

Aufgabenkatalog

Alle Klausur-, Haus- und Präsenzaufgaben mit minimalen Lösungen
Analyse von Algorithmen und Komplexität – CAU Kiel

Inhalt

1	NP-Zugehörigkeit	2
2	Reduktionen: NP-Schwere und NP-Vollständigkeit	12
3	Vorlesungsbeweise	50
4	Wahr oder falsch?	64
5	ETH-Schranken	67
6	Approximation: anwenden, Güte beweisen, Worst Case	91
7	Turingmaschinen	106

1 NP-Zugehörigkeit

A1 Knapsack $\in$ NP

Präsenz 9.1

Gegeben: n Gegenstände mit Gewichten $w_1, \dots, w_n$ und Profiten $p_1, \dots, p_n$, Kapazität K , Zielprofit P . **Entscheide:** Gibt es $S \subseteq [n]$ mit $\sum_{i \in S} w_i \leq K$ und $\sum_{i \in S} p_i \geq P$?

Zeigen Sie $\text{KNAPSACK} \in \text{NP}$ auf zwei Wege: (1) nichtdeterministischer Polynomialzeit-Algorithmus, (2) polynomieller Verifizierer. Wie hängen beide zusammen?

Lösung.

(1) NTM: Für jeden Gegenstand wird nichtdeterministisch entschieden, ob er in den Rucksack kommt. Danach wird geprüft, ob $\sum_{i \in S} w_i \leq K$ und $\sum_{i \in S} p_i \geq P$ gilt; genau dann wird akzeptiert.

Existiert eine gültige Füllung mit genug Profit, gibt es eine Folge von Entscheidungen, die sie auswählt – diese wird akzeptiert; sonst lehnt jeder Rechenweg ab. n Rateschritte plus Summation und Vergleich in $O(n)$ – also polynomiell.

(2) Verifizierer: Zertifikat ist $S \subseteq [n]$. Berechne $\sum_{i \in S} w_i$ und $\sum_{i \in S} p_i$, vergleiche mit K und P , akzeptiere entsprechend.

Existiert eine gültige Füllung, beschreibt ein Zertifikat genau sie und wird akzeptiert; sonst wird jedes abgelehnt. Summen und Vergleiche in $O(n)$ – ein polynomieller Verifizierer.

Zusammenhang: Die Folge der Rateschritte der NTM *ist* das Zertifikat. Beide Wege bestimmen mit der Extra-Ressource einen Kandidaten und prüfen ihn dann deterministisch. $\square$

Beispiel A1 – Knapsack $\in$ NP

Instanz. Drei Gegenstände als (w_i, p_i) :

$$(2, 3), \quad (3, 4), \quad (5, 6); \quad K = 5, \quad P = 7.$$

Gültiges Zertifikat $S = \{1, 2\}$: Der Verifizierer rechnet

$$\sum_{i \in S} w_i = 2 + 3 = 5 \leq K, \quad \sum_{i \in S} p_i = 3 + 4 = 7 \geq P,$$

beide Bedingungen erfüllt – er *akzeptiert*.

Ungültige Zertifikate.

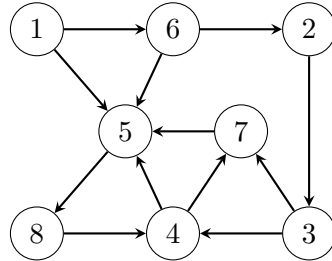
- $S = \{1, 2, 3\}$: Gewicht $2 + 3 + 5 = 10 > 5$ – Kapazität verletzt, verworfen.
- $S = \{3\}$: Profit $6 < 7$ – Zielprofit verfehlt, verworfen.

Verbindung zur NTM. Die NTM trifft pro Gegenstand eine Ja/Nein-Entscheidung; die Folge (ja, ja, nein) wählt genau $S = \{1, 2\}$ aus. Dieses Entscheidungsprotokoll *ist* das Zertifikat – ein akzeptierender Rechenweg der NTM und ein akzeptiertes Zertifikat des Verifizierers sind dasselbe Objekt.

Problem VertexCover: Gegeben ein Graph $G = (V, E)$ und eine Zahl $k \in \mathbb{N}_0$. Ein Vertex Cover ist eine Knotenmenge $C \subseteq V$, die von jeder Kante mindestens einen Endpunkt enthält. Gibt es ein Vertex Cover mit $|C| \leq k$?

Gegeben der folgende gerichtete Graph mit $V = \{1, \dots, 8\}$ und Kanten

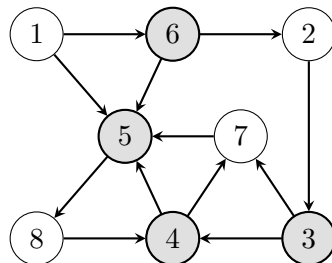
$1 \rightarrow 6, 6 \rightarrow 2, 2 \rightarrow 3, 1 \rightarrow 5, 6 \rightarrow 5, 7 \rightarrow 5, 5 \rightarrow 8, 8 \rightarrow 4, 4 \rightarrow 5, 3 \rightarrow 4, 3 \rightarrow 7, 4 \rightarrow 7$.



1. Finden Sie ein minimales Vertex Cover (Orientierung irrelevant).
2. Zeigen Sie VERTEXCOVER $\in$ NP durch einen nichtdeterministischen Polynomialzeit-Algorithmus (Korrektheit + Laufzeit).

Lösung.

1. Ein minimales Vertex Cover ist $C = \{3, 4, 5, 6\}$ (Größe 4) – jede Kante hat mindestens einen markierten Endpunkt. Kleiner geht es nicht: Ohne Knoten 5 decken drei Knoten höchstens $4 + 3 + 3 = 10 < 12$ Kanten; mit Knoten 5 bleiben 7 Kanten, die zwei weitere Knoten (je ≤ 3 davon) nicht decken können.



(Kantenliste aus der Text-Rekonstruktion des Original-Blatts; die Original-Abbildung hatte Optimum 5. Für die hier abgedruckte Instanz gilt 4.)

2. Der Algorithmus arbeitet wie folgt:

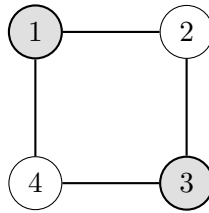
- Für jeden Knoten $v \in V$ wird nichtdeterministisch entschieden, ob er in C aufgenommen wird.
- Prüfe $|C| \leq k$; lehne sonst ab.
- Prüfe für jede Kante $\{u, v\} \in E$, ob $u \in C$ oder $v \in C$. Akzeptiere genau dann, wenn alle Kanten gedeckt sind.

Korrektheit: Existiert ein Vertex Cover der Größe $\leq k$, gibt es eine Folge von Entscheidungen, die genau diese Menge wählt – sie wird akzeptiert. Existiert keines, lehnt jeder Rechenweg ab.

Laufzeit: Auswahl der Knoten und Prüfung aller Kanten laufen in $O(|V| + |E|)$, also polynomiell. Damit VERTEXCOVER $\in$ NP. $\square$

Beispiel A2 – VertexCover $\in$ NP

Instanz. Viereck mit $V = \{1, 2, 3, 4\}$ und Kanten $\{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 1\}$; Schranke $k = 2$.



Gültiges Zertifikat $C = \{1, 3\}$ (grau markiert). Der Verifizierer prüft jede Kante auf einen Endpunkt in C :

$$\{1, 2\}: 1 \in C, \quad \{2, 3\}: 3 \in C, \quad \{3, 4\}: 3 \in C, \quad \{4, 1\}: 1 \in C.$$

Jede Kante gedeckt und $|C| = 2 \leq k$ – er *akzeptiert*.

Ungültige Zertifikate.

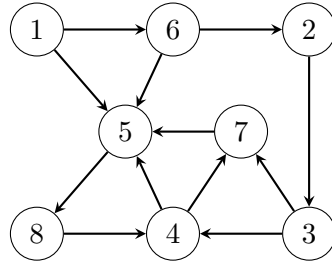
- $C = \{1, 2\}$: Kante $\{3, 4\}$ ungedeckt ($3 \notin C, 4 \notin C$) – verworfen.
- $C = \{2\}$: zwar $|C| \leq k$, aber $\{3, 4\}$ und $\{4, 1\}$ ungedeckt – verworfen.

Verbindung zur NTM. Die NTM rät pro Knoten „in C “ / „nicht in C “; die Folge (ja, nein, ja, nein) wählt genau $C = \{1, 3\}$ und liefert den akzeptierenden Rechenweg.

Problem FeedbackVertexSet: Gegeben ein gerichteter Graph $G = (V, E)$ und eine Zahl $k \in \mathbb{N}_0$. Gibt es eine Knotenmenge $X \subseteq V$ mit $|X| \leq k$, sodass $G \setminus X$ (der Graph nach Entfernen von X samt anliegender Kanten) kreisfrei ist?

Gegeben der folgende gerichtete Graph mit $V = \{1, \dots, 8\}$ und Kanten

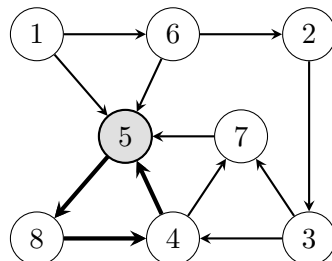
$1 \rightarrow 6, 6 \rightarrow 2, 2 \rightarrow 3, 1 \rightarrow 5, 6 \rightarrow 5, 7 \rightarrow 5, 5 \rightarrow 8, 8 \rightarrow 4, 4 \rightarrow 5, 3 \rightarrow 4, 3 \rightarrow 7, 4 \rightarrow 7$.



- Finden Sie ein minimales Feedback Vertex Set im gegebenen Graphen.
- Zeigen Sie $\text{FEEDBACKVERTEXSET} \in \text{NP}$ durch einen polynomiellen Verifizierer (Korrektheit + Laufzeit).

Lösung.

1. Ein minimales FVS ist $X = \{5\}$ (Größe 1). Der Graph enthält genau zwei Kreise: $5 \rightarrow 8 \rightarrow 4 \rightarrow 5$ (dick markiert) und $5 \rightarrow 8 \rightarrow 4 \rightarrow 7 \rightarrow 5$. Beide laufen durch 5 – ohne 5 bleibt kein Kreis übrig. (Auch $\{4\}$ oder $\{8\}$ ist möglich.)



(Kantenliste aus der Text-Rekonstruktion des Original-Blatts; die Original-Abbildung hatte Optimum 2. Für die hier abgedruckte Instanz gilt 1.)

2. Zertifikat ist eine Teilmenge $X \subseteq V$. Der Verifizierer:

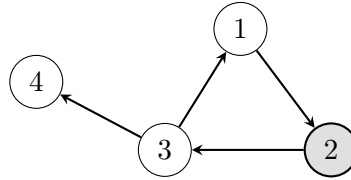
- Prüfe $|X| \leq k$; lehne sonst ab.
- Konstruiere $G' = G \setminus X$.
- Prüfe mit dem Algorithmus zur topologischen Sortierung aus der Vorlesung, ob G' kreisfrei ist: Er findet genau dann eine Sortierung, wenn kein Kreis existiert. Akzeptiere entsprechend.

Korrektheit: Existiert ein FVS der Größe $\leq k$, beschreibt ein Zertifikat genau dieses; G' ist dann kreisfrei und es wird akzeptiert. Sonst bleibt bei jedem Zertifikat ein Kreis und es wird abgelehnt.

Laufzeit: Größentest $O(|V|)$, Konstruktion und topologische Sortierung $O(|V| + |E|)$ – polynomiell. Damit $\text{FEEDBACKVERTEXSET} \in \text{NP}$. $\square$

Beispiel A3 – FeedbackVertexSet $\in$ NP

Instanz. Gerichteter Graph mit $V = \{1, 2, 3, 4\}$ und Kanten $1 \rightarrow 2$, $2 \rightarrow 3$, $3 \rightarrow 1$, $3 \rightarrow 4$; Schranke $k = 1$. Einziger Kreis: $1 \rightarrow 2 \rightarrow 3 \rightarrow 1$.



Gültiges Zertifikat $X = \{2\}$ (grau markiert). In $G \setminus X$ bleiben nur $3 \rightarrow 1$ und $3 \rightarrow 4$; die topologische Sortierung $(3, 1, 4)$ gelingt – $G \setminus X$ ist kreisfrei und $|X| = 1 \leq k$ – *akzeptiert*.

Ungültige Zertifikate.

- $X = \{4\}$: 4 liegt auf keinem Kreis, der Kreis $1 \rightarrow 2 \rightarrow 3 \rightarrow 1$ bleibt – die topologische Sortierung scheitert – verworfen.
- $X = \emptyset$: $|X| \leq k$, aber der Kreis bleibt – verworfen.

Problem Longest Path: Gegeben ein ungerichteter Graph $G = (V, E)$ und eine Zahl k . Gibt es einen *einfachen* Pfad in G (kein Knoten wird doppelt besucht) mit Länge $\geq k$?

Zeigen Sie auf verschiedene Weisen, dass das Problem in NP liegt:

- a) Geben Sie ein Zertifikat an.
- b) Beschreiben Sie einen Verifizierer; Laufzeit konkret.
- c) Beschreiben Sie eine NTM (in Worten); Laufzeit konkret.

Lösung.

- a) Eine Folge von $k + 1$ paarweise verschiedenen Knoten $(v_0, \dots, v_k)$ – der behauptete Pfad.
- b) Der Verifizierer prüft zwei Bedingungen:

- Alle v_i sind paarweise verschieden.
- $\{v_i, v_{i+1}\} \in E$ für alle $i < k$.

Akzeptiere genau dann, wenn beides gilt.

Korrektheit: Hat G einen einfachen Pfad der Länge $\geq k$, bilden dessen erste $k + 1$ Knoten ein akzeptiertes Zertifikat. Wird umgekehrt ein Zertifikat akzeptiert, ist es ein einfacher Pfad der Länge k .

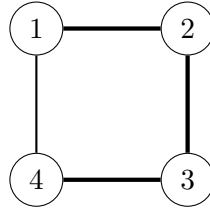
Laufzeit: Verschiedenheit mit $O(k^2)$ paarweisen Vergleichen; Kanten mit k Adjazenzmatrix-Lookups à $O(1)$. Gesamt $O(k^2) \subseteq O(|V|^2)$, polynomiell.

c) Rate nacheinander $k + 1$ Knoten auf ein Arbeitsband, prüfe dann deterministisch Verschiedenheit und Kanten wie in b).

Laufzeit: $O(k \log |V|)$ nichtdeterministische Rateschritte plus $O(|V|^2)$ fürs Prüfen – insgesamt polynomiell. $\square$

Beispiel A4 – Longest Path $\in$ NP

Instanz. Viereck mit $V = \{1, 2, 3, 4\}$ und Kanten $\{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 1\}$; Schranke $k = 3$ (gesucht: einfacher Pfad mit $k+1 = 4$ Knoten).



Gültiges Zertifikat $(1, 2, 3, 4)$ (fetter Pfad). Der Verifizierer prüft: alle vier Knoten paarweise verschieden, und die Kanten $\{1, 2\}, \{2, 3\}, \{3, 4\}$ existieren – *akzeptiert* (einfacher Pfad der Länge $3 \geq k$).

Ungültige Zertifikate.

- $(1, 2, 4, 3)$: Kante $\{2, 4\}$ existiert nicht – verworfen.
- $(1, 2, 2, 3)$: Knoten 2 doppelt (nicht paarweise verschieden) – verworfen.

Problem Dominating Set: Gegeben ein ungerichteter Graph $G = (V, E)$ und $k \in \mathbb{N}_0$. Gibt es $D \subseteq V$ mit $|D| \leq k$, sodass jeder Knoten in D liegt oder zu einem Knoten in D benachbart ist?

Zeigen Sie: (a) Zertifikat, (b) polynomieller Verifizierer mit konkreter Laufzeit, (c) polynomielle NTM mit konkreter Laufzeit.

Lösung.

a) Die Knotenmenge $D \subseteq V$.

b) Der Verifizierer prüft zwei Bedingungen:

- $|D| \leq k$.
- Für jeden Knoten $v \in V$: $v \in D$ oder ein Nachbar von v liegt in D – dazu wird die Zeile von v in der Adjazenzmatrix durchlaufen.

Akzeptiere genau dann, wenn beides gilt.

Korrektheit: Existiert ein Dominating Set der Größe $\leq k$, wird genau dieses akzeptiert. Sonst bleibt bei jedem Zertifikat ein Knoten undominiert und es wird abgelehnt.

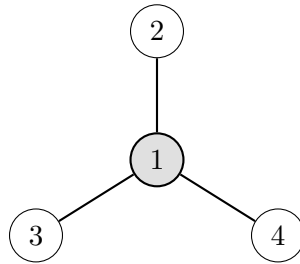
Laufzeit: Größentest $O(|V|)$; Dominanztest $O(|V|)$ pro Knoten. Gesamt $O(|V|^2)$, polynomiell.

c) Rate für jeden Knoten nichtdeterministisch, ob er in D liegt, prüfe dann deterministisch Größe und Dominanz wie in b).

Laufzeit: $O(|V|)$ Rateschritte plus $O(|V|^2)$ fürs Prüfen, also $O(|V|^2)$, polynomiell. $\square$

Beispiel A5 – Dominating Set $\in$ NP

Instanz. Stern mit Zentrum 1 und Blättern 2, 3, 4; Kanten $\{1, 2\}, \{1, 3\}, \{1, 4\}$; Schranke $k = 1$.



Gültiges Zertifikat $D = \{1\}$ (grau markiert). Der Verifizierer prüft jeden Knoten: $1 \in D$; die Blätter 2, 3, 4 sind je zu $1 \in D$ benachbart. Alle Knoten dominiert und $|D| = 1 \leq k$ – *akzeptiert*.

Ungültige Zertifikate.

- $D = \{2\}$: 2 dominiert nur sich selbst und 1; die Blätter 3 und 4 bleiben undominiert – verworfen.
- $D = \emptyset$: kein Knoten dominiert – verworfen.

2 Reduktionen: NP-Schwere und NP-Vollständigkeit

A6 VertexCover ist NP-vollständig

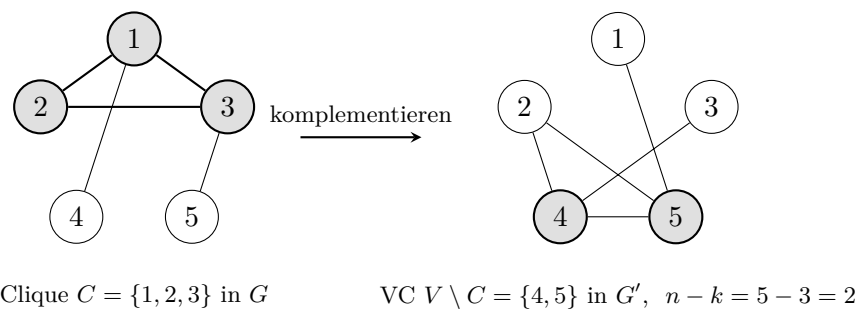
Präsenz 10.2

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

Lösung.

VERTEXCOVER $\in$ NP ist bekannt (Hausaufgabe 9.1).

Konstruktion: Sei $(G = (V, E), k)$ eine CLIQUE-Instanz mit $n = |V|$ (o.B.d.A. $k \leq n$). Invertiere G zum Komplementgraphen G' (genau die Nicht-Kanten von G). Gib $(G', n - k)$ aus.



Beweis Clique nach VertexCover: Sei C eine Clique in G mit $|C| \geq k$. Dann ist $V' := V \setminus C$ ein Vertex Cover von G' mit $|V'| \leq n - k$: In G' gibt es keine Kante innerhalb von C (in G waren alle da), also hat jede Kante von G' einen Endpunkt in V' .

Beweis VertexCover nach Clique: Sei V' ein Vertex Cover von G' mit $|V'| \leq n - k$. Dann ist $C := V \setminus V'$ eine Clique in G mit $|C| \geq k$: Für $\{u, v\} \subseteq C$ gilt $\{u, v\} \notin E'$ (sonst wäre V' kein VC), also $\{u, v\} \in E$.

Laufzeit: Invertieren naiv in $O(|V|^2)$, polynomiell.

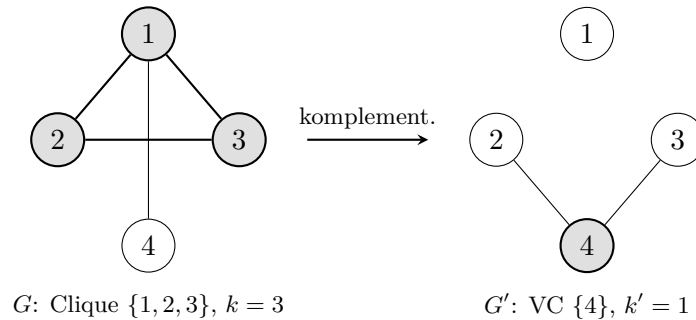
Da CLIQUE NP-vollständig ist und VERTEXCOVER $\in$ NP, ist VERTEXCOVER NP-vollständig.

□

Beispiel A6 – VertexCover NP-vollständig

Reduktion $\text{CLIQUE} \leq_p \text{VERTEXCOVER}$: komplementiere G zu G' und setze $k' = n - k$. Wir verfolgen eine winzige Instanz auf $V = \{1, 2, 3, 4\}$ mit den Kanten $\{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 4\}$ ($n = 4$) durch die Konstruktion.

Ja-Instanz. G hat die Clique $C = \{1, 2, 3\}$, also ist (G, k) mit $k = 3$ eine Ja-Instanz. Das Komplement G' enthält genau die Nicht-Kanten $\{2, 4\}, \{3, 4\}$. Ausgabe (G', k') mit $k' = n - k = 4 - 3 = 1$. Es ist $V \setminus C = \{4\}$ ein Vertex Cover von G' (Knoten 4 deckt beide Kanten) mit $|\{4\}| = 1 \leq k'$ – die Bildinstanz bleibt *Ja*.



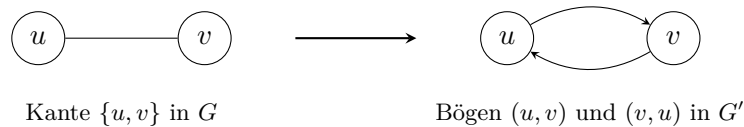
Nein-Instanz. Auf demselben G ist (G, k) mit $k = 4$ eine Nein-Instanz: G hat nur 4 Kanten, also keinen K_4 . Die Konstruktion liefert dasselbe G' , aber $k' = n - k = 4 - 4 = 0$. Ein Vertex Cover der Größe 0 ist die leere Menge – sie deckt die Kanten $\{2, 4\}, \{3, 4\}$ von G' nicht. Also existiert kein VC mit $|C| \leq 0$: die Bildinstanz bleibt *Nein*.

Zeigen Sie, dass `FEEDBACKVERTEXSET` NP-vollständig ist. *Hinweis:* Reduzieren Sie `VERTEXCOVER` auf `FEEDBACKVERTEXSET`.

Lösung.

`FEEDBACKVERTEXSET` $\in$ NP ist bekannt (Hausaufgabe 9.2, topologische Sortierung).

Konstruktion: Sei $(G = (V, E), k)$ eine `VERTEXCOVER`-Instanz. Erzeuge $G' = (V, E')$ mit $E' := \{(u, v), (v, u) \mid \{u, v\} \in E\}$ – jede ungerichtete Kante wird zu zwei antiparallelen Bögen. Gib (G', k) aus.



Beweis VertexCover nach FVS: Sei C ein VC mit $|C| \leq k$. Dann hat $G \setminus C$ keine Kanten, also hat $G' \setminus C$ keine Bögen und damit keine Kreise.

Beweis FVS nach VertexCover: Sei X ein FVS mit $|X| \leq k$. Angenommen, X wäre kein VC. Dann gäbe es $\{u, v\} \in E$ mit $u, v \notin X$; dann wäre (u, v, u) ein Kreis in $G' \setminus X$ – Widerspruch. Also ist X ein VC.

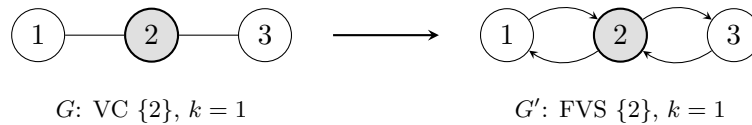
Laufzeit: $O(|E|)$, polynomiell.

Da `VERTEXCOVER` NP-vollständig ist und `FEEDBACKVERTEXSET` $\in$ NP, ist `FEEDBACKVERTEXSET` NP-vollständig. $\square$

Beispiel A7 – FeedbackVertexSet NP-vollständig

Reduktion $\text{VERTEXCOVER} \leq_p \text{FEEDBACKVERTEXSET}$: jede ungerichtete Kante $\{u, v\}$ wird zu zwei antiparallelen Bögen $(u, v), (v, u)$ (einem 2-Kreis); k bleibt. Instanz: der Pfad 1–2–3.

Ja-Instanz. Der Pfad hat das Vertex Cover $C = \{2\}$, also ist (G, k) mit $k = 1$ eine Ja-Instanz. In G' entstehen die 2-Kreise $1 \rightleftarrows 2$ und $2 \rightleftarrows 3$. Die Menge $X = \{2\}$ trifft beide (jeder 2-Kreis läuft über 2), ist also ein Feedback Vertex Set mit $|X| = 1 \leq k$ – die Bildinstanz bleibt *Ja*.



Nein-Instanz. Auf demselben Pfad ist (G, k) mit $k = 0$ eine Nein-Instanz (kein VC der Größe 0, es gibt Kanten). Die Konstruktion liefert dasselbe G' mit zwei 2-Kreisen; ein FVS der Größe 0 (die leere Menge) bricht keinen davon. Also kein FVS mit $|X| \leq 0$: die Bildinstanz bleibt *Nein*.

Problem k -CliqueUniversal: Gegeben ein Graph $G = (V, E)$ und $k \geq 1$, wobei ein $u \in V$ existiert, das mit allen anderen Knoten verbunden ist. **Entscheide:** Existiert eine Clique C mit $|C| \geq k$?

Zeigen Sie, dass k -CLIQUEUNIVERSAL NP-vollständig ist.

Lösung.

∈ **NP:** Jeder Verifizierer für CLIQUE eignet sich, da nur die Instanzmenge eingeschränkt ist: Zertifikat C , prüfe alle Paare und $|C| \geq k$. Laufzeit $O(|V|^2)$, polynomiell.

Konstruktion: Sei $(G = (V, E), k)$ eine CLIQUE-Instanz. Setze $\bar{k} = k + 1$ und $\bar{G} = (V \cup \{u\}, E \cup \{\{u, v\} \mid v \in V\})$ für einen neuen Knoten u . u ist universell, also ist $(\bar{G}, \bar{k})$ eine gültige Instanz.

Beweis Clique nach CliqueUniversal: Sei C eine Clique in G mit $|C| \geq k$. Da u mit allen verbunden ist, ist $C \cup \{u\}$ eine Clique in $\bar{G}$ mit $|C \cup \{u\}| \geq k + 1 = \bar{k}$.

Beweis CliqueUniversal nach Clique: Sei $\bar{C}$ eine Clique in $\bar{G}$ mit $|\bar{C}| \geq k + 1$. Dann ist $\bar{C} \setminus \{u\} \subseteq V$ eine Clique in G (Teilmenge einer Clique, u ist der einzige neue Knoten) mit $|\bar{C} \setminus \{u\}| \geq k$.

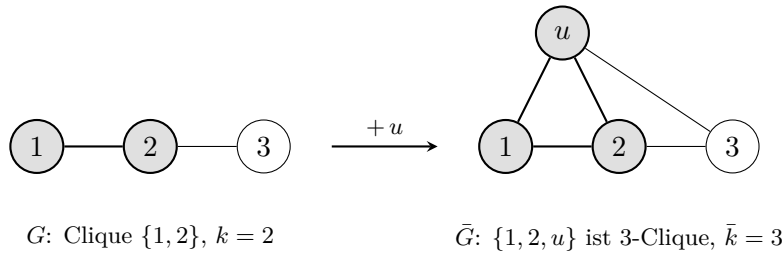
Laufzeit: $O(|V|)$ für Knoten und neue Kanten, polynomiell.

Da CLIQUE NP-vollständig ist und k -CLIQUEUNIVERSAL ∈ NP, ist k -CLIQUEUNIVERSAL NP-vollständig. $\square$

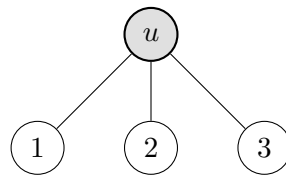
Beispiel A8 – k -CliqueUniversal NP-vollständig

Reduktion $\text{CLIQUE} \leq_p k\text{-CLIQUEUNIVERSAL}$: füge einen universellen Knoten u hinzu (verbunden mit allen) und setze $\bar{k} = k + 1$.

