird

- $N = 2n$ ILP-Variablen, je eine x_ℓ pro Literal ℓ ,
 - pro Klausel C_i die Ungleichung $-\sum_{\ell \in C_i} x_\ell \leq -1$,
 - pro Variable v die Ungleichungen $x_v + x_{\bar{v}} \leq 1$ und $-x_v - x_{\bar{v}} \leq -1$.
- a) Geben Sie die Zeilenzahl M und die Spaltenzahl N der ILP-Instanz in Abhängigkeit von n und m an.
- b) Beweisen Sie die ETH-Schranken für ILP-FEASIBILITY bzgl. M und N basierend auf der Reduktion.

Hinweis: Unter der ETH ist 3-SAT nicht in $2^{o(n)} \cdot |I|^{O(1)}$ lösbar, mit dem Sparsification-Lemma auch nicht in $2^{o(m)} \cdot |I|^{O(1)}$.

Lösung.

a) $M = m + 2n$; wegen $n \leq 3m$ also $M = O(m)$. $N = 2n$.

b)

Bzgl. M :

- Angenommen, ILP-FEASIBILITY wäre in $2^{o(M)} \cdot |I|^{O(1)}$ lösbar.
- Wegen $M = O(m)$ wird daraus ein $2^{o(m)}$ -Algorithmus für 3-SAT.
- Widerspruch zur ETH nach dem Sparsification-Lemma.

Bzgl. N :

- Angenommen, ILP-FEASIBILITY wäre in $2^{o(N)} \cdot |I|^{O(1)}$ lösbar.
- Wegen $N = 2n$ wird daraus ein $2^{o(n)}$ -Algorithmus für 3-SAT.
- Direkter Widerspruch zur ETH.

Bounds: kein $2^{o(M)}$, kein $2^{o(N)}$. □