

Komplexitätstheorie Reduktionsmethode, erste NP-vollständige Probleme

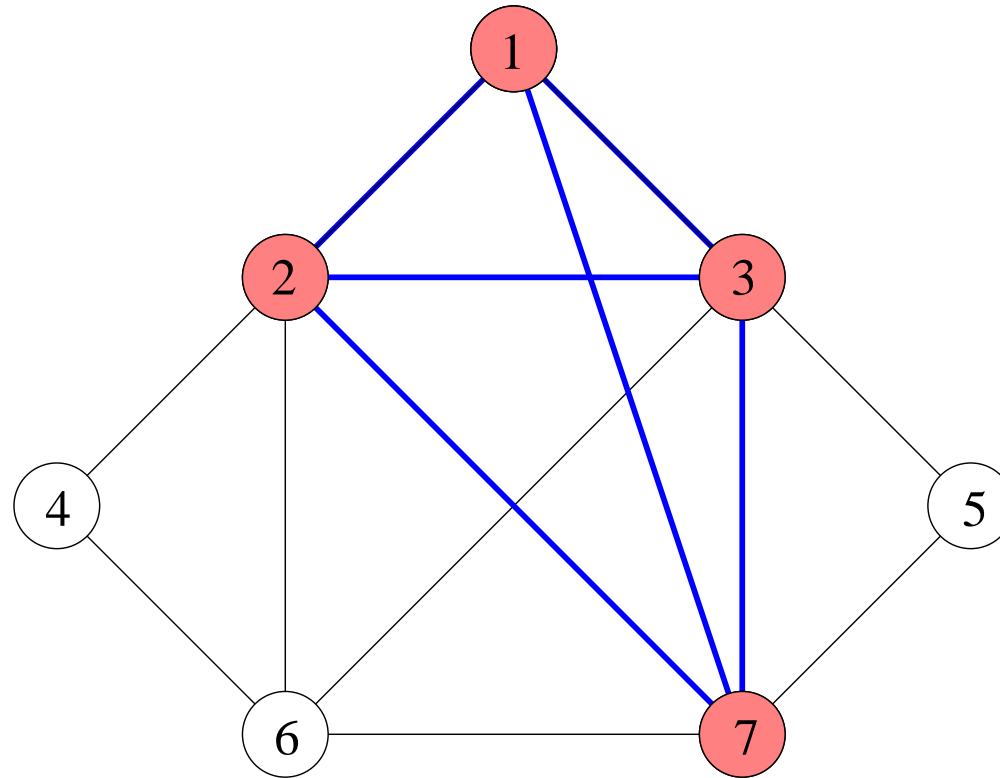
Martin Dietzfelbinger

11.+18.+25. Mai 2009

Kapitel 2

Grundlegende NP-vollständige Probleme

2.1 Das Cliquesproblem und seine Verwandten



Graph mit (maximal großer) **Clique**.

MAXCLIQUE: Gegeben ein ungerichteter Graph $G = (V, E)$, finde eine Clique V' in G mit möglichst vielen Knoten.

Ein Optimierungsproblem.

Zugehöriges Entscheidungsproblem: **CLIQUE:**

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

Frage: gibt es eine Clique V' in G mit (mindestens) k Knoten?

Satz 2.1.1

CLIQUE ist **NP**-vollständig.

Satz 2.1.1

CLIQUE ist **NP**-vollständig.

Beweis: (i) CLIQUE \in **NP**:

Satz 2.1.1

CLIQUE ist **NP**-vollständig.

Beweis: (i) CLIQUE \in **NP**: schon gesehen.

Satz 2.1.1

CLIQUE ist **NP**-vollständig.

Beweis: (i) CLIQUE \in **NP**: schon gesehen.

(ii) Mit Reduktionsmethode.

Satz 2.1.1

CLIQUE ist **NP**-vollständig.

Beweis: (i) CLIQUE \in **NP**: schon gesehen.

(ii) Mit Reduktionsmethode.

Behauptung: 3-SAT \leq_p CLIQUE.

Müssen: Funktion f definieren

Müssen: Funktion f definieren

$$f: \varphi \mapsto (G_\varphi, k_\varphi)$$

Müssen: Funktion f definieren

$$f: \varphi \mapsto (G_\varphi, k_\varphi)$$

φ : 3-KNF-Formel

Müssen: Funktion f definieren

$$f: \varphi \mapsto (G_\varphi, k_\varphi)$$

φ : 3-KNF-Formel

G_φ Graph,

Müssen: Funktion f definieren

$$f: \varphi \mapsto (G_\varphi, k_\varphi)$$

φ : 3-KNF-Formel

G_φ Graph, $k_\varphi \in \mathbb{N}$

Müssen: Funktion f definieren

$$f: \varphi \mapsto (G_\varphi, k_\varphi)$$

φ : 3-KNF-Formel

G_φ Graph, $k_\varphi \in \mathbb{N}$ mit

- f polynomialzeitberechenbar,

Müssen: Funktion f definieren

$$f: \varphi \mapsto (G_\varphi, k_\varphi)$$

φ : 3-KNF-Formel

G_φ Graph, $k_\varphi \in \mathbb{N}$ mit

- f polynomialzeitberechenbar, und
- Für jede 3-KNF-Formel φ gilt:

Müssen: Funktion f definieren

$$f: \varphi \mapsto (G_\varphi, k_\varphi)$$

φ : 3-KNF-Formel

G_φ Graph, $k_\varphi \in \mathbb{N}$ mit

- f polynomialzeitberechenbar, und
- Für jede 3-KNF-Formel φ gilt:

$$\varphi \in 3\text{-SAT}$$

Müssen: Funktion f definieren

$$f: \varphi \mapsto (G_\varphi, k_\varphi)$$

φ : 3-KNF-Formel

G_φ Graph, $k_\varphi \in \mathbb{N}$ mit

- f polynomialzeitberechenbar, und
- Für jede 3-KNF-Formel φ gilt:

$$\varphi \in 3\text{-SAT} \iff$$

Müssen: Funktion f definieren

$$f: \varphi \mapsto (G_\varphi, k_\varphi)$$

φ : 3-KNF-Formel

G_φ Graph, $k_\varphi \in \mathbb{N}$ mit

- f polynomialzeitberechenbar, und
- Für jede 3-KNF-Formel φ gilt:

$$\varphi \in \text{3-SAT} \iff f(\varphi) = (G_\varphi, k_\varphi) \in \text{CLIQUE.}$$

Müssen: Funktion f definieren

$$f: \varphi \mapsto (G_\varphi, k_\varphi)$$

φ : 3-KNF-Formel

G_φ Graph, $k_\varphi \in \mathbb{N}$ mit

- f polynomialzeitberechenbar, und
- Für jede 3-KNF-Formel φ gilt:

$$\varphi \in \text{3-SAT} \iff f(\varphi) = (G_\varphi, k_\varphi) \in \text{CLIQUE.}$$

(Syntaxcheck: $f(x) = 0$ falls x keine 3-KNF-Formel.)

Beispiel:

Beispiel:

$$\varphi = (X_1 \vee \overline{X}_2 \vee \overline{X}_4) \wedge (X_2 \vee X_1 \vee X_3) \wedge (\overline{X}_1 \vee \overline{X}_2 \vee X_4).$$

Beispiel:

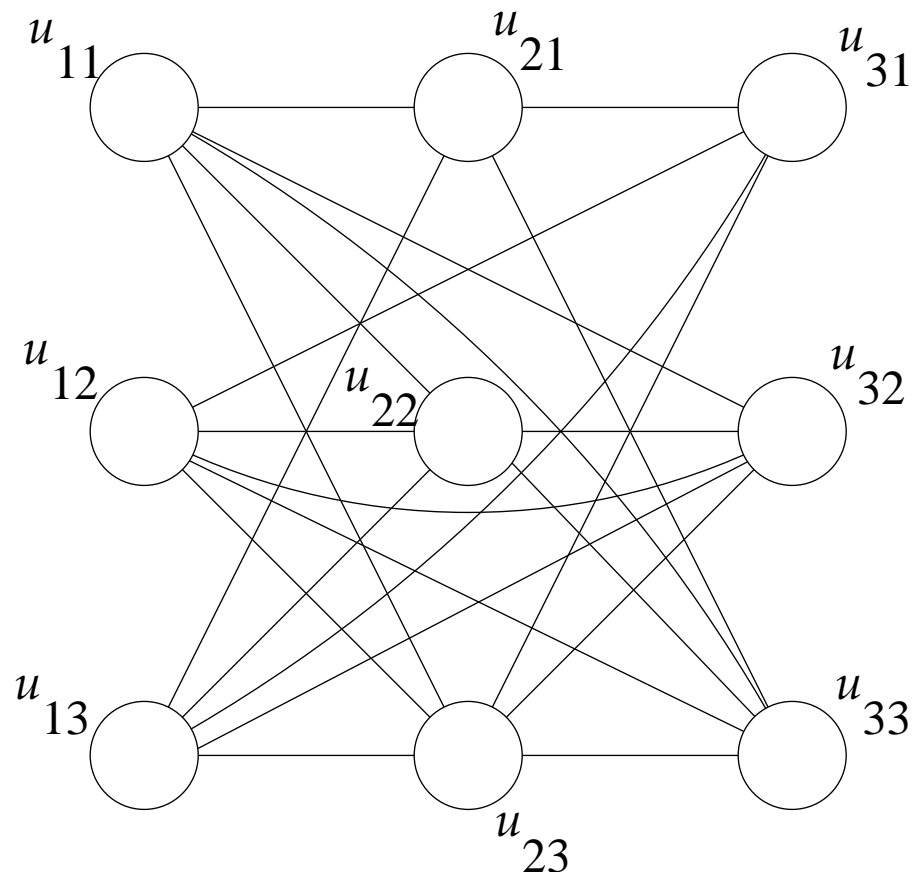
$$\varphi = (X_1 \vee \bar{X}_2 \vee \bar{X}_4) \wedge (X_2 \vee X_1 \vee X_3) \wedge (\bar{X}_1 \vee \bar{X}_2 \vee X_4).$$

G_φ :

Beispiel:

$$\varphi = (X_1 \vee \bar{X}_2 \vee \bar{X}_4) \wedge (X_2 \vee X_1 \vee X_3) \wedge (\bar{X}_1 \vee \bar{X}_2 \vee X_4).$$

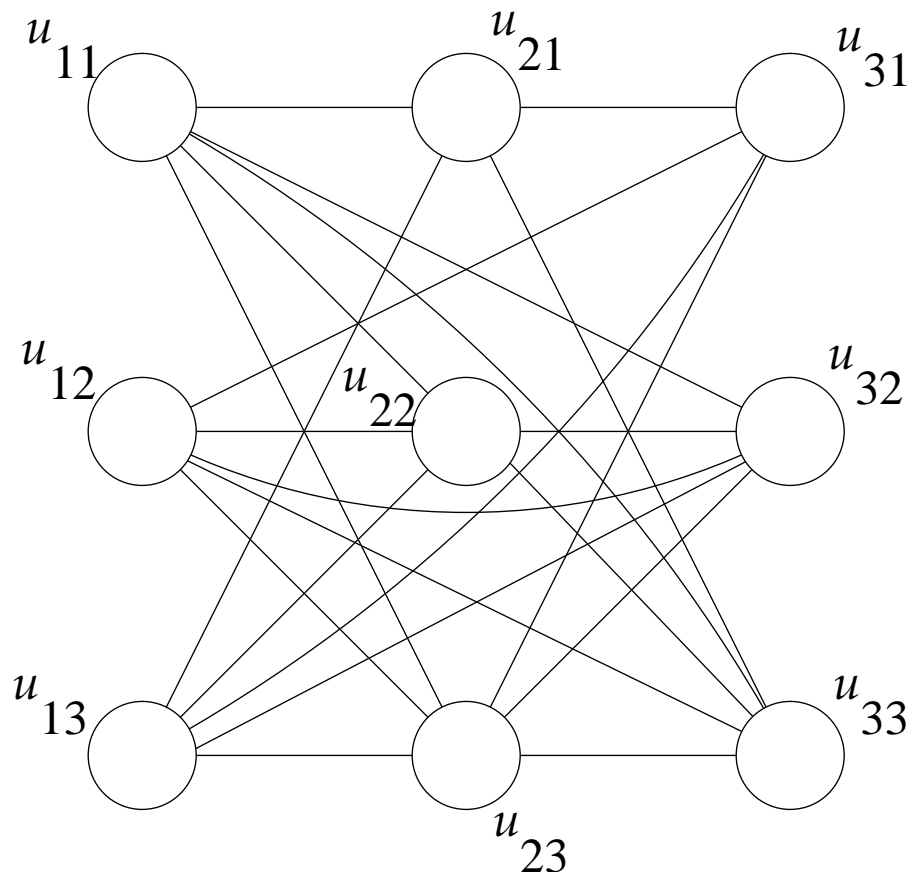
G_φ :



Beispiel:

$$\varphi = (X_1 \vee \bar{X}_2 \vee \bar{X}_4) \wedge (X_2 \vee X_1 \vee X_3) \wedge (\bar{X}_1 \vee \bar{X}_2 \vee X_4).$$

G_φ :



$k_\varphi := 3$ (Anzahl der Klauseln)

Formal: Gegeben

$$\varphi = C_1 \wedge \dots \wedge C_r$$

Formal: Gegeben

$$\varphi = C_1 \wedge \cdots \wedge C_r$$

mit

$$C_j = (l_{j1} \vee l_{j2} \vee l_{j3}), \quad 1 \leq j \leq r.$$

Formal: Gegeben

$$\varphi = C_1 \wedge \cdots \wedge C_r$$

mit

$$C_j = (l_{j1} \vee l_{j2} \vee l_{j3}), \quad 1 \leq j \leq r.$$

Graph G_φ hat $3r$ Knoten

Formal: Gegeben

$$\varphi = C_1 \wedge \cdots \wedge C_r$$

mit

$$C_j = (l_{j1} \vee l_{j2} \vee l_{j3}), \quad 1 \leq j \leq r.$$

Graph G_φ hat $3r$ Knoten

$$u_{j1}, u_{j2}, u_{j3}, \quad 1 \leq j \leq r.$$

Formal: Gegeben

$$\varphi = C_1 \wedge \cdots \wedge C_r$$

mit

$$C_j = (l_{j1} \vee l_{j2} \vee l_{j3}), \quad 1 \leq j \leq r.$$

Graph G_φ hat $3r$ Knoten

$$u_{j1}, u_{j2}, u_{j3}, \quad 1 \leq j \leq r.$$

Ein Knoten für jede **Literal-Position** in φ .