Ja-Instanz. G sei der Pfad 1–2–3 mit der 2-Clique $\{1, 2\}$, also (G, k) mit $k = 2$ eine Ja-Instanz. In $\bar{G}$ ist u mit 1, 2, 3 verbunden, $\bar{k} = 3$. Dann ist $\{1, 2, u\}$ ein Dreieck ($\{1, 2\} \in E$ plus die neuen Kanten $\{1, u\}, \{2, u\}$), also eine 3-Clique $\geq \bar{k}$ – die Bildinstanz bleibt *Ja*.



Nein-Instanz. G seien drei isolierte Knoten 1, 2, 3 (keine Kanten), (G, k) mit $k = 2$ eine Nein-Instanz (keine 2-Clique). Der universelle u liefert $\bar{k} = 3$, aber die größte Clique in $\bar{G}$ ist $\{u, i\}$ der Größe $2 < 3$ (die i sind paarweise nicht benachbart). Also keine 3-Clique: die Bildinstanz bleibt *Nein*.



$\bar{G}$: max. Clique $\{u, i\}$ hat Größe $2 < \bar{k} = 3$

Problem Clique-Nomember: Gegeben $G = (V, E)$, ein Knoten $v \in V$ und k . Gibt es eine k -Clique, die v *nicht* enthält?

Zeigen Sie NP-Vollständigkeit durch Reduktion von CLIQUE; CLIQUE-NOMEMBER $\in$ NP dürfen Sie annehmen.

Lösung.

Konstruktion: Sei $(G = (V, E), k)$ eine CLIQUE-Instanz. Sei $v \notin V$ ein neuer, isolierter Knoten. Gib $(G' = (V \cup \{v\}, E), v, k)$ aus.

Beweis Clique nach Clique-Nomember: Sei C eine k -Clique in G . C ist auch k -Clique in G' (keine Kanten entfernt), und $v \notin C$ wegen $v \notin V$.

Beweis Clique-Nomember nach Clique: Sei C eine k -Clique in G' ohne v . Da außer dem isolierten v nichts geändert wurde, liegt C komplett in G – eine k -Clique in G .

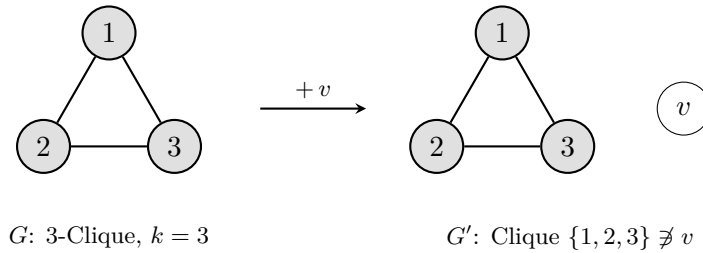
Laufzeit: $O(1)$ zusätzlich, polynomiell.

Da CLIQUE NP-vollständig ist und CLIQUE-NOMEMBER $\in$ NP, ist CLIQUE-NOMEMBER NP-vollständig. $\square$

Beispiel A9 – Clique-Nomember NP-vollständig

Reduktion $\text{CLIQUE} \leq_p \text{CLIQUE-NOMEMBER}$: füge einen neuen *isolierten* Knoten v hinzu und gib (G', v, k) aus. Gefragt ist eine k -Clique, die v nicht enthält.

Ja-Instanz. G sei das Dreieck $\{1, 2, 3\}$, also (G, k) mit $k = 3$ eine Ja-Instanz. In G' ist v isoliert. Die 3-Clique $\{1, 2, 3\}$ liegt ganz in G und enthält v nicht ($v \notin \{1, 2, 3\}$) – die Bildinstanz bleibt *Ja*.



Nein-Instanz. G sei der Pfad 1–2–3, (G, k) mit $k = 3$ eine Nein-Instanz (keine 3-Clique). In G' bleibt v isoliert; da nur der isolierte v hinzukam, gibt es weiterhin keine 3-Clique – erst recht keine ohne v . Die Bildinstanz bleibt *Nein*.

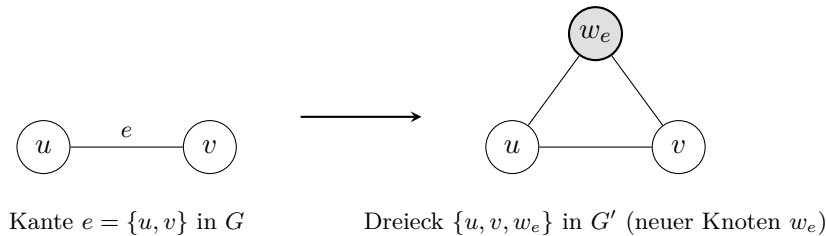
Problem Δ -Cover: Gegeben $G = (V, E)$ und k . Gibt es $C_\Delta \subseteq V$ mit $|C_\Delta| \leq k$, sodass jedes Dreieck (3-Clique) von G mindestens einen Knoten in C_Δ hat?

Zeigen Sie, dass Δ -COVER NP-vollständig ist.

Lösung.

∈ **NP**: Zertifikat $C_\Delta \subseteq V$. Prüfe $|C_\Delta| \leq k$ und für jedes Knotentripel, ob es ein Dreieck bildet und dann getroffen wird. Existiert eine Dreiecksüberdeckung, wird sie akzeptiert, sonst jedes Zertifikat abgelehnt. Laufzeit: $O(|V|^3)$ Tripel mit je $O(|V|)$ Prüfung – $O(|V|^4)$, polynomiell.

Konstruktion: Sei $(G = (V, E), k)$ eine VERTEXCOVER-Instanz. Bilde $G' = (V', E')$ mit $V' = V \cup \{w_e \mid e \in E\}$ und $E' = E \cup \{\{w_e, u\}, \{w_e, v\} \mid e = \{u, v\} \in E\}$ – jede Kante wird zu einem Dreieck $\{w_e, u, v\}$ aufgeblasen. Gib (G', k) aus.



Beweis VertexCover nach Δ -Cover: Sei C ein VC mit $|C| \leq k$. Jedes Dreieck in G' ist entweder ein Gadget-Dreieck $\{w_e, u, v\}$ – dann gilt $u \in C$ oder $v \in C$, da C die Kante e deckt – oder ein aus G geerbtes Dreieck $\{a, b, c\}$ (wegen $E \subseteq E'$) – dann wird z. B. die Kante $\{a, b\}$ von C gedeckt, also liegt ein Eckknoten in C . Jedes Dreieck ist getroffen.

Beweis Δ -Cover nach VertexCover: Sei C_Δ eine Dreiecksüberdeckung von G' mit $|C_\Delta| \leq k$. O.B.d.A. gilt $C_\Delta \subseteq V$: Enthält C_Δ ein w_e , ersetze es durch e_1 – das einzige Dreieck durch w_e ist $\{w_e, e_1, e_2\}$, es bleibt getroffen, und die Menge wird nicht größer. Nun trifft C_Δ jedes Gadget-Dreieck $\{w_e, u, v\}$ in u oder v – also hat jede Kante $e \in E$ einen Endpunkt in C_Δ : ein Vertex Cover.

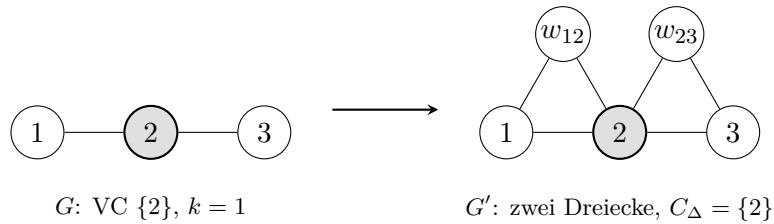
Laufzeit: Pro Kante ein Knoten und zwei Kanten, $O(|E|)$, polynomiell.

Da VERTEXCOVER NP-vollständig ist und Δ -COVER ∈ NP, ist Δ -COVER NP-vollständig. $\square$

Beispiel A10 – Δ -Cover NP-vollständig

Reduktion $\text{VERTEXCOVER} \leq_p \Delta\text{-COVER}$: jede Kante $e = \{u, v\}$ wird durch einen neuen Knoten w_e zum Dreieck $\{u, v, w_e\}$ aufgeblasen; k bleibt. Instanz: der Pfad 1–2–3 mit Kanten $e_{12} = \{1, 2\}$, $e_{23} = \{2, 3\}$.

Ja-Instanz. Der Pfad hat das Vertex Cover $C = \{2\}$, also (G, k) mit $k = 1$ eine Ja-Instanz. G' erhält w_{12}, w_{23} und damit genau die Dreiecke $\{1, 2, w_{12}\}$ und $\{2, 3, w_{23}\}$. Beide enthalten 2, also trifft $C_\Delta = \{2\}$ jedes Dreieck mit $|C_\Delta| = 1 \leq k$ – die Bildinstanz bleibt *Ja*.



Nein-Instanz. Auf demselben Pfad ist (G, k) mit $k = 0$ eine Nein-Instanz (kein VC der Größe 0). G' hat dieselben zwei Dreiecke; ein Δ -Cover der Größe 0 (die leere Menge) überdeckt keines. Also kein Δ -Cover mit $|C_\Delta| \leq 0$: die Bildinstanz bleibt *Nein*.

Problem SubsetSumCardinality: Gegeben $c_1, \dots, c_n \in \mathbb{N}_{>0}$ (n gerade) und K . Gibt es S mit $|S| = n/2$ und $\sum_{i \in S} c_i = K$? Zeigen Sie NP-Vollständigkeit; $\in \text{NP}$ dürfen Sie annehmen.

Lösung.

Konstruktion: Sei $(c_1, \dots, c_n, K)$ eine SUBSETSUM-Instanz. Setze $c'_i := c_i + 1$ für $i \leq n$ (Shift) und $c'_i := 1$ für $i \in \{n+1, \dots, 2n\}$ (n Padding-Items). Gib $(c'_1, \dots, c'_{2n}, K+n)$ aus $(2n$ Items, verlangt $|S| = n)$.

Beweis SubsetSum nach Cardinality: Sei S mit $\sum_{i \in S} c_i = K$. Dann $\sum_{i \in S} c'_i = K + |S|$. Fülle mit $S' = \{n+1, \dots, 2n - |S|\}$ auf: $|S \cup S'| = n$ und

$$\sum_{i \in S \cup S'} c'_i = K + |S| + (n - |S|) = K + n.$$

Beweis Cardinality nach SubsetSum: Sei S' mit $|S'| = n$ und $\sum_{i \in S'} c'_i = K + n$. Setze $S := S' \cap \{1, \dots, n\}$. Die $n - |S|$ Padding-Items in S' tragen je 1 bei, also $\sum_{i \in S} (c_i + 1) = K + n - (n - |S|) = K + |S|$, d. h. $\sum_{i \in S} c_i = K$.

Laufzeit: $O(n)$, polynomiell.

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

Beispiel A11 – SubsetSumCardinality NP-vollständig

Reduktion $\text{SUBSETSUM} \leq_p \text{SUBSETSUMCARDINALITY}$: aus $(c_1, \dots, c_n, K)$ mache $c'_i = c_i + 1$ (Shift) für $i \leq n$ und n Padding-Items $c'_i = 1$ für $i \in \{n+1, \dots, 2n\}$; Zielwert $K + n$, verlangte Kardinalität n (also $n/2$ der $2n$ Items). Instanz: $c = (2, 3, 5)$, $n = 3$.

Ja-Instanz. $K = 5$ ist eine Ja-Instanz: $\{1, 2\}$ mit $2 + 3 = 5$. Die Konstruktion liefert $c' = (3, 4, 6)$ und die Padding-Items $(1, 1, 1)$ als Items 4, 5, 6; Zielwert $K + n = 5 + 3 = 8$, Kardinalität 3. Nimm die Bildmenge $\{1, 2\}$ und fülle mit einem Padding-Item auf: $\{1, 2, 4\}$ mit $3 + 4 + 1 = 8$ und genau 3 Elementen – die Bildinstanz bleibt *Ja*.

Nein-Instanz. $K = 4$ ist eine Nein-Instanz: keine Teilmenge von $(2, 3, 5)$ summiert zu 4. Dieselbe Konstruktion ergibt Items $(3, 4, 6, 1, 1, 1)$, Zielwert $K + n = 4 + 3 = 7$, Kardinalität 3. Ein 3-elementiges Subset mit j Original- und $3 - j$ Padding-Items hat Summe $(\text{Originalsumme} + j) + (3 - j) = \text{Originalsumme} + 3$; für Summe 7 wäre eine Originalsumme 4 nötig – die es nicht gibt. Kein 3-Subset trifft 7 (mögliche Summen: 3, 5, 6, 8, 10, 11, 13): die Bildinstanz bleibt *Nein*.

$\text{HALT}_{\text{TM}} := \{\langle M, w \rangle \mid M \text{ ist DTM und hält auf } w\}$. Zeigen Sie: HALT_{TM} ist NP-schwer.

Lösung.

Wir zeigen $\text{SAT} \leq_p \text{HALT}_{\text{TM}}$; da SAT NP-schwer ist, folgt mit der Transitivität die NP-Schwere.

Konstruktion: Zu einer KNF-Formel φ konstruiere die DTM M_φ , die (unabhängig von ihrer Eingabe) alle Belegungen der Variablen von φ systematisch aufzählt, jede auswertet und *hält*, sobald eine erfüllende gefunden ist – sonst läuft sie endlos weiter. Gib $\langle M_\varphi, \varepsilon \rangle$ aus.

Korrektheit: Ist φ erfüllbar, erreicht die Aufzählung eine erfüllende Belegung und M_φ hält. Ist φ unerfüllbar, hält M_φ nie. Also $\varphi \in \text{SAT} \iff \langle M_\varphi, \varepsilon \rangle \in \text{HALT}_{\text{TM}}$.

Laufzeit: M_φ läuft womöglich ewig – aber die *Konstruktion* von $\langle M_\varphi, \varepsilon \rangle$ aus φ ist polynomiell (Formel plus feste Schleifen-Schablone), und nur darauf kommt es an.

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

Beispiel A12 – HALT_{TM} NP-schwer

Reduktion $\text{SAT} \leq_p \text{HALT}_{\text{TM}}$: zu einer Formel φ baue die DTM M_φ , die alle Belegungen von φ systematisch aufzählt, jede auswertet und *hält*, sobald eine erfüllende gefunden ist – sonst läuft sie ewig. Ausgabe $\langle M_\varphi, \varepsilon \rangle$.

Ja-Instanz. $\varphi = (x_1 \vee x_2) \wedge (\neg x_1)$ ist erfüllbar. M_φ zählt auf: $x_1x_2 = 00$ gibt $C_1 = (0 \vee 0) = \text{falsch}$; $x_1x_2 = 01$ gibt $(0 \vee 1) = \text{wahr}$ und $(\neg 0) = \text{wahr}$ – beide Klauseln wahr, also hält M_φ hier. Damit $\langle M_\varphi, \varepsilon \rangle \in \text{HALT}_{\text{TM}}$: die Bildinstanz bleibt *Ja*.

Nein-Instanz. $\varphi = (x_1) \wedge (\neg x_1)$ ist unerfüllbar: $x_1 = 0$ scheitert an der Klausel (x_1) , $x_1 = 1$ an $(\neg x_1)$. M_φ findet nie eine erfüllende Belegung und hält nie, also $\langle M_\varphi, \varepsilon \rangle \notin \text{HALT}_{\text{TM}}$: die Bildinstanz bleibt *Nein*.

Pointe. Obwohl M_φ selbst womöglich *nie* hält, ist die *Konstruktion* des Paares $\langle M_\varphi, \varepsilon \rangle$ aus φ (Formel plus feste Schleifen-Schablone) in Polynomialzeit möglich – und nur darauf kommt es bei der Reduktion an.

Problem CliqueAndIndependentSet: Gegeben $G = (V, E)$ und $k \geq 1$. Gibt es in G sowohl ein Independent Set der Größe k als auch eine Clique der Größe k ? Zeigen Sie die NP-Schwere durch Reduktion eines bekannten NP-schweren Problems.

Lösung.

Konstruktion: Sei $(G = (V, E), k)$ eine CLIQUE-Instanz (o.B.d.A. $k \geq 2$). Füge k neue isolierte Knoten $u_1, \dots, u_k$ hinzu: $G' = (V \cup \{u_1, \dots, u_k\}, E)$. Gib (G', k) aus.

Beweis Clique nach CliqueAndIndependentSet: Sei C eine k -Clique in G . Dann ist C auch k -Clique in G' , und $\{u_1, \dots, u_k\}$ ist ein Independent Set der Größe k (isolierte Knoten). Beides existiert.

Beweis CliqueAndIndependentSet nach Clique: G' enthalte eine k -Clique C' . Wegen $k \geq 2$ hat jeder Knoten in C' mindestens einen Nachbarn in C' ; die isolierten u_i haben keine Nachbarn, also $C' \subseteq V$ – eine k -Clique in G .

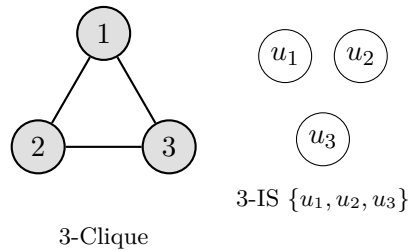
Laufzeit: $O(k) \subseteq O(|V|)$, polynomiell.

Da CLIQUE NP-schwer ist, ist CLIQUEANDINDEPENDENTSET NP-schwer. □

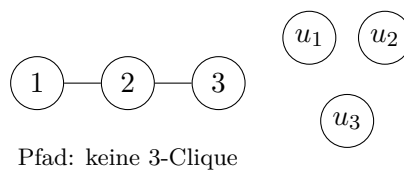
Beispiel A13 – CliqueAndIndependentSet NP-schwer

Reduktion $\text{CLIQUE} \leq_p \text{CLIQUEANDINDEPENDENTSET}$: füge k neue isolierte Knoten $u_1, \dots, u_k$ hinzu; k bleibt. Gefragt sind in G' zugleich eine k -Clique und ein k -Independent-Set.

Ja-Instanz. G sei das Dreieck $\{1, 2, 3\}$, also (G, k) mit $k = 3$ eine Ja-Instanz. G' erhält u_1, u_2, u_3 . Dann ist $\{1, 2, 3\}$ eine 3-Clique und $\{u_1, u_2, u_3\}$ ein 3-Independent-Set (isoliert, paarweise nicht benachbart) – beides existiert, die Bildinstanz bleibt *Ja*.



Nein-Instanz. G sei der Pfad 1–2–3, (G, k) mit $k = 3$ eine Nein-Instanz (keine 3-Clique). G' erhält wieder u_1, u_2, u_3 ; ein 3-Independent-Set existiert nun zwar (z. B. $\{u_1, u_2, u_3\}$), aber eine 3-Clique gibt es weiterhin nicht (Pfad hat kein Dreieck, die u_i sind isoliert). Da nicht beides existiert, bleibt die Bildinstanz *Nein*.



Problem $(a_1=1)$ -SubsetSum: n Items mit Größen $a_i \in \mathbb{N}_{>0}$, wobei $a_1 = 1$, und Zielwert T . Gibt es $I \subseteq [n]$ mit $\sum_{i \in I} a_i = T$? Zeigen Sie die NP-Schwere durch Reduktion eines bekannten NP-schweren Problems.

Lösung.

Konstruktion: Sei $(c_1, \dots, c_n, K)$ eine SUBSETSUM-Instanz. Erzeuge die Items $a_1 = 1$ und $a_{i+1} = 2c_i$ für $i \in [n]$, mit Zielwert $T = 2K$. ($a_1 = 1$ ist erfüllt, die Instanz ist gültig.)

Beweis SubsetSum nach $(a_1=1)$ -SubsetSum: Sei S mit $\sum_{i \in S} c_i = K$. Dann hat $\{i+1 \mid i \in S\}$ die Summe $2K = T$.

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, T ist gerade – enthielte I das Item $a_1 = 1$, wäre die Summe ungerade. Also $1 \notin I$, und $S := \{i-1 \mid i \in I\}$ erfüllt $\sum_{i \in S} 2c_i = 2K$, d. h. $\sum_{i \in S} c_i = K$.

Laufzeit: $O(n)$, polynomiell.

Da SUBSETSUM NP-schwer ist, ist $(a_1=1)$ -SUBSETSUM NP-schwer. $\square$

Beispiel A14 – $(a_1=1)$ -SubsetSum

Reduktion. Aus einer SUBSETSUM-Instanz $(c_1, \dots, c_n, K)$ entsteht die neue Instanz mit $a_1 = 1$, $a_{i+1} = 2c_i$ und $T = 2K$. Das Original-Item c_i wandert also nach a_{i+1} ; das aufgesetzte Item $a_1 = 1$ ist reiner Ballast, damit die Vorbedingung $a_1 = 1$ erfüllt ist.

Ja-Instanz bleibt Ja. $c = (2, 3, 5)$, $K = 5$ mit Lösung $c_1 + c_2 = 2 + 3 = 5$. Transformiert: $a = (1, 4, 6, 10)$, denn $a_2 = 2 \cdot 2$, $a_3 = 2 \cdot 3$, $a_4 = 2 \cdot 5$, und $T = 2 \cdot 5 = 10$. Die Original-Auswahl $\{c_1, c_2\}$ wird zu $\{a_2, a_3\}$ mit $4 + 6 = 10 = T$. Das Item $a_1 = 1$ bleibt ungenutzt. **Ja.**

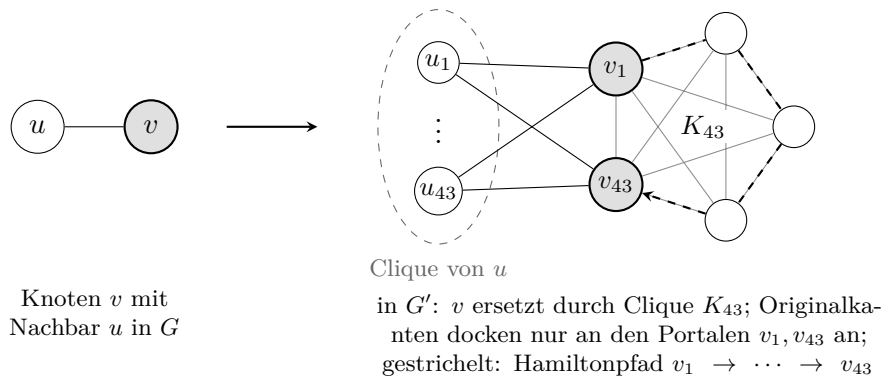
Nein-Instanz bleibt Nein. $c = (2, 3, 5)$, $K = 4$: keine Teilmenge von $\{2, 3, 5\}$ ergibt 4. Transformiert: $a = (1, 4, 6, 10)$, $T = 8$. *Paritätsargument:* $T = 8$ ist gerade, und jedes a_i mit $i \geq 2$ ist gerade; nähme man $a_1 = 1$ hinzu, würde die Summe ungerade. Also muss a_1 draußen bleiben – doch aus $\{4, 6, 10\}$ ergibt sich nie 8 (4, 6, 10, 4+6=10, 4+10=14, 6+10=16, 20). **Nein.**

Genau dieses Paritätsargument ist die Rückrichtung des Beweises: Eine Lösung mit Summe $2K$ kann $a_1 = 1$ nie verwenden und entspricht daher einer c -Auswahl mit Summe K .

Problem Hitchhiker's-HamiltonianCycle: Gegeben $G = (V, E)$ mit $\deg(v) \geq 42$ für alle $v \in V$. Gibt es einen Hamiltonkreis? Zeigen Sie die NP-Schwere durch Reduktion eines bekannten NP-schweren Problems.

Lösung.

Konstruktion (Reduktion von HAMILTONIANCYCLE): Ersetze jeden Knoten v durch eine Clique K_{43} mit Knoten $v_1, \dots, v_{43}$. Für jede Originalkante $\{u, v\} \in E$ füge die vier Kanten $\{u_1, v_1\}, \{u_1, v_{43}\}, \{u_{43}, v_1\}, \{u_{43}, v_{43}\}$ hinzu (Portale sind v_1 und v_{43}).



Grade: innere Clique-Knoten haben Grad 42; die Portale v_1, v_{43} haben Grad ≥ 42 . Die Instanz ist also gültig.

Beweis HK nach Hitchhiker's-HK: Sei $v^{(1)}, \dots, v^{(n)}, v^{(1)}$ ein Hamiltonkreis in G . Ersetze jeden Besuch von v durch einen Hamiltonpfad durch dessen Clique von Portal zu Portal ($v_1 \rightarrow \dots \rightarrow v_{43}$; in einer Clique existiert er immer). Die Übergänge nutzen die Portalkanten – ein Hamiltonkreis in G' .

Beweis Hitchhiker's-HK nach HK: Ein Hamiltonkreis in G' besucht in jeder Knoten-Clique alle 43 Knoten. Die inneren Knoten haben nur Clique-Nachbarn, Außenkanten gibt es nur an den zwei Portalen – der Kreis durchläuft jede Clique also als *ein* Segment von Portal zu Portal. Zieht man jede Clique zu einem Knoten zusammen, entsteht ein Hamiltonkreis in G .

Laufzeit: $O(43^2 \cdot |V| + |E|)$, polynomiell.

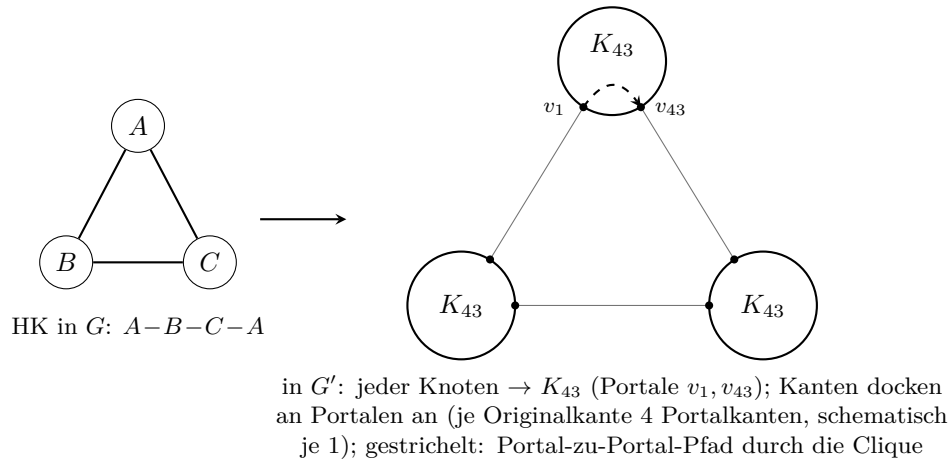
Da HAMILTONIANCYCLE NP-schwer ist, ist HITCHHIKER'S-HAMILTONIANCYCLE NP-schwer.

□

Beispiel A15 – Hitchhiker’s-HamiltonianCycle

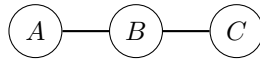
Reduktion. Jeder Knoten von G wird zu einer Clique K_{43} mit den Portalen v_1, v_{43} ; jede Originalkante dockt nur an den Portalen an. Ein Hamiltonkreis durchläuft jede Clique als *ein* Portal-zu-Portal-Segment.

Ja-Instanz bleibt Ja. Basis: Dreieck A, B, C mit Hamiltonkreis $A-B-C-A$.



Der HK $A-B-C-A$ wird zu einem HK in G' : In jeder Clique ein Portal-zu-Portal-Pfad, dazwischen die Portalkanten. **Ja.**

Nein-Instanz bleibt Nein. Basis: Pfad $A-B-C$ (keine Kante $A-C$), also *kein* Hamiltonkreis.



Ohne HK in G gibt es keinen in G' : Ein Kreis müsste jede Clique einmal Portal-zu-Portal durchlaufen; das Zusammenziehen der Cliques ergäbe einen HK im Pfad – den gibt es nicht. **Nein.**

Problem AtMostTwoPerSize-SubsetSum: Zielwert $T > 0$, n Items mit Größen $a_i \in \mathbb{N}_{>0}$; jede Größe tritt höchstens zweimal auf. Gibt es S mit $\sum_{i \in S} a_i = T$? Zeigen Sie die NP-Schwere durch Reduktion eines bekannten NP-schweren Problems.

Hinweis: Bei 3-EXACTCOVER sind ein Universum U mit $|U| = 3m$ und Dreiermengen $S_1, \dots, S_n \subseteq U$ gegeben; gefragt ist eine Auswahl von Mengen, die jedes Element *genau einmal* überdeckt. Die Vorlesung reduziert 3-EC auf SUBSETSUM über die Kodierung $c_j = \sum_{u_i \in S_j} (n+1)^{i-1}$ mit Ziel $K = \sum_{j=0}^{3m-1} (n+1)^j$.