Formal: Gegeben

$$\varphi = C_1 \wedge \cdots \wedge C_r$$

mit

$$C_j = (l_{j1} \vee l_{j2} \vee l_{j3}), \quad 1 \leq j \leq r.$$

Graph G_φ hat $3r$ Knoten

$$u_{j1}, u_{j2}, u_{j3}, \quad 1 \leq j \leq r.$$

Ein Knoten für jede **Literal-Position** in φ .

$$k_\varphi = r.$$

$3r$ Knoten

$$u_{j1}, u_{j2}, u_{j3}, \quad 1 \leq j \leq r.$$

$3r$ Knoten

$$u_{j1}, u_{j2}, u_{j3}, \quad 1 \leq j \leq r.$$

Anordnung in r Spalten

$3r$ Knoten

$$u_{j1}, u_{j2}, u_{j3}, \quad 1 \leq j \leq r.$$

Anordnung in r Spalten ($\hat{=}$ Klauseln)

$3r$ Knoten

$$u_{j1}, u_{j2}, u_{j3}, \quad 1 \leq j \leq r.$$

Anordnung in r Spalten ($\hat{=}$ Klauseln) mit je 3 Knoten.

$3r$ Knoten

$$u_{j1}, u_{j2}, u_{j3}, \quad 1 \leq j \leq r.$$

Anordnung in r Spalten ($\hat{=}$ Klauseln) mit je 3 Knoten.

Kanten E_φ :

$3r$ Knoten

$$u_{j1}, u_{j2}, u_{j3}, \quad 1 \leq j \leq r.$$

Anordnung in r Spalten ($\hat{=}$ Klauseln) mit je 3 Knoten.

Kanten E_φ :

Keine Kante innerhalb einer Spalte.

$3r$ Knoten

$$u_{j1}, u_{j2}, u_{j3}, 1 \leq j \leq r.$$

Anordnung in r Spalten ($\hat{=}$ Klauseln) mit je 3 Knoten.

Kanten E_φ :

Keine Kante innerhalb einer Spalte.

$j \neq j'$:

$3r$ Knoten

$$u_{j1}, u_{j2}, u_{j3}, 1 \leq j \leq r.$$

Anordnung in r Spalten ($\hat{=}$ Klauseln) mit je 3 Knoten.

Kanten E_φ :

Keine Kante innerhalb einer Spalte.

$j \neq j'$: Kante zwischen u_{js} und $u_{j's'}$

$3r$ Knoten

$$u_{j1}, u_{j2}, u_{j3}, 1 \leq j \leq r.$$

Anordnung in r Spalten ($\hat{=}$ Klauseln) mit je 3 Knoten.

Kanten E_φ :

Keine Kante innerhalb einer Spalte.

$j \neq j'$: Kante zwischen u_{js} und $u_{j's'}$

existiert genau dann wenn

$3r$ Knoten

$$u_{j1}, u_{j2}, u_{j3}, 1 \leq j \leq r.$$

Anordnung in r Spalten ($\hat{=}$ Klauseln) mit je 3 Knoten.

Kanten E_φ :

Keine Kante innerhalb einer Spalte.

$j \neq j'$: Kante zwischen u_{js} und $u_{j's'}$

existiert genau dann wenn

nicht l_{js} und $l_{j's'}$ **entgegengesetzte Literale** sind.

$3r$ Knoten

$$u_{j1}, u_{j2}, u_{j3}, \quad 1 \leq j \leq r.$$

Anordnung in r Spalten ($\hat{=}$ Klauseln) mit je 3 Knoten.

Kanten E_φ :

Keine Kante innerhalb einer Spalte.

$j \neq j'$: Kante zwischen u_{js} und $u_{j's'}$

existiert genau dann wenn

nicht l_{js} und $l_{j's'}$ **entgegengesetzte Literale** sind.

(Entgegengesetzt sind z.B. X_3 und \overline{X}_3 .)

$3r$ Knoten

$$u_{j1}, u_{j2}, u_{j3}, \quad 1 \leq j \leq r.$$

Anordnung in r Spalten ($\hat{=}$ Klauseln) mit je 3 Knoten.

Kanten E_φ :

Keine Kante innerhalb einer Spalte.

$j \neq j'$: Kante zwischen u_{js} und $u_{j's'}$

existiert genau dann wenn

nicht l_{js} und $l_{j's'}$ **entgegengesetzte Literale** sind.

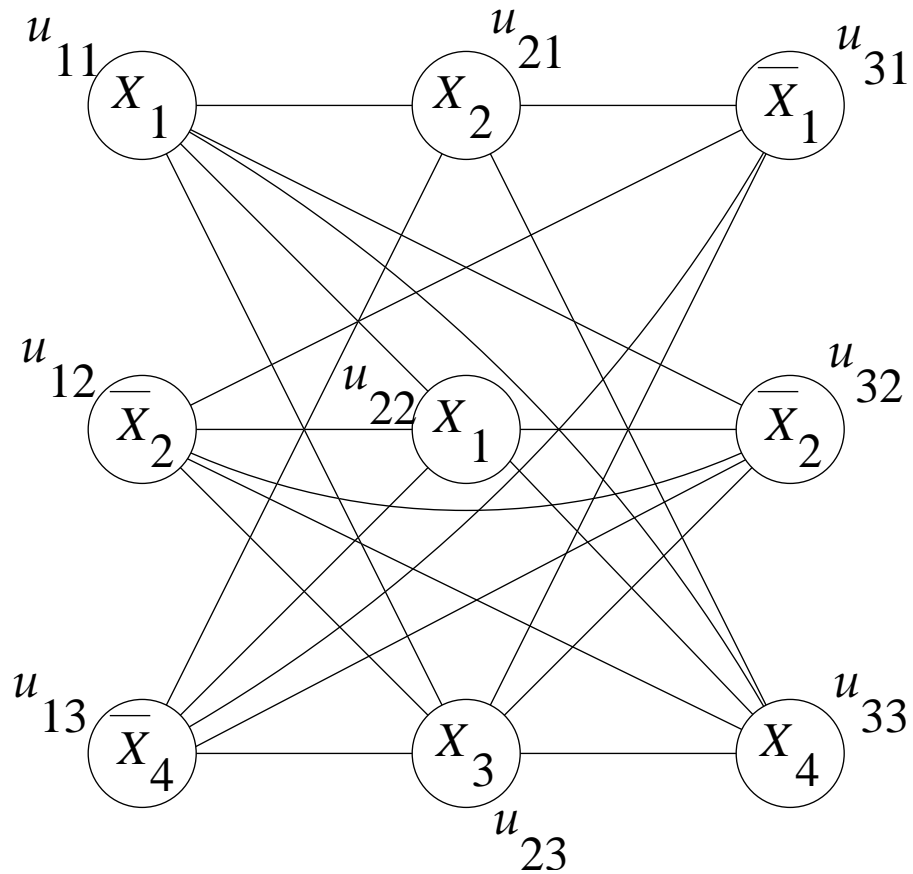
(Entgegengesetzt sind z.B. X_3 und \overline{X}_3 .)

Die im Bild eingetragenen Literale sind **nicht** Teil des Graphen G_φ ; sie dienen nur zur Orientierung.

Beispiel:

$$\varphi = (X_1 \vee \bar{X}_2 \vee \bar{X}_4) \wedge (X_2 \vee X_1 \vee X_3) \wedge (\bar{X}_1 \vee \bar{X}_2 \vee X_4).$$

G_φ :

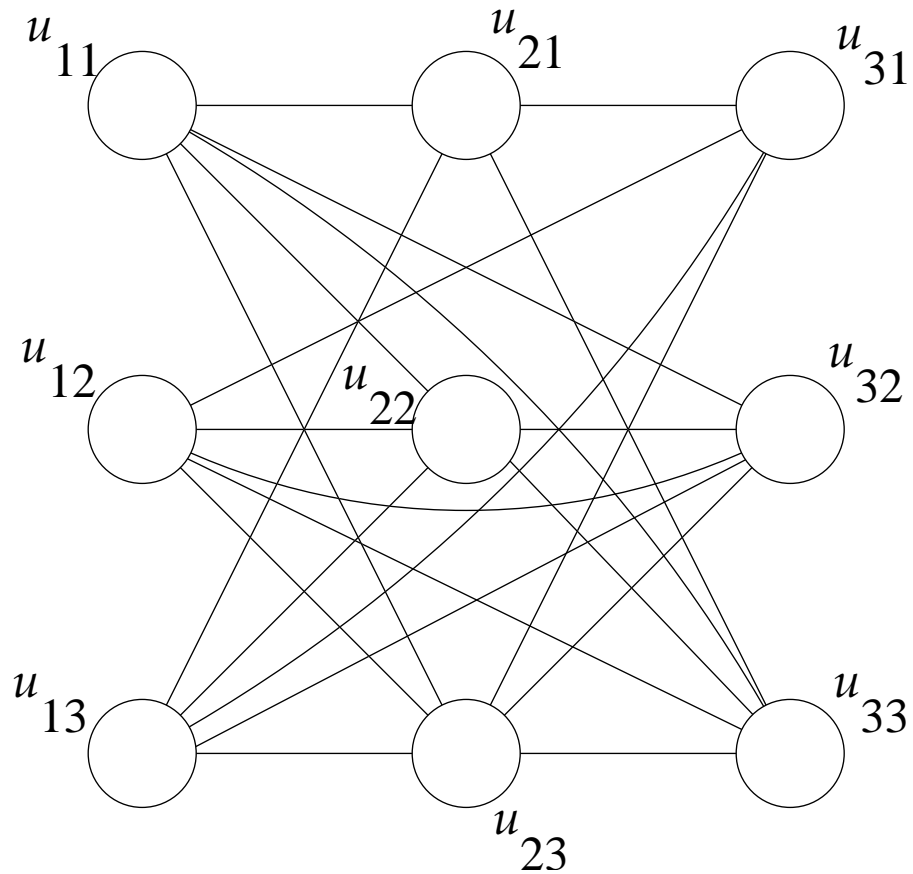


$$k_\varphi = 3$$

Beispiel:

$$\varphi = (X_1 \vee \bar{X}_2 \vee \bar{X}_4) \wedge (X_2 \vee X_1 \vee X_3) \wedge (\bar{X}_1 \vee \bar{X}_2 \vee X_4).$$

G_φ :



$$k_\varphi = 3$$

Nicht schwer zu sehen:

Die Funktion $f: \varphi \mapsto (G_\varphi, k_\varphi)$
ist in polynomieller Zeit berechenbar.

Nicht schwer zu sehen:

Die Funktion $f: \varphi \mapsto (G_\varphi, k_\varphi)$
ist in polynomieller Zeit berechenbar.

Behauptung:

Nicht schwer zu sehen:

Die Funktion $f: \varphi \mapsto (G_\varphi, k_\varphi)$
ist in polynomieller Zeit berechenbar.

Behauptung: Für jede 3-KNF-Formel φ gilt:

Nicht schwer zu sehen:

Die Funktion $f: \varphi \mapsto (G_\varphi, k_\varphi)$
ist in polynomieller Zeit berechenbar.

Behauptung: Für jede 3-KNF-Formel φ gilt:

$$\varphi \in \text{3-SAT} \Leftrightarrow f(\varphi) \in \text{CLIQUE},$$

Nicht schwer zu sehen:

Die Funktion $f: \varphi \mapsto (G_\varphi, k_\varphi)$
ist in polynomieller Zeit berechenbar.

Behauptung: Für jede 3-KNF-Formel φ gilt:

$$\varphi \in \text{3-SAT} \Leftrightarrow f(\varphi) \in \text{CLIQUE},$$

das heißt:

Nicht schwer zu sehen:

Die Funktion $f: \varphi \mapsto (G_\varphi, k_\varphi)$
ist in polynomieller Zeit berechenbar.

Behauptung: Für jede 3-KNF-Formel φ gilt:

$$\varphi \in \text{3-SAT} \Leftrightarrow f(\varphi) \in \text{CLIQUE},$$

das heißt:

$$\varphi \text{ ist erfüllbar} \Leftrightarrow G_\varphi \text{ hat Clique der Größe } k_\varphi.$$

Zu zeigen: φ ist erfüllbar $\Leftrightarrow (G_\varphi, k_\varphi) \in \text{CLIQUE}$.

Zu zeigen: φ ist erfüllbar $\Leftrightarrow (G_\varphi, k_\varphi) \in \text{CLIQUE}$.

„ \Rightarrow “:

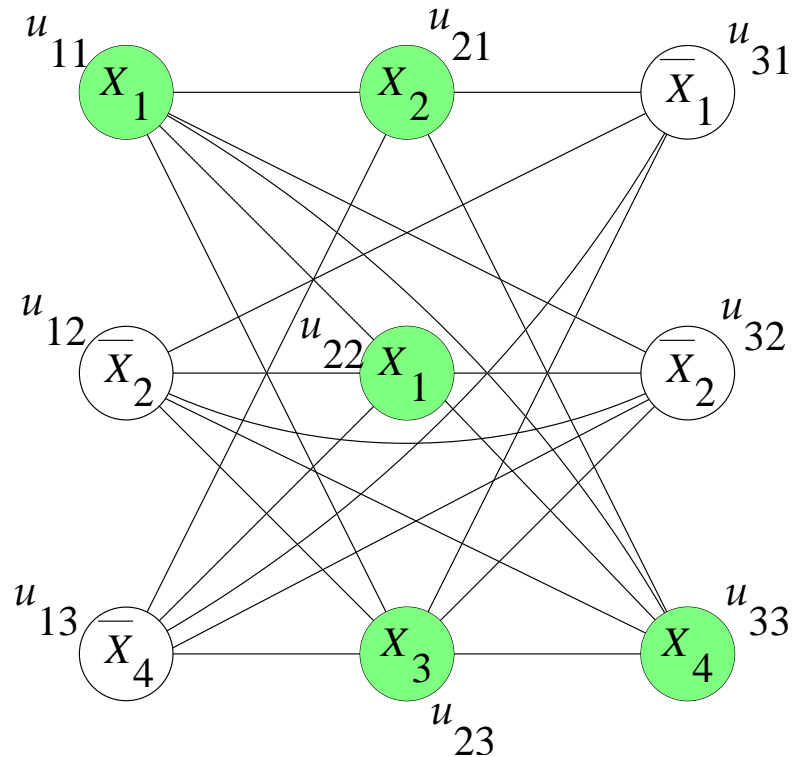
Zu zeigen: φ ist erfüllbar $\Leftrightarrow (G_\varphi, k_\varphi) \in \text{CLIQUE}$.