Lösung.

Konstruktion (Reduktion von 3-EXACTCOVER): Nutze die Vorlesungs-Reduktion $3\text{-EC} \leq_p \text{SUBSETSUM}$: Für eine 3-EC-Instanz mit Universum $|U| = 3m$ und Mengen $S_1, \dots, S_n$ setze $c_j = \sum_{u_i \in S_j} (n+1)^{i-1}$ und $K = \sum_{j=0}^{3m-1} (n+1)^j$. O.B.d.A. sind die Mengen S_j paarweise verschieden (doppelte Mengen ändern die Existenz einer exakten Überdeckung nicht – lösche Duplikate vorab).

Verschiedene Mengen ergeben verschiedene Bitvektoren und damit paarweise verschiedene Größen c_j – jede Größe tritt genau einmal auf, die Zusatzbedingung ist erfüllt.

	Ziffer zu u_i (Basis $n+1=4$)						c_j
	u_6	u_5	u_4	u_3	u_2	u_1	
$S_1 = \{u_1, u_2, u_3\}$	0	0	0	1	1	1	21
$S_2 = \{u_3, u_4, u_5\}$	0	1	1	1	0	0	336
$S_3 = \{u_4, u_5, u_6\}$	1	1	1	0	0	0	1344
Ziel K	1	1	1	1	1	1	1365

Beispiel mit $|U| = 6$, $n = 3$: verschiedene Mengen $\Rightarrow$ verschiedene Ziffernvektoren $\Rightarrow$ paarweise verschiedene c_j ; $c_1 + c_3 = 1365 = K$ entspricht der exakten Überdeckung $\{S_1, S_3\}$.

Korrektheit (beide Richtungen): Wie in der Vorlesungs-Reduktion: Wegen Basis $n+1$ entsteht kein Übertrag, also gilt $\sum_{j \in S} c_j = K$ genau dann, wenn die gewählten Mengen jedes Element genau einmal überdecken – eine exakte Überdeckung existiert genau dann, wenn die SubsetSum-Instanz eine Lösung hat.

Laufzeit: Duplikat-Entfernung und Kodierung in Polynomialzeit.

Da 3-EXACTCOVER NP-schwer ist, ist ATMOSTTWOPERSIZE-SUBSETSUM NP-schwer. $\square$

Beispiel A16 – AtMostTwoPerSize-SubsetSum

Reduktion ($3\text{-EXACTCOVER} \leq_p \text{SUBSETSUM}$). Universum $U = \{u_1, \dots, u_6\}$ ($m = 2$, also $|U| = 6$), $n = 3$ Mengen. Kodiere jede Menge in Basis $n + 1 = 4$: $c_j = \sum_{u_i \in S_j} 4^{i-1}$, Ziel $K = \sum_{j=0}^5 4^j = 1365$ (in Basis 4: 111111).

Ja-Instanz bleibt Ja. $S_1 = \{u_1, u_2, u_3\}$, $S_2 = \{u_4, u_5, u_6\}$, $S_3 = \{u_1, u_2, u_4\}$.

	Ziffer zu u_i (Basis 4)						c_j
	u_6	u_5	u_4	u_3	u_2	u_1	
$S_1 = \{u_1, u_2, u_3\}$	0	0	0	1	1	1	21
$S_2 = \{u_4, u_5, u_6\}$	1	1	1	0	0	0	1344
$S_3 = \{u_1, u_2, u_4\}$	0	0	1	0	1	1	69
Ziel K	1	1	1	1	1	1	1365

Verschiedene Mengen $\Rightarrow$ verschiedene Ziffernvektoren $\Rightarrow c_1, c_2, c_3$ paarweise verschieden (jede Größe tritt genau einmal auf – ≤ 2 -mal ist erfüllt). Die exakte Überdeckung $\{S_1, S_2\}$ liefert $c_1 + c_2 = 21 + 1344 = 1365 = K$ (kein Übertrag in Basis 4). **Ja.**

Nein-Instanz bleibt Nein. Gleiches U , aber nur $S_1 = \{u_1, u_2, u_3\}$, $S_3 = \{u_1, u_2, u_4\}$: u_5, u_6 sind unüberdeckbar, keine exakte Überdeckung. Größen $c_1 = 21$, $c_3 = 69$; jede Teilsumme $(21, 69, 90)$ bleibt unter $K = 1365$. **Nein.**

Beweisen Sie die NP-Vollständigkeit von SUBSETSUM, bei dem jede Itemgröße durch 3 oder durch 7 teilbar ist.

Lösung.

∈ **NP**: Zertifikat S ; summiere und vergleiche mit K , wie beim gewöhnlichen SUBSETSUM. Laufzeit $O(n)$, polynomiell.

Konstruktion: Sei $(c_1, \dots, c_n, K)$ eine SUBSETSUM-Instanz. Setze $c'_i := 3c_i$ und $K' := 3K$. Alle Größen sind durch 3 teilbar – die Instanz ist gültig.

Beweis hin: Sei S mit $\sum_{i \in S} c_i = K$. Dann $\sum_{i \in S} c'_i = 3K = K'$.

Beweis zurück: Sei S mit $\sum_{i \in S} c'_i = K'$. Division durch 3 liefert $\sum_{i \in S} c_i = K$.

Laufzeit: $O(n)$, polynomiell.

Da SUBSETSUM NP-vollständig ist und die Variante in NP liegt, ist sie NP-vollständig. $\square$

Beispiel A17 – SubsetSum mit Teilbarkeit

Reduktion. Aus $(c_1, \dots, c_n, K)$ entsteht $c'_i = 3c_i$, $K' = 3K$; dann ist jede Größe durch 3 teilbar (also die Zusatzbedingung „durch 3 oder 7 teilbar“ erfüllt).

Ja-Instanz bleibt Ja. $c = (2, 3, 5)$, $K = 5$ mit $c_1 + c_2 = 2 + 3 = 5$. Transformiert: $c' = (6, 9, 15)$, $K' = 15$. Dieselbe Auswahl: $6 + 9 = 15 = K'$ (alle Größen durch 3 teilbar). **Ja.**

Nein-Instanz bleibt Nein. $c = (2, 3, 5)$, $K = 4$: keine Teilmenge ergibt 4. Transformiert: $c' = (6, 9, 15)$, $K' = 12$. Teilsummen von $\{6, 9, 15\}$: 6, 9, 15, 15, 21, 24, 30 – 12 ist nicht dabei. **Nein.**

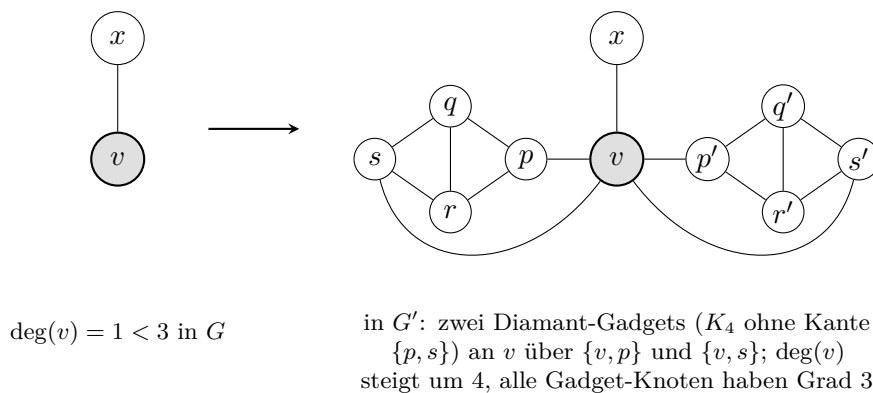
Rückrichtung: Aus $\sum_{i \in S} c'_i = K'$ folgt durch Division durch 3 sofort $\sum_{i \in S} c_i = K$.

Beweisen Sie die NP-Vollständigkeit von 3-COLOR, wobei jeder Knoten im Graphen mindestens Grad 3 hat.

Lösung.

∈ **NP**: Zertifikat ist die Färbung $f : V \rightarrow \{1, 2, 3\}$; prüfe für jede Kante $f(u) \neq f(v)$. Laufzeit $O(|E|)$, polynomiell.

Konstruktion (Reduktion von 3-COLOR): Benutze das Diamant-Gadget D : Knoten p, q, r, s mit Kanten pq, pr, qr, qs, rs (K_4 ohne ps). Hänge an jeden Knoten v mit $\deg(v) < 3$ zwei eigene Gadgets an, verbunden über $\{v, p\}$ und $\{v, s\}$ je Gadget. Dann steigt $\deg(v)$ um 4, und alle Gadget-Knoten haben Grad 3 (p, s : zwei Gadget-Kanten + v ; q, r : drei Gadget-Kanten).



Beweis hin: Sei f eine 3-Färbung von G . Erweitere sie auf jedes Gadget: Färbe $p = s = c$ für ein $c \neq f(v)$; dann q, r mit den beiden übrigen Farben (q, r sind zu p und s adjazent, aber p, s tragen dieselbe Farbe – zwei Farben bleiben frei). Alle Kanten sind gültig.

Beweis zurück: Sei f' eine 3-Färbung von G' . Die Einschränkung auf V ist eine 3-Färbung von G , denn $E \subseteq E'$.

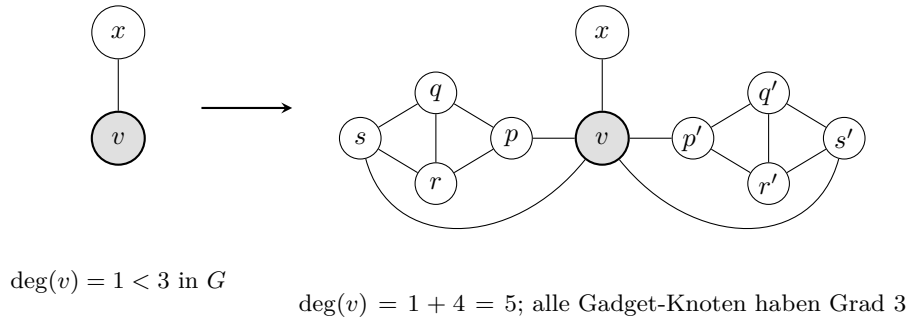
Laufzeit: $O(|V|)$ Gadgets konstanter Größe, polynomiell.

Da 3-COLOR NP-vollständig ist und die Variante in NP liegt, ist sie NP-vollständig. $\square$

Beispiel A18 – 3-COLOR mit Minimalgrad 3

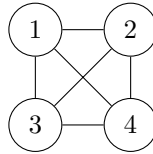
Reduktion. An jeden Knoten v mit $\deg(v) < 3$ werden *zwei* Diamant-Gadgets ($D = K_4$ ohne Kante $\{p, s\}$; Kanten pq, pr, qr, qs, rs) angehängt, verbunden über $\{v, p\}$ und $\{v, s\}$. Das hebt $\deg(v)$ um 4; alle Gadget-Knoten bekommen Grad 3.

Ja-Instanz bleibt Ja. G = eine Kante $\{x, v\}$, also $\deg(v) = 1$.



Gültige 3-Färbung (Farben 1, 2, 3): $f(v) = 1$, $f(x) = 2$; je Gadget $p = s = 2$, $q = 1$, $r = 3$ (analog $p', s' = 2$, $q' = 1$, $r' = 3$). Alle Kanten korrekt. **Ja.**

Nein-Instanz bleibt Nein. $G = K_4$ ist nicht 3-färbbar – und hat schon überall Grad 3, es werden also *keine* Gadgets angehängt, $G' = K_4$.



K_4 : Grade alle 3, nicht 3-färbbar

$G' = K_4$ ist nicht 3-färbbar. **Nein.**

Sei L die Menge der SUBSETSUM-Instanzen, bei denen keine Itemgröße eine Zweierpotenz ist. Zeigen Sie, dass L NP-vollständig ist.

Lösung.

∈ **NP**: Zertifikat S ; summiere und vergleiche mit K . Laufzeit $O(n)$, polynomiell.

Konstruktion: Sei $(c_1, \dots, c_n, K)$ eine SUBSETSUM-Instanz. Setze $c'_i := 3c_i$ und $K' := 3K$. Jede Größe $c'_i \geq 3$ ist durch 3 teilbar und damit keine Zweierpotenz (Zweierpotenzen haben nur den Primfaktor 2) – die Instanz liegt in der eingeschränkten Menge.

Beweis hin: $\sum_{i \in S} c_i = K \Rightarrow \sum_{i \in S} c'_i = 3K = K'$.

Beweis zurück: $\sum_{i \in S} c'_i = K' \Rightarrow$ Division durch 3: $\sum_{i \in S} c_i = K$.

Laufzeit: $O(n)$, polynomiell.

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

Beispiel A19 – SubsetSum ohne Zweierpotenzen

Reduktion. Aus $(c_1, \dots, c_n, K)$ entsteht $c'_i = 3c_i$, $K' = 3K$. Jedes c'_i hat den Primfaktor 3, ist also keine Zweierpotenz (diese haben nur den Primfaktor 2) – die Instanz liegt in der eingeschränkten Menge, auch wenn die Ausgangsgrößen Zweierpotenzen enthalten.

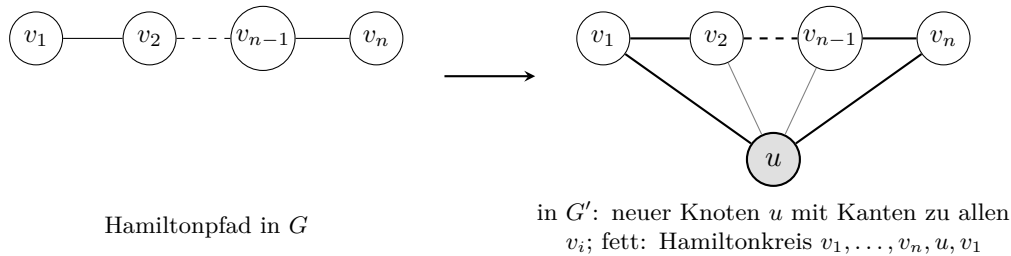
Ja-Instanz bleibt Ja. $c = (1, 4, 6)$ (mit $1 = 2^0$, $4 = 2^2$ Zweierpotenzen!), $K = 5$, Lösung $c_1 + c_2 = 1 + 4 = 5$. Transformiert: $c' = (3, 12, 18)$, $K' = 15$. Dieselbe Auswahl: $3 + 12 = 15 = K'$ – und 3, 12, 18 ist keine Zweierpotenz. **Ja.**

Nein-Instanz bleibt Nein. $c = (1, 4, 6)$, $K = 8$: keine Teilmenge von $\{1, 4, 6\}$ ergibt 8. Transformiert: $c' = (3, 12, 18)$, $K' = 24$. Teilsummen von $\{3, 12, 18\}$: 3, 12, 18, 15, 21, 30, 33 – 24 ist nicht dabei. **Nein.**

Geben Sie eine polynomielle Reduktion von HAMILTONIANPATH auf HAMILTONIANCYCLE an.

Lösung.

Konstruktion: Sei $G = (V, E)$ gegeben. Füge einen neuen universellen Knoten u hinzu: $G' = (V \cup \{u\}, E \cup \{\{u, v\} \mid v \in V\})$.



Beweis HamPath nach HamCycle: Sei $v_1, \dots, v_n$ ein Hamiltonpfad in G . Dann ist $v_1, \dots, v_n, u, v_1$ ein Hamiltonkreis in G' – die Kanten $\{v_n, u\}$ und $\{u, v_1\}$ existieren, da u universell ist.

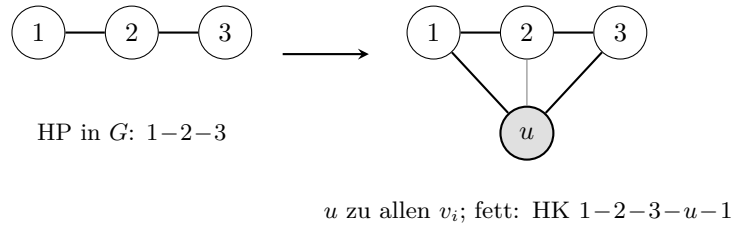
Beweis HamCycle nach HamPath: Sei C ein Hamiltonkreis in G' . Er besucht u genau einmal, zwischen zwei Nachbarn $v_i, v_j \in V$. Entfernt man u aus dem Kreis, bleibt ein Pfad von v_i nach v_j , der alle Knoten aus V genau einmal besucht und nur Kanten aus E benutzt – ein Hamiltonpfad in G .

Laufzeit: $O(|V|)$ neue Kanten, polynomiell. □

Beispiel A20 – HamiltonianPath $\leq_p$ HamiltonianCycle

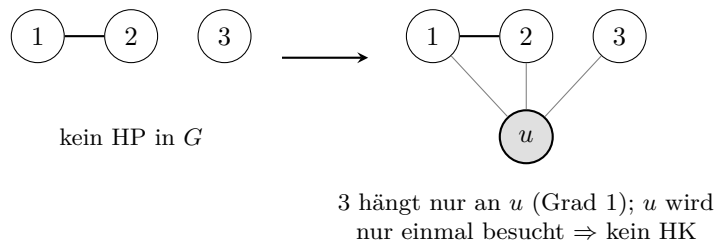
Reduktion. Füge einen universellen Knoten u hinzu, verbunden mit allen $v \in V$. Ein Hamiltonpfad in G schließt sich über u zum Hamiltonkreis in G' .

Ja-Instanz bleibt Ja. $G = \text{Pfad } 1-2-3$ (hat HP).



Ja.

Nein-Instanz bleibt Nein. $G = \text{Kante } \{1,2\} \text{ plus isolierter Knoten } 3$ (kein HP).

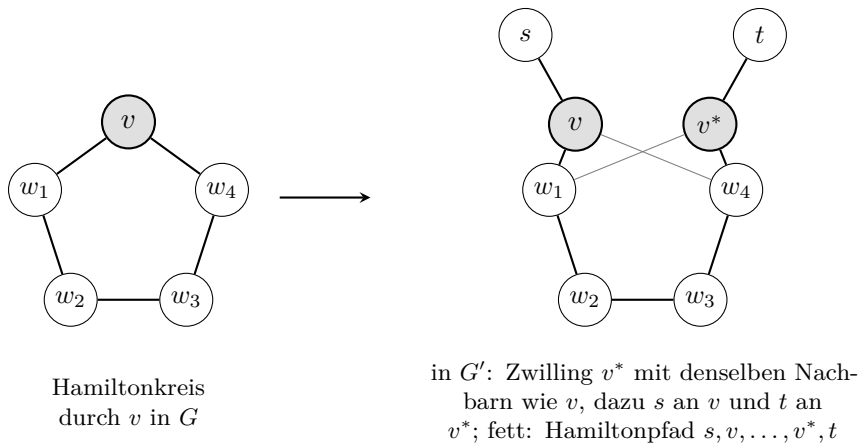


Knoten 3 ist im Kreis nur über u erreichbar, doch u liegt in einem Kreis zwischen genau zwei Nachbarn – 3 mit Grad 1 kann nicht auf einem Kreis liegen. **Nein.**

Geben Sie eine polynomielle Reduktion von HAMILTONIANCYCLE auf HAMILTONIANPATH an.

Lösung.

Konstruktion: Sei $G = (V, E)$ gegeben, o.B.d.A. $|V| \geq 3$. Wähle einen beliebigen Knoten v . Füge hinzu: einen Zwilling v^* mit denselben Nachbarn wie v , einen Grad-1-Knoten s mit Kante $\{s, v\}$ und einen Grad-1-Knoten t mit Kante $\{t, v^*\}$.



Beweis HamCycle nach HamPath: Sei $v, w_1, \dots, w_{n-1}, v$ ein Hamiltonkreis in G . Dann ist $s, v, w_1, \dots, w_{n-1}, v^*, t$ ein Hamiltonpfad in G' – die Kante $\{w_{n-1}, v^*\}$ existiert, weil v^* dieselben Nachbarn wie v hat.

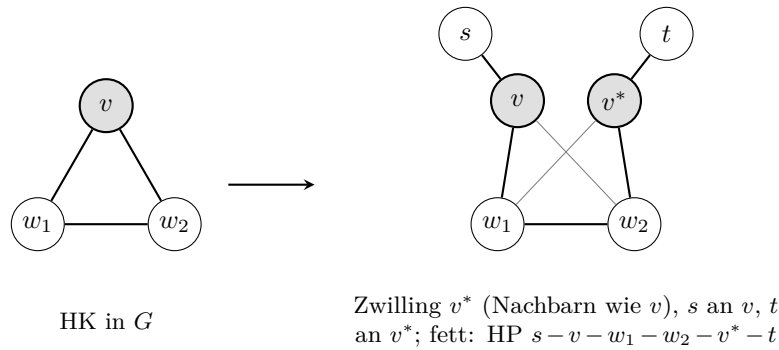
Beweis HamPath nach HamCycle: Sei P ein Hamiltonpfad in G' . Die Knoten s und t haben Grad 1, also sind sie die beiden Endpunkte: $P = s, v, x_1, \dots, x_{n-1}, v^*, t$, wobei $x_1, \dots, x_{n-1}$ alle Knoten aus $V \setminus \{v\}$ sind. Ersetzt man v^* durch v , wird aus dem Mittelstück $v, x_1, \dots, x_{n-1}, v$ ein Hamiltonkreis in G – die letzte Kante existiert, da $\{x_{n-1}, v^*\} \in E'$ nur für Nachbarn von v möglich ist.

Laufzeit: $O(|V|)$, polynomiell. □

Beispiel A21 – $\text{HamiltonianCycle} \leq_p \text{HamiltonianPath}$

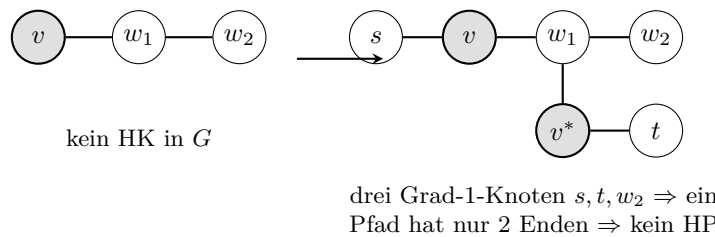
Reduktion. Wähle einen Knoten v ; füge einen Zwilling v^* mit denselben Nachbarn wie v hinzu, dazu Grad-1-Pendants s an v und t an v^* . Ein HK durch v wird zum HP, indem er bei v/v^* aufgetrennt und mit s, t verlängert wird.

Ja-Instanz bleibt Ja. $G = \text{Dreieck } \{v, w_1, w_2\}$ (hat HK).



Ja.

Nein-Instanz bleibt Nein. $G = \text{Pfad } v-w_1-w_2$ (kein HK). Dann ist v^* nur zu w_1 benachbart (einziger Nachbar von v).



v^* ist nur über w_1 erreichbar; jeder Versuch, v^* zu erreichen, bräuchte w_1 ein zweites Mal. Formal: s, t, w_2 haben alle Grad 1, ein Pfad kann aber höchstens 2 Endpunkte haben. **Nein.**

Problem TSP: Gegeben n Städte mit beliebigen Distanzen $d(u, v) > 0$ – *allgemeines* TSP, die Dreiecksungleichung wird *nicht* vorausgesetzt. Gesucht ist eine Rundtour minimaler Länge durch alle Städte. Ein Algorithmus A hat Güte α , wenn $A(I) \leq \alpha \cdot \text{OPT}(I)$ für alle Instanzen I gilt.

Zeigen Sie: Für kein $\alpha > 1$ gibt es einen Approximationsalgorithmus mit Güte α für TSP, außer $P = NP$.

Lösung.

Angenommen, es gäbe einen polynomiellen Algorithmus A mit $A(I) \leq \alpha \cdot \text{OPT}(I)$ für ein festes $\alpha > 1$. Wir lösen damit HAMILTONIANCYCLE in Polynomialzeit:

Konstruktion: Zu $G = (V, E)$ mit $n = |V|$ baue die TSP-Instanz mit Städten V und

$$d(u, v) = \begin{cases} 1 & \{u, v\} \in E, \\ \alpha n + 1 & \text{sonst.} \end{cases}$$

Fall 1: G hat einen Hamiltonkreis. Dann ist $\text{OPT} = n$ (Tour nur über echte Kanten), also $A(I) \leq \alpha n$.

Fall 2: G hat keinen Hamiltonkreis. Dann benutzt jede Tour mindestens eine Nicht-Kante, also $\text{OPT} \geq (\alpha n + 1) + (n - 1) > \alpha n$, und damit auch $A(I) > \alpha n$.

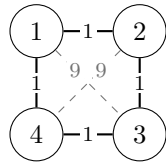
Der Vergleich „ $A(I) \leq \alpha n$?“ entscheidet also HAMILTONIANCYCLE in Polynomialzeit. Da HAMILTONIANCYCLE NP-vollständig ist, folgt $P = NP$. Also existiert ein solcher Algorithmus nur, wenn $P = NP$. $\square$

Beispiel A22 – TSP ist nicht approximierbar

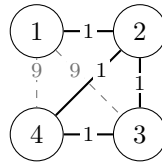
Reduktion (Gap: HK $\rightarrow$ TSP). Kante $\rightarrow$ Distanz 1, Nicht-Kante $\rightarrow \alpha n + 1$. Fest gewählt: $\alpha = 2$, $n = 4$, also $\alpha n + 1 = 9$ und Schranke $\alpha n = 8$. Der Test „ $A(I) \leq 8$?“ entscheidet HK.

Ja-Instanz. $G =$ Kreis 1–2–3–4–1 (hat HK). $\text{OPT} = n = 4$ (Tour nur über echte Kanten), also $A(I) \leq \alpha n = 8$.

Nein-Instanz. Kanten $\{1, 2\}, \{2, 3\}, \{3, 4\}, \{2, 4\}$; Knoten 1 hat Grad 1, kein HK. Jede Tour muss bei 1 eine Nicht-Kante (= 9) benutzen, also $\text{OPT} \geq 9 + 3 \cdot 1 = 12 > 8$, d. h. $A(I) > 8$.



Ja: $\text{OPT} = 4$



Nein: $\text{OPT} \geq 12$

Distanzmatrix Nein-Instanz (symm.): $d_{12}=d_{23}=d_{34}=d_{24}=1$, $d_{13}=d_{14}=9$. Solid = 1, gestrichelt = $\alpha n + 1 = 9$. Der Gütetest trennt Ja (≤ 8) von Nein (> 8) und würde HK in Polynomialzeit lösen.

Problem k -CLIQUE-DEG-3: Gegeben $G = (V, E)$ mit $\deg(v) \geq 3$ für alle $v \in V$, und $k \in \mathbb{N}_{>0}$. Existiert eine Clique C mit $|C| \geq k$?

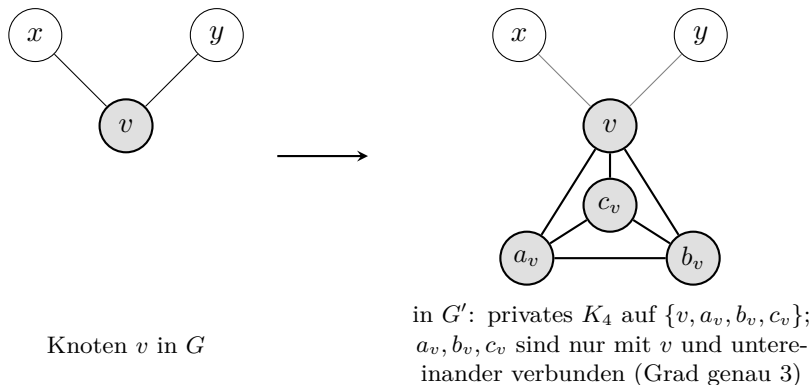
- Zeigen Sie, dass k -CLIQUE für $k \in \{1, 2, 3, 4\}$ polynomiell lösbar ist.
- Zeigen Sie, dass k -CLIQUE-DEG-3 NP-vollständig ist.

Lösung.

a) Für festes $k \leq 4$: Teste alle $\binom{|V|}{k} = O(|V|^k)$ Knotenmengen der Größe k ; pro Menge prüfe die ≤ 6 Kantenpaare in $O(1)$ (Adjazenzmatrix). Laufzeit $O(|V|^4)$, polynomiell. ($k = 1$: jeder Knoten ist eine 1-Clique.)

b) $\in NP$: Zertifikat C ; prüfe alle Paare und $|C| \geq k$ – die Instanzmenge ist nur eingeschränkt. Laufzeit $O(|V|^2)$, polynomiell.