„ \Rightarrow “: Gegeben: Belegung v mit $v(\varphi) = 1$.

Zu zeigen: φ ist erfüllbar $\Leftrightarrow (G_\varphi, k_\varphi) \in \text{CLIQUE}$.

„ \Rightarrow “: Gegeben: Belegung v mit $v(\varphi) = 1$.

Beispiel: $v(X_1) = v(X_2) = v(X_3) = v(X_4) = 1$.

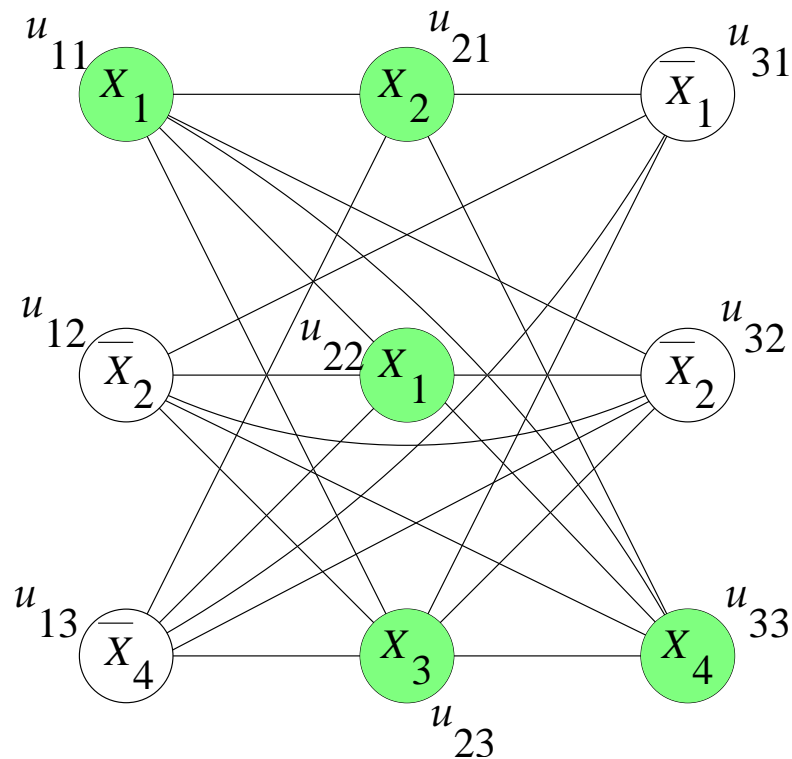


In jeder Spalte gibt es ein wahres Literal.

Zu zeigen: φ ist erfüllbar $\Leftrightarrow (G_\varphi, k_\varphi) \in \text{CLIQUE}$.

„ \Rightarrow “: Gegeben: Belegung v mit $v(\varphi) = 1$.

Beispiel: $v(X_1) = v(X_2) = v(X_3) = v(X_4) = 1$.

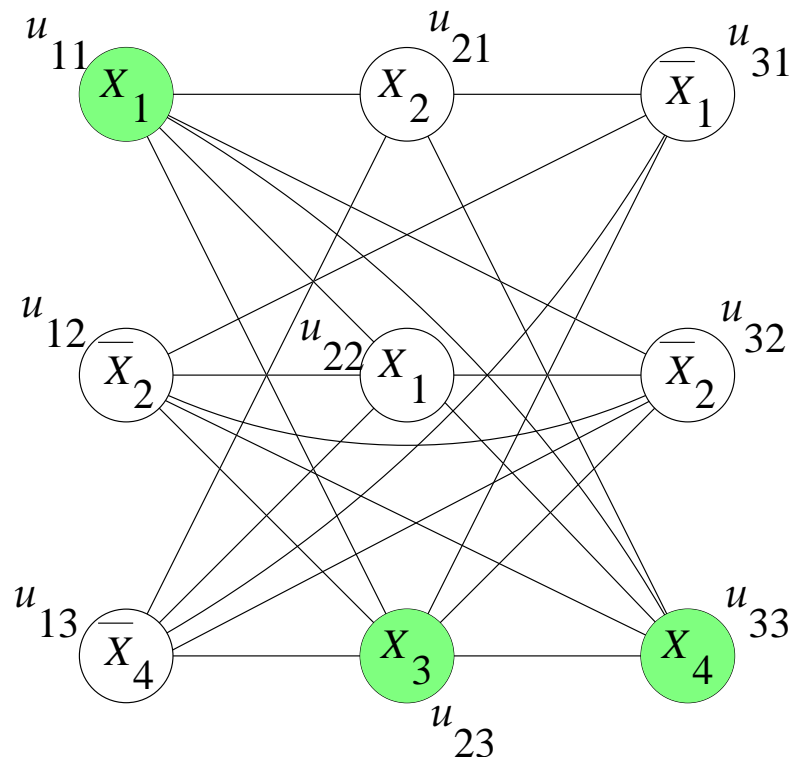


In jeder Spalte gibt es ein wahres Literal. Keine Clique!

Zu zeigen: φ ist erfüllbar $\Leftrightarrow (G_\varphi, k_\varphi) \in \text{CLIQUE}$.

„ \Rightarrow “: Gegeben: Belegung v mit $v(\varphi) = 1$.

Beispiel: $v(X_1) = v(X_2) = v(X_3) = v(X_4) = 1$.



Aus jeder Spalte ein wahres Literal **wählen!** \rightarrow Clique.

Allgemein:

In jeder Klausel C_j

Allgemein:

In jeder Klausel C_j **gibt es ein** Literal l_{j,s_j}

Allgemein:

In jeder Klausel C_j **gibt es ein** Literal l_{j,s_j} mit $v(l_{j,s_j}) = 1$.

Allgemein:

In jeder Klausel C_j **gibt es ein** Literal l_{j,s_j} mit $v(l_{j,s_j}) = 1$.

Dann ist

Allgemein:

In jeder Klausel C_j **gibt es ein** Literal l_{j,s_j} mit $v(l_{j,s_j}) = 1$.

Dann ist

$$V'$$

Allgemein:

In jeder Klausel C_j **gibt es ein** Literal l_{j,s_j} mit $v(l_{j,s_j}) = 1$.

Dann ist

$$V' := \{u_{j,s_j} \mid$$

Allgemein:

In jeder Klausel C_j **gibt es ein** Literal l_{j,s_j} mit $v(l_{j,s_j}) = 1$.

Dann ist

$$V' := \{u_{j,s_j} \mid 1 \leq j \leq r\}$$

Allgemein:

In jeder Klausel C_j **gibt es ein** Literal l_{j,s_j} mit $v(l_{j,s_j}) = 1$.

Dann ist

$$V' := \{u_{j,s_j} \mid 1 \leq j \leq r\}$$

eine Clique in G_φ mit r Knoten.

Allgemein:

In jeder Klausel C_j **gibt es ein** Literal l_{j,s_j} mit $v(l_{j,s_j}) = 1$.

Dann ist

$$V' := \{u_{j,s_j} \mid 1 \leq j \leq r\}$$

eine Clique in G_φ mit r Knoten.

Denn:

Allgemein:

In jeder Klausel C_j **gibt es ein** Literal l_{j,s_j} mit $v(l_{j,s_j}) = 1$.

Dann ist

$$V' := \{u_{j,s_j} \mid 1 \leq j \leq r\}$$

eine Clique in G_φ mit r Knoten.

Denn: l_{j,s_j} und $l_{j',s_{j'}}$ haben unter v den Wert 1

Allgemein:

In jeder Klausel C_j **gibt es ein** Literal l_{j,s_j} mit $v(l_{j,s_j}) = 1$.

Dann ist

$$V' := \{u_{j,s_j} \mid 1 \leq j \leq r\}$$

eine Clique in G_φ mit r Knoten.

Denn: l_{j,s_j} und $l_{j',s_{j'}}$ haben unter v den Wert 1

\Rightarrow können nicht entgegengesetzte Literale sein

Allgemein:

In jeder Klausel C_j **gibt es ein** Literal l_{j,s_j} mit $v(l_{j,s_j}) = 1$.

Dann ist

$$V' := \{u_{j,s_j} \mid 1 \leq j \leq r\}$$

eine Clique in G_φ mit r Knoten.

Denn: l_{j,s_j} und $l_{j',s_{j'}}$ haben unter v den Wert 1

\Rightarrow können nicht entgegengesetzte Literale sein

$\Rightarrow (u_{j,s_j}, u_{j',s_{j'}}) \in E_\varphi$.

Zu zeigen: φ ist erfüllbar $\Leftrightarrow (G_\varphi, k_\varphi) \in \text{CLIQUE}$.

Zu zeigen: φ ist erfüllbar $\Leftrightarrow (G_\varphi, k_\varphi) \in \text{CLIQUE}$.

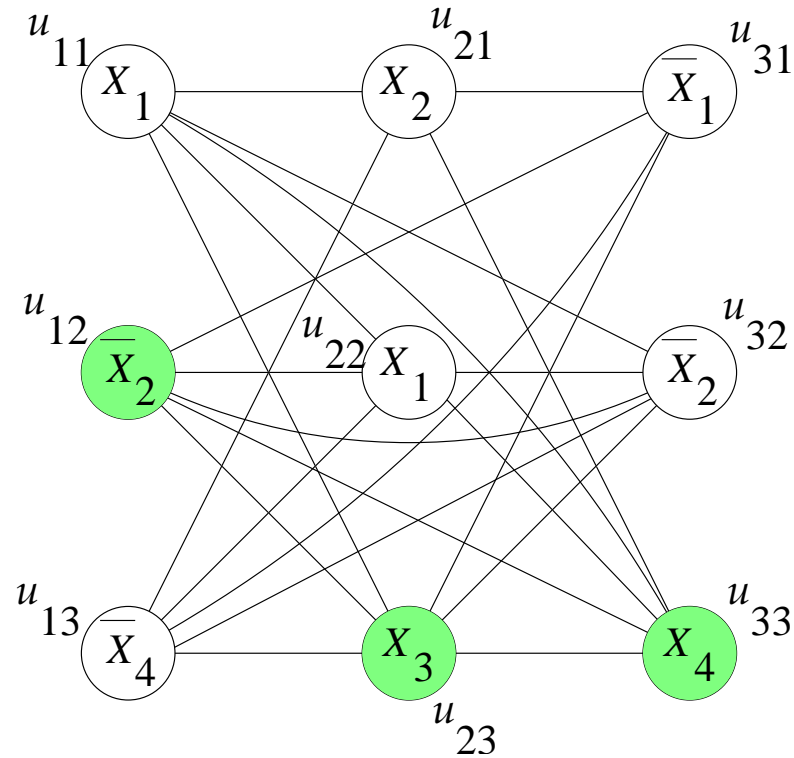
„ \Leftarrow “:

Zu zeigen: φ ist erfüllbar $\Leftrightarrow (G_\varphi, k_\varphi) \in \text{CLIQUE}$.

„ \Leftarrow “: Gegeben: Clique V' in G_φ mit r Knoten.

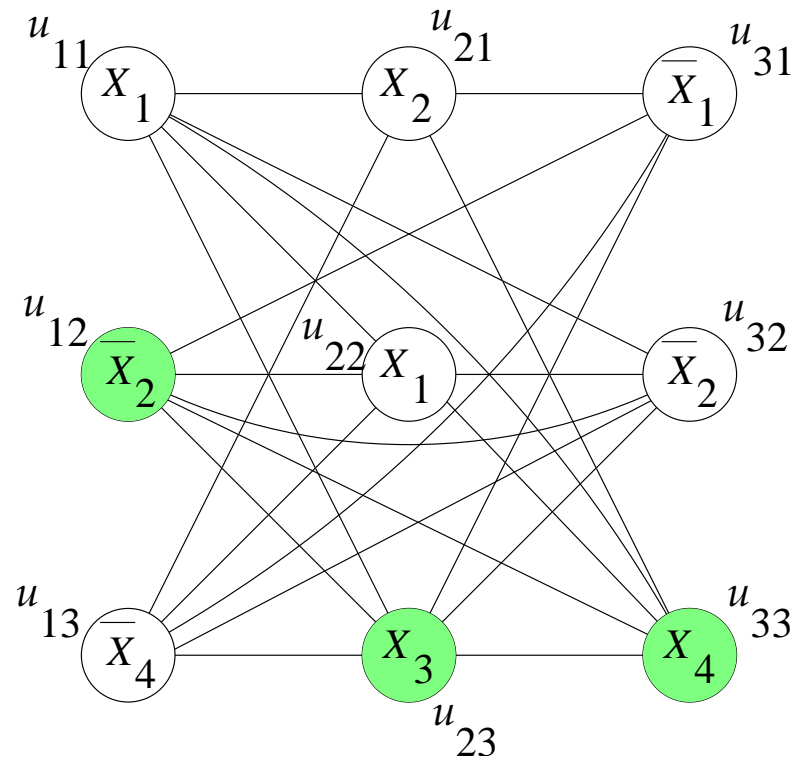
Zu zeigen: φ ist erfüllbar $\Leftrightarrow (G_\varphi, k_\varphi) \in \text{CLIQUE}$.

„ \Leftarrow “: Gegeben: Clique V' in G_φ mit r Knoten. *Beispiel:*



Zu zeigen: φ ist erfüllbar $\Leftrightarrow (G_\varphi, k_\varphi) \in \text{CLIQUE}$.

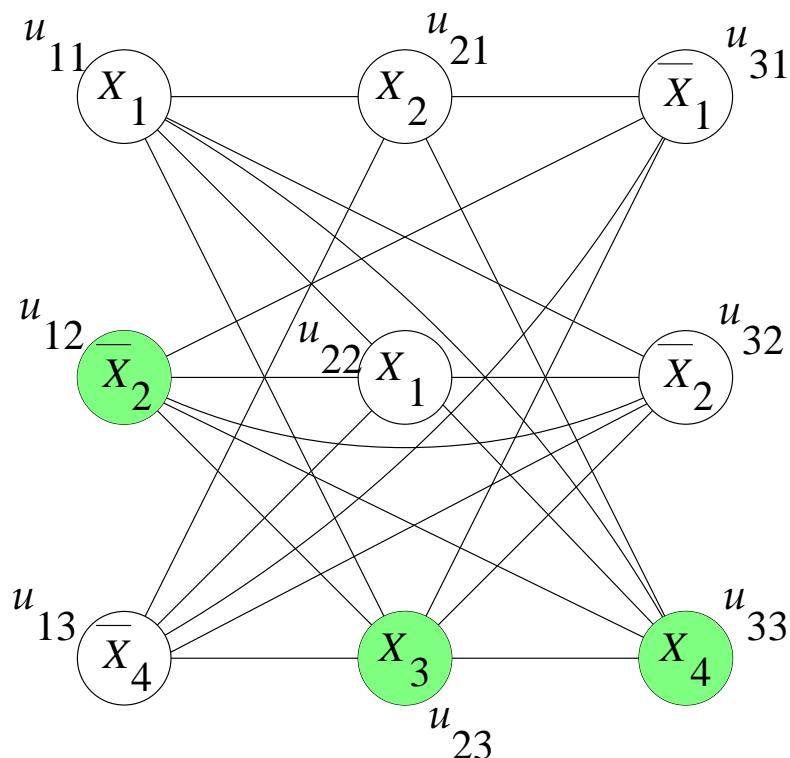
„ \Leftarrow “: Gegeben: Clique V' in G_φ mit r Knoten. *Beispiel:*



Definiere Belegung: $v(X_3) = v(X_4) = 1$, die anderen auf 0.

Zu zeigen: φ ist erfüllbar $\Leftrightarrow (G_\varphi, k_\varphi) \in \text{CLIQUE}$.

„ \Leftarrow “: Gegeben: Clique V' in G_φ mit r Knoten. *Beispiel:*



Definiere Belegung: $v(X_3) = v(X_4) = 1$, die anderen auf 0.

Erfüllend für

$$\varphi = (X_1 \vee \bar{X}_2 \vee \bar{X}_4) \wedge (X_2 \vee X_1 \vee X_3) \wedge (\bar{X}_1 \vee \bar{X}_2 \vee X_4).$$

Zu zeigen: φ ist erfüllbar $\Leftrightarrow (G_\varphi, k_\varphi) \in \text{CLIQUE}$.

„ \Leftarrow “:

Zu zeigen: φ ist erfüllbar $\Leftrightarrow (G_\varphi, k_\varphi) \in \text{CLIQUE}$.

„ \Leftarrow “: Gegeben: Clique V' in G_φ mit r Knoten.

\Rightarrow r Knoten liegen in verschiedenen Spalten

Zu zeigen: φ ist erfüllbar $\Leftrightarrow (G_\varphi, k_\varphi) \in \text{CLIQUE}$.

„ \Leftarrow “: Gegeben: Clique V' in G_φ mit r Knoten.

\Rightarrow r Knoten liegen in verschiedenen Spalten

\Rightarrow für jedes j gibt es genau ein $s_j \in \{1, 2, 3\}$,

Zu zeigen: φ ist erfüllbar $\Leftrightarrow (G_\varphi, k_\varphi) \in \text{CLIQUE}$.

„ \Leftarrow “: Gegeben: Clique V' in G_φ mit r Knoten.

\Rightarrow r Knoten liegen in verschiedenen Spalten

\Rightarrow für jedes j gibt es genau ein $s_j \in \{1, 2, 3\}$, so dass

$$V' = \{u_{j,s_j} \mid 1 \leq j \leq r\}.$$

Zu zeigen: φ ist erfüllbar $\Leftrightarrow (G_\varphi, k_\varphi) \in \text{CLIQUE}$.

„ \Leftarrow “: Gegeben: Clique V' in G_φ mit r Knoten.

\Rightarrow r Knoten liegen in verschiedenen Spalten

\Rightarrow für jedes j gibt es genau ein $s_j \in \{1, 2, 3\}$, so dass

$$V' = \{u_{j,s_j} \mid 1 \leq j \leq r\}.$$

Definiere Belegung:

Zu zeigen: φ ist erfüllbar $\Leftrightarrow (G_\varphi, k_\varphi) \in \text{CLIQUE}$.

„ \Leftarrow “: Gegeben: Clique V' in G_φ mit r Knoten.

\Rightarrow r Knoten liegen in verschiedenen Spalten

\Rightarrow für jedes j gibt es genau ein $s_j \in \{1, 2, 3\}$, so dass

$$V' = \{u_{j,s_j} \mid 1 \leq j \leq r\}.$$

Definiere Belegung:

$$v(X_i) :=$$

Zu zeigen: φ ist erfüllbar $\Leftrightarrow (G_\varphi, k_\varphi) \in \text{CLIQUE}$.

„ \Leftarrow “: Gegeben: Clique V' in G_φ mit r Knoten.

\Rightarrow r Knoten liegen in verschiedenen Spalten

\Rightarrow für jedes j gibt es genau ein $s_j \in \{1, 2, 3\}$, so dass

$$V' = \{u_{j,s_j} \mid 1 \leq j \leq r\}.$$

Definiere Belegung:

$$v(X_i) := \begin{cases} 1 & \text{falls } X_i = l_{j,s_j} \\ & \end{cases}$$

Zu zeigen: φ ist erfüllbar $\Leftrightarrow (G_\varphi, k_\varphi) \in \text{CLIQUE}$.

„ \Leftarrow “: Gegeben: Clique V' in G_φ mit r Knoten.

\Rightarrow r Knoten liegen in verschiedenen Spalten

\Rightarrow für jedes j gibt es genau ein $s_j \in \{1, 2, 3\}$, so dass

$$V' = \{u_{j,s_j} \mid 1 \leq j \leq r\}.$$

Definiere Belegung:

$$v(X_i) := \begin{cases} 1 & \text{falls } X_i = l_{j,s_j} \text{ für ein } j, \\ & \end{cases}$$

Zu zeigen: φ ist erfüllbar $\Leftrightarrow (G_\varphi, k_\varphi) \in \text{CLIQUE}$.

„ \Leftarrow “: Gegeben: Clique V' in G_φ mit r Knoten.

\Rightarrow r Knoten liegen in verschiedenen Spalten

\Rightarrow für jedes j gibt es genau ein $s_j \in \{1, 2, 3\}$, so dass

$$V' = \{u_{j,s_j} \mid 1 \leq j \leq r\}.$$

Definiere Belegung:

$$v(X_i) := \begin{cases} 1 & \text{falls } X_i = l_{j,s_j} \text{ für ein } j, \\ 0 & \text{sonst.} \end{cases}$$

$$v(X_i) := \begin{cases} 1 & \text{falls } X_i = l_{j,s_j} \text{ für ein } j, \\ 0 & \text{sonst.} \end{cases}$$

Sei nun j beliebig, $1 \leq j \leq r$.



$$v(X_i) := \begin{cases} 1 & \text{falls } X_i = l_{j,s_j} \text{ für ein } j, \\ 0 & \text{sonst.} \end{cases}$$

Sei nun j beliebig, $1 \leq j \leq r$.

1. Fall: $l_{j,s_j} = X_i$.



$$v(X_i) := \begin{cases} 1 & \text{falls } X_i = l_{j,s_j} \text{ für ein } j, \\ 0 & \text{sonst.} \end{cases}$$

Sei nun j beliebig, $1 \leq j \leq r$.

1. Fall: $l_{j,s_j} = X_i$.

Dann $v(X_i) = 1$,



$$v(X_i) := \begin{cases} 1 & \text{falls } X_i = l_{j,s_j} \text{ für ein } j, \\ 0 & \text{sonst.} \end{cases}$$

Sei nun j beliebig, $1 \leq j \leq r$.

1. Fall: $l_{j,s_j} = X_i$.

Dann $v(X_i) = 1$, also $v(C_j) = 1$.



$$v(X_i) := \begin{cases} 1 & \text{falls } X_i = l_{j,s_j} \text{ für ein } j, \\ 0 & \text{sonst.} \end{cases}$$

Sei nun j beliebig, $1 \leq j \leq r$.

1. Fall: $l_{j,s_j} = X_i$.

Dann $v(X_i) = 1$, also $v(C_j) = 1$.

2. Fall: $l_{j,s_j} = \overline{X_i}$.



$$v(X_i) := \begin{cases} 1 & \text{falls } X_i = l_{j,s_j} \text{ für ein } j, \\ 0 & \text{sonst.} \end{cases}$$

Sei nun j beliebig, $1 \leq j \leq r$.

1. Fall: $l_{j,s_j} = X_i$.

Dann $v(X_i) = 1$, also $v(C_j) = 1$.

2. Fall: $l_{j,s_j} = \overline{X_i}$.

Dann kann nicht $v(X_i) = 1$ sein.



$$v(X_i) := \begin{cases} 1 & \text{falls } X_i = l_{j,s_j} \text{ für ein } j, \\ 0 & \text{sonst.} \end{cases}$$

Sei nun j beliebig, $1 \leq j \leq r$.

1. Fall: $l_{j,s_j} = X_i$.

Dann $v(X_i) = 1$, also $v(C_j) = 1$.

2. Fall: $l_{j,s_j} = \overline{X_i}$.

Dann kann nicht $v(X_i) = 1$ sein.

Sonst wäre $l_{j',s_{j'}} = X_i$ für einen Knoten $u_{j',s_{j'}}$ in Clique V' .



$$v(X_i) := \begin{cases} 1 & \text{falls } X_i = l_{j,s_j} \text{ für ein } j, \\ 0 & \text{sonst.} \end{cases}$$

Sei nun j beliebig, $1 \leq j \leq r$.

1. Fall: $l_{j,s_j} = X_i$.

Dann $v(X_i) = 1$, also $v(C_j) = 1$.

2. Fall: $l_{j,s_j} = \overline{X_i}$.

Dann kann nicht $v(X_i) = 1$ sein.

Sonst wäre $l_{j',s_{j'}} = X_i$ für einen Knoten $u_{j',s_{j'}}$ in Clique V' .

Das ist unmöglich, weil



$$v(X_i) := \begin{cases} 1 & \text{falls } X_i = l_{j,s_j} \text{ für ein } j, \\ 0 & \text{sonst.} \end{cases}$$

Sei nun j beliebig, $1 \leq j \leq r$.

1. Fall: $l_{j,s_j} = X_i$.

Dann $v(X_i) = 1$, also $v(C_j) = 1$.

2. Fall: $l_{j,s_j} = \overline{X_i}$.

Dann kann nicht $v(X_i) = 1$ sein.

Sonst wäre $l_{j',s_{j'}} = X_i$ für einen Knoten $u_{j',s_{j'}}$ in Clique V' .

Das ist unmöglich, weil $(u_{j,s_j}, u_{j',s_{j'}})$ keine Kante in E_φ ist.

□

$$v(X_i) := \begin{cases} 1 & \text{falls } X_i = l_{j,s_j} \text{ für ein } j, \\ 0 & \text{sonst.} \end{cases}$$

Sei nun j beliebig, $1 \leq j \leq r$.

1. Fall: $l_{j,s_j} = X_i$.

Dann $v(X_i) = 1$, also $v(C_j) = 1$.

2. Fall: $l_{j,s_j} = \overline{X_i}$.

Dann kann nicht $v(X_i) = 1$ sein.

Sonst wäre $l_{j',s_{j'}} = X_i$ für einen Knoten $u_{j',s_{j'}}$ in Clique V' .

Das ist unmöglich, weil $(u_{j,s_j}, u_{j',s_{j'}})$ keine Kante in E_φ ist.

Also $v(X_i) = 0$ und $v(l_{j,s_j}) = 1$; □

$$v(X_i) := \begin{cases} 1 & \text{falls } X_i = l_{j,s_j} \text{ für ein } j, \\ 0 & \text{sonst.} \end{cases}$$

Sei nun j beliebig, $1 \leq j \leq r$.

1. Fall: $l_{j,s_j} = X_i$.

Dann $v(X_i) = 1$, also $v(C_j) = 1$.

2. Fall: $l_{j,s_j} = \overline{X_i}$.

Dann kann nicht $v(X_i) = 1$ sein.

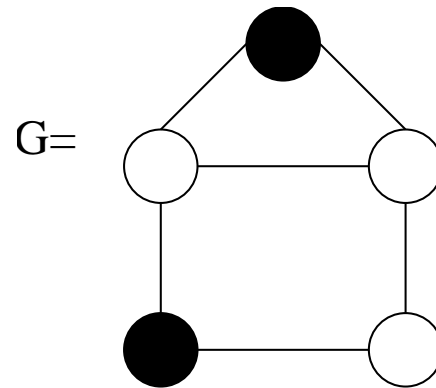
Sonst wäre $l_{j',s_{j'}} = X_i$ für einen Knoten $u_{j',s_{j'}}$ in Clique V' .

Das ist unmöglich, weil $(u_{j,s_j}, u_{j',s_{j'}})$ keine Kante in E_φ ist.