Konstruktion (Reduktion von k -CLIQUE): Falls $k \leq 4$, löse die Instanz mit a) in Polynomialzeit und gib eine feste triviale Ja- bzw. Nein-Instanz von k -CLIQUE-DEG-3 aus. Sonst ($k \geq 5$): Hänge an jeden Knoten v drei private Knoten a_v, b_v, c_v und verbinde $\{v, a_v, b_v, c_v\}$ vollständig (K_4). Dann gilt $\deg(v) \geq 3$ für alle Knoten (Gadget-Knoten haben Grad genau 3), die Instanz ist gültig; k bleibt.



Beweis hin: Eine k -Clique in G bleibt eine k -Clique in G' (keine Kanten entfernt).

Beweis zurück: Sei C' eine k -Clique in G' mit $k \geq 5$. Jede Clique, die einen Gadget-Knoten enthält, liegt komplett in einem $\{v, a_v, b_v, c_v\}$ und hat Größe $\leq 4 < k$. Also $C' \subseteq V$ – eine k -Clique in G .

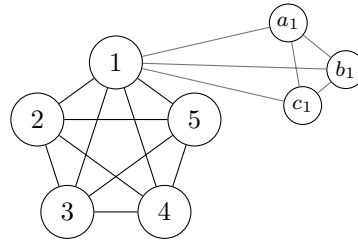
Laufzeit: $O(|V|)$ Gadgets konstanter Größe (bzw. $O(|V|^4)$ im Fall $k \leq 4$), polynomiell.

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

Beispiel A23 – k -CLIQUE-DEG-3

Reduktion (von k -CLIQUE, Fall $k \geq 5$). An jeden Knoten v ein privates K_4 auf $\{v, a_v, b_v, c_v\}$ anhängen; dann $\deg(v) \geq 3$ überall, k bleibt. (Fall $k \leq 4$ separat: Brute-Force über $O(|V|^k)$ Knotenmengen.) Gadget-Cliquen haben Größe ≤ 4 , zerstören also keine k -Clique mit $k \geq 5$ und erzeugen auch keine.

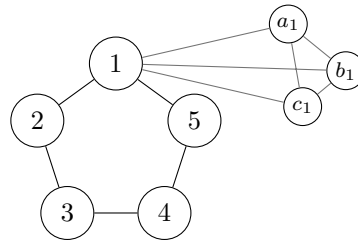
Ja-Instanz bleibt Ja. Basis K_5 auf $\{1, \dots, 5\}$, $k = 5$.



K_5 (5-Clique überlebt), an Knoten 1 ein privates K_4 (schematisch)

Die 5-Clique $\{1, \dots, 5\}$ bleibt in G' erhalten. **Ja.**

Nein-Instanz bleibt Nein. Basis C_5 (5-Kreis, größte Clique = 2), $k = 5$.



C_5 : größte Clique 2; Gadget- K_4 liefert nur Cliquen $\leq 4 < 5$

Keine 5-Clique in G' : Gadget-Cliquen sind ≤ 4 , im C_5 -Teil ≤ 2 . **Nein.**

Problem k -COLOR-PRECOLORING: Gegeben $G = (V, E)$, $k \in \mathbb{N}_{>0}$ und paarweise verschiedene Knoten $v_1, \dots, v_k \in V$. Existiert eine Färbung $f : V \rightarrow [k]$ mit $f(v_i) = i$ für alle i und $f(u) \neq f(v)$ für alle $\{u, v\} \in E$?

Zeigen Sie, dass k -COLOR-PRECOLORING NP-vollständig ist.

Lösung.

∈ **NP:** Zertifikat ist die Färbung f . Prüfe $f(v_i) = i$ für alle i und $f(u) \neq f(v)$ für jede Kante. Existiert eine gültige Färbung, wird sie akzeptiert, sonst jedes Zertifikat abgelehnt. Laufzeit $O(k + |E|)$, polynomiell.

Konstruktion (Reduktion von k -COLOR): Sei (G, k) eine k -COLOR-Instanz. Füge k neue Knoten $v_1, \dots, v_k$ hinzu, die untereinander vollständig verbunden sind (K_k), aber keine Kanten nach G haben. Gib $(G' = G \dot{\cup} K_k, k, v_1, \dots, v_k)$ aus.

Beweis k -Color nach Precoloring: Sei f eine k -Färbung von G . Erweitere sie durch $f(v_i) = i$ – die v_i sind nur untereinander verbunden und paarweise verschieden gefärbt. f ist gültig und erfüllt die Vorgabe.

Beweis Precoloring nach k -Color: Sei f eine gültige Färbung von G' mit $f(v_i) = i$. Die Einschränkung von f auf V ist eine k -Färbung von G , denn alle G -Kanten liegen in E' .

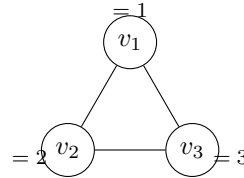
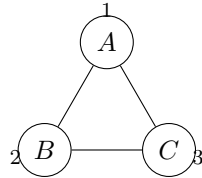
Laufzeit: $O(k^2)$ für die neue Clique, polynomiell.

Da k -COLOR NP-vollständig ist und k -COLOR-PRECOLORING ∈ NP, ist k -COLOR-PRECOLORING NP-vollständig. $\square$

Beispiel A24 – k -COLOR-PRECOLORING

Reduktion (von k -COLOR, hier $k = 3$). Zu G ein disjunktes K_k mit Knoten $v_1, \dots, v_k$ hinzufügen (keine Kanten nach G) und diese vorfärben: $f(v_i) = i$. Da das K_k isoliert ist, erzwingt die Vorfärbung nichts in G .

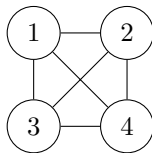
Ja-Instanz bleibt Ja. $G = \text{Dreieck } \{A, B, C\}$ (3-färbbar).



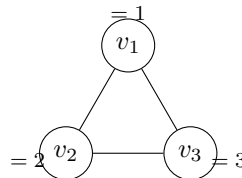
G : 3-Färbung $A=1, B=2, C=3$ disjunktes K_3 , vorgefärbt $f(v_i) = i$

$G' = G \dot{\cup} K_3$: Die 3-Färbung von G plus $f(v_i) = i$ ist gültig. **Ja.**

Nein-Instanz bleibt Nein. $G = K_4$ (nicht 3-färbbar).



$G = K_4$: nicht 3-färbbar



disjunktes vorgefärbtes K_3

$G' = K_4 \dot{\cup} K_3$: Der K_4 -Teil scheitert bereits an 3 Farben. **Nein.**

3 Vorlesungsbeweise

A25 Transitivität von Polynomialzeitreduktionen

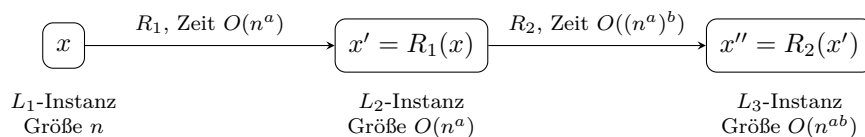
Präsenz 9.2

Zeigen Sie: Für alle Entscheidungsprobleme L_1, L_2, L_3 gilt $L_1 \leq_p L_2 \wedge L_2 \leq_p L_3 \implies L_1 \leq_p L_3$.

Lösung.

Sei R_1 eine Reduktion von L_1 auf L_2 mit Laufzeit $O(n^a)$ und R_2 eine Reduktion von L_2 auf L_3 mit Laufzeit $O(n^b)$.

Laufzeit: Sei x eine L_1 -Instanz der Größe n . $R_1(x)$ braucht $O(n^a)$ Zeit; die Ausgabe x' hat damit Größe $O(n^a)$ (mehr kann in der Zeit nicht geschrieben werden). $R_2(x')$ braucht dann $O((n^a)^b) = O(n^{ab})$ Zeit. Die Verkettung ist also polynomiell.



gesamt polynomiell: $O(n^{ab})$; $x \in L_1 \iff x' \in L_2 \iff x'' \in L_3$

Korrektheit: Mit $x'' = R_2(R_1(x))$ gilt

$$x \in L_1 \iff x' \in L_2 \quad \text{und} \quad x' \in L_2 \iff x'' \in L_3,$$

also $x \in L_1 \iff x'' \in L_3$. Damit ist $R_2 \circ R_1$ eine polynomielle Reduktion von L_1 auf L_3 . $\square$

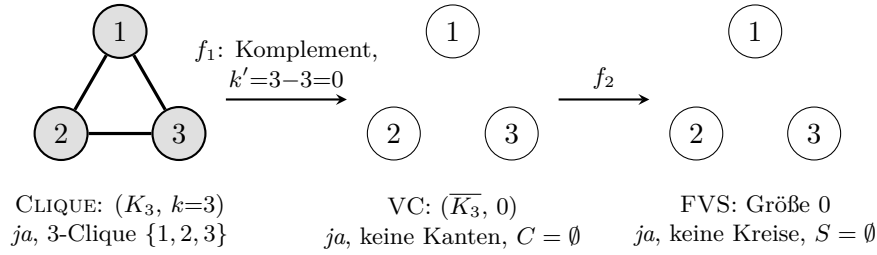
Beispiel A25 – Transitivität konkret durchgerechnet

Laufzeit mit Zahlen. Sei R_1 eine Reduktion mit Laufzeit $O(n^a)$, $a = 2$, und R_2 mit $O(n^b)$, $b = 3$. Eingabe x der Größe $n = 4$:

$$R_1(x) \leq 4^2 = 16 \text{ Schritte} \Rightarrow |x'| \leq 16, \quad R_2(x') \leq 16^3 = 4096 \text{ Schritte.}$$

Gesamt $\leq 16 + 4096 = 4112$ Schritte $= O(n^{ab}) = O(n^6)$ – polynomiell.

Konkrete Reduktionskette $\text{CLIQUE} \leq_p \text{VC} \leq_p \text{FVS}$ an der Ja-Instanz $x = (K_3, 3)$ (Standardreduktionen $f_1(G, k) = (\overline{G}, |V| - k)$, dann $\text{VC} \leq_p \text{FVS}$):



Alle drei sind Ja-Instanzen: $x \in \text{CLIQUE} \iff x' \in \text{VC} \iff x'' \in \text{FVS}$. Damit ist $f_2 \circ f_1$ eine polynomielle Reduktion $\text{CLIQUE} \leq_p \text{FVS}$.

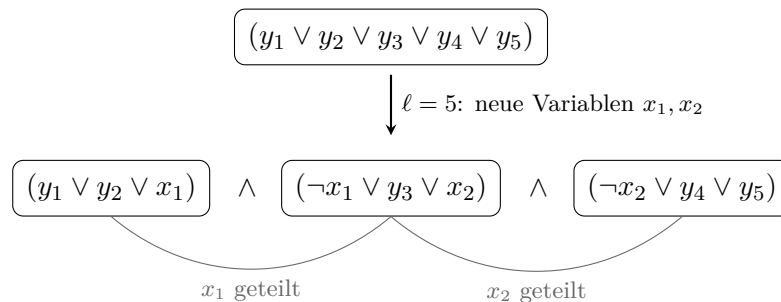
Beweisen Sie: 3-SAT ist NP-vollständig. Zeigen Sie dafür $\text{SAT} \leq_p \text{3-SAT}$; $\text{3-SAT} \in \text{NP}$ dürfen Sie annehmen.

Lösung.

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). Kurze Klauseln bleiben unverändert.



Nachbarklauseln teilen die Hilfsvariable: links positiv (x_i), rechts negiert ($\neg x_i$)

Beweis hin: Sei die Originalklausel erfüllt, etwa y_i wahr. Setze $x_1 = \dots = x_{i-2} = \text{wahr}$ und $x_{i-1} = \dots = x_{\ell-3} = \text{falsch}$. Dann ist jede Kettenklausel erfüllt: die ersten durch ihr x -Literal, die Klausel mit y_i durch y_i , die restlichen durch ihr $\neg x$ -Literal.

Beweis zurück: Sei die Kette erfüllt und angenommen, alle y_i wären falsch. Die erste Klausel erzwingt $x_1 = \text{wahr}$, damit 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 wird nur linear vergrößert ($|\bar{\alpha}| \leq c|\alpha|$), die Transformation ist polynomiell.

Da SAT NP-vollständig ist, $\text{SAT} \leq_p \text{3-SAT}$ gilt und $\text{3-SAT} \in \text{NP}$, ist 3-SAT NP-vollständig.

□

Beispiel A26 – SAT $\leq_p$ 3-SAT: eine 5er-Klausel zerlegt

Konstruktion. Die Klausel $C = (y_1 \vee y_2 \vee y_3 \vee y_4 \vee y_5)$ wird ($\ell = 5$, neue Variablen x_1, x_2) ersetzt durch

$$\underbrace{(y_1 \vee y_2 \vee x_1)}_{K_1} \wedge \underbrace{(\neg x_1 \vee y_3 \vee x_2)}_{K_2} \wedge \underbrace{(\neg x_2 \vee y_4 \vee y_5)}_{K_3}.$$

Hin (C erfüllt $\Rightarrow$ Kette erfüllt). Es sei $y_3 = \text{wahr}$ (Rest *falsch*). Regel (y_i mit $i = 3$): $x_1 = \text{wahr}$, $x_2 = \text{falsch}$.

$$K_1 = (y_1 \vee y_2 \vee x_1) = (F \vee F \vee \mathbf{T}) = \text{wahr} \quad (\text{durch } x_1)$$

$$K_2 = (\neg x_1 \vee y_3 \vee x_2) = (F \vee \mathbf{T} \vee F) = \text{wahr} \quad (\text{durch } y_3)$$

$$K_3 = (\neg x_2 \vee y_4 \vee y_5) = (\mathbf{T} \vee F \vee F) = \text{wahr} \quad (\text{durch } \neg x_2)$$

Zurück per Widerspruch (C unerfüllt $\Rightarrow$ Kette unerfüllt). Setze alle $y_i = \text{falsch}$ und nimm an, die Kette sei erfüllbar:

$$K_1 = (F \vee F \vee x_1) \quad \Rightarrow \quad x_1 = \text{wahr}$$

$$K_2 = (\neg x_1 \vee F \vee x_2) = (F \vee F \vee x_2) \quad \Rightarrow \quad x_2 = \text{wahr}$$

$$K_3 = (\neg x_2 \vee F \vee F) = (F \vee F \vee F) = \text{falsch} \quad (\perp)$$

Widerspruch: bei $y_1 = \dots = y_5 = \text{falsch}$ ist die Kette *nicht* erfüllbar. Also erzwingt jede erfüllende Belegung der Kette ein wahres y_i – genau dann ist C erfüllt. Kette und C sind erfüllbarkeitsäquivalent.

Beweisen Sie: $\text{SAT} \leq k\text{-CLIQUE}$. *Hinweis: NICHT 3-SAT, sondern allgemeines SAT!*

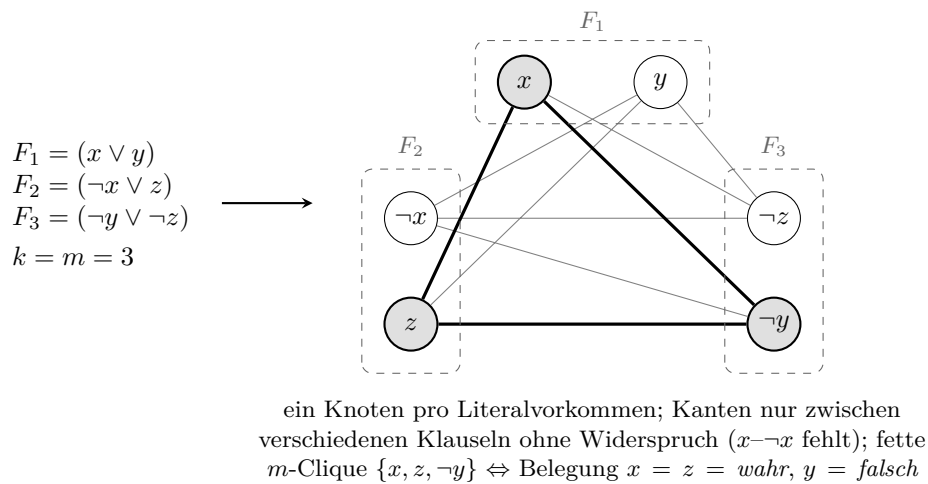
Lösung.

Konstruktion: Sei $F = F_1 \wedge \dots \wedge F_m$ eine KNF mit Klauseln $F_i = (y_{i1} \vee \dots \vee y_{i\ell_i})$. Baue $G = (V, E)$:

$$V = \{ [i, j] \mid 1 \leq i \leq m, 1 \leq j \leq \ell_i \} \quad (\text{ein Knoten pro Literalvorkommen})$$

$$E = \{ \{ [i, j], [i', j'] \} \mid i \neq i', y_{ij} \neq \neg y_{i'j'} \} \quad (\text{verschiedene Klausel, kein Widerspruch})$$

Setze $k = m$.



Beweis SAT nach Clique: Sei ψ eine erfüllende Belegung. Für jedes i existiert ein r_i mit $\psi(y_{ir_i}) = \text{wahr}$. Setze $C = \{ [i, r_i] \mid i \in [m] \}$. Wäre $\{ [i, r_i], [j, r_j] \} \notin E$ für $i \neq j$, müsste $y_{ir_i} = \neg y_{jr_j}$ gelten – aber beide sind unter ψ wahr, Widerspruch. Also ist C eine m -Clique.

Beweis Clique nach SAT: Sei C eine m -Clique. Da Knoten derselben Klausel nie verbunden sind, hat C die Form $\{ [1, r_1], \dots, [m, r_m] \}$. Setze $\psi(y_{ir_i}) = \text{wahr}$ für alle i – widerspruchsfrei, da $y_{ir_i} \neq \neg y_{jr_j}$ (sonst keine Kante). Dann ist jede Klausel F_i durch y_{ir_i} erfüllt.

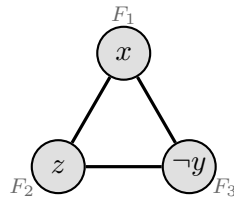
Laufzeit: $O(L^2)$ für L Literalvorkommen, polynomiell. $\square$

Beispiel A27 – SAT $\leq_p$ k -Clique: beide Richtungen an Zahlen

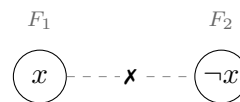
Wir nutzen $F_1 = (x \vee y)$, $F_2 = (\neg x \vee z)$, $F_3 = (\neg y \vee \neg z)$, $k = m = 3$.

SAT $\rightarrow$ Clique. Die Belegung $x = z = \text{wahr}$, $y = \text{falsch}$ wählt pro Klausel ein wahres Literal: $[F_1: x]$, $[F_2: z]$, $[F_3: \neg y]$. Je zwei stammen aus verschiedenen Klauseln und sind widerspruchsfrei ($x, z, \neg y$ paarweise verträglich) – also paarweise verbunden: eine 3-Clique.

Clique $\rightarrow$ SAT. Nein-Instanz $F = (x) \wedge (\neg x)$, $k = m = 2$: der Graph hat nur $[F_1: x]$ und $[F_2: \neg x]$; da x und $\neg x$ sich widersprechen, gibt es *keine* Kante – keine 2-Clique. Also ist F unerfüllbar.



3-Clique $\{x, z, \neg y\}$
(Ja-Instanz)



keine Kante

$F = (x) \wedge (\neg x)$: keine 2-Clique
(Nein-Instanz)

Beweisen Sie: Das Problem k -CLIQUE ist NP-vollständig.

Lösung.

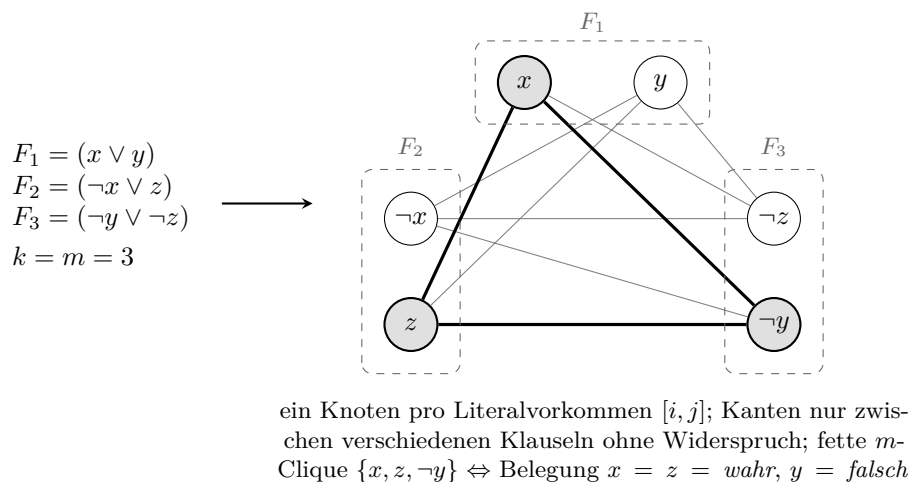
Teil 1: k -Clique $\in$ NP. Zertifikat ist eine Knotenmenge $C \subseteq V$. Der Verifizierer prüft zwei Bedingungen:

- Für alle zweielementigen $\{u, v\} \subseteq C$ gilt $\{u, v\} \in E$.
- $|C| \geq k$.

Existiert eine k -Clique, wird sie akzeptiert, sonst jedes Zertifikat abgelehnt. Laufzeit: Paartests $O(|V|^2)$, Größentest $O(|V|)$ – polynomiell.

Teil 2: k -Clique ist NP-schwer durch $\text{SAT} \leq_p k\text{-CLIQUE}$:

Konstruktion: Für $F = F_1 \wedge \dots \wedge F_m$ setze $V = \{[i, j] \mid j\text{-tes Literal in Klausel } i\}$ und $E = \{\{[i, j], [i', j']\} \mid i \neq i', y_{ij} \neq \neg y_{i'j'}\}$, $k = m$.



Beweis hin: Erfüllende Belegung ψ liefert pro Klausel ein wahres Literal y_{ir_i} ; die Knoten $[i, r_i]$ sind paarweise verbunden (verschiedene Klauseln, kein Widerspruch unter ψ) – eine m -Clique.

Beweis zurück: Eine m -Clique enthält aus jeder Klausel genau einen Knoten; die zugehörigen Literale sind widerspruchsfrei und können alle wahr gesetzt werden – jede Klausel ist erfüllt.

Laufzeit: polynomiell in der Formellänge.

Da SAT NP-vollständig ist, $\text{SAT} \leq_p k\text{-CLIQUE}$ gilt und $k\text{-CLIQUE} \in \text{NP}$, ist $k\text{-CLIQUE}$ NP-vollständig. $\square$

Beispiel A28 – k -Clique $\in$ NP: ein Zertifikat prüfen

Instanz aus der Reduktion: $F_1 = (x \vee y)$, $F_2 = (\neg x \vee z)$, $F_3 = (\neg y \vee \neg z)$, Graph G wie in A27, $k = m = 3$.

Gültiges Zertifikat $C = \{[F_1: x], [F_2: z], [F_3: \neg y]\}$. Der Verifizierer testet alle $\binom{3}{2} = 3$ Paare auf Adjazenz und $|C| \geq k$:

Paar $\{u, v\}$	versch. Klauseln?	kein Widerspruch? $\Rightarrow$ Kante
$\{x, z\}$	$F_1 \neq F_2$ ✓	x, z verträglich ✓
$\{x, \neg y\}$	$F_1 \neq F_3$ ✓	$x, \neg y$ verträglich ✓
$\{z, \neg y\}$	$F_2 \neq F_3$ ✓	$z, \neg y$ verträglich ✓

Alle drei Paare adjazent *und* $|C| = 3 \geq k = 3 \Rightarrow$ **akzeptiert**.

Ungültiges Zertifikat $C' = \{[F_1: x], [F_2: \neg x]\}$: das Paar $\{x, \neg x\}$ ist ein Widerspruch (ℓ und $\neg\ell$), also *keine* Kante – C' ist keine Clique $\Rightarrow$ **verworfen**.

Laufzeit: $\binom{|C|}{2}$ Paartests plus Größentest, $O(|V|^2)$ – polynomiell. Da für jede Ja-Instanz ein solches Zertifikat existiert, gilt k -CLIQUE $\in$ NP.

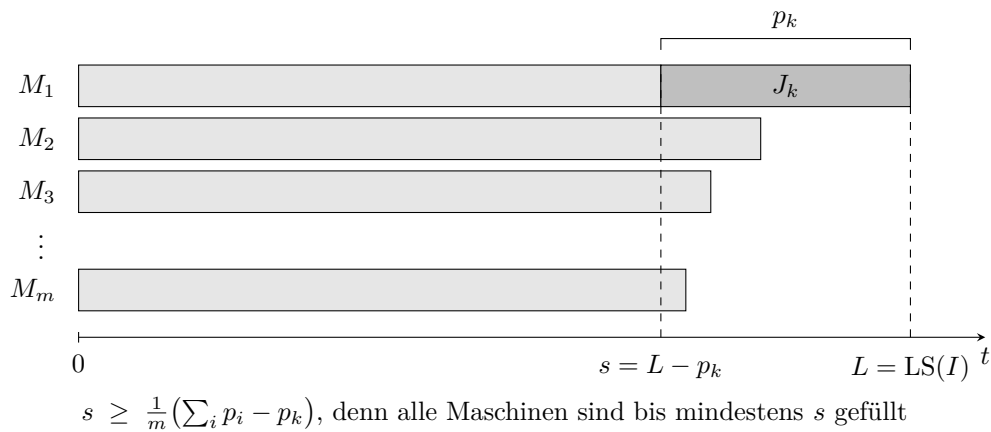
$P \parallel C_{\max}$: n Jobs mit Zeiten $p_1, \dots, p_n$, m Maschinen; minimiere $C_{\max}$. ListScheduling legt jeden Job der Reihe nach auf die Maschine mit minimaler Last. Zeigen Sie: Für alle I gilt $LS(I)/OPT(I) \leq 2 - 1/m$.

Lösung.

O.B.d.A. habe Maschine M_1 am Ende die höchste Last $L = LS(I)$. Sei J_k der letzte Job, der M_1 zugeordnet wurde.

Kernbeobachtung: Bei der Zuordnung von J_k hatte M_1 die *kleinste* Last $L - p_k$ – also hatten alle m Maschinen mindestens Last $L - p_k$. Daraus folgt

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



Zwei Schranken für das Optimum: $OPT(I) \geq \frac{1}{m} \sum_i p_i$ (Gesamtlast auf m Maschinen) und $OPT(I) \geq p_k$ (jeder Job muss irgendwo laufen).

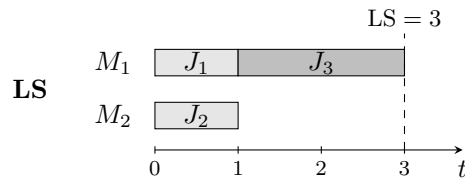
Kombination:

$$OPT(I) \geq \frac{1}{m} \sum_i p_i \geq \frac{m(L - p_k) + p_k}{m} = L - \left(1 - \frac{1}{m}\right)p_k.$$

Mit $p_k \leq OPT(I)$: $OPT(I) \geq LS(I) - \left(1 - \frac{1}{m}\right)OPT(I)$, also $LS(I) \leq \left(2 - \frac{1}{m}\right)OPT(I)$. $\square$

Beispiel A29 – ListScheduling-Güte $2 - \frac{1}{m}$ scharf: $m = 2$, Jobs $(1, 1, 2)$

Ablauf von ListScheduling (jeder Job auf die Maschine kleinster Last): $J_1 \rightarrow M_1$ (Last 1), $J_2 \rightarrow M_2$ (Last 1), $J_3 \rightarrow M_1$ (kleinste Last) $\Rightarrow$ LS = 3.



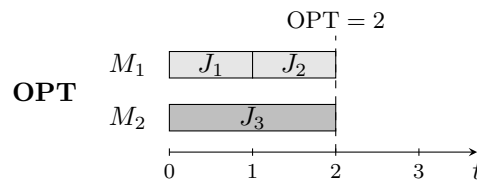
Beweis-Ungleichungen mit Zahlen. Letzter Job auf M_1 ist $J_k = J_3$, $p_k = 2$. Bei seiner Zuordnung hatte M_1 die kleinste Last $L - p_k = 3 - 2 = 1$, also alle Maschinen ≥ 1 :

$$\sum_i p_i = 1 + 1 + 2 = 4 \geq m(L - p_k) + p_k = 2 \cdot 1 + 2 = 4.$$

Zwei Optimumsschranken: $\text{OPT} \geq \frac{1}{m} \sum_i p_i = \frac{4}{2} = 2$ und $\text{OPT} \geq p_k = 2$. Kombination:

$$\text{OPT} \geq \frac{m(L - p_k) + p_k}{m} = L - \left(1 - \frac{1}{m}\right)p_k = 3 - \frac{1}{2} \cdot 2 = 2, \quad \frac{\text{LS}}{\text{OPT}} = \frac{3}{2} = 2 - \frac{1}{m}.$$

Das reale Optimum $\text{OPT} = 2$ (Jobs 1, 1 auf M_1 , Job 2 auf M_2) bestätigt die Schranke – sie ist *scharf*:



Beweisen Sie: Jede Sprache L , die eine NDTM M in nichtdeterministischer Polynomialzeit T akzeptiert, besitzt einen polynomiellen Verifizierer.