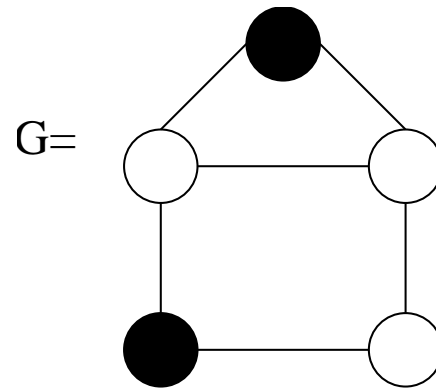
Also $v(X_i) = 0$ und $v(l_{j,s_j}) = 1$; also $v(C_j) = 1$. □

„Unabhängige Menge“ / „Independent Set“:

„Unabhängige Menge“ / „Independent Set“:

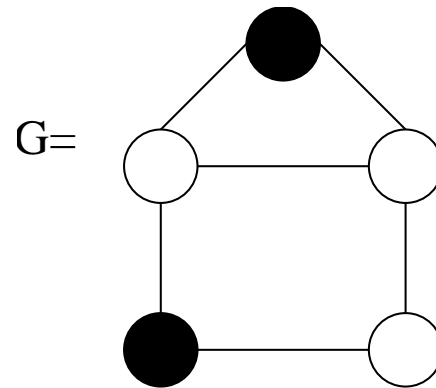


„Unabhängige Menge“ / „Independent Set“:



Definition

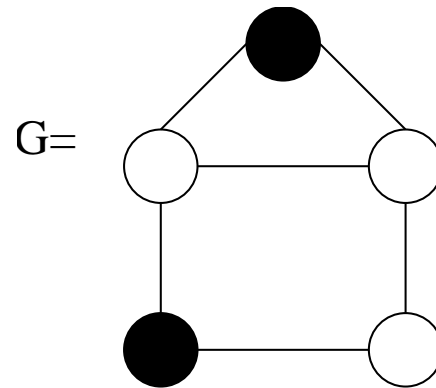
„Unabhängige Menge“ / „Independent Set“:



Definition

$G = (V, E)$ sei ein Graph.

„Unabhängige Menge“ / „Independent Set“:

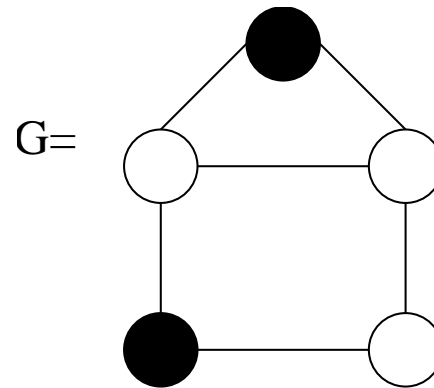


Definition

$G = (V, E)$ sei ein Graph.

$V' \subseteq V$ heißt **unabhängig**

„Unabhängige Menge“ / „Independent Set“:

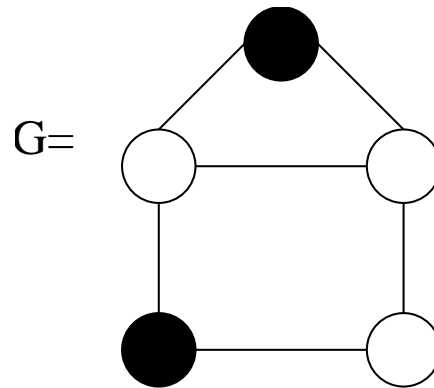


Definition

$G = (V, E)$ sei ein Graph.

$V' \subseteq V$ heißt **unabhängig**
(auch „stabil“, engl. „**independent set**“) in G ,

„Unabhängige Menge“ / „Independent Set“:



Definition

$G = (V, E)$ sei ein Graph.

$V' \subseteq V$ heißt **unabhängig**

(auch „stabil“, engl. „**independent set**“) in G ,

wenn es **innerhalb von V' keine Kante gibt.**

Das Independent-Set-Problem: Ein Maximierungsproblem.

Das Independent-Set-Problem: Ein Maximierungsproblem.

MAXIS: Gegeben Graph $G = (V, E)$,

Das Independent-Set-Problem: Ein Maximierungsproblem.

MAXIS: Gegeben Graph $G = (V, E)$,
finde unabhängige Menge $V' \subseteq V$

Das Independent-Set-Problem: Ein Maximierungsproblem.

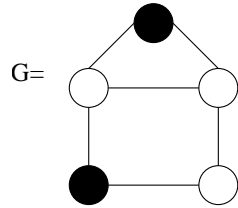
MAXIS: Gegeben Graph $G = (V, E)$,

finde unabhängige Menge $V' \subseteq V$ mit $|V'|$ möglichst groß.

Das Independent-Set-Problem: Ein Maximierungsproblem.

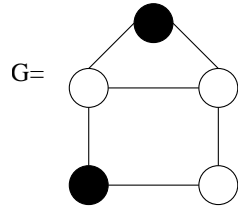
MAXIS: Gegeben Graph $G = (V, E)$,
finde unabhängige Menge $V' \subseteq V$ mit $|V'|$ möglichst groß.

Im Beispiel



Das Independent-Set-Problem: Ein Maximierungsproblem.

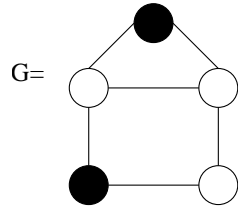
MAXIS: Gegeben Graph $G = (V, E)$,
finde unabhängige Menge $V' \subseteq V$ mit $|V'|$ möglichst groß.



Im Beispiel ist 2 größtmöglich.

Das Independent-Set-Problem: Ein Maximierungsproblem.

MAXIS: Gegeben Graph $G = (V, E)$,
finde unabhängige Menge $V' \subseteq V$ mit $|V'|$ möglichst groß.



Im Beispiel ist 2 größtmöglich.

Entscheidungsvariante IS:

Gegeben Graph $G = (V, E)$ und Zahl $1 \leq k \leq |V|$.

Frage: gibt es in G eine unabhängige Menge mit $\geq k$ Knoten?

Formalisierung:

Formalisierung:

IS

Formalisierung:

$$\text{IS} := \left\{ (G, k) \mid G = (V, E) \text{ Graph, } 1 \leq k \leq m; \right. \\ \text{es gibt } V' \subseteq V \text{ mit } |V'| \geq k \\ \left. \text{und } \forall v, w \in V' : v \neq w \Rightarrow (v, w) \notin E \right\} .$$

Formalisierung:

$$\text{IS} := \left\{ (G, k) \mid G = (V, E) \text{ Graph, } 1 \leq k \leq m; \right. \\ \text{es gibt } V' \subseteq V \text{ mit } |V'| \geq k \\ \left. \text{und } \forall v, w \in V' : v \neq w \Rightarrow (v, w) \notin E \right\} .$$

Satz 2.1.2

IS ist **NP**-vollständig.

Formalisierung:

$$\text{IS} := \left\{ (G, k) \mid G = (V, E) \text{ Graph, } 1 \leq k \leq m; \right. \\ \text{es gibt } V' \subseteq V \text{ mit } |V'| \geq k \\ \left. \text{und } \forall v, w \in V' : v \neq w \Rightarrow (v, w) \notin E \right\} .$$

Satz 2.1.2

IS ist **NP**-vollständig.

Beweis: (i) $\text{IS} \in \mathbf{NP}$: Wie üblich für Entscheidungsvarianten von **NP**-Optimierungsproblemen, wie bei **CLIQUE**.

Satz 2.1.2

IS ist **NP**-vollständig.

Satz 2.1.2

IS ist **NP**-vollständig.

Beweis: (ii) IS ist **NP**-schwer. Reduktionsmethode!

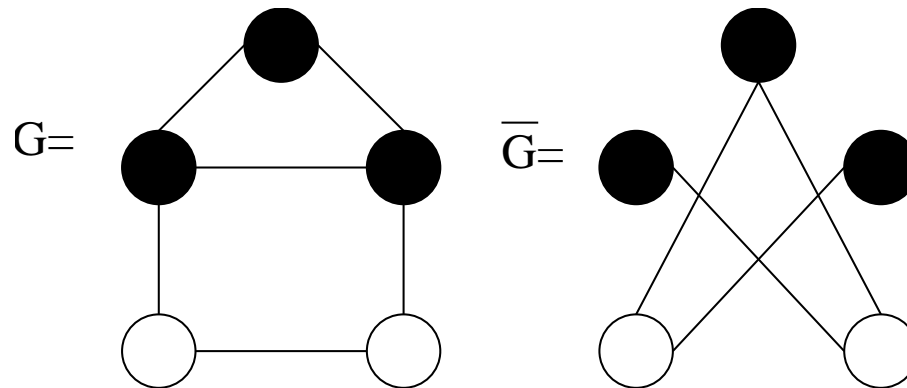
Wir zeigen: $\text{CLIQUE} \leq_p \text{IS}$.

Satz 2.1.2

IS ist **NP**-vollständig.

Beweis: (ii) IS ist **NP**-schwer. Reduktionsmethode!

Wir zeigen: $\text{CLIQUE} \leq_p \text{IS}$.

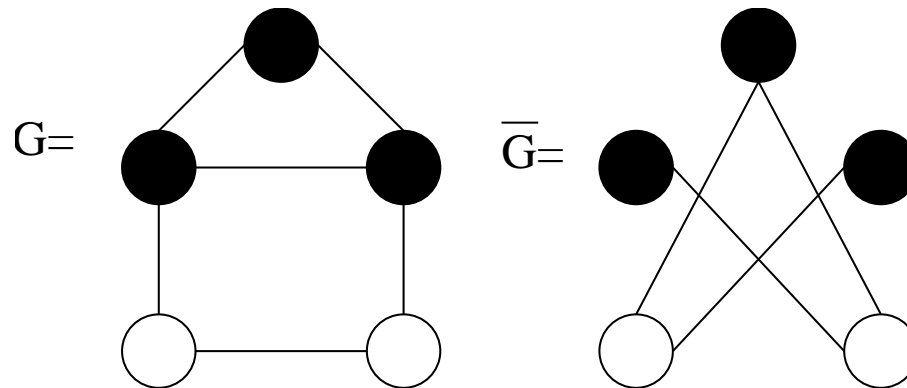


Satz 2.1.2

IS ist **NP**-vollständig.

Beweis: (ii) IS ist **NP**-schwer. Reduktionsmethode!

Wir zeigen: $\text{CLIQUE} \leq_p \text{IS}$.



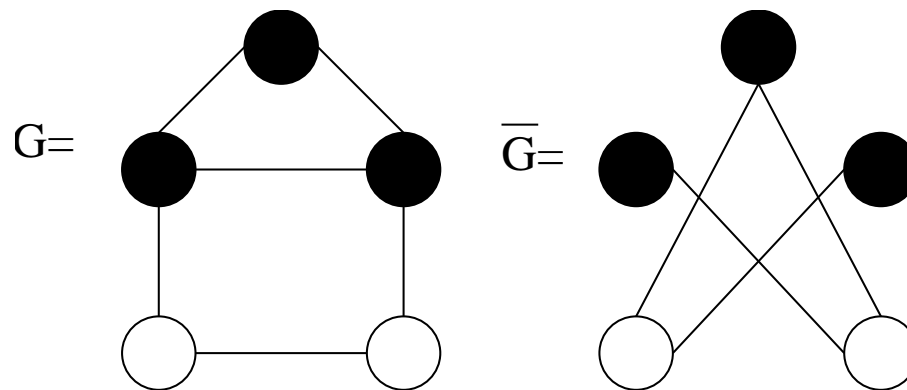
„Komplementgraph“ $\bar{G} = (V, \bar{E})$ zu $G = (V, E)$:

Satz 2.1.2

IS ist **NP**-vollständig.

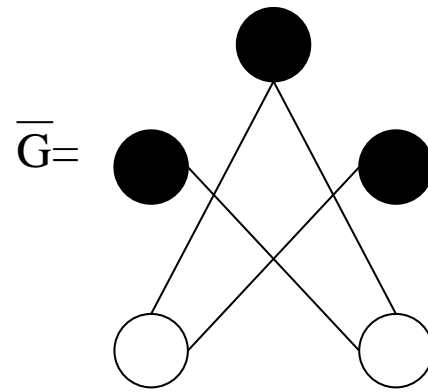
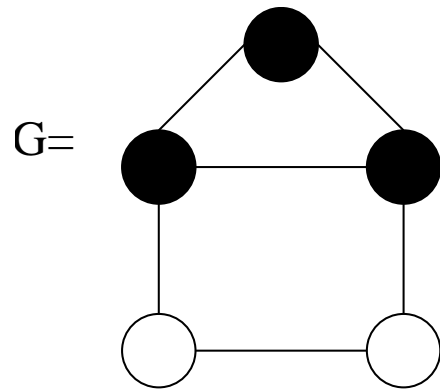
Beweis: (ii) IS ist **NP**-schwer. Reduktionsmethode!

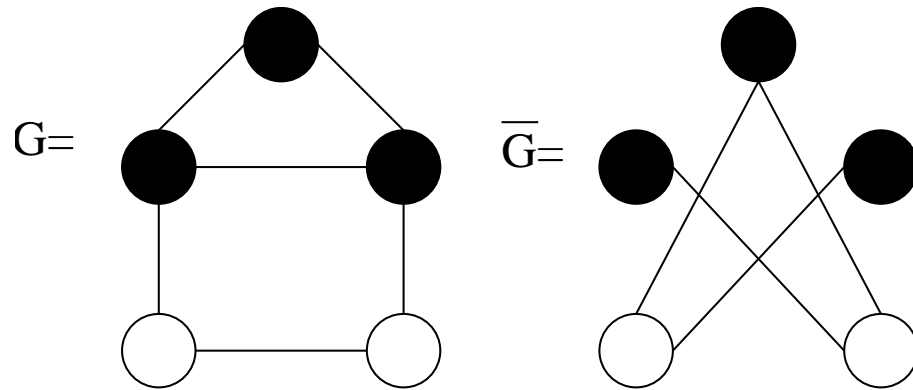
Wir zeigen: $\text{CLIQUE} \leq_p \text{IS}$.