- a) Geben Sie zunächst für die Sprache der folgenden NDTM einen analog arbeitenden Verifizierer an: $M(G, k)$ rät eine Knotenmenge $C \subseteq V$, verwirft, falls eine Kante ungedeckt ist oder $|C| > k$, sonst akzeptiert sie.
- b) Führen Sie den allgemeinen Beweis.

Lösung.

a) Verifizierer $V(G, k; c)$: Das Zertifikat c beschreibt eine Knotenmenge $C \subseteq V$. Prüfe deterministisch: für alle $\{u, v\} \in E$, ob $u \in C$ oder $v \in C$; und ob $|C| \leq k$. Akzeptiere genau dann. Das Raten der NDTM ist durch das gegebene Zertifikat ersetzt; die Prüfschritte sind identisch. Laufzeit $O(|V| + |E|)$.

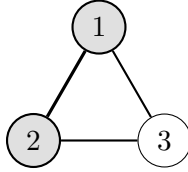
b) Sei L von der NDTM M in Zeit T akzeptiert. Definiere den Verifizierer $V(x, c)$: Interpretiere c als Folge von höchstens $T(|x|)$ nichtdeterministischen Entscheidungen. Simuliere M auf x deterministisch; bei jedem Schritt mit mehreren möglichen Übergängen wähle den, den c vorgibt. Akzeptiere genau dann, wenn diese Rechnung innerhalb von $T(|x|)$ Schritten akzeptierend endet.

Korrektheit: Ist $x \in L$, so hat M einen akzeptierenden Rechenweg der Länge $\leq T(|x|)$; die Folge seiner Entscheidungen ist ein Zertifikat, das V akzeptiert. Ist $x \notin L$, akzeptiert kein Rechenweg von M – also führt kein Zertifikat zu einer akzeptierenden Simulation.

Laufzeit und Zertifikatlänge: Die Simulation umfasst $\leq T(|x|)$ Schritte mit konstantem Aufwand pro Schritt; das Zertifikat hat Länge $O(T(|x|))$. Beides polynomiell – V ist ein polynomieller Verifizierer für L . $\square$

Beispiel A30 – NDTM $\subseteq$ Verifizierer: VC am Dreieck, $k = 2$

Die NDTM $M(G, k)$ rät $C \subseteq V$ Knoten für Knoten (ein Bit pro Knoten). Am Dreieck G mit $V = \{1, 2, 3\}$, $E = \{\{1, 2\}, \{1, 3\}, \{2, 3\}\}$ und $k = 2$:



Zertifikat $c = 110 \Leftrightarrow C = \{1, 2\}$
(Knoten 1, 2 hervorgehoben)

Das Ratebit-Wort *ist* das Zertifikat: die i -te Verzweigung der NDTM wird durch das i -te Bit von c ersetzt und deterministisch nachsimuliert.

Gültiges $c = 110$ ($C = \{1, 2\}$). Der Verifizierer prüft jede Kante gedeckt und $|C| \leq k$:

$$\{1, 2\}: 1 \in C \checkmark \quad \{1, 3\}: 1 \in C \checkmark \quad \{2, 3\}: 2 \in C \checkmark \quad |C| = 2 \leq 2 \checkmark$$

$\Rightarrow$ **akzeptiert**: $\{1, 2\}$ ist eine Knotenüberdeckung.

Ungültiges $c = 100$ ($C = \{1\}$): Kante $\{2, 3\}$ ist ungedeckt ($2 \notin C$, $3 \notin C$) $\Rightarrow$ **verworfen**.

Allgemein: Jeder akzeptierende Rechenweg von M liefert über seine Verzweigungsfolge ein Zertifikat, das der Verifizierer akzeptiert – und umgekehrt. Zertifikatlänge $O(T(|x|))$, polynomiell.

Beweisen Sie: Sei L_0 eine NP-vollständige Sprache. Falls $L_0 \in P$, dann gilt $P = NP$.

Lösung.

Es gilt stets $P \subseteq NP$ (ein Polynomialzeit-Löser ist ein Verifizierer, der das Zertifikat ignoriert). Zu zeigen bleibt $NP \subseteq P$.

Sei A ein Algorithmus, der L_0 in Zeit $O(n^c)$ entscheidet, und sei $L \in NP$ beliebig. Da L_0 NP-vollständig ist, gilt $L \leq_p L_0$; sei f die zugehörige Reduktion mit Laufzeit $O(n^a)$.

Algorithmus für L : Berechne zu einer Eingabe x (Größe n) zuerst $f(x)$ – Zeit $O(n^a)$, Ausgabegröße $O(n^a)$. Wende dann A auf $f(x)$ an – Zeit $O((n^a)^c) = O(n^{ac})$. Gesamt polynomiell.

Korrektheit: $x \in L \iff f(x) \in L_0$ (Reduktionseigenschaft), und A entscheidet $f(x) \in L_0$ korrekt.

Also $L \in P$ für jedes $L \in NP$, d. h. $NP \subseteq P$. Zusammen mit $P \subseteq NP$ folgt $P = NP$. $\square$

Beispiel A31 – L_0 NP-vollständig $\wedge L_0 \in P \Rightarrow P = NP$ konkret

Annahme: $L_0 = 3\text{-SAT}$ ist NP-vollständig *und* es gäbe einen Algorithmus A , der 3-SAT in $O(n^3)$ entscheidet.

Ein beliebiges $L \in \text{NP}$ wird polynomiell. Nimm $L = \text{CLIQUE}$. Da L_0 NP-vollständig ist, existiert eine Reduktion $f : \text{CLIQUE} \leq_p 3\text{-SAT}$, etwa mit Laufzeit $O(n^2)$. Für die Eingabe $x = (K_3, 3)$ der Größe $n = 10$:

- (1) $f(x)$ berechnen: $\leq 10^2 = 100$ Schritte, $|f(x)| = O(n^2) = 100$.
- (2) A auf $f(x)$: $\leq (10^2)^3 = 10^6$ Schritte.

Gesamt $\leq 100 + 10^6$ Schritte $= O(n^6)$ – polynomiell, und korrekt, denn $x \in \text{CLIQUE} \iff f(x) \in 3\text{-SAT}$ (Reduktionseigenschaft) und A entscheidet $f(x)$ richtig.

Schluss. CLIQUE war beliebig; dasselbe Argument gilt für *jedes* $L \in \text{NP}$ (nur f wechselt). Also $\text{NP} \subseteq \text{P}$. Mit $\text{P} \subseteq \text{NP}$ folgt $\text{P} = \text{NP}$.

4 Wahr oder falsch?

Diskutieren Sie:

- (i) Für jede Sprache, die eine NDTM in polynomieller Zeit akzeptiert, existiert eine DTM, die sie in polynomieller Zeit akzeptiert.
- (ii) A ist NP-schwer $\implies A \in P$.

Lösung.

(i) Gilt genau dann, wenn $P = NP$: Dann liegen alle NDTM-polynomiell lösbaren Probleme ($= NP$) auch in P . Falls $P \neq NP$, gilt sie nicht – ein Problem aus $NP \setminus P$ wäre sonst per DTM-Simulation in P .

(ii) Falsch. Es gibt NP-schwere Probleme, die nicht einmal in NP liegen, z. B. das Halteproblem – diese liegen erst recht nicht in P . (NP-schwer ist eine untere Schranke, $\in NP$ eine obere; beides zusammen ist NP-vollständig.) $\square$

Wahr oder falsch?

- a) Für $L \in \text{NP}$ gilt: aus $L \leq_p \text{3-SAT}$ folgt, dass L NP-vollständig ist.
- b) Gilt $L \leq_p \text{3-SAT}$ und $\text{3-SAT} \leq_p L$, dann ist L NP-vollständig.
- c) Sei L NP-vollständig. Dann gilt $L \in P \iff P = \text{NP}$.
- d) Es ist möglich, dass $\text{3-SAT} \in P$ und $\text{CLIQUE} \notin P$.

Lösung.

a) Falsch. Gegenbeispiel $L = \emptyset \in \text{NP}$: $L \leq_p \text{3-SAT}$ sagt nur, dass sich L auf 3-SAT reduzieren lässt – nicht umgekehrt, also keine Schwere.

b) Wahr. $\text{3-SAT} \leq_p L$ gibt die NP-Schwere (3-SAT ist NP-vollständig, Transitivität); $L \leq_p \text{3-SAT}$ gibt $L \in \text{NP}$. Das ist die Definition von NP-vollständig.

c) Wahr. Jedes $L' \in \text{NP}$ reduziert auf L ; liegt L in P , dann auch jedes L' – also $P = \text{NP}$. Umgekehrt folgt aus $P = \text{NP}$ sofort $L \in P$, da $L \in \text{NP}$.

d) Falsch. $\text{CLIQUE} \in \text{NP}$ reduziert auf das NP-vollständige 3-SAT; aus $\text{3-SAT} \in P$ folgt also $\text{CLIQUE} \in P$. □

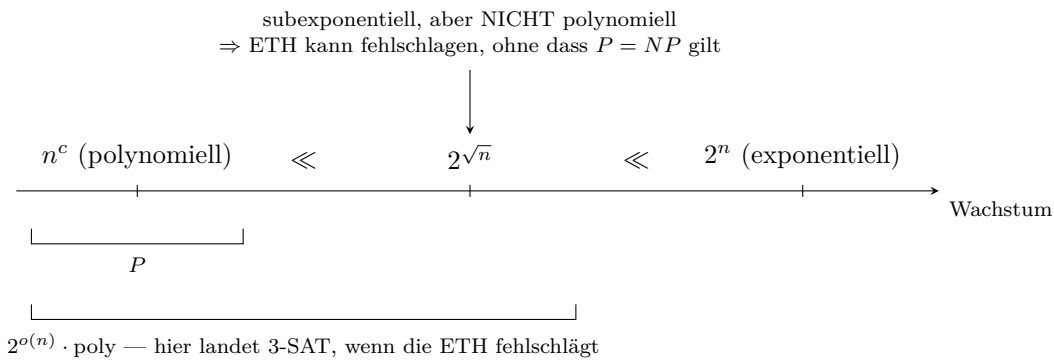
Die *Exponentialzeithypothese* (ETH) besagt: 3-SAT mit n Variablen ist nicht in Zeit $2^{o(n)} \cdot |I|^{O(1)}$ lösbar.

Beweisen oder widerlegen Sie: Wenn die ETH fehlschlägt, gilt $P = NP$.

Lösung.

Die Aussage ist falsch (nicht beweisbar; die Implikation gilt nicht).

„ETH fehlschlägt“ bedeutet nur: 3-SAT ist in $2^{o(n)} \cdot \text{poly}$ lösbar – also subexponentiell. Subexponentiell ist nicht polynomiell: Eine Laufzeit wie $2^{\sqrt{n}}$ erfüllt $2^{o(n)}$, ist aber superpolynomiell. Ein Fehlschlagen der ETH wäre also damit verträglich, dass 3-SAT (und damit alle NP-Probleme) zwar in $2^{\sqrt{n}}$, aber nicht in Polynomialzeit lösbar sind – $P \neq NP$ bliebe möglich.



Die Umkehrung gilt dagegen: Aus $P = NP$ folgt $3\text{-SAT} \in P \subseteq 2^{o(n)}$, also wäre die ETH falsch. Die behauptete Richtung folgt daraus nicht. $\square$

5 ETH-Schranken

A35 Lower Bounds: VertexCover

Hausaufgabe 12.1, 5 P.

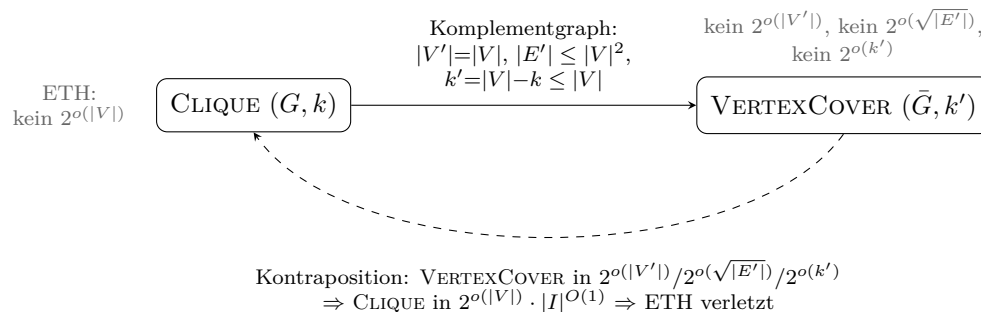
Betrachten Sie die Reduktion $\text{CLIQUE} \leq_p \text{VERTEXCOVER}$ (Komplementgraph, $k' = n - k$).

1. Geben Sie $|V'|$, $|E'|$ und k' in Abhängigkeit der Clique-Instanz an.
2. Zeigen Sie Lower Bounds für VERTEXCOVER bzgl. $|V'|$, $|E'|$ und k' basierend auf der ETH.

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

Lösung.

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



2. Für CLIQUE gilt unter der ETH: kein $2^{o(|V|)} \cdot |I|^{O(1)}$ -Algorithmus.

- **Bzgl. $|V'|$:** Angenommen, VERTEXCOVER wäre in $2^{o(|V'|)} \cdot |I|^{O(1)}$ lösbar. Wegen $|V'| = |V|$ ergäbe Reduktion + Algorithmus einen $2^{o(|V|)}$ -Algorithmus für CLIQUE – nur möglich, wenn die ETH falsch ist.
- **Bzgl. $|E'|$:** Angenommen, VERTEXCOVER wäre in $2^{o(\sqrt{|E'|})} \cdot |I|^{O(1)}$ lösbar. Wegen $|E'| \leq |V|^2$ ist $\sqrt{|E'|} = O(|V|)$ – es ergäbe sich 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 $k' \leq |V|$ ergäbe sich ein $2^{o(|V|)}$ -Algorithmus für CLIQUE . Widerspruch zur ETH. $\square$

Beispiel A35 – Lower Bounds VertexCover: konkrete Zahlen

Wir setzen eine kleine CLIQUE-Instanz ein: G mit $|V| = 5$, $|E| = 6$, gesucht $k = 3$. Die Reduktion bildet den Komplementgraphen $\bar{G}$ und setzt $k' = |V| - k$.

Parameter der VC-Instanz $(\bar{G}, k')$:

$$|V'| = |V| = 5, \quad |E'| = \binom{5}{2} - |E| = 10 - 6 = 4, \quad k' = |V| - k = 5 - 3 = 2.$$

Kontrapositions-Schluss. Ein zu schneller VC-Löser ergäbe via dieser Reduktion einen zu schnellen CLIQUE-Löser – und $2^{o(|V|)}$ ist für CLIQUE unter der ETH verboten:

- $|V'| = |V| = 5$: Ein $2^{o(|V'|)}$ -Löser wäre ein $2^{o(5)} = 2^{o(|V|)}$ -Löser für CLIQUE. Also *kein* $2^{o(|V'|)}$.
- $\sqrt{|E'|} = \sqrt{4} = 2 = O(|V|)$: Ein $2^{o(\sqrt{|E'|})}$ -Löser wäre ein $2^{o(|V|)}$ -Löser für CLIQUE. Also *kein* $2^{o(\sqrt{|E'|})}$.
- $k' = 2 \leq |V| = 5$: Ein $2^{o(k')}$ -Löser wäre ein $2^{o(|V|)}$ -Löser für CLIQUE. Also *kein* $2^{o(k')}$.

Alle drei Schranken für VERTEXCOVER folgen aus dem einen verbotenen $2^{o(|V|)}$ -Algorithmus für CLIQUE.

Reduktion 3-SAT $\leq_p$ HITTINGSET: $U = \{x_1, \bar{x}_1, \dots, x_n, \bar{x}_n\}$; $F_i = \{x_i, \bar{x}_i\}$ pro Variable; $F_{n+j} = C_j$ pro Klausel; $k = n$.

1. Geben Sie $|U|$, r und k in Abhängigkeit von n und m an.
2. Beweisen Sie Lower Bounds bzgl. $|U|$, r und k unter der ETH.

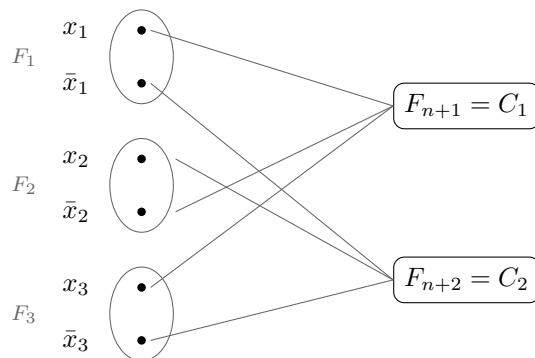
Hinweis: Unter der ETH ist 3-SAT nicht in $2^{o(n)} \cdot |I|^{O(1)}$ lösbar; mit dem Sparsification-Lemma auch nicht in $2^{o(m)} \cdot |I|^{O(1)}$. Für 3-SAT-Formeln gilt $n \leq 3m$.

Lösung.

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

$$U = \{x_1, \bar{x}_1, \dots\}, |U| = 2n$$

$$C_1 = (x_1 \vee \bar{x}_2 \vee x_3), C_2 = (\bar{x}_1 \vee x_2 \vee \bar{x}_3)$$



$$\begin{aligned} |U| &= 2n \\ r &= n + m \text{ Mengen} \\ k &= n \end{aligned}$$

$$\begin{aligned} &\text{kein } 2^{o(|U|)}, \text{ kein } 2^{o(r)}, \\ &\text{kein } 2^{o(k)} \end{aligned}$$

2.

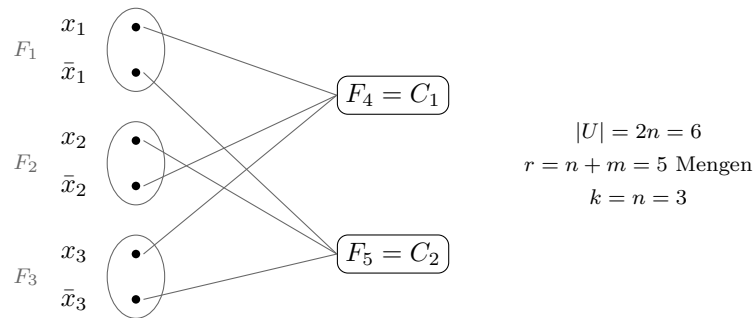
- **Bzgl. $|U|$:** Angenommen, HITTINGSET wäre in $2^{o(|U|)} \cdot |I|^{O(1)}$ lösbar. Wegen $|U| = 2n$ ergäbe Reduktion + Algorithmus einen $2^{o(n)}$ -Algorithmus für 3-SAT – direkter Widerspruch zur ETH.
- **Bzgl. r :** Angenommen, HITTINGSET wäre in $2^{o(r)} \cdot |I|^{O(1)}$ lösbar. Wegen $r = n + m$ und $n \leq 3m$ ist $r = O(m)$ – es ergäbe sich ein $2^{o(m)}$ -Algorithmus für 3-SAT. Nach dem **Sparsification-Lemma** geht das nur, wenn die ETH falsch ist.
- **Bzgl. k :** Angenommen, HITTINGSET wäre in $2^{o(k)} \cdot |I|^{O(1)}$ lösbar. Wegen $k = n$ ergäbe sich ein $2^{o(n)}$ -Algorithmus für 3-SAT – direkter Widerspruch zur ETH. $\square$

Beispiel A36 – Lower Bounds HittingSet: konkrete Instanz

Wir setzen die 3-SAT-Formel mit $n = 3$, $m = 2$ ein:

$$C_1 = (x_1 \vee \bar{x}_2 \vee x_3), \quad C_2 = (\bar{x}_1 \vee x_2 \vee \bar{x}_3).$$

Die Reduktion liefert die HITTINGSET-Instanz $U = \{x_1, \bar{x}_1, x_2, \bar{x}_2, x_3, \bar{x}_3\}$ mit den Mengen $F_i = \{x_i, \bar{x}_i\}$ (pro Variable) und $F_{n+j} = C_j$ (pro Klausel), $k = n$.



Kontrapositions-Schluss (ETH: 3-SAT nicht in $2^{o(n)}$; mit Sparsification auch nicht in $2^{o(m)}$; $n \leq 3m$):

- $|U| = 2n = 6$: Ein $2^{o(|U|)}$ -Löser wäre ein $2^{o(2n)} = 2^{o(n)}$ -Löser für 3-SAT – direkter ETH-Verstoß. Kein $2^{o(|U|)}$.
- $r = n + m = 5$: Wegen $n \leq 3m$ ist $r = O(m)$; ein $2^{o(r)}$ -Löser wäre ein $2^{o(m)}$ -Löser – Verstoß laut Sparsification-Lemma. Kein $2^{o(r)}$.
- $k = n = 3$: Ein $2^{o(k)}$ -Löser wäre ein $2^{o(n)}$ -Löser – direkter ETH-Verstoß. Kein $2^{o(k)}$.

Welche Lower Bounds ergeben sich unter der ETH (mit Sparsification-Lemma) aus den Reduktionen (1) $3\text{-SAT} \leq_p k\text{-CLIQUE}$, (2) $k\text{-CLIQUE} \leq_p k\text{-IS}$ (Komplementgraph), (3) $\text{SAT} \leq_p 3\text{-DM}$, (4) $3\text{-DM} \leq_p 3\text{-EC}$, (5) $3\text{-EC} \leq_p \text{SUBSETSUM}$? Schätzen Sie zuerst die Parameter der jeweils erzeugten Instanz ab.

Lösung.

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

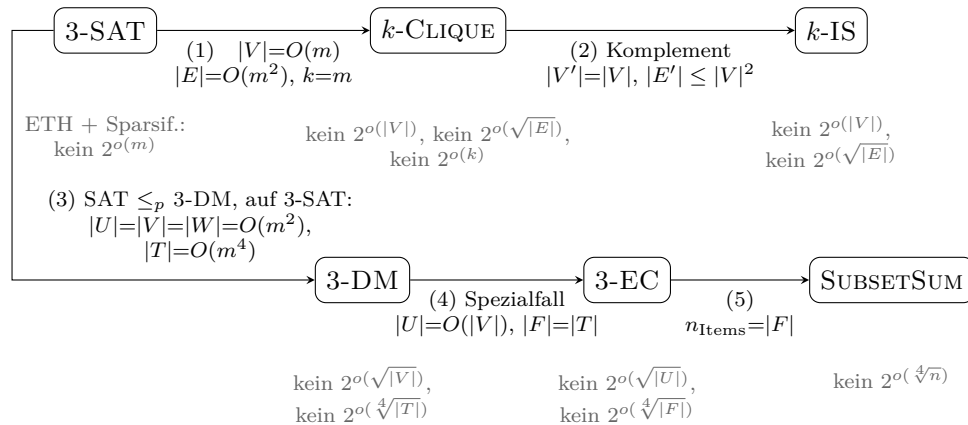
(1) Erzeugt $|V| = O(m)$, $|E| = O(m^2)$, $k = m$. Ein $2^{o(|V|)}$ -, $2^{o(\sqrt{|E|})}$ - oder $2^{o(k)}$ -Algorithmus für CLIQUE ergäbe jeweils $2^{o(m)}$ für 3-SAT – Widerspruch. Bounds: kein $2^{o(|V|)}$, kein $2^{o(\sqrt{|E|})}$, kein $2^{o(k)}$.

(2) Erzeugt $|V'| = |V|$, $|E'| \leq |V|^2$. Ein zu schneller IS-Algorithmus ergäbe via Komplementbildung denselben für CLIQUE. Bounds: kein $2^{o(|V|)}$, kein $2^{o(\sqrt{|E|})}$.

(3) Erzeugt $|U| = |V| = |W| = O(mn) \subseteq O(m^2)$ und $|T| = O(m^2n^2) \subseteq O(m^4)$ (angewandt auf 3-SAT-Instanzen). Ein $2^{o(\sqrt{|V|})}$ -Algorithmus ergäbe $2^{o(m)}$ für 3-SAT; ebenso ein $2^{o(\sqrt[4]{|T|})}$ -Algorithmus. Bounds: kein $2^{o(\sqrt{|V|})}$, kein $2^{o(\sqrt[4]{|T|})}$.

(4) 3-DM ist Spezialfall von 3-EC mit $|U| = O(|V|)$ und $|F| = |T|$. Die Bounds übertragen sich: kein $2^{o(\sqrt{|U|})}$, kein $2^{o(\sqrt[4]{|F|})}$.

(5) Erzeugt $n_{\text{Items}} = |F|$. Ein $2^{o(\sqrt[4]{n})}$ -Algorithmus für SUBSETSUM ergäbe $2^{o(\sqrt[4]{|F|})}$ für 3-EC – nach (4) verboten. Bound: kein $2^{o(\sqrt[4]{n})}$. $\square$



Beispiel A37 – Lower Bounds entlang der Kette: $m = 4$ durchpropagiert

Wir starten mit einer 3-SAT-Formel mit $m = 4$ Klauseln (unter der ETH mit Sparsification kein $2^{o(m)} = 2^{o(4)}$ -Löser) und verfolgen die von den Reduktionen erzeugten Instanzgrößen. Zur Illustration setzen wir die O -Schranken als konkrete Werte an ($|V| = 3m$ usw.):

Schritt	erzeugte Instanz	Größen bei $m = 4$
(1) $3\text{-SAT} \leq_p k\text{-CLIQUE}$	$ V = O(m), E = O(m^2), k = m$	$ V = 12, E = 16, k = 4$
(3) $\text{SAT} \leq_p 3\text{-DM}$	$ U = O(m^2), T = O(m^4)$	$ U = 16, T = 256$
(4) $3\text{-DM} \leq_p 3\text{-EC}$	$ U = O(V), F = T $	$ U = 16, F = 256$
(5) $3\text{-EC} \leq_p \text{SUBSETSUM}$	$n_{\text{Items}} = F $	$n = 256$

Exponenten-Verkettung (Kontraposition). Ein $2^{o(\sqrt[4]{n})}$ -Löser für SUBSETSUM hätte bei $n = 256$ den Exponenten $\sqrt[4]{256} = 4 = m$. Über die Kette:

$$2^{o(\sqrt[4]{n})}\text{-SubsetSum} \Rightarrow 2^{o(\sqrt[4]{|F|})}\text{-3-EC} \Rightarrow 2^{o(\sqrt[4]{|T|})}\text{-3-DM} \Rightarrow 2^{o(m)}\text{-3-SAT},$$

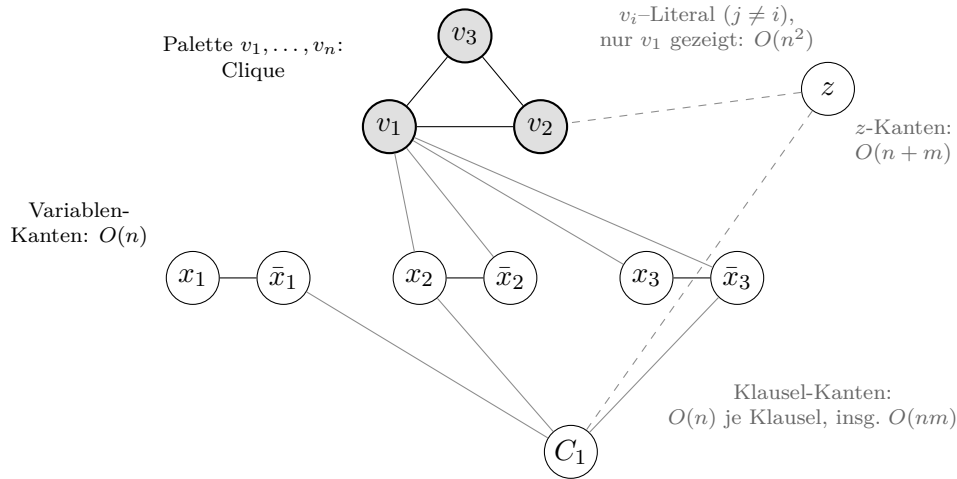
denn $\sqrt[4]{|T|} = \sqrt[4]{m^4} = m$. Ein $2^{o(m)}$ -Löser für 3-SAT ist unter der ETH (Sparsification) verboten – also existiert keiner der Löser in der Kette. Analog liefert Schritt (1) über $\sqrt{|E|} = \sqrt{m^2} = m$ und $k = m$ die Clique-Schranken kein $2^{o(\sqrt{|E|})}$, kein $2^{o(k)}$.