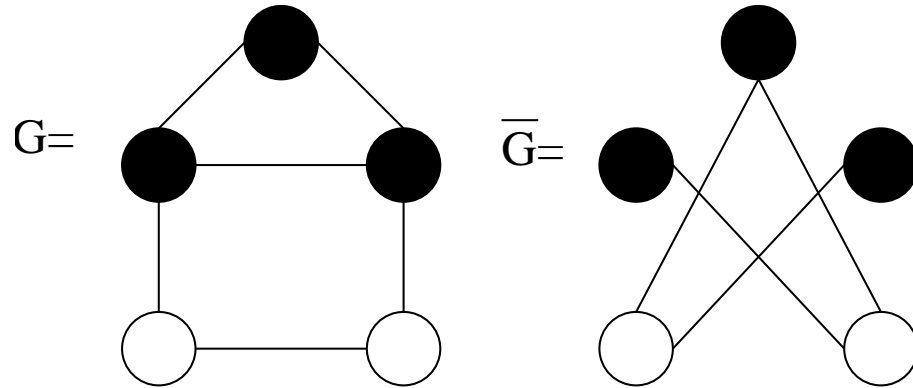
„Komplementgraph“ $\bar{G} = (V, \bar{E})$ zu $G = (V, E)$:

$$(v, w) \in \bar{E} \Leftrightarrow (v, w) \notin E$$



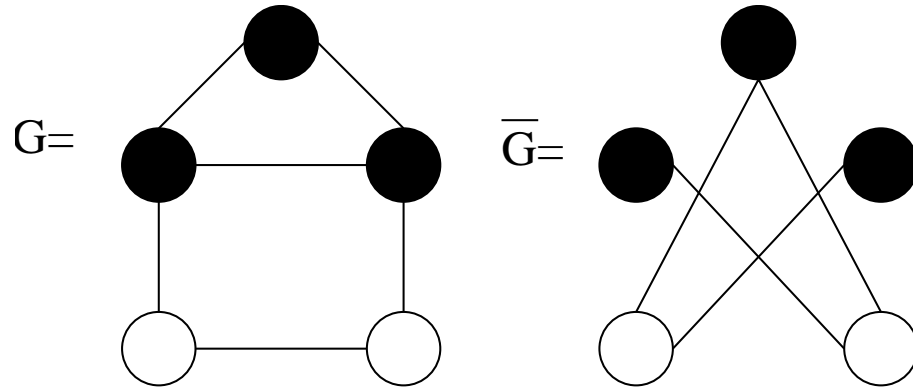


Klar: V' Clique in $G \Leftrightarrow V'$ unabhängig in \bar{G}



Klar: V' Clique in $G \Leftrightarrow V'$ unabhängig in \bar{G}

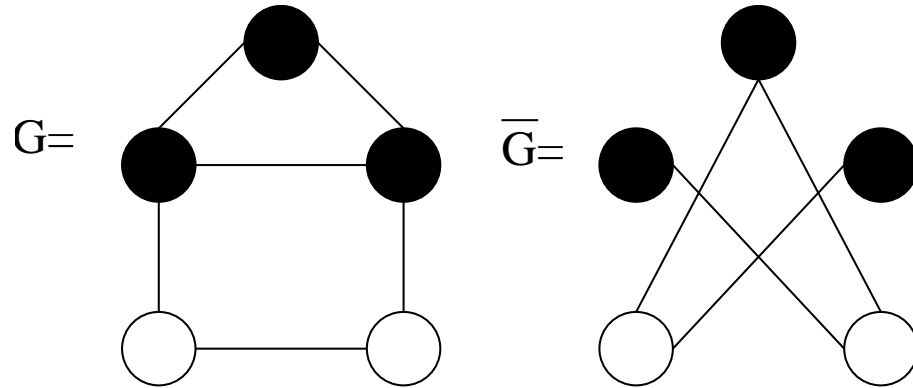
Reduktionsfunktion: $f: (G, k) \mapsto (\bar{G}, k)$



Klar: V' Clique in $G \Leftrightarrow V'$ unabhängig in \overline{G}

Reduktionsfunktion: $f: (G, k) \mapsto (\overline{G}, k)$

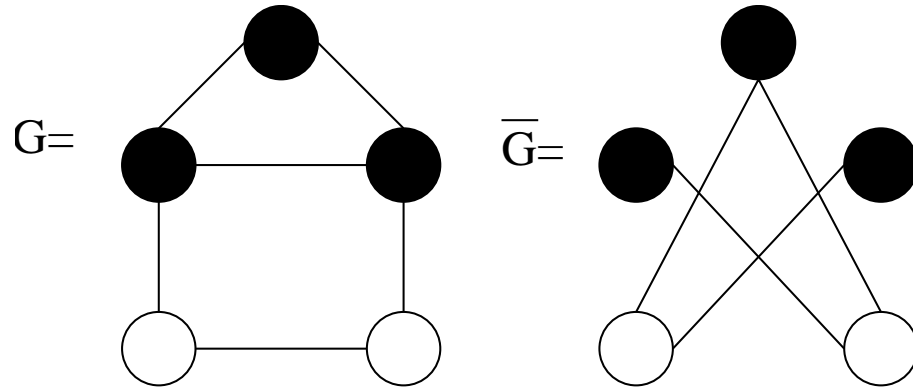
(Inputs w , die nicht die Form (G, k) haben,



Klar: V' Clique in $G \Leftrightarrow V'$ unabhängig in \bar{G}

Reduktionsfunktion: $f: (G, k) \mapsto (\bar{G}, k)$

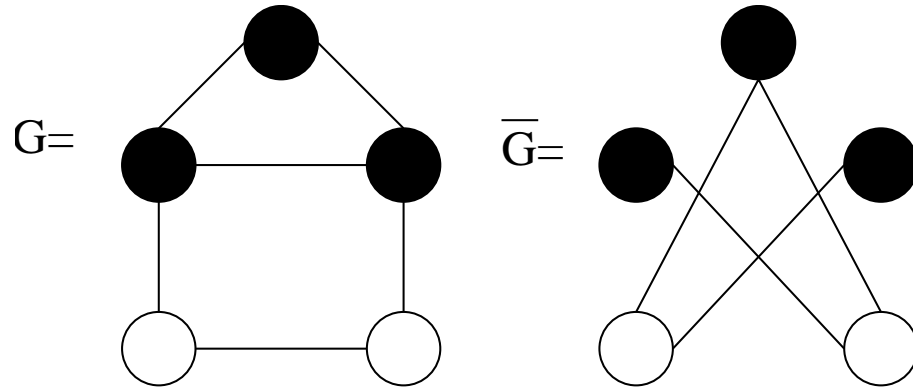
(Inputs w , die nicht die Form (G, k) haben, werden von f z.B. auf 0 abgebildet.)



Klar: V' Clique in $G \Leftrightarrow V'$ unabhängig in \overline{G}

Reduktionsfunktion: $f: (G, k) \mapsto (\overline{G}, k)$

(Inputs w , die nicht die Form (G, k) haben, werden von f z.B. auf 0 abgebildet.) – Leicht: $f \in \mathbf{FP}$.

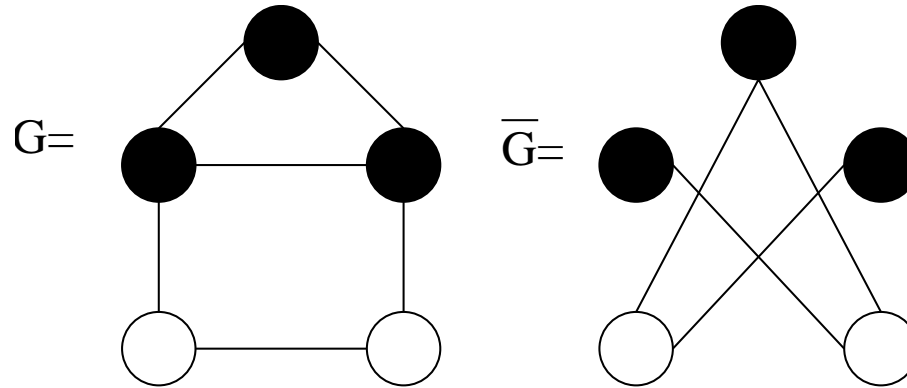


Klar: V' Clique in $G \Leftrightarrow V'$ unabhängig in \overline{G}

Reduktionsfunktion: $f: (G, k) \mapsto (\overline{G}, k)$

(Inputs w , die nicht die Form (G, k) haben, werden von f z.B. auf 0 abgebildet.) – Leicht: $f \in \mathbf{FP}$.

Und:

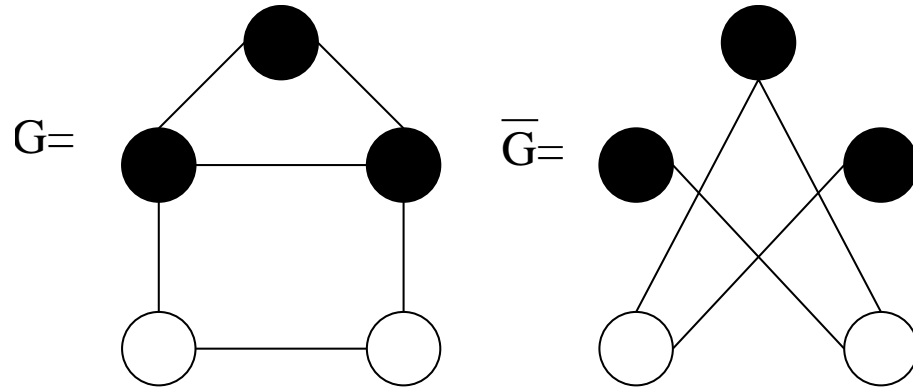


Klar: V' Clique in $G \Leftrightarrow V'$ unabhängig in \overline{G}

Reduktionsfunktion: $f: (G, k) \mapsto (\overline{G}, k)$

(Inputs w , die nicht die Form (G, k) haben, werden von f z.B. auf 0 abgebildet.) – Leicht: $f \in \mathbf{FP}$.

Und: Für **jeden Input** (G, k) gilt:



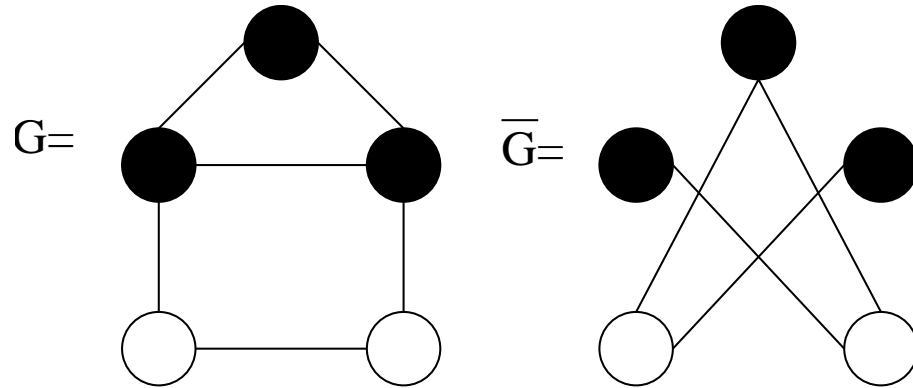
Klar: V' Clique in $G \Leftrightarrow V'$ unabhängig in \overline{G}

Reduktionsfunktion: $f: (G, k) \mapsto (\overline{G}, k)$

(Inputs w , die nicht die Form (G, k) haben, werden von f z.B. auf 0 abgebildet.) – Leicht: $f \in \mathbf{FP}$.

Und: Für **jeden Input** (G, k) gilt:

$$(G, k) \in \text{CLIQUE}$$



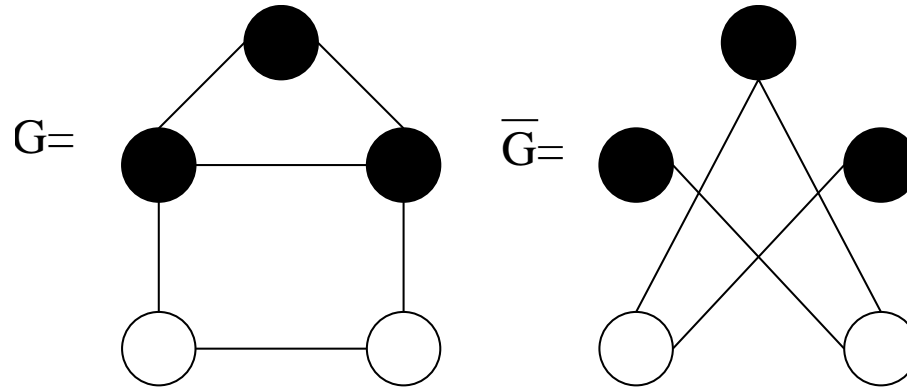
Klar: V' Clique in $G \Leftrightarrow V'$ unabhängig in \overline{G}

Reduktionsfunktion: $f: (G, k) \mapsto (\overline{G}, k)$

(Inputs w , die nicht die Form (G, k) haben, werden von f z.B. auf 0 abgebildet.) – Leicht: $f \in \mathbf{FP}$.

Und: Für **jeden Input** (G, k) gilt:

$$(G, k) \in \text{CLIQUE} \Leftrightarrow f((G, k)) = (\overline{G}, k) \in \text{IS}$$



Klar: V' Clique in $G \Leftrightarrow V'$ unabhängig in \overline{G}

Reduktionsfunktion: $f: (G, k) \mapsto (\overline{G}, k)$

(Inputs w , die nicht die Form (G, k) haben, werden von f z.B. auf 0 abgebildet.) – Leicht: $f \in \mathbf{FP}$.

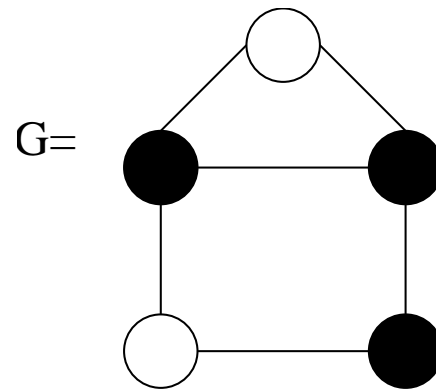
Und: Für **jeden Input** (G, k) gilt:

$$(G, k) \in \text{CLIQUE} \Leftrightarrow f((G, k)) = (\overline{G}, k) \in \text{IS}$$

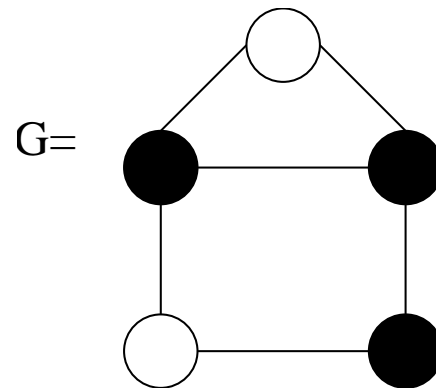
also $\text{CLIQUE} \leq_p \text{IS}$ via f .