Die Reduktion $3\text{-SAT} \leq_p k\text{-COLOR}$ erzeugt $V = \{x_i, \bar{x}_i, v_i \mid i \in [n]\} \cup \{C_j \mid j \in [m]\} \cup \{z\}$ mit den Kanten des Vorlesungs-Gadgets. Welche Lower Bounds ergeben sich unter der ETH bzgl. (i) $|V|$ und (ii) $|E|$?

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

Lösung.

Parameter: $|V| = 3n + m + 1$; wegen $n \leq 3m$ gilt $|V| = O(m)$. Für die Kanten: v_i -Clique und v_i -Literal-Kanten $O(n^2)$, Variablen-Kanten $O(n)$, Klausel-Kanten $O(nm)$, z -Kanten $O(n + m)$ – insgesamt $|E| = O(n^2 + nm + m) = O(m^2)$.



schematisch — $|V| = 3n + m + 1 = O(m)$, $|E| = O(n^2 + nm + m) = O(m^2) \Rightarrow$ kein $2^{o(|V|)}$, kein $2^{o(\sqrt{|E|})}$

(i) Angenommen, $k\text{-COLOR}$ wäre in $2^{o(|V|)} \cdot |I|^{O(1)}$ lösbar. Wegen $|V| = O(m)$ ergäbe sich ein $2^{o(m)}$ -Algorithmus für 3-SAT – nach dem Sparsification-Lemma nur möglich, wenn die ETH falsch ist. Bound: kein $2^{o(|V|)}$.

(ii) Angenommen, $k\text{-COLOR}$ wäre in $2^{o(\sqrt{|E|})} \cdot |I|^{O(1)}$ lösbar. Wegen $|E| = O(m^2)$ ist $\sqrt{|E|} = O(m)$ – es ergäbe sich ein $2^{o(m)}$ -Algorithmus für 3-SAT . Widerspruch (Sparsification-Lemma). Bound: kein $2^{o(\sqrt{|E|})}$. $\square$

Beispiel A38 – Lower Bounds k -Color: konkrete Zahlen

Wir setzen $n = 3$, $m = 2$ ein. Die Reduktion erzeugt die Knotenmenge $V = \{x_i, \bar{x}_i, v_i \mid i \in [3]\} \cup \{C_1, C_2\} \cup \{z\}$:

$$|V| = 3n + m + 1 = 9 + 2 + 1 = 12, \quad |E| = O(m^2).$$

Wegen $n \leq 3m$ (hier $3 \leq 6$) ist $|V| = 3n + m + 1 = O(m)$ und $\sqrt{|E|} = O(m)$.

Kontrapositions-Schluss (ETH mit Sparsification: kein $2^{o(m)}$ für 3-SAT):

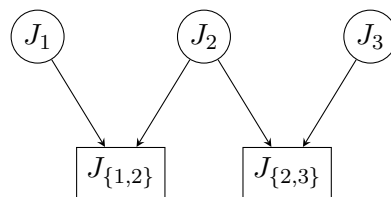
- $|V| = 12 = O(m)$: Ein $2^{o(|V|)}$ -Löser für k -COLOR wäre ein $2^{o(m)}$ -Löser für 3-SAT – Sparsification-Verstoß. Kein $2^{o(|V|)}$.
- $\sqrt{|E|} = O(m)$: Ein $2^{o(\sqrt{|E|})}$ -Löser wäre ebenso ein $2^{o(m)}$ -Löser für 3-SAT – Verstoß. Kein $2^{o(\sqrt{|E|})}$.

Beide Schranken folgen aus dem verbotenen $2^{o(m)}$ -Algorithmus für 3-SAT.

Reduktion von k -CLIQUE: $n = 4|V| + 3|E|$ Jobs (Knoten-Jobs $p = 1$, Kanten-Jobs $p = 2$, Dummy-Jobs $p = 1$), Präzedenzen $(J_i, J_{\{i,j\}})$, Makespan $T = 2|V| + 2|E|$. Nutzen Sie die bekannten Clique-Bounds (zusammenhängender Graph): kein $2^{o(|V|)}$, kein $2^{o(\sqrt{|E|})}$. Welche Lower Bounds ergeben sich bzgl. (i) n und (ii) T ?

Lösung.

Parameter: Da der Clique-Graph zusammenhängend ist, gilt $|V| \leq |E| + 1$, also $n = 4|V| + 3|E| = O(|E|)$ und ebenso $T = 2|V| + 2|E| = O(|E|)$.



- Knoten-Job je $v \in V$: $p = 1$
- Kanten-Job je $e \in E$: $p = 2$,
Vorgänger $J_i \rightarrow J_{\{i,j\}}$
dazu Dummy-Jobs ($p = 1$)

$$n = 4|V| + 3|E| = O(|E|)$$

$$T = 2|V| + 2|E| = O(|E|)$$

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

(i) Angenommen, das Scheduling-Problem wäre in $2^{o(\sqrt{n})} \cdot |I|^{O(1)}$ lösbar. Wegen $n = O(|E|)$ ist $\sqrt{n} = O(\sqrt{|E|})$ – Reduktion + Algorithmus ergäben einen $2^{o(\sqrt{|E|})}$ -Algorithmus für CLIQUE, den es unter der ETH nicht gibt. Bound: kein $2^{o(\sqrt{n})}$.

(ii) Analog mit $T = O(|E|)$: Ein $2^{o(\sqrt{T})}$ -Algorithmus ergäbe $2^{o(\sqrt{|E|})}$ für CLIQUE. Bound: kein $2^{o(\sqrt{T})}$. □

Beispiel A39 – Lower Bounds Scheduling: Dreieck einsetzen

Als CLIQUE-Instanz nehmen wir den Dreiecksgraphen K_3 : $|V| = 3$, $|E| = 3$ (zusammenhängend). Die Reduktion erzeugt

$$n = 4|V| + 3|E| = 12 + 9 = 21 \text{ Jobs}, \quad T = 2|V| + 2|E| = 6 + 6 = 12 \text{ (Makespan)}.$$

Da der Graph zusammenhängend ist ($|V| \leq |E| + 1$), wachsen $n = 4|V| + 3|E|$ und $T = 2|V| + 2|E|$ in der Reduktionsfamilie nur *linear* in $|E|$; an dieser Instanz: $n = 21$, $T = 12$ bei $|E| = 3$.

Kontrapositions-Schluss (Clique-Bounds für zusammenhängende Graphen: kein $2^{o(|V|)}$, kein $2^{o(\sqrt{|E|})}$):

- $\sqrt{n} = O(\sqrt{|E|})$: Ein $2^{o(\sqrt{n})}$ -Scheduler wäre via Reduktion ein $2^{o(\sqrt{|E|})}$ -Löser für CLIQUE – ETH-Verstoß. Kein $2^{o(\sqrt{n})}$.
- $\sqrt{T} = O(\sqrt{|E|})$: analog ergäbe ein $2^{o(\sqrt{T})}$ -Scheduler einen $2^{o(\sqrt{|E|})}$ -Löser für CLIQUE. Kein $2^{o(\sqrt{T})}$.

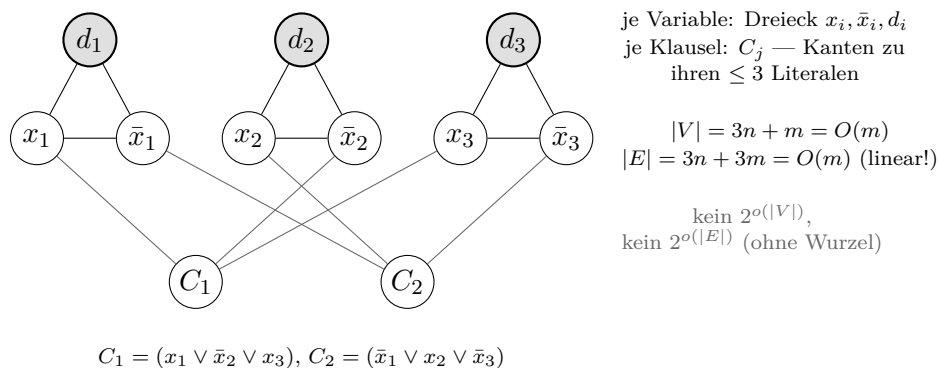
Ohne Wurzel (also kein $2^{o(n)}$, kein $2^{o(T)}$) folgt *nichts*: die Clique-Schranke ist selbst nur $2^{o(\sqrt{|E|})}$, und $n, T = O(|E|)$ liefern nur $\sqrt{n}, \sqrt{T} = O(\sqrt{|E|})$.

Reduktion $3\text{-SAT} \leq_p \text{DOMINATINGSET}$: pro Variable die Knoten $x_i, \bar{x}_i, d_i$ mit Dreieckskanten; pro Klausel ein Knoten C_j mit Kanten zu seinen Literalen. Welche Lower Bounds ergeben sich unter der ETH bzgl. (i) $|V|$ und (ii) $|E|$?

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

Lösung.

Parameter: $|V| = 3n + m$; wegen $n \leq 3m$ gilt $|V| = O(m)$. Kanten: $3n$ Dreieckskanten plus höchstens $3m$ Klausel-Literal-Kanten (je Klausel ≤ 3 Literale), also $|E| = 3n + 3m = O(m)$ – linear in m .



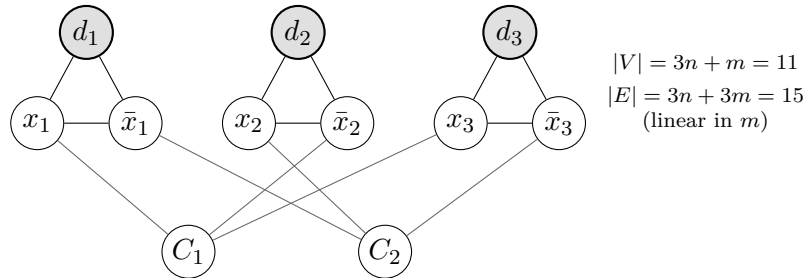
(i) Angenommen, DOMINATINGSET wäre in $2^{o(|V|)} \cdot |I|^{O(1)}$ lösbar. Wegen $|V| = O(m)$ ergäbe sich ein $2^{o(m)}$ -Algorithmus für 3-SAT – nach dem Sparsification-Lemma Widerspruch zur ETH. Bound: kein $2^{o(|V|)}$.

(ii) Angenommen, DOMINATINGSET wäre in $2^{o(|E|)} \cdot |I|^{O(1)}$ lösbar. Wegen $|E| = O(m)$ ergäbe sich ebenfalls ein $2^{o(m)}$ -Algorithmus für 3-SAT – Widerspruch (Sparsification-Lemma). Bound: kein $2^{o(|E|)}$ (hier ohne Wurzel, da die Kantenzahl nur linear wächst). $\square$

Beispiel A40 – Lower Bounds DominatingSet: konkrete Instanz

Wir setzen $n = 3$, $m = 2$ ein, mit $C_1 = (x_1 \vee \bar{x}_2 \vee x_3)$, $C_2 = (\bar{x}_1 \vee x_2 \vee \bar{x}_3)$. Pro Variable ein Dreieck $x_i, \bar{x}_i, d_i$; pro Klausel ein Knoten C_j mit Kanten zu seinen Literalen.

$$|V| = 3n + m = 9 + 2 = 11, \quad |E| = 3n + 3m = 9 + 6 = 15 \quad (\text{linear in } m!).$$



Kontrapositions-Schluss (ETH mit Sparsification: kein $2^{o(m)}$ für 3-SAT; $n \leq 3m$):

- $|V| = 11 = O(m)$: Ein $2^{o(|V|)}$ -Löser für DOMINATINGSET wäre ein $2^{o(m)}$ -Löser für 3-SAT – Verstoß. Kein $2^{o(|V|)}$.
- $|E| = 15 = O(m)$ *linear*: Ein $2^{o(|E|)}$ -Löser (*ohne* Wurzel!) wäre ebenso ein $2^{o(m)}$ -Löser – Verstoß. Kein $2^{o(|E|)}$.

Besonderheit: Weil $|E|$ hier nur linear (statt quadratisch) in m wächst, gilt die Schranke *ohne* Wurzel – $2^{o(|E|)}$ statt $2^{o(\sqrt{|E|})}$.

Reduktion von CLIQUE (mit $c \cdot n \leq k \leq n$ erlaubt): $S_{i,i} = \{(a, a) \mid a \in V\}$, $S_{i,j} = \{(a, b) \mid a \neq b, \{a, b\} \in E\}$ für $i \neq j$; k übernommen. Welche Lower Bounds ergeben sich bzgl. (i) $X = \sum_{(i,j)} |S_{i,j}|$ und (ii) $Y = \max_{(i,j)} |S_{i,j}|$?

Lösung.

Parameter (mit $N := |V|$, $k = \Theta(N)$): Diagonale Mengen haben N Tupel, Nebendiagonal-Mengen $2|E| \leq N^2$ Tupel. Also $X \leq k \cdot N + k^2 \cdot N^2 = O(N^4)$ und $Y \leq \max(N, N^2) = O(N^2)$.

$k \times k$ Zellen (hier $k = 3$), k von CLIQUE übernommen, $k = \Theta(N)$, $N = |V|$

	$j = 1$	$j = 2$	$j = 3$
$i = 1$	$\{(a, a) \mid a \in V\}$	$\{(a, b) \mid \{a, b\} \in E\}$	$\{(a, b) \mid \{a, b\} \in E\}$
$i = 2$	$\{(a, b) \mid \{a, b\} \in E\}$	$\{(a, a) \mid a \in V\}$	$\{(a, b) \mid \{a, b\} \in E\}$
$i = 3$	$\{(a, b) \mid \{a, b\} \in E\}$	$\{(a, b) \mid \{a, b\} \in E\}$	$\{(a, a) \mid a \in V\}$

$$|S_{i,i}| = N, \quad |S_{i,j}| = 2|E| \leq N^2 \Rightarrow X = \sum_{(i,j)} |S_{i,j}| = O(N^4), \quad Y = \max_{(i,j)} |S_{i,j}| = O(N^2)$$

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

(i) Angenommen, GRIDTILING wäre in $2^{o(\sqrt[4]{X})} \cdot |I|^{O(1)}$ lösbar. Wegen $X = O(N^4)$ ist $\sqrt[4]{X} = O(N)$ – es ergäbe sich ein $2^{o(N)}$ -Algorithmus für CLIQUE, den es unter der ETH nicht gibt. Bound: kein $2^{o(\sqrt[4]{X})}$.

(ii) Angenommen, GRIDTILING wäre in $2^{o(\sqrt{Y})} \cdot |I|^{O(1)}$ lösbar. Wegen $Y = O(N^2)$ ist $\sqrt{Y} = O(N)$ – wieder ein $2^{o(N)}$ -Algorithmus für CLIQUE. Widerspruch. Bound: kein $2^{o(\sqrt{Y})}$. $\square$

Beispiel A41 – Lower Bounds GridTiling: 3×3 -Instanz

CLIQUE-Instanz: $N = |V| = 4$, $|E| = 5$, gesucht $k = 3$. Die Reduktion baut ein $k \times k = 3 \times 3$ -Gitter: Diagonalzellen $S_{i,i} = \{(a, a)\}$ mit $|S_{i,i}| = N = 4$, Nebendiagonalzellen $S_{i,j} = \{(a, b) : \{a, b\} \in E\}$ mit $|S_{i,j}| = 2|E| = 10$.

	$j = 1$	$j = 2$	$j = 3$	
$i = 1$	4	10	10	grau (Diagonale): $ S_{i,i} = N = 4$ weiß: $ S_{i,j} = 2 E = 10$ 3×3 -Gitter, $k = 3 = \Theta(N)$
$i = 2$	10	4	10	
$i = 3$	10	10	4	

Parameter: $X = \sum_{(i,j)} |S_{i,j}| = 3 \cdot 4 + 6 \cdot 10 = 72 = O(N^4)$ und $Y = \max_{(i,j)} |S_{i,j}| = \max(4, 10) = 10 = O(N^2)$.

Kontrapositions-Schluss (ETH: kein $2^{o(N)}$ für CLIQUE):

- $\sqrt[4]{X} = \sqrt[4]{72} \approx 2,9 = O(N)$: Ein $2^{o(\sqrt[4]{X})}$ -Löser für GRIDTILING wäre ein $2^{o(N)}$ -Löser für CLIQUE – ETH-Verstoß. Kein $2^{o(\sqrt[4]{X})}$.
- $\sqrt{Y} = \sqrt{10} \approx 3,2 = O(N)$: Ein $2^{o(\sqrt{Y})}$ -Löser wäre ebenso ein $2^{o(N)}$ -Löser für CLIQUE. Kein $2^{o(\sqrt{Y})}$.

Reduktion 3-SAT $\leq_p$ SUBSETSUM mit Dezimalziffern: pro Variable zwei Items a_i, b_i ($s = 10^{i-1} + \sum 10^{n+j-1}$ über die Klauseln mit x_i bzw. $\bar{x}_i$), pro Klausel zwei Items c_j, d_j mit $s = 10^{n+j-1}$; Ziel $B = \sum_j 3 \cdot 10^{n+j-1} + \sum_i 10^{i-1}$. Welche Lower Bounds ergeben sich bzgl. (i) der Anzahl unterschiedlicher Itemgrößen d und (ii) der Kodierungslänge $L = \log \Delta$ der größten Itemgröße?

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

Lösung.

Parameter: Es gibt $2n + 2m$ Items; die Klausel-Items c_j, d_j teilen sich je eine Größe, also $d \leq 2n + m$. Wegen $n \leq 3m$ gilt $d = O(m)$. Die größte Größe ist $\Delta < 10^{n+m+1}$, also $L = \log \Delta = O(n + m) = O(m)$.

$$\text{Beispiel } n = 3, m = 2: \quad \varphi = \underbrace{(x_1 \vee x_2 \vee x_3)}_{C_1} \wedge \underbrace{(\bar{x}_1 \vee \bar{x}_2 \vee x_3)}_{C_2}$$

Item	10^4 (C_2)	10^3 (C_1)	10^2 (x_3)	10^1 (x_2)	10^0 (x_1)	
a_1 (x_1)	0	1	0	0	1	$10^0 + 10^3$
b_1 ($\bar{x}_1$)	1	0	0	0	1	$10^0 + 10^4$
a_2 (x_2)	0	1	0	1	0	$10^1 + 10^3$
b_2 ($\bar{x}_2$)	1	0	0	1	0	$10^1 + 10^4$
a_3 (x_3)	1	1	1	0	0	$10^2 + 10^3 + 10^4$
b_3 ($\bar{x}_3$)	0	0	1	0	0	10^2
$c_1 = d_1$	0	1	0	0	0	10^3 (je zweimal)
$c_2 = d_2$	1	0	0	0	0	10^4 (je zweimal)
B	3	3	1	1	1	Ziel

$2n + 2m$ Items, aber $d \leq 2n + m = O(m)$ Größen; $\Delta < 10^{n+m+1}$, also $L = \log \Delta = O(m) \Rightarrow$ kein $2^{o(d)}$, kein $2^{o(L)}$

(i) Angenommen, SUBSETSUM wäre in $2^{o(d)} \cdot |I|^{O(1)}$ lösbar. Wegen $d = O(m)$ ergäbe Reduktion + Algorithmus einen $2^{o(m)}$ -Algorithmus für 3-SAT – nach dem Sparsification-Lemma Widerspruch zur ETH. Bound: kein $2^{o(d)}$.

(ii) Angenommen, SUBSETSUM wäre in $2^{o(L)} \cdot |I|^{O(1)}$ lösbar. Wegen $L = O(m)$ ergäbe sich ebenso ein $2^{o(m)}$ -Algorithmus für 3-SAT – Widerspruch (Sparsification-Lemma). Bound: kein $2^{o(L)}$. $\square$

Beispiel A42 – Lower Bounds SubsetSum: Parameter der $n=3, m=2$ -Instanz

Wir nutzen die Instanz aus der Musterlösung ($n = 3, m = 2, \varphi = (x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3)$; siehe Ziffern-Tabelle dort) und setzen die Parameter ein.

Anzahl Items: $2n + 2m = 6 + 4 = 10$ (je Variable a_i, b_i , je Klausel c_j, d_j).

Verschiedene Itemgrößen: Die Klausel-Items c_j, d_j teilen sich je *eine* Größe (10^{n+j-1}), also

$$d \leq 2n + m = 6 + 2 = 8 \quad (< 10).$$

Wegen $n \leq 3m$ ist $d = O(m)$.

Kodierungslänge: Die größte Größe ist $\Delta < 10^{n+m+1} = 10^6$, also $L = \log \Delta = O(n + m) = O(m)$; konkret $L < 6$ Dezimalstellen.

Kontrapositions-Schluss (ETH mit Sparsification: kein $2^{o(m)}$ für 3-SAT):

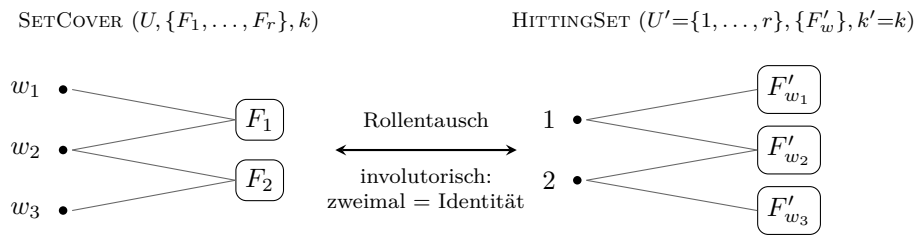
- $d = 8 = O(m)$: Ein $2^{o(d)}$ -Löser für SUBSETSUM wäre ein $2^{o(m)}$ -Löser für 3-SAT – Sparsification-Verstoß. Kein $2^{o(d)}$.
- $L = O(m)$: Ein $2^{o(L)}$ -Löser wäre ebenso ein $2^{o(m)}$ -Löser – Verstoß. Kein $2^{o(L)}$.

Schärfe der Reduktion: obwohl 10 Items vorliegen, gibt es nur $d \leq 8$ Größen, und die Zahlen bleiben mit $L < 6$ Stellen kurz – beides linear in m .

Gegeben: HITTINGSET ist unter der ETH nicht in $2^{o(r')} \langle I \rangle^{O(1)}$ und nicht in $2^{o(|U'|)} \langle I \rangle^{O(1)}$ lösbar. Die Reduktion $\text{SETCOVER} \leq_p \text{HITTINGSET}$ vertauscht die Rollen: $U' = \{1, \dots, r\}$ und pro $w \in U$ die Menge $F'_w = \{v \mid w \in F_v\}$, $k' = k$. Welche Lower Bounds ergeben sich für SETCOVER bzgl. (i) $|U|$ und (ii) r ?

Lösung.

Die Abbildung ist involutorisch (Elemente und Mengen tauschen die Plätze; zweimal angewandt entsteht die Ausgangsinstanz). Sie ist damit zugleich eine Reduktion $\text{HITTINGSET} \leq_p \text{SETCOVER}$, bei der aus einer HittingSet-Instanz mit Universum U' und r' Mengen eine SetCover-Instanz mit $|U| = r'$ und $r = |U'|$ entsteht.



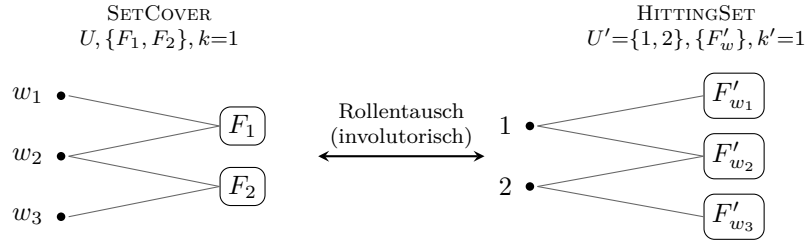
$$F'_w = \{v \mid w \in F_v\}: \text{ aus } (U', r') \text{ wird } |U| = r', r = |U'| \Rightarrow \text{kein } 2^{o(|U|)}, \text{kein } 2^{o(r)}$$

(i) Angenommen, SETCOVER wäre in $2^{o(|U|)} \langle I \rangle^{O(1)}$ lösbar. Über die Reduktion entstünde ein Algorithmus für HITTINGSET in $2^{o(r')} \langle I \rangle^{O(1)}$ – laut Voraussetzung unter der ETH ausgeschlossen. Bound: kein $2^{o(|U|)}$.

(ii) Angenommen, SETCOVER wäre in $2^{o(r)} \langle I \rangle^{O(1)}$ lösbar. Über die Reduktion entstünde ein $2^{o(|U'|)}$ -Algorithmus für HITTINGSET – ebenfalls ausgeschlossen. Bound: kein $2^{o(r)}$. $\square$

Beispiel A43 – Lower Bounds SetCover: Rollentausch konkret

SETCOVER-Instanz: $U = \{w_1, w_2, w_3\}$, $F_1 = \{w_1, w_2\}$, $F_2 = \{w_2, w_3\}$, $k = 1$ (also $r = 2$ Mengen). Der involutorische Rollentausch $F'_w = \{v : w \in F_v\}$, $U' = \{1, \dots, r\}$ liefert die HITTINGSET-Instanz.



Inzidenz: $F'_{w_1} = \{1\}$, $F'_{w_2} = \{1, 2\}$, $F'_{w_3} = \{2\}$. Somit

$$|U| = r' = 3, \quad r = |U'| = 2.$$

Kontrapositions-Schluss (Voraussetzung: HITTINGSET unter der ETH nicht in $2^{o(r')}$ und nicht in $2^{o(|U'|)}$):

- $|U| = r'$: Ein $2^{o(|U|)}$ -Löser für SETCOVER wäre via Rollentausch ein $2^{o(r')}$ -Löser für HITTINGSET – verboten. Kein $2^{o(|U|)}$.
- $r = |U'|$: Ein $2^{o(r)}$ -Löser wäre ein $2^{o(|U'|)}$ -Löser für HITTINGSET – verboten. Kein $2^{o(r)}$.

Reduktion $3\text{-SAT} \leq_p \text{ILP-FEASIBILITY}$: $N = 2n$ ILP-Variablen (eine pro Literal); 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 $-x_v - x_{\bar{v}} \leq -1$. Welche Lower Bounds ergeben sich bzgl. (i) M (Zeilenzahl) und (ii) N (Spaltenzahl)?

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

Lösung.

Parameter: $M = m + 2n$; wegen $n \leq 3m$ gilt $M = O(m)$. Und $N = 2n$.

(i) Angenommen, ILP-FEASIBILITY wäre in $2^{o(M)} \langle I \rangle^{O(1)}$ lösbar. Wegen $M = O(m)$ ergäbe Reduktion + Algorithmus einen $2^{o(m)}$ -Algorithmus für 3-SAT – nach dem Sparsification-Lemma Widerspruch zur ETH. Bound: kein $2^{o(M)}$.

(ii) Angenommen, ILP-FEASIBILITY wäre in $2^{o(N)} \langle I \rangle^{O(1)}$ lösbar. Wegen $N = 2n$ ergäbe sich ein $2^{o(n)}$ -Algorithmus für 3-SAT – direkter Widerspruch zur ETH. Bound: kein $2^{o(N)}$. $\square$

Beispiel A44 – Lower Bounds ILP-Feasibility: Ungleichungen einsetzen

$n = 3, m = 2$ mit $C_1 = (x_1 \vee x_2 \vee x_3), C_2 = (\bar{x}_1 \vee x_2 \vee \bar{x}_3)$. Pro Literal eine 0/1-Variable $x_{(\cdot)}$; also $N = 2n = 6$ Variablen $x_{x_1}, x_{\bar{x}_1}, \dots, x_{x_3}, x_{\bar{x}_3}$.

Ungleichungen ($M = m + 2n = 2 + 6 = 8$ Zeilen):

$$\text{je Klausel:} \quad - (x_{x_1} + x_{x_2} + x_{x_3}) \leq -1, \quad - (x_{\bar{x}_1} + x_{x_2} + x_{\bar{x}_3}) \leq -1;$$

$$\text{je Variable } v: \quad x_{x_v} + x_{\bar{x}_v} \leq 1, \quad -x_{x_v} - x_{\bar{x}_v} \leq -1 \quad (v = 1, 2, 3).$$

Das sind $m = 2$ Klausel- plus $2n = 6$ Variablen-Zeilen, zusammen $M = 8$.

Kontrapositions-Schluss (ETH: kein $2^{o(n)}$ für 3-SAT; mit Sparsification kein $2^{o(m)}$; $n \leq 3m$):

- $M = 8 = O(m)$: Ein $2^{o(M)}$ -Löser für ILP-FEASIBILITY wäre ein $2^{o(m)}$ -Löser für 3-SAT – Sparsification-Verstoß. Kein $2^{o(M)}$.
- $N = 2n = 6$: Ein $2^{o(N)}$ -Löser wäre ein $2^{o(2n)} = 2^{o(n)}$ -Löser für 3-SAT – direkter ETH-Verstoß. Kein $2^{o(N)}$.

NAE-4-SAT: Klauseln mit ≤ 4 Literalen; gesucht ist eine Belegung, unter der jede Klausel zwei *verschieden* belegte Literale enthält. Reduktion von 3-SAT: neue Variable w ; jede Klausel C_i wird zu $C_i \cup \{w\}$. **SetSplitting:** Grundmenge S , Teilmengen $F_1, \dots, F_r$; gesucht eine Partition $S = S_1 \dot{\cup} S_2$, die jede F_i schneidet. Reduktion von NAE-4-SAT: $S = \{x_i, \bar{x}_i\}$; Teilmengen $\{x_i, \bar{x}_i\}$ pro Variable und C_i pro Klausel.

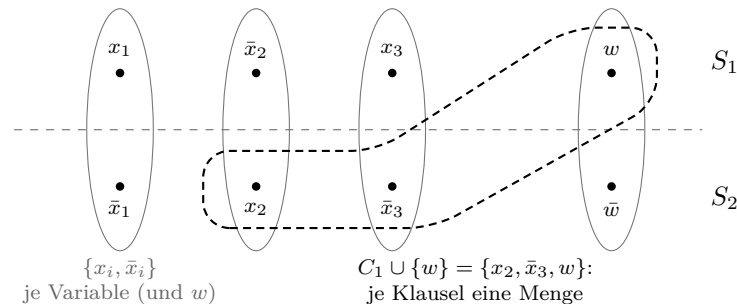
- a) Parameter der NAE-4-SAT-Instanz; b) Lower Bounds dafür;
c) Parameter der SetSplitting-Instanz; d) Lower Bounds dafür.

Lösung.

a) Variablen $n' = n + 1$, Klauseln $m' = m$.

b) Angenommen, NAE-4-SAT wäre in $2^{o(n')} \cdot |I|^{O(1)}$ lösbar. Wegen $n' = n + 1$ ergäbe Reduktion + Algorithmus einen $2^{o(n)}$ -Algorithmus für 3-SAT – direkter Widerspruch zur ETH. Angenommen, NAE-4-SAT wäre in $2^{o(m')} \cdot |I|^{O(1)}$ lösbar. Wegen $m' = m$ ergäbe sich ein $2^{o(m)}$ -Algorithmus für 3-SAT – nach dem Sparsification-Lemma Widerspruch zur ETH. Bounds: kein $2^{o(n')}$, kein $2^{o(m')}$.

c) Mit n' Variablen und m' Klauseln der NAE-Instanz: $|S| = 2n'$, $r = n' + m'$.



$|S| = 2n'$, $r = n' + m'$ Mengen; jede Menge schneidet S_1 und $S_2 \Rightarrow$ kein $2^{o(|S|)}$, kein $2^{o(r)}$

d) Angenommen, SETSPLITTING wäre in $2^{o(|S|)} \cdot |I|^{O(1)}$ lösbar. Wegen $|S| = 2n'$ ergäbe sich ein $2^{o(n')}$ -Algorithmus für NAE-4-SAT – nach b) ausgeschlossen. Angenommen, SETSPLITTING wäre in $2^{o(r)} \cdot |I|^{O(1)}$ lösbar. Jede NAE-Klausel hat ≤ 4 Literale, also $n' \leq 4m' + 1 = O(m')$ und damit $r = n' + m' = O(m')$ – es ergäbe sich ein $2^{o(m')}$ -Algorithmus für NAE-4-SAT, nach b) ausgeschlossen. Bounds: kein $2^{o(|S|)}$, kein $2^{o(r)}$. $\square$

Beispiel A45 – Lower Bounds NAE-4-SAT und SetSplitting: Kette einsetzen

Start: 3-SAT mit $n = 3$, $m = 2$, $C_1 = (x_1 \vee x_2 \vee x_3)$, $C_2 = (\bar{x}_1 \vee x_2 \vee \bar{x}_3)$.

(a) **NAE-4-SAT**. Neue Variable w ; jede Klausel wird zu $C_i \cup \{w\}$: $C'_1 = (x_1 \vee x_2 \vee x_3 \vee w)$, $C'_2 = (\bar{x}_1 \vee x_2 \vee \bar{x}_3 \vee w)$. Parameter: $n' = n + 1 = 4$, $m' = m = 2$.

(b) **Bounds NAE-4-SAT** (ETH: kein $2^{o(n)}/2^{o(m)}$ für 3-SAT). Ein $2^{o(n')}$ -Löser ergäbe (da $n' = n + 1$) einen $2^{o(n)}$ -3-SAT-Löser (direkter Verstoß); ein $2^{o(m')}$ -Löser ergäbe (da $m' = m$) einen $2^{o(m)}$ -3-SAT-Löser (Sparsification-Verstoß). Kein $2^{o(n')}$, kein $2^{o(m')}$.

(c) **SetSplitting**. $S = \{x_i, \bar{x}_i \mid i \in [3]\} \cup \{w, \bar{w}\}$, also

$$|S| = 2n' = 8, \quad r = n' + m' = 4 + 2 = 6.$$

Mengen: die $n' = 4$ Paare $\{x_i, \bar{x}_i\}$ bzw. $\{w, \bar{w}\}$ und die $m' = 2$ Klauseln C'_i . Die gesuchte Partition $S = S_1 \dot{\cup} S_2$ (wahr/falsch) schneidet jede Menge.

(d) **Bounds SetSplitting** (nach (b) ausgeschlossen):

- $|S| = 8 = 2n'$: Ein $2^{o(|S|)}$ -Löser ergäbe einen $2^{o(n')}$ -NAE-Löser – verboten. Kein $2^{o(|S|)}$.
- $r = 6$: jede NAE-Klausel hat ≤ 4 Literale, also $n' \leq 4m' + 1 = O(m')$ und $r = n' + m' = O(m')$; ein $2^{o(r)}$ -Löser ergäbe einen $2^{o(m')}$ -NAE-Löser – verboten. Kein $2^{o(r)}$.

CoverClique: Gibt es k disjunkte Cliques, die alle Knoten überdecken? Reduktion von k -COLOR: $(G, k) \mapsto (G' = (V, \bar{E}), k)$ (Komplementgraph). *Hinweis:* k -COLOR ist unter der ETH nicht in $2^{o(|V|)} \cdot |I|^{O(1)}$ lösbar. **ΔCover:** Reduktion von VERTEXCOVER (aus der Übung): $V' = V \cup \{v_e \mid e \in E\}$, $E' = E \cup \{\{v_e, e_1\}, \{v_e, e_2\}\}$.

- a) Parameter der CoverClique-Instanz; b) Lower Bounds dafür;
c) Parameter der ΔCover-Instanz; d) Lower Bounds dafür.

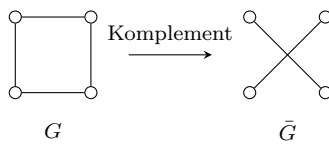
Lösung.

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

b) Angenommen, COVERCLIQUE wäre in $2^{o(|V'|)} \cdot |I|^{O(1)}$ lösbar. Wegen $|V'| = |V|$ ergäbe sich ein $2^{o(|V|)}$ -Algorithmus für k -COLOR – laut Hinweis unter der ETH ausgeschlossen. Angenommen, COVERCLIQUE wäre in $2^{o(\sqrt{|\bar{E}|})} \cdot |I|^{O(1)}$ lösbar. Wegen $|\bar{E}| \leq |V|^2$ ist $\sqrt{|\bar{E}|} = O(|V|)$ – wieder ein $2^{o(|V|)}$ -Algorithmus für k -COLOR. Bounds: kein $2^{o(|V'|)}$, kein $2^{o(\sqrt{|\bar{E}|})}$.

c) $|V'| = |V| + |E|$, $|E'| = |E| + 2|E| = 3|E|$.

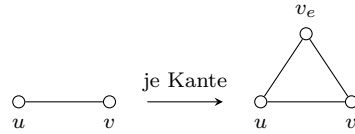
$$k\text{-COLOR} \leq_p \text{COVERCLIQUE}$$



$$|V'| = |V|, |\bar{E}| \leq |V|^2$$

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

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



$$|V'| = |V| + |E| = O(|V|^2), |E'| = 3|E|$$

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

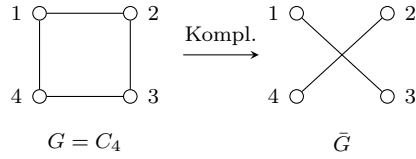
d) Für VERTEXCOVER gilt unter der ETH (Hausaufgabe 12.1): kein $2^{o(|V|)}$ - und kein $2^{o(\sqrt{|E|})}$ -Algorithmus. Angenommen, ΔCOVER wäre in $2^{o(\sqrt{|V'|})} \cdot |I|^{O(1)}$ lösbar. Wegen $|V'| = |V| + |E| \leq |V| + |V|^2 = O(|V|^2)$ ist $\sqrt{|V'|} = O(|V|)$ – es ergäbe sich ein $2^{o(|V|)}$ -Algorithmus für VERTEXCOVER. Widerspruch. Angenommen, ΔCOVER wäre in $2^{o(\sqrt{|E'|})} \cdot |I|^{O(1)}$ lösbar. Wegen $|E'| = 3|E| = O(|E|)$ ist $\sqrt{|E'|} = O(\sqrt{|E|})$ – es ergäbe sich ein $2^{o(\sqrt{|E|})}$ -Algorithmus für VERTEXCOVER. Widerspruch. Bounds: kein $2^{o(\sqrt{|V'|})}$, kein $2^{o(\sqrt{|E'|})}$. $\square$

Beispiel A46 – Lower Bounds CoverClique und Δ Cover: Mini-Gadgets

(a) **CoverClique** (aus k -COLOR, Komplementgraph). Instanz $G = C_4$ (4-Kreis), $k = 2$: $|V| = 4$, $|E| = 4$. Komplement $\bar{G}$ hat die beiden Nicht-Kanten $\{1, 3\}, \{2, 4\}$, also $|V'| = |V| = 4$, $|\bar{E}| = 2$.

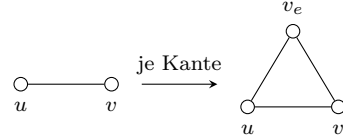
(c) **Δ Cover** (aus VERTEXCOVER). Instanz: eine Kante $\{u, v\}$ ($|V| = 2$, $|E| = 1$). Pro Kante ein neuer Knoten v_e mit Dreieck: $V' = \{u, v, v_e\}$, $E' = \{uv, uv_e, vv_e\}$, also $|V'| = |V| + |E| = 3$, $|E'| = 3|E| = 3$.

k -COLOR $\leq_p$ COVERCLIQUE



$$|V'|=4, |\bar{E}|=2$$

VERTEXCOVER $\leq_p$ Δ COVER



$$|V'|=3, |E'|=3$$

(b) **Bounds CoverClique** (k -COLOR nicht in $2^{o(|V|)}$): $|V'| = |V| \Rightarrow$ ein $2^{o(|V'|)}$ -Löser wäre ein $2^{o(|V|)}$ - k -Color-Löser; $\sqrt{|\bar{E}|} \leq \sqrt{|V|^2} = O(|V|) \Rightarrow 2^{o(\sqrt{|\bar{E}|})}$ ebenso. Kein $2^{o(|V'|)}$, kein $2^{o(\sqrt{|\bar{E}|})}$.

(d) **Bounds Δ Cover** (VERTEXCOVER nicht in $2^{o(|V|)}$, nicht in $2^{o(\sqrt{|E|})}$): $|V'| = |V| + |E| = O(|V|^2)$, also $\sqrt{|V'|} = O(|V|) \Rightarrow$ ein $2^{o(\sqrt{|V'|})}$ -Löser wäre ein $2^{o(|V|)}$ -VC-Löser; $|E'| = 3|E| = O(|E|)$, also $\sqrt{|E'|} = O(\sqrt{|E|}) \Rightarrow$ ein $2^{o(\sqrt{|E'|})}$ -Löser wäre ein $2^{o(\sqrt{|E|})}$ -VC-Löser. Kein $2^{o(\sqrt{|V'|})}$, kein $2^{o(\sqrt{|E'|})}$.

6 Approximation: anwenden, Güte beweisen, Worst Case

A47 Knapsack: Greedy und ModifiedGreedy anwenden Klausur SS23-H, A1, 3+2+5 P.

Rucksack-Instanz mit Kapazität $B = 16$, Items als (p_i, w_i) :

$$I = [(1, 4), (1, 1), (1, 3), (11, 13), (3, 3), (5, 6), (1, 5)]$$

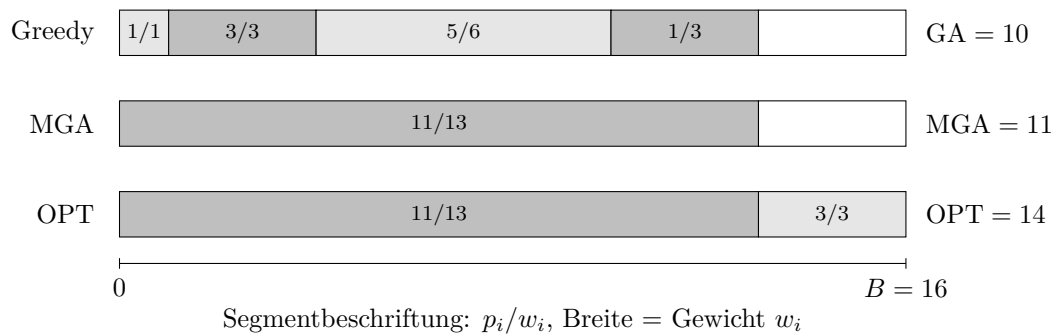
- Wenden Sie Greedy an (mit Begründung der Auswahl). Lösungswert?
- Wenden Sie ModifiedGreedy an. Wie ändert sich die Lösung?
- Güte von ModifiedGreedy? Idee zur Verbesserung auf $\frac{3}{2}$ bzw. $\frac{4}{3}$?

Hinweis: Greedy sortiert die Items absteigend nach Profitdichte p_i/w_i und packt in dieser Reihenfolge jedes Item ein, das noch passt. ModifiedGreedy gibt das Bessere aus Greedy-Lösung und profitreichstem Einzel-Item aus.

Lösung.

a) Profitdichten p_i/w_i : 0,25; 1; 0,33; 0,85; 1; 0,83; 0,2. Absteigend sortiert: (1, 1), (3, 3), (11, 13), (5, 6), (1, 3), (1, 4), (1, 5). Einpacken (Restkapazität beachten): (1, 1) [Last 1], (3, 3) [4], (11, 13) passt nicht ($17 > 16$), (5, 6) [10], (1, 3) [13], (1, 4) und (1, 5) passen nicht. GA = $1 + 3 + 5 + 1 = 10$.

b) Bestes Einzel-Item ist (11, 13) mit Profit $11 > 10$, also gibt ModifiedGreedy dieses aus: MGA = 11. (Optimal wäre $\{(11, 13), (3, 3)\}$ mit Gewicht 16 und Profit 14.)



c) MGA hat Güte 2. Verbesserung durch k -Enumeration (Sahni): Probiere alle Vorauswahlen mit $\leq k$ Items, fülle mit Greedy auf, nimm das Beste – Güte $1 + 1/k$, also $\frac{3}{2}$ für $k = 2$ und $\frac{4}{3}$ für $k = 3$. $\square$

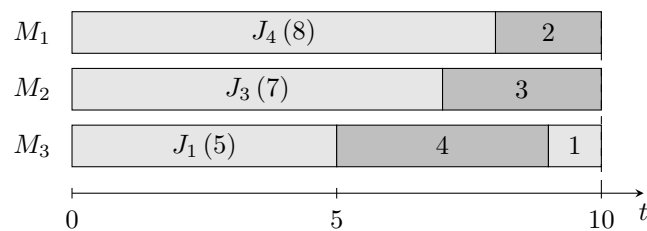
Jobs $p = (5, 3, 7, 8, 1, 4, 2)$, $m = 3$.

- Wenden Sie LPT an (Zwischenschritte, Reihenfolge, Schedule grafisch).
- Approximative Güte von LPT?
- Instanz für $m = 2$, bei der ListScheduling Güte $2 - 1/m$ erreicht (mit OPT und LS-Wert).
- Dasselbe für $m = 3$.

Hinweis: ListScheduling legt jeden Job in der gegebenen Reihenfolge auf die aktuell am wenigsten belastete Maschine. LPT ist ListScheduling nach absteigender Sortierung der Bearbeitungszeiten.

Lösung.

a) Absteigend sortiert: 8, 7, 5, 4, 3, 2, 1 (also $J_4, J_3, J_1, J_6, J_2, J_7, J_5$). Platzierung auf die jeweils leerste Maschine: $8 \rightarrow M_1$ [8], $7 \rightarrow M_2$ [7], $5 \rightarrow M_3$ [5], $4 \rightarrow M_3$ [9], $3 \rightarrow M_2$ [10], $2 \rightarrow M_1$ [10], $1 \rightarrow M_3$ [10].

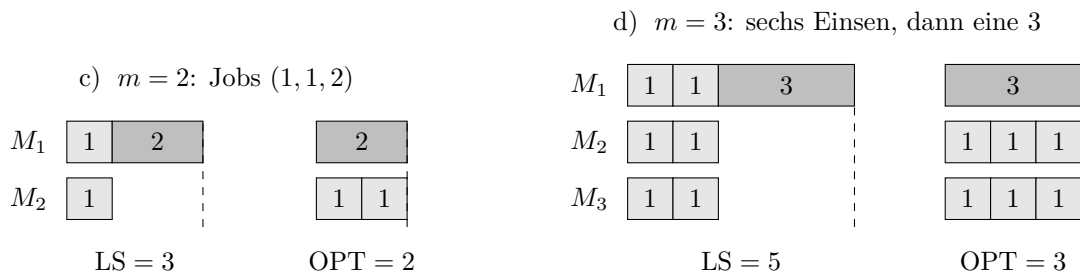


Makespan = 10 (hier sogar optimal, da $\sum p_j = 30 = 3 \cdot 10$).

b) LPT hat Güte $\frac{4}{3} - \frac{1}{3m}$.

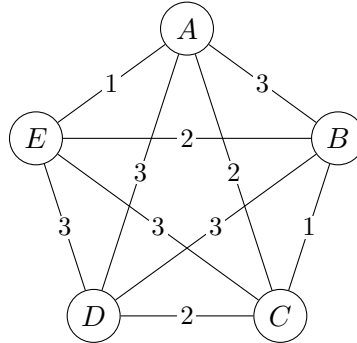
c) $m = 2$: Jobs (1, 1, 2) in dieser Reihenfolge. LS verteilt die Einsen auf beide Maschinen, die 2 kommt obendrauf: LS = 3. Optimal: $\{2\} / \{1, 1\}$ mit OPT = 2. Rate $3/2 = 2 - \frac{1}{2}$.

d) $m = 3$: sechs Einser-Jobs, dann ein Job der Größe 3. LS verteilt die Einsen gleichmäßig (je 2), die 3 kommt obendrauf: LS = 5. Optimal: 3 allein, die Einsen auf zwei Maschinen: OPT = 3. Rate $5/3 = 2 - \frac{1}{3}$.



□

Gegeben ein vollständiger Graph auf $\{A, B, C, D, E\}$ mit symmetrischen Distanzen $d(A, E) = 1$, $d(B, C) = 1$, $d(A, C) = 2$, $d(C, D) = 2$, $d(B, E) = 2$, übrige Distanzen 3 (metrisch). (Instanz ersetzt; Original-Gewichte der Klausur nicht überliefert.)



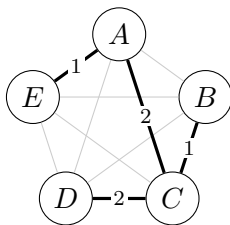
- Wenden Sie Δ TSP1 ab Startknoten C an; geben Sie alle Zwischenschritte an.
- Was berechnet Δ TSP1, und welche Bedingungen muss der Graph erfüllen?
- Begründen Sie die approximative Güte 2.

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

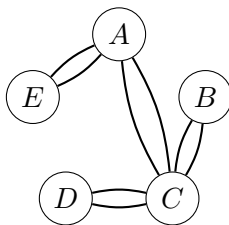
Lösung.

a)

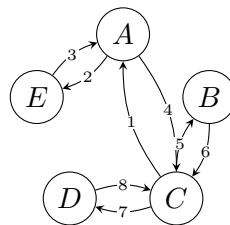
- MST: $A-E$ (1), $B-C$ (1), $A-C$ (2), $C-D$ (2); Gewicht 6.
- Kanten verdoppeln: Multigraph mit Gewicht 12, alle Grade gerade.
- Eulerkreis ab C : $[C, A, E, A, C, B, C, D, C]$, Länge 12.
- Abkürzen (besuchte Knoten überspringen): Tour $[C, A, E, B, D, C]$, Länge 10.



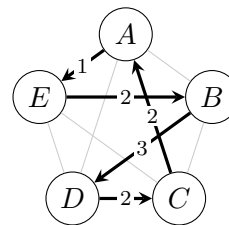
(1) MST, Gewicht 6



(2) verdoppelt, Gewicht 12



(3) Eulerkreis ab C

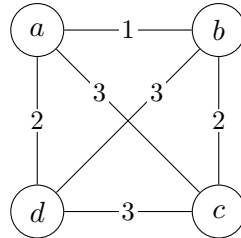


(4) Tour, Länge 10

b) Δ TSP1 berechnet eine Rundreise (TSP-Tour) mit Güte 2. Der Graph muss vollständig sein, die Distanzen symmetrisch und die Dreiecksungleichung $d(i, j) \leq d(i, k) + d(k, j)$ erfüllen – sonst kann das Abkürzen die Tour verlängern.

c) Drei Schritte: (1) $w(T) \leq \text{OPT}$, denn eine optimale Tour ohne eine Kante ist ein Spannbaum und T ist minimal. (2) Der Eulerkreis hat Länge $2w(T) \leq 2\text{OPT}$. (3) Abkürzen verlängert wegen der Dreiecksungleichung nicht. Also $d(R) \leq 2\text{OPT}$. $\square$

Vollständiger Graph mit Knoten a, b, c, d und symmetrischen Distanzen $d(a, b) = 1$, $d(b, c) = 2$, $d(a, d) = 2$, $d(a, c) = 3$, $d(c, d) = 3$, $d(b, d) = 3$.



- Was berechnet Christofides' Algorithmus, welche Güte hat er, und welche Eigenschaft müssen die Distanzen neben der Symmetrie erfüllen?
- Wenden Sie den Algorithmus mit Startknoten a an; geben Sie alle Zwischengraphen an – auch Schritte, die nichts ändern.

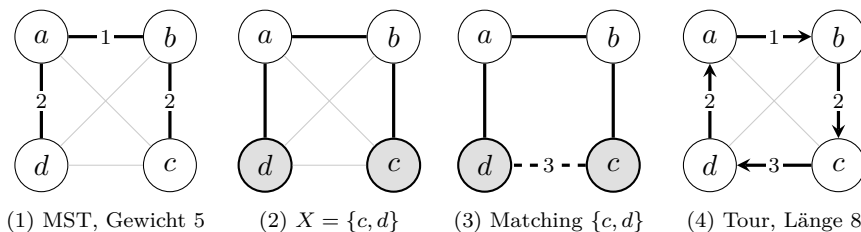
Hinweis (Christofides): (1) MST T berechnen; (2) X = Knoten mit ungeradem Grad in T ; (3) minimales perfektes Matching K auf X ; (4) Eulerkreis im Multigraphen $T + K$; (5) Abkürzen.

Lösung.

a) Christofides berechnet eine TSP-Rundreise mit Güte $\frac{3}{2}$. Die Distanzen müssen zusätzlich die Dreiecksungleichung erfüllen.

b)

- MST (Kruskal): $\{a, b\}$ (1), $\{b, c\}$ (2), $\{a, d\}$ (2); Gewicht 5.
- Ungerade Grade im MST: $a: 2, b: 2, c: 1, d: 1 \Rightarrow X = \{c, d\}$.
- Minimales perfektes Matching auf X : einzige Möglichkeit $K = \{\{c, d\}\}$, Kosten 3.
- Multigraph $T + K$: alle Grade gerade. Eulerkreis ab a : $[a, b, c, d, a]$, Kosten $1 + 2 + 3 + 2 = 8$.
- Abkürzen: kein Knoten kommt doppelt vor – dieser Schritt ändert nichts, wird aber geprüft. Tour $R = [a, b, c, d, a]$, Länge 8 (hier sogar optimal).



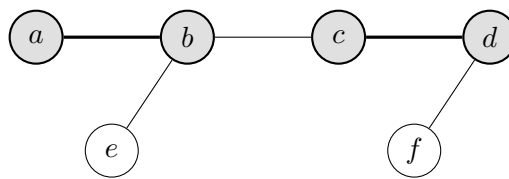
□

Algorithmus **2ApproxCV**: Gehe alle Kanten durch; sind beide Endpunkte noch nicht in C , füge beide zu C hinzu. Beweisen Sie die Güte 2.

Lösung.

Sei A die Menge der Kanten, für die beide Endpunkte aufgenommen wurden. Dann gilt $|C| = 2|A|$.

Die Kanten in A sind paarweise knotendisjunkt: Hätten zwei Kanten aus A einen gemeinsamen Endpunkt, wäre bei der später betrachteten dieser Endpunkt schon in C gewesen – sie wäre nicht aufgenommen worden. A ist also ein Matching.



dick = A , gefüllt = C : $|C| = 2|A| = 4$

Sei C^* ein minimales Vertex Cover. C^* enthält von jeder Kante aus A mindestens einen Endpunkt, und da die Kanten aus A knotendisjunkt sind, braucht es für jede einen *eigenen* Knoten: $|C^*| \geq |A|$.

Zusammen: $|C| = 2|A| \leq 2|C^*|$. □

MAX-3-SAT: Gegeben eine Formel φ in KNF mit drei Literalen pro Klausel; gesucht eine Belegung β , die die Anzahl $v(\beta)$ der erfüllten Klauseln maximiert.

Algorithmus A : Werte β_0 (alle false) und β_1 (alle true) aus, gib die bessere Belegung zurück.

- a) Zeigen Sie $v(A(\varphi)) \geq \frac{1}{2}v(\text{OPT}(\varphi))$ für alle φ .
- b) Geben Sie eine Formel an, bei der A genau die Hälfte der maximal erfüllbaren Klauseln erfüllt (Begründung).

Lösung.

a) Jede Klausel enthält ein positives oder ein negatives Literal, wird also von β_1 oder β_0 erfüllt. Angenommen $v(A(\varphi)) < \frac{1}{2}v(\text{OPT}(\varphi))$ für ein φ mit m Klauseln. Wegen $v(\text{OPT}) \leq m$ folgt $v(\beta_0) < \frac{m}{2}$ und $v(\beta_1) < \frac{m}{2}$. Da aber jede Klausel von mindestens einer der beiden erfüllt wird, gilt $v(\beta_0) + v(\beta_1) \geq m$ – also erfüllt eine der beiden $\geq \frac{m}{2}$ Klauseln. Widerspruch. Somit hat A Güte 2.

b) $\varphi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_0 \vee \neg x_2 \vee \neg x_3)$. Mit $x_0 = \text{falsch}$, $x_1 = x_2 = x_3 = \text{wahr}$ sind beide Klauseln erfüllt: $\text{OPT} = 2$. Aber β_0 erfüllt nur die zweite Klausel, β_1 nur die erste: $v(A(\varphi)) = 1 = \frac{1}{2} \cdot \text{OPT}$. $\square$

APPROXIMATESUBSETSUM: Zahlen $a_1, \dots, a_n \leq T$ mit $\sum a_i > T$; maximiere $\sum_{j \in S} a_j \leq T$. Algorithmus GA: sortiere absteigend, nimm der Reihe nach, *stoppe* beim ersten nicht mehr passenden Element.

- a) Konstruktionsvorschrift für Instanzen, mit denen GA beliebig nah an Güte 2 kommt.
- b) Zeigen Sie $\text{GA}(I) \geq \text{OPT}(I)/2$.