Problem „**Vertex Cover**“: Ein Minimierungsproblem.

Problem „**Vertex Cover**“: Ein Minimierungsproblem.

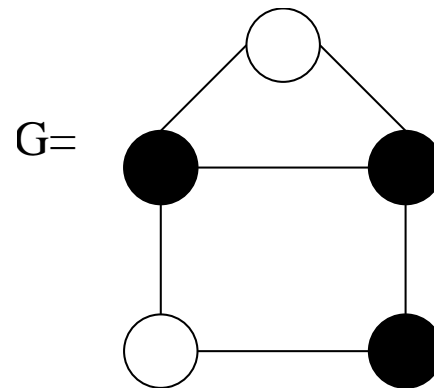


Problem „**Vertex Cover**“: Ein Minimierungsproblem.



Definition

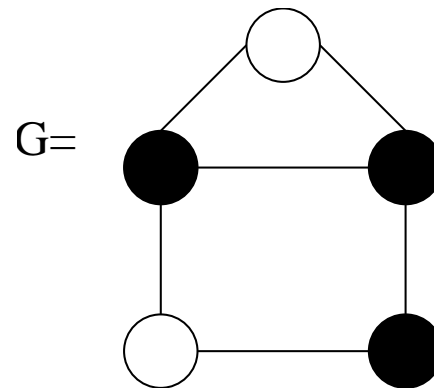
Problem „**Vertex Cover**“: Ein Minimierungsproblem.



Definition

$G = (V, E)$ sei ein Graph.

Problem „**Vertex Cover**“: Ein Minimierungsproblem.

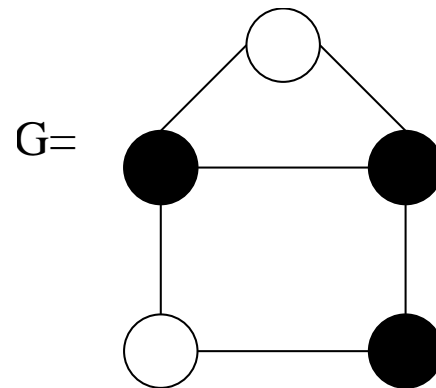


Definition

$G = (V, E)$ sei ein Graph.

Ein Knoten v **überdeckt** eine Kante (u, w) ,

Problem „**Vertex Cover**“: Ein Minimierungsproblem.

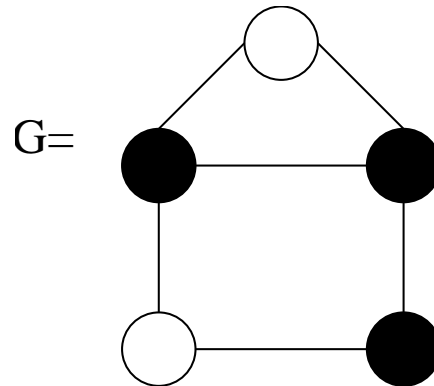


Definition

$G = (V, E)$ sei ein Graph.

Ein Knoten v **überdeckt** eine Kante (u, w) , falls $v \in \{u, w\}$.

Problem „**Vertex Cover**“: Ein Minimierungsproblem.



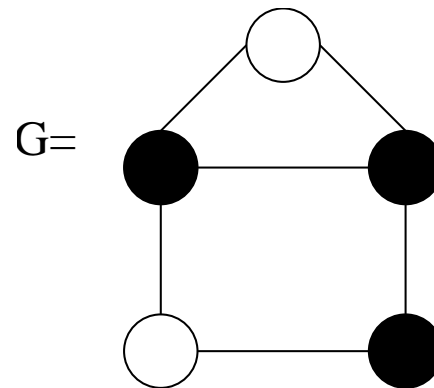
Definition

$G = (V, E)$ sei ein Graph.

Ein Knoten v **überdeckt** eine Kante (u, w) , falls $v \in \{u, w\}$.

$V' \subseteq V$ ist **Knotenüberdeckung** von G ,

Problem „**Vertex Cover**“: Ein Minimierungsproblem.



Definition

$G = (V, E)$ sei ein Graph.

Ein Knoten v **überdeckt** eine Kante (u, w) , falls $v \in \{u, w\}$.

$V' \subseteq V$ ist **Knotenüberdeckung** von G , wenn jede Kante in E von einem Knoten in V' überdeckt wird.

Knotenüberdeckungsproblem/Vertex-Cover-Problem MINVC :

Knotenüberdeckungsproblem/Vertex-Cover-Problem MINVC:

Zu Graph $G = (V, E)$

Knotenüberdeckungsproblem/Vertex-Cover-Problem MINVC:

Zu Graph $G = (V, E)$ finde Knotenüberdeckung $V' \subseteq V$

Knotenüberdeckungsproblem/Vertex-Cover-Problem MINVC:

Zu Graph $G = (V, E)$ finde Knotenüberdeckung $V' \subseteq V$
mit $|V'|$ möglichst klein.

Knotenüberdeckungsproblem/Vertex-Cover-Problem MINVC:

Zu Graph $G = (V, E)$ finde Knotenüberdeckung $V' \subseteq V$
mit $|V'|$ möglichst klein.

Im Beispiel ist 3 kleinstmöglich.

Knotenüberdeckungsproblem/Vertex-Cover-Problem MINVC:

Zu Graph $G = (V, E)$ finde Knotenüberdeckung $V' \subseteq V$
mit $|V'|$ möglichst klein.

Im Beispiel ist 3 kleinstmöglich.

Entscheidungsvariante VC:

Knotenüberdeckungsproblem/Vertex-Cover-Problem MINVC:

Zu Graph $G = (V, E)$ finde Knotenüberdeckung $V' \subseteq V$
mit $|V'|$ möglichst klein.

Im Beispiel ist 3 kleinstmöglich.

Entscheidungsvariante VC: Gegeben Graph $G = (V, E)$ mit
 $V = \{1, \dots, m\}$

Knotenüberdeckungsproblem/Vertex-Cover-Problem MINVC:

Zu Graph $G = (V, E)$ finde Knotenüberdeckung $V' \subseteq V$
mit $|V'|$ möglichst klein.

Im Beispiel ist 3 kleinstmöglich.

Entscheidungsvariante VC: Gegeben Graph $G = (V, E)$ mit
 $V = \{1, \dots, m\}$ **und Zahl $k \leq m$.**

Frage: Gibt es eine Knotenüberdeckung für G mit $\leq k$
Knoten?

Formalisierung als Sprache:

Formalisierung als Sprache:

VC

Formalisierung als Sprache:

$$\text{VC} := \left\{ (G, k) \mid G = (V, E) \text{ Graph, } 1 \leq k \leq |V|; \right. \\ \text{es gibt } V' \subseteq V \text{ mit } |V'| \leq k \\ \left. \text{und } \forall (v, w) \in E : v \in V' \vee w \in V' \right\} .$$

Formalisierung als Sprache:

$$\text{VC} := \left\{ (G, k) \mid G = (V, E) \text{ Graph, } 1 \leq k \leq |V|; \right. \\ \left. \begin{array}{l} \text{es gibt } V' \subseteq V \text{ mit } |V'| \leq k \\ \text{und } \forall (v, w) \in E : v \in V' \vee w \in V' \end{array} \right\} .$$

Satz 2.1.3

VC ist **NP**-vollständig.

Formalisierung als Sprache:

$$\text{VC} := \left\{ (G, k) \mid G = (V, E) \text{ Graph, } 1 \leq k \leq |V|; \right. \\ \left. \begin{array}{l} \text{es gibt } V' \subseteq V \text{ mit } |V'| \leq k \\ \text{und } \forall (v, w) \in E : v \in V' \vee w \in V' \end{array} \right\} .$$

Satz 2.1.3

VC ist **NP**-vollständig.

Beweis: (i) $\text{VC} \in \mathbf{NP}$: Wie üblich für Entscheidungsvarianten von **NP**-Optimierungsproblemen, wie bei **CLIQUE**.

Satz 2.1.3

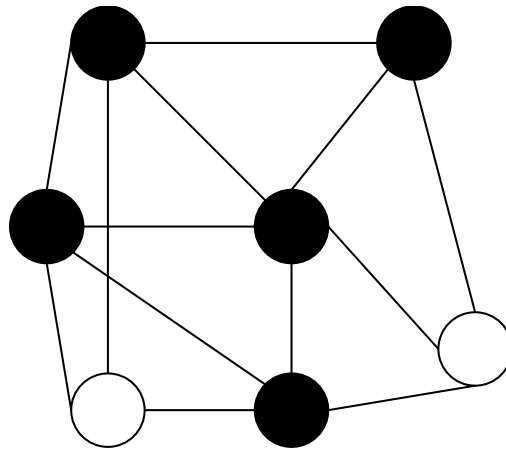
VC ist **NP**-vollständig.

Beweis: (ii) VC ist **NP**-schwer: Wir zeigen $IS \leq_p VC$.

Satz 2.1.3

VC ist **NP**-vollständig.

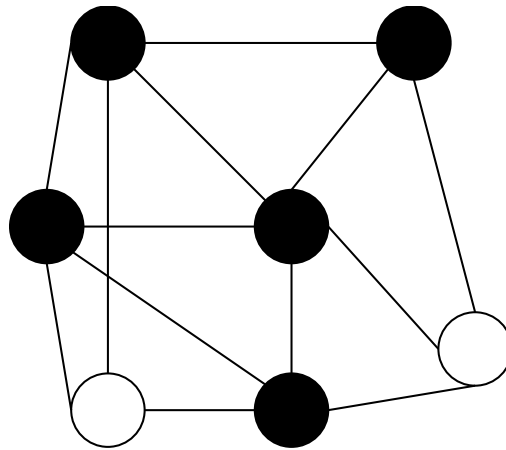
Beweis: (ii) VC ist **NP**-schwer: Wir zeigen $IS \leq_p VC$.



Satz 2.1.3

VC ist **NP**-vollständig.

Beweis: (ii) VC ist **NP**-schwer: Wir zeigen $IS \leq_p VC$.

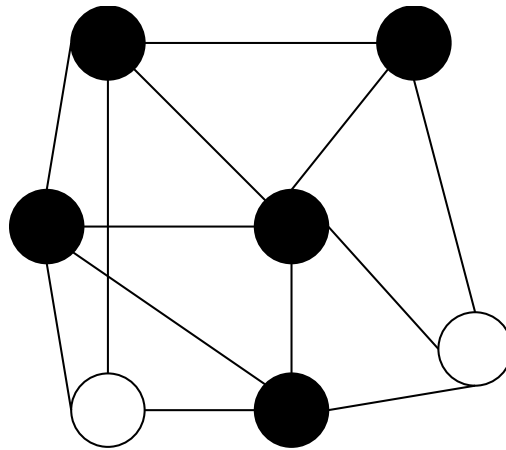


Schwarz: Knotenüberdeckung

Satz 2.1.3

VC ist **NP**-vollständig.

Beweis: (ii) VC ist **NP**-schwer: Wir zeigen $IS \leq_p VC$.



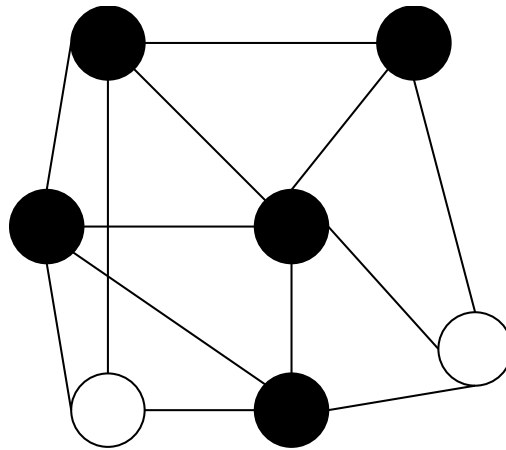
Schwarz: Knotenüberdeckung

Weiß: Unabhängige Menge

Satz 2.1.3

VC ist **NP**-vollständig.

Beweis: (ii) VC ist **NP**-schwer: Wir zeigen $IS \leq_p VC$.



Schwarz: Knotenüberdeckung

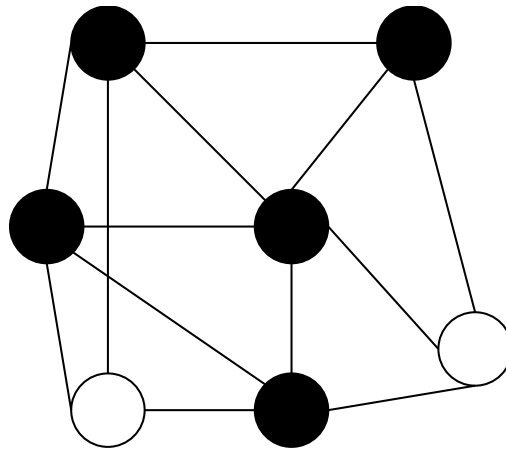
Weiß: Unabhängige Menge

Hilfsbehauptung (leicht zu sehen):

Satz 2.1.3

VC ist **NP**-vollständig.

Beweis: (ii) VC ist **NP**-schwer: Wir zeigen $IS \leq_p VC$.



Schwarz: Knotenüberdeckung

Weiß: Unabhängige Menge

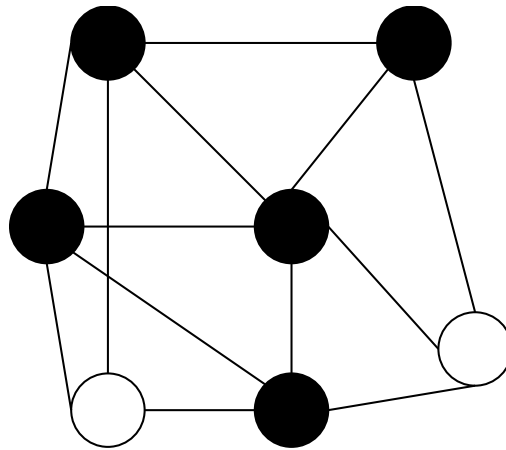
Hilfsbehauptung (leicht zu sehen):

V' unabhängig in G

Satz 2.1.3

VC ist **NP**-vollständig.