Lösung.

a) Für gerades T : $a_1 = \frac{T}{2} + 1$, $a_2 = a_3 = \frac{T}{2}$. GA nimmt a_1 und stoppt, denn $a_1 + a_2 > T$: $\text{GA} = \frac{T}{2} + 1$. Optimal ist $a_2 + a_3 = T$. Rate $\frac{T}{T/2+1} \rightarrow 2$ für $T \rightarrow \infty$.

b) Sei a_ℓ das erste Element, das nicht mehr passte (existiert wegen $\sum a_i > T$). Dann gilt

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

Wegen der absteigenden Sortierung enthält die GA-Auswahl ein Element $a_j \geq a_\ell$, also $\text{GA}(I) \geq a_\ell$. Einsetzen: $2 \text{GA}(I) \geq \text{GA}(I) + a_\ell > T \geq \text{OPT}(I)$, somit $\text{GA}(I) \geq \text{OPT}(I)/2$. $\square$

RoundRobin sortiert die Jobs absteigend und verteilt sie reihum (J_j auf Maschine $((j - 1) \bmod m) + 1$).

- a) Zeigen Sie per Widerspruch, dass RoundRobin Güte 2 hat.
- b) **Bonus:** Konstruktionsvorschrift für Instanzen mit $\text{Rate} \rightarrow 2 - \frac{1}{m}$ bei beliebigem festen m (mit OPT- und RR-Wert).

Lösung.

a) Angenommen $\text{RR}(I) > 2 \text{OPT}(I)$. Sei M_i die Maschine mit maximaler Last; sie trägt die Jobs $J_i, J_{i+m}, J_{i+2m}, \dots$. Wegen der absteigenden Sortierung gilt für $l \geq 1$: $p_{i+lm} \leq p_j$ für alle $j \leq lm$, insbesondere

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

Summieren über $l \geq 1$ ergibt $\text{RR}(I) - p_i \leq \frac{1}{m} \sum_j p_j \leq \text{OPT}(I)$. Mit $p_i \leq p_1 \leq \text{OPT}(I)$ folgt $\text{RR}(I) \leq 2 \text{OPT}(I)$ – Widerspruch zur Annahme.

b) Ein Job der Größe m und $m(m-1)$ Jobs der Größe 1 (absteigend sortiert steht der große zuerst). RoundRobin gibt Maschine 1 den großen Job und danach reihum jede m -te Eins – $m-1$ weitere: $\text{RR} = m + (m-1) = 2m-1$. Optimal: großer Job allein, die $m(m-1)$ Einsen auf die übrigen $m-1$ Maschinen (je m): $\text{OPT} = m$. $\text{Rate} \frac{2m-1}{m} = 2 - \frac{1}{m}$. (Für $m=3$: $\text{RR} = 5$, $\text{OPT} = 3$, $\text{Rate} \frac{5}{3}$.)

M_1	3	1	1
M_2	1	1	
M_3	1	1	

RoundRobin: $\text{RR} = 5$

M_1	3		
M_2	1	1	1
M_3	1	1	1

Optimum: $\text{OPT} = 3$

□

MIN-EDGE-COVER: Finde eine kleinste Kantenmenge C , sodass jeder Knoten zu einer Kante aus C inzident ist (G zusammenhängend). Algorithmus A : Gehe die Knoten in beliebiger Reihenfolge durch; ist der aktuelle Knoten v unbedeckt, füge eine beliebige zu v inzidente Kante hinzu.

- Zeigen Sie $A(G) \leq 2 \text{OPT}(G)$.
- Kreisgraph G_n ($n \geq 4$ gerade): Größe eines Min-Edge-Covers? Schlechteste Approximationsrate von A ? Geben Sie Reihenfolge und Kantenwahl an.

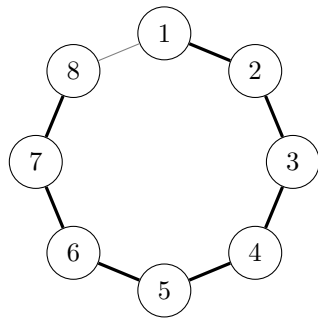
Lösung.

a) Der Algorithmus fügt höchstens eine Kante pro Knoten hinzu, also $A(G) \leq |V|$. Jede Kante deckt höchstens zwei Knoten, also braucht jedes Edge Cover mindestens $|V|/2$ Kanten: $\text{OPT}(G) \geq |V|/2$. Zusammen: $A(G) \leq |V| \leq 2 \text{OPT}(G)$.

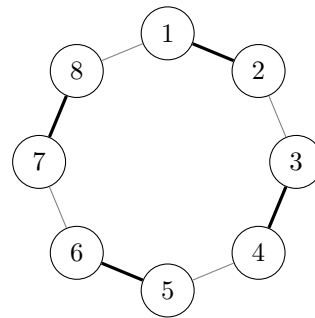
b) Im Kreis G_n bilden $n/2$ paarweise disjunkte Kanten (jede zweite Kreiskante) ein Edge Cover – weniger geht nicht, also $\text{OPT} = n/2$.

Schlechteste Ausführung: Reihenfolge $1, 3, 4, 5, \dots, n$. Knoten 1 wählt $\{1, 2\}$; Knoten 3 wählt $\{3, 2\}$ (2 ist schon gedeckt – die Kante deckt nur einen neuen Knoten); Knoten 4 wählt $\{4, 3\}$, Knoten 5 wählt $\{5, 4\}$, $\dots$, Knoten n wählt $\{n, n-1\}$. Nach der ersten Kante deckt jede weitere nur einen neuen Knoten: insgesamt $1 + (n-2) = n-1$ Kanten. Rate

$$\frac{n-1}{n/2} = 2 - \frac{2}{n} \rightarrow 2.$$



A : $n-1 = 7$ Kanten



OPT : $n/2 = 4$ Kanten

□

Leitergraph: K_n ($n \bmod 4 = 2$), Kanten $E_2 = \{\{i, i+2\}\} \cup \{\{2i+1, 2i+2\}\}$ mit Gewicht 1; alle übrigen Distanzen = kürzester Weg. Zeigen Sie: (a) $\text{OPT} = n$; (b) es gibt eine Ausführung von ΔTSP2 mit Tourlänge $(n-1) + n/2$.

Hinweis (ΔTSP2 , Christofides): (1) MST T ; (2) X = Knoten mit ungeradem Grad in T ; (3) minimales perfektes Matching K auf X ; (4) Eulerkreis in $T+K$; (5) Abkürzen. Bei Gleichständen (mehrere MSTs, mehrere Eulerkreise) darf die Ausführung beliebig – auch ungünstigst – gewählt werden.

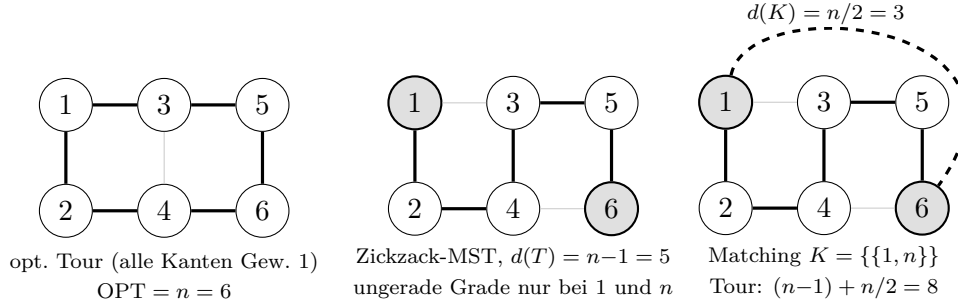
Lösung.

a) Die Tour „oben hin, unten zurück“ $-1, 3, 5, \dots, n-1, n, n-2, \dots, 2, 1-$ benutzt n Kanten vom Gewicht 1, also $\text{OPT} \leq n$. Weniger geht nicht (n Knoten brauchen n Tourkanten vom Gewicht ≥ 1): $\text{OPT} = n$.

b) Bei Gleichständen darf die Ausführung (MST-Wahl, Knotenreihenfolge) adversariell gewählt werden. Wähle den Zickzack-MST

$$1-2, 2-4, 4-3, 3-5, 5-6, 6-8, 8-7, \dots, \{n-1, n\}$$

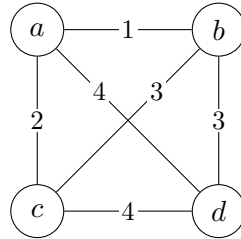
mit $n-1$ Kanten vom Gewicht 1: $d(T) = n-1$. In diesem Baum haben nur die Knoten 1 und n ungeraden Grad, also ist das Matching $K = \{\{1, n\}\}$ – Kosten = kürzester Weg von 1 nach $n = n/2$. Die Tour aus Eulerkreis und Abkürzen hat damit Länge $d(T) + d(K) = (n-1) + n/2$.



$$\text{Rate: } \frac{n-1+n/2}{n} = \frac{3}{2} - \frac{1}{n} \rightarrow \frac{3}{2}.$$

□

Vollständiger Graph auf $\{a, b, c, d\}$ mit $d(a, b) = 1$, $d(a, c) = 2$, $d(b, c) = 3$, $d(b, d) = 3$, $d(a, d) = 4$, $d(c, d) = 4$. (Instanz ersetzt; Original-Gewichtszuordnung nicht überliefert.)



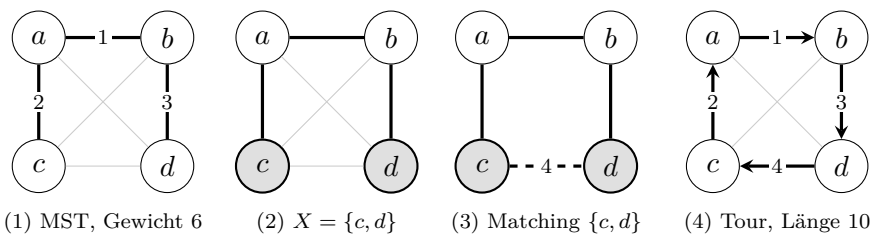
- Wenden Sie Christofides (Δ TSP2) mit Startknoten a an; alle Zwischengraphen angeben, auch Schritte ohne Änderung.
- Nennen Sie die zwei Zusatzeigenschaften der Eingabe, die Christofides braucht, mit je einer verletzenden Eingabe. Geben Sie die Güte an.

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

Lösung.

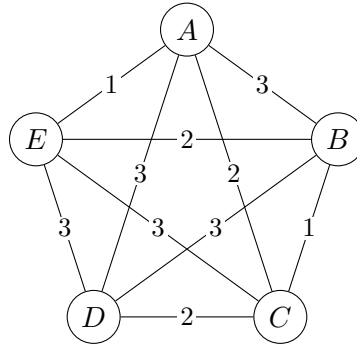
a)

- MST (Kruskal): $\{a, b\}$ (1), $\{a, c\}$ (2), $\{b, d\}$ (3); Gewicht 6.
- Ungerade Grade im MST: $a: 2, b: 2, c: 1, d: 1 \Rightarrow X = \{c, d\}$.
- Minimales perfektes Matching auf X : einzige Möglichkeit $K = \{\{c, d\}\}$, Kosten 4.
- Multigraph $T + K$: alle Grade gerade. Eulerkreis ab a : $[a, b, d, c, a]$, Kosten $1 + 3 + 4 + 2 = 10$.
- Abkürzen: kein Knoten doppelt – keine Änderung (wird aber geprüft). Tour $R = [a, b, d, c, a]$, Länge 10 (hier optimal: die Alternativen $[a, b, c, d, a]$ und $[a, c, b, d, a]$ kosten je 12).



b) Benötigt werden *Symmetrie* ($d(u, v) = d(v, u)$) und die *Dreiecksungleichung* ($d(u, v) \leq d(u, w) + d(w, v)$). Verletzende Eingaben: $d(a, b) = 1$, $d(b, a) = 5$ (asymmetrisch); bzw. $d(a, b) = 10$, $d(a, c) = d(c, b) = 1$ (Δ -Ungleichung verletzt). Güte: $\frac{3}{2}$. $\square$

Vollständiger Graph auf $\{A, B, C, D, E\}$ mit $d(A, E) = 1$, $d(B, C) = 1$, $d(A, C) = 2$, $d(C, D) = 2$, $d(B, E) = 2$, übrige Distanzen 3. (*Instanz ersetzt; Original-Gewichtszuordnung nicht überliefert.*)



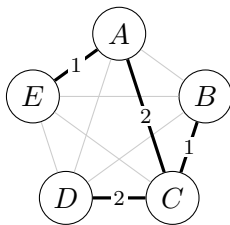
- Wenden Sie Δ TSP1 ab Startknoten C an (alle Zwischengraphen, Länge der Rundreise).
- Welche Güte hat Δ TSP1?
- Ist das Entscheidungsproblem zum TSP in P ? Kurze Begründung.
- Wahr oder falsch: „Güte 1,5 bedeutet $\frac{d(R)}{\text{OPT}(I)} \geq 1,5$ für eine Instanz I .“ Kurze Begründung.

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

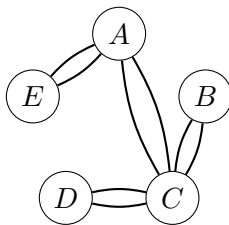
Lösung.

a)

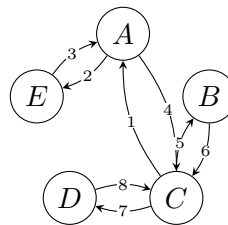
- MST: $A-E$ (1), $B-C$ (1), $A-C$ (2), $C-D$ (2); Gewicht 6.
- Verdoppeln: Multigraph, Gewicht 12.
- Eulerkreis ab C : $[C, A, E, A, C, B, C, D, C]$.
- Abkürzen: Tour $[C, A, E, B, D, C]$, Länge 10.



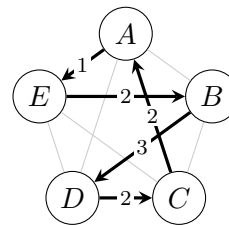
(1) MST, Gewicht 6



(2) verdoppelt, Gewicht 12



(3) Eulerkreis ab C



(4) Tour, Länge 10

b) Güte 2.

c) Nein (außer $P = NP$): Das TSP-Entscheidungsproblem ist NP-vollständig – HAMILTONIANCYCLE reduziert darauf (Kanten Distanz 1, Nicht-Kanten $|V|+1$, $L = |V|$). Läge es in P , folgte $P = NP$.

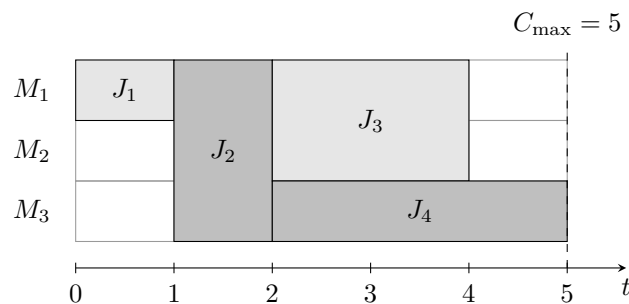
d) Falsch. Multiplikative Güte α ist eine obere Schranke für alle Instanzen: $d(R) \leq \alpha \cdot \text{OPT}(I)$ für jede Instanz I – nicht eine untere Schranke $\geq$ für eine Instanz. $\square$

Jobs belegen $q_j \in [m]$ Maschinen gleichzeitig für Zeit p_j . ListScheduling platziert die Jobs der Reihe nach zum frühestmöglichen Zeitpunkt auf den q_j am wenigsten belasteten Maschinen.

- Anwendung: $m = 3$; Jobs (p, q) : $(1, 1), (1, 3), (2, 2), (3, 1)$. Geben Sie den Schedule an.
- Worst-Case-Güte für $q_j = 1$?
- Zeigen Sie $LS(I) \leq OPT(I) + \frac{1}{2}p_{\max}$ für alle I mit $\frac{m}{3} < q_j \leq \frac{m}{2}$.
- Bonus:** Güte für $\frac{m}{k+1} < q_j \leq \frac{m}{k}$ ($k \geq 2$)? Beweis.

Lösung.

a) Lasten $C = (0, 0, 0)$. Job 1 $(1, 1)$: startet bei 0 auf einer Maschine, $C = (1, 0, 0)$. Job 2 $(1, 3)$: braucht alle 3 Maschinen, frühester Start $\max C = 1$, läuft $[1, 2]$: $C = (2, 2, 2)$. Job 3 $(2, 2)$: Start 2 auf zwei Maschinen, $[2, 4]$: $C = (4, 4, 2)$. Job 4 $(3, 1)$: leerste Maschine hat Last 2, Start 2, $[2, 5]$: $C = (5, 4, 4)$. **Makespan 5.**



b) Für $q_j = 1$ ist es das klassische ListScheduling: Güte $2 - \frac{1}{m}$.

c) Sei j ein Job, der den Makespan bestimmt, mit Startzeit $s = LS - p_j$. Zu jedem Zeitpunkt $t < s$ sind weniger als $q_j \leq \frac{m}{2}$ Maschinen frei (sonst wäre j früher platziert worden), also mehr als $\frac{m}{2}$ belegt. Da jeder Job höchstens $\frac{m}{2}$ Maschinen belegt, laufen dann mindestens zwei Jobs; da jeder *mehr* als $\frac{m}{3}$ belegt, laufen höchstens zwei. Also laufen in $[0, s)$ stets *genau zwei* Jobs. Damit ist

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

Im Optimum laufen ebenfalls nie drei Jobs gleichzeitig ($3 \cdot \frac{m}{3} < \sum q$ überschritte m), also $\sum_i p_i \leq 2 \text{OPT}$. Einsetzen: $s \leq \frac{1}{2}(\sum_i p_i - p_j) \leq \text{OPT} - \frac{1}{2}p_j$, folglich

$$LS = s + p_j \leq \text{OPT} + \frac{1}{2}p_j \leq \text{OPT} + \frac{1}{2}p_{\max}.$$

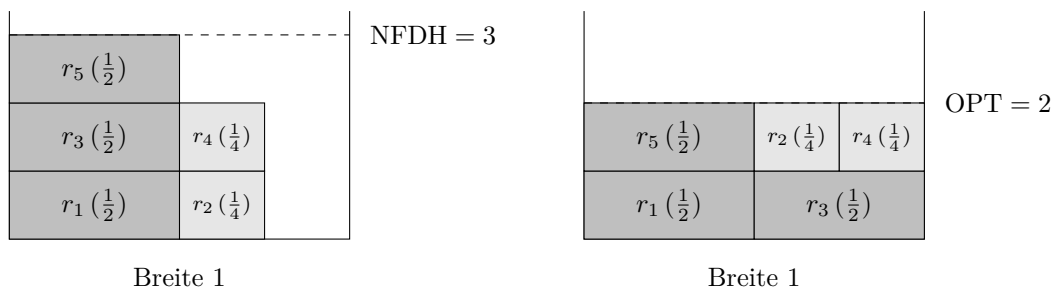
d) Güte $2 - \frac{1}{k}$. Analog: Vor dem Start von j sind mehr als $m - \frac{m}{k} = \frac{(k-1)m}{k}$ Maschinen belegt; jeder Job belegt $\leq \frac{m}{k}$, also laufen $\geq k$ Jobs – und $\leq k$, da jeder $> \frac{m}{k+1}$ belegt. Genau k Jobs: $ks \leq \sum_i p_i - p_j$. Im Optimum laufen $\leq k$ Jobs parallel, also $\sum_i p_i \leq k \text{OPT}$. Daraus $s \leq \text{OPT} - \frac{1}{k}p_j$ und $LS \leq \text{OPT} + (1 - \frac{1}{k})p_j \leq \text{OPT} + (1 - \frac{1}{k})p_{\max} \leq (2 - \frac{1}{k}) \text{OPT}$ (mit $p_{\max} \leq \text{OPT}$). $\square$

STRIPPACKING: Rechtecke (nach Höhe absteigend sortiert) in einen Streifen der Breite 1 packen, Höhe minimieren. NFDH packt stufenweise von links; passt ein Rechteck nicht mehr, beginnt eine neue Stufe.

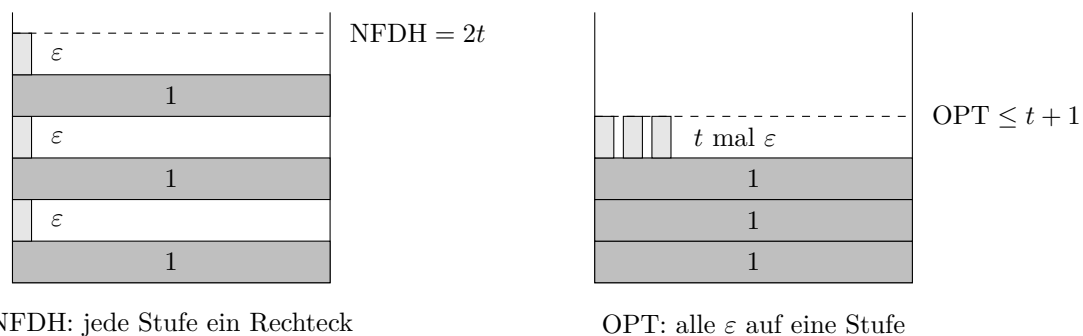
- a) Instanz: fünf Rechtecke der Höhe 1 mit Breiten $\frac{1}{2}, \frac{1}{4}, \frac{1}{2}, \frac{1}{4}, \frac{1}{2}$; NFDH braucht Höhe 3. Geben Sie eine optimale Packung und die Approximationsrate an.
- b) Zeigen Sie: Es gibt L_n mit $\lim_{n \rightarrow \infty} \text{NFDH}(L_n)/\text{OPT}(L_n) = 2$.
- c) Zeigen Sie für Instanzen mit $h(r_i) = 1$: $\text{NFDH}(L) \leq 2 \text{OPT}(L) + 1$.
- d) **Bonus:** Zeigen Sie dieselbe Schranke für $h(r_i) \leq 1$.

Lösung.

a) Optimal: Stufe 1 mit $\frac{1}{2} + \frac{1}{2} = 1$ (Rechtecke 1 und 3), Stufe 2 mit $\frac{1}{2} + \frac{1}{4} + \frac{1}{4} = 1$ (Rechtecke 5, 2, 4) – Höhe $\text{OPT} = 2$. Rate: $\frac{3}{2}$.



b) L_n ($n = 2t$): alle Höhen 1; Breiten abwechselnd 1 und ε . NFDH: Jedes Breite-1-Rechteck füllt seine Stufe komplett, jedes ε -Rechteck eröffnet eine neue Stufe, auf die das nächste Breite-1-Rechteck nicht mehr passt – $2t$ Stufen, $\text{NFDH} = 2t$. Optimal: die t Breite-1-Rechtecke je eigene Stufe, alle t ε -Rechtecke zusammen auf eine Stufe – $\text{OPT} \leq t + 1$. Rate $\frac{2t}{t+1} \rightarrow 2$.



c) Seien $W_1, \dots, W_S$ die Gesamtbreiten der NFDH-Stufen; $\text{NFDH} = S$. Für $i < S$ gilt $W_i + W_{i+1} > 1$: Das erste Rechteck der Stufe $i+1$ (Breite $\leq W_{i+1}$) passte nicht mehr auf Stufe i . Summieren über $i = 1, \dots, S-1$:

$$2 \sum_i W_i \geq \sum_{i=1}^{S-1} (W_i + W_{i+1}) > S - 1.$$

Die Gesamtfläche ist $\sum_i W_i$ (Höhen 1) und passt in einen Streifen der Breite 1: $\text{OPT} \geq \sum_i W_i$. Also $2 \text{OPT} > S - 1$, d. h. $\text{NFDH} = S \leq 2 \text{OPT} + 1$.

d) Sei H_i die Höhe der Stufe i (= Höhe ihres ersten Rechtecks); wegen der Sortierung gilt $H_1 \geq H_2 \geq \dots$ und jedes Rechteck der Stufe i hat Höhe $\geq H_{i+1}$. Also ist die Fläche der Stufe i mindestens $W_i \cdot H_{i+1}$, und mit $W_i + W_{i+1} > 1$:

$$\sum_{i \geq 2} H_i = \sum_{i \geq 1} H_{i+1} < \sum_{i \geq 1} (W_i + W_{i+1}) H_{i+1} \leq 2 \sum_{i \geq 1} W_i H_{i+1} \leq 2 \cdot \text{Fläche} \leq 2 \text{OPT}.$$

Mit $H_1 \leq 1$ folgt $\text{NFDH} = \sum_i H_i \leq 2 \text{OPT} + 1$. □

7 Turingmaschinen

A61 TM für Palindrome

Hausaufgabe 10.2, 5 P.

Entwerfen Sie eine Turingmaschine (Einband-TM) für $L = \{w \in \Sigma^* \mid w \text{ ist ein Palindrom}\}$ und geben Sie die Laufzeit an (mit Begründung von Korrektheit und Laufzeit).

Lösung.

Die Maschine:

- (1) Falls kein Wort auf dem Band steht, akzeptiere.
- (2) Falls nur ein Buchstabe auf dem Band steht, akzeptiere.
- (3) Lies den ersten Buchstaben, bewege den Kopf zum letzten. Sind beide verschieden, verwirf. Sonst ersetze beide durch ein neues Symbol x (erst den letzten, dann zurück zum ersten).
- (4) Gehe zu Schritt (1).

Korrektheit: Die Maschine vergleicht wiederholt den ersten und den letzten (unmarkierten) Buchstaben und streicht beide. Bei Ungleichheit ist das Wort kein Palindrom – verwerfen. Sonst endet sie in der Mitte mit einem oder keinem Buchstaben – in beiden Fällen ein Palindrom, akzeptieren.

Laufzeit: Die Schritte (1)–(3) kosten je $O(n)$. Pro Durchlauf werden zwei Buchstaben gestrichen, also gibt es höchstens $\lfloor n/2 \rfloor$ Durchläufe. Insgesamt $O(n^2)$. $\square$

Beispiel A61 – TM für Palindrome

Eingabe $w = abba$ (**Palindrom**). Die Maschine vergleicht die äußersten unmarkierten Buchstaben und ersetzt gleiche Paare durch x ; der Bandinhalt entwickelt sich:

a b b a	erstes a = letztes a $\Rightarrow$ beide $\rightarrow x$
x b b x	b = b $\Rightarrow$ beide $\rightarrow x$
x x x x	keine unmarkierten Buchstaben $\Rightarrow$ akzeptiert

Nicht-Palindrom $w = ab$. Erster Buchstabe a, letzter b: verschieden $\Rightarrow$ **verworfen** (Band bleibt a b).

Ungerades Palindrom $w = aba$.

a b a	a = a $\Rightarrow$ beide $\rightarrow x$
x b x	ein einzelner Buchstabe (b) in der Mitte $\Rightarrow$ akzeptiert

Entwerfen Sie eine Turingmaschine (Einband-TM) für $L = \{0^{2^n} \mid n \in \mathbb{N}\}$ über $\Sigma = \{0\}$; begründen Sie Korrektheit und Laufzeit. Diskutieren Sie die Laufzeit im Vergleich zu einem RAM-Algorithmus.

Lösung.**Die Maschine:**

- (1) Falls genau eine 0 auf dem Band steht, akzeptiere.
- (2) Laufe über das Band und ersetze jede zweite 0 durch x .
- (3) Falls das letzte Symbol eine 0 war (ungerade Anzahl), verwirf.
- (4) Gehe zu Schritt (1).

Korrektheit: Eine Zahl ist genau dann eine Zweierpotenz, wenn wiederholtes Halbieren bei 1 endet, ohne je auf eine ungerade Zahl > 1 zu treffen. Genau das prüft die Maschine: Jeder Durchlauf halbiert die Anzahl der Nullen; eine ungerade Anzahl > 1 führt zum Verwerfen.

Laufzeit: Jeder Durchlauf kostet $O(n)$; die Anzahl halbiert sich pro Durchlauf, also $O(\log n)$ Durchläufe. Insgesamt $O(n \log n)$.

RAM-Vergleich: Ein RAM-Algorithmus zählt die Nullen ($O(n)$) und prüft, ob die Anzahl eine Zweierpotenz ist ($O(\log n)$) – Laufzeit $O(n)$. Der wahlfreie Speicherzugriff erlaubt es, die Aufgabe schneller zu lösen als die kopfbewegungsgebundene Turingmaschine. $\square$

Beispiel A62 – TM für $L = \{0^{2^n}\}$

Eingabe 0000 ($= 2^2$, **in** L). Jeder Durchlauf ersetzt jede zweite 0 durch x , halbiert also die Anzahl der Nullen:

0 0 0 0	4 Nullen (gerade): jede zweite $\rightarrow x$, weiter
0 x 0 x	2 Nullen (gerade): jede zweite $\rightarrow x$, weiter
0 x x x	genau eine 0 $\Rightarrow$ akzeptiert

Eingabe 000 ($= 3$, **nicht in** L).

0 0 0	3 Nullen (ungerade): jede zweite $\rightarrow x$
0 x 0	letztes Symbol ist eine 0 (ungerade Anzahl) $\Rightarrow$ verworfen

Genau das prüft die Maschine: fortgesetztes Halbieren; trifft sie auf eine ungerade Anzahl > 1 , verwirft sie, sonst endet sie bei einer einzigen 0.