Beweis: (ii) VC ist **NP**-schwer: Wir zeigen $IS \leq_p VC$.



Schwarz: Knotenüberdeckung

Weiß: Unabhängige Menge

Hilfsbehauptung (leicht zu sehen):

V' unabhängig in $G \iff V - V'$ Knotenüberdeckung in G

Also:

Also:

$G = (V, E)$ hat unabhängige Menge der Größe $\geq k \iff$

Also:

$G = (V, E)$ hat unabhängige Menge der Größe $\geq k \iff$

$G = (V, E)$ hat Knotenüberdeckung der Größe $\leq |V| - k$

Also:

$G = (V, E)$ hat unabhängige Menge der Größe $\geq k \iff$

$G = (V, E)$ hat Knotenüberdeckung der Größe $\leq |V| - k$

Reduktionsfunktion $f: ((V, E), k) \mapsto ((V, E), |V| - k)$

Also:

$G = (V, E)$ hat unabhängige Menge der Größe $\geq k \iff$

$G = (V, E)$ hat Knotenüberdeckung der Größe $\leq |V| - k$

Reduktionsfunktion $f: ((V, E), k) \mapsto ((V, E), |V| - k)$

(klar: in **FP** !)

Also:

$G = (V, E)$ hat unabhängige Menge der Größe $\geq k \iff$

$G = (V, E)$ hat Knotenüberdeckung der Größe $\leq |V| - k$

Reduktionsfunktion $f: ((V, E), k) \mapsto ((V, E), |V| - k)$

(klar: in **FP** !) erfüllt

Also:

$G = (V, E)$ hat unabhängige Menge der Größe $\geq k \iff$

$G = (V, E)$ hat Knotenüberdeckung der Größe $\leq |V| - k$

Reduktionsfunktion $f: ((V, E), k) \mapsto ((V, E), |V| - k)$

(klar: in **FP** !) erfüllt

$$((V, E), k) \in \text{IS} \iff$$

Also:

$G = (V, E)$ hat unabhängige Menge der Größe $\geq k \iff$

$G = (V, E)$ hat Knotenüberdeckung der Größe $\leq |V| - k$

Reduktionsfunktion $f: ((V, E), k) \mapsto ((V, E), |V| - k)$

(klar: in **FP** !) erfüllt

$$((V, E), k) \in \text{IS} \iff f(((V, E), k)) \in \text{VC}.$$

Also:

$G = (V, E)$ hat unabhängige Menge der Größe $\geq k \iff$

$G = (V, E)$ hat Knotenüberdeckung der Größe $\leq |V| - k$

Reduktionsfunktion $f: ((V, E), k) \mapsto ((V, E), |V| - k)$

(klar: in **FP** !) erfüllt

$$((V, E), k) \in \text{IS} \iff f(((V, E), k)) \in \text{VC}.$$

Also: $\text{IS} \leq_p \text{VC}$ via f .

Problem „**Hitting Set**“ MINHITTINGSET: Ein Minimierungsproblem.

Problem „**Hitting Set**“ MINHITTINGSET: Ein Minimierungsproblem.
(„*to hit*“: treffen).

Eingabe: Endliche Menge A , Teilmengen C_1, \dots, C_m von A .

Aufgabe: Finde eine möglichst kleine Menge $B \subseteq A$ mit:
 $B \cap C_j \neq \emptyset$ für $1 \leq j \leq m$.
(B „trifft“ alle Mengen C_j in mindestens einem Punkt.)

Problem „**Hitting Set**“ MINHITTINGSET: Ein Minimierungsproblem.
(„*to hit*“: treffen).

Eingabe: Endliche Menge A , Teilmengen C_1, \dots, C_m von A .

Aufgabe: Finde eine möglichst kleine Menge $B \subseteq A$ mit:

$B \cap C_j \neq \emptyset$ für $1 \leq j \leq m$.

(B „trifft“ alle Mengen C_j in mindestens einem Punkt.)

Entscheidungsvariante: HITTINGSET.

Eingabe: A, C_1, \dots, C_m wie oben und $k \leq m$.

Frage: Gibt es eine Teilmenge $B \subseteq A$ mit (höchstens) k Elementen, die jede Menge C_j trifft?

Übung: Beschreibe diese Probleme formal als Optimierungsprobleme und als Suchvariante/Entscheidungsvariante.

Satz 2.1.5 HITTINGSET ist **NP**-vollständig.

Satz 2.1.5 HITTINGSET ist **NP**-vollständig.

Beweis: (i) HITTINGSET \in **NP**: Wie üblich für Entscheidungsvarianten von **NP**-Optimierungsproblemen, wie bei CLIQUE.

Satz 2.1.5 HITTINGSET ist **NP**-vollständig.

Beweis: (i) HITTINGSET \in **NP**: Wie üblich für Entscheidungsvarianten von **NP**-Optimierungsproblemen, wie bei CLIQUE.

(ii) HITTINGSET ist **NP**-schwer:

Wir zeigen $VC \leq_p$ HITTINGSET.

Satz 2.1.5 HITTINGSET ist **NP**-vollständig.

Beweis: (i) HITTINGSET \in **NP**: Wie üblich für Entscheidungsvarianten von **NP**-Optimierungsproblemen, wie bei CLIQUE.

(ii) HITTINGSET ist **NP**-schwer:

Wir zeigen $VC \leq_p$ HITTINGSET.

Reduktionsfunktion: $f: (V, E, k) \mapsto (A, C_1, \dots, C_m, k)$,

wobei $A = \{1, \dots, n\} = V$;

Satz 2.1.5 HITTINGSET ist **NP**-vollständig.

Beweis: (i) HITTINGSET \in **NP**: Wie üblich für Entscheidungsvarianten von **NP**-Optimierungsproblemen, wie bei CLIQUE.

(ii) HITTINGSET ist **NP**-schwer:

Wir zeigen $VC \leq_p$ HITTINGSET.

Reduktionsfunktion: $f: (V, E, k) \mapsto (A, C_1, \dots, C_m, k)$,

wobei $A = \{1, \dots, n\} = V$;

C_1, \dots, C_m sind Paarmengen, die genau den m Kanten in E entsprechen; Schranke k wird übernommen.

Satz 2.1.5 HITTINGSET ist **NP**-vollständig.

Beweis: (i) HITTINGSET \in **NP**: Wie üblich für Entscheidungsvarianten von **NP**-Optimierungsproblemen, wie bei CLIQUE.

(ii) HITTINGSET ist **NP**-schwer:

Wir zeigen $VC \leq_p$ HITTINGSET.

Reduktionsfunktion: $f: (V, E, k) \mapsto (A, C_1, \dots, C_m, k)$,

wobei $A = \{1, \dots, n\} = V$;

C_1, \dots, C_m sind Paarmengen, die genau den m Kanten in E entsprechen; Schranke k wird übernommen.

Diese Funktion ist in polynomieller Zeit berechenbar und sie hat die Reduktionseigenschaft, weil eine Knotenüberdeckung in (V, E) genau dasselbe ist wie ein „hitting set“ in (A, C_1, \dots, C_m) . \square

Ein Input für MINHITTINGSET lässt sich in normalisierter Form als 0-1-Matrix M darstellen.

M hat n Spalten und m Zeilen.

Die Spalten entsprechen den Elementen von $A = \{1, \dots, n\}$; die Zeilen stellen die charakteristischen Funktionen von C_1, \dots, C_m dar:

$a_{ji} = 1$ falls $i \in C_j$, und $a_{ji} = 0$ falls $i \notin C_j$.

Gesucht ist eine Auswahl von möglichst wenigen Spalten, so dass jede der m Zeilen in einer der ausgewählten Spalten eine 1 hat.

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Zeilen 5, 6, 7, 8 sind eine legale Lösung:
in jeder Spalte eine 1.

Problem „**Set Cover**“

MINSETCOVER: Ein Minimierungsproblem.

Problem „Set Cover“

MINSETCOVER: Ein Minimierungsproblem.

Eingabe: Endliche Menge A , Teilmengen C_1, \dots, C_m von A .

Aufgabe: Finde eine möglichst kleine Menge $I \subseteq \{1, \dots, m\}$ (eine Auswahl der C_1, \dots, C_m) mit:

$$A = \bigcup_{i \in I} C_i.$$

($C_i, i \in I$, „überdeckt“ ganz A .)

Problem „Set Cover“

MINSETCOVER: Ein Minimierungsproblem.

Eingabe: Endliche Menge A , Teilmengen C_1, \dots, C_m von A .

Aufgabe: Finde eine möglichst kleine Menge $I \subseteq \{1 \dots, m\}$ (eine Auswahl der C_1, \dots, C_m) mit:

$$A = \bigcup_{i \in I} C_i.$$

(C_i , $i \in I$, „überdeckt“ ganz A .)

Sinnvoll ist diese Aufgabe nur, wenn $A = C_1 \cup \dots \cup C_m$ ist, aber das verlangt man nicht unbedingt.

Entscheidungsvariante: SETCOVER.

Eingabe: A, C_1, \dots, C_m wie oben und $k \leq m$.

Frage: Gibt es eine Menge $I \subseteq \{1, \dots, m\}$ mit $|I| \leq k$, derart dass $A = \bigcup_{i \in I} C_i$?

Übung: Beschreibe diese Probleme formal als Optimierungsprobleme und als Suchvariante/Entscheidungsvariante.

Satz 2.1.6 SETCOVER ist **NP**-vollständig.

Satz 2.1.6 SETCOVER ist **NP**-vollständig.

Beweis: (i) SETCOVER \in **NP**: Wie üblich für Entscheidungsvarianten von **NP**-Optimierungsproblemen, wie bei CLIQUE.

(ii) SETCOVER ist **NP**-schwer:
Wir zeigen $VC \leq_p SETCOVER$.

(ii) SETCOVER ist **NP**-schwer:

Wir zeigen $VC \leq_p \text{SETCOVER}$.

Reduktionsfunktion: $f: (V, E, k) \mapsto (A, C_1, \dots, C_n, k)$.

Dabei: $V = \{1, \dots, n\}$.

(ii) SETCOVER ist **NP**-schwer:

Wir zeigen $VC \leq_p \text{SETCOVER}$.

Reduktionsfunktion: $f: (V, E, k) \mapsto (A, C_1, \dots, C_n, k)$.

Dabei: $V = \{1, \dots, n\}$.

$A = E$; **(!!!)**

(ii) SETCOVER ist **NP**-schwer:

Wir zeigen $VC \leq_p \text{SETCOVER}$.

Reduktionsfunktion: $f: (V, E, k) \mapsto (A, C_1, \dots, C_n, k)$.

Dabei: $V = \{1, \dots, n\}$.

$A = E$; **(!!!)**

$C_i = \{e \in E \mid \text{Knoten } i \text{ liegt auf } e\}$, für $1 \leq i \leq n$.

Diese Funktion ist in polynomieller Zeit berechenbar und sie hat die Reduktionseigenschaft, weil eine Knotenüberdeckung in (V, E) genau dasselbe ist wie eine Teilmenge $I \subseteq \{1, \dots, n\}$ mit $E = \bigcup_{i \in I} C_i$.

(ii) SETCOVER ist **NP**-schwer:

Wir zeigen $VC \leq_p \text{SETCOVER}$.

Reduktionsfunktion: $f: (V, E, k) \mapsto (A, C_1, \dots, C_n, k)$.

Dabei: $V = \{1, \dots, n\}$.

$A = E$; **(!!!)**

$C_i = \{e \in E \mid \text{Knoten } i \text{ liegt auf } e\}$, für $1 \leq i \leq n$.

Diese Funktion ist in polynomieller Zeit berechenbar und sie hat die Reduktionseigenschaft, weil eine Knotenüberdeckung in (V, E) genau dasselbe ist wie eine Teilmenge $I \subseteq \{1, \dots, n\}$ mit $E = \bigcup_{i \in I} C_i$.

(Ein Knoten $i \in V$ überdeckt genau die Kanten in C_i !) \square

Ein Input für MINSETCOVER lässt sich in normalisierter Form als 0-1-Matrix M darstellen.

M hat n Spalten und m Zeilen.

Die Spalten entsprechen den Elementen von $A = \{1, \dots, n\}$; die Zeilen stellen die charakteristischen Funktionen von C_1, \dots, C_m dar:

$a_{ji} = 1$ falls $i \in C_j$, und $a_{ji} = 0$ falls $i \notin C_j$.

Gesucht ist eine Auswahl von möglichst wenigen **Zeilen**, so dass jede der n **Spalten** in einer der ausgewählten Zeilen eine 1 hat.

Das haben wir schon gesehen!

Das haben wir schon gesehen! – Fast!

Das haben wir schon gesehen! – Fast!

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Das haben wir schon gesehen! – Fast!

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Zeilen 2, 3, 5, 8 sind eine legale Lösung: in jeder Spalte eine 1.

Das haben wir schon gesehen! – Fast!

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Zeilen 2, 3, 5, 8 sind eine legale Lösung: in jeder Spalte eine 1.

MINHITTINGSET und MINSETCOVER sind eigentlich dasselbe Problem – man übersetzt das eine in das andere Problem durch Transponieren einer 0-1-Matrix.

Haben:

NP-vollständige Sprachen

SAT, 3-SAT, CLIQUE, IS, VC, HITTINGSET und
SETCOVER.

Haben:

NP-vollständige Sprachen

SAT, 3-SAT, CLIQUE, IS, VC, HITTINGSET und
SETCOVER.

Nur der Anfang von Tausenden von NP-vollständigen
Problemen.

Haben:

NP-vollständige Sprachen

SAT, 3-SAT, CLIQUE, IS, VC, HITTINGSET und
SETCOVER.

Nur der Anfang von Tausenden von NP-vollständigen
Problemen.

Zugehörige **NP**-Suchprobleme: SAT, 3-SAT

Zugehörige **NP**-Optimierungsprobleme: CLIQUE, IS, VC,
MINHITTINGSET und MINSETCOVER

haben vermutlich keinen Polynomialzeitalgorithmus.