

Komplexitätstheorie

WiSe 2011/12 in Trier

Henning Fernau
Universität Trier
fernau@uni-trier.de

Komplexitätstheorie Gesamtübersicht

- Organisatorisches / Einführung
Motivation / Erinnerung / Fragestellungen
- Diskussion verschiedener Komplexitätsklassen:
Zeitkomplexität
Platzkomplexität
- zugehörige Reduktionsbegriffe
- vollständige Probleme
- Anpassung von Klassenbegriffen und Reduktionen

Umfangreiche Liste NP-vollständiger Probleme

- Garey, M.R., Johnson, D.S., Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco 1979 sowie als Fortsetzung davon:
- Johnson, D.S., The NP-completeness column: an ongoing guide, seit 1981 in der Zeitschrift “Journal of Algorithms”, später in “ACM Transactions on Algorithms”

Noch einmal als Merksatz:

Solange es nicht gelungen ist, $P = NP$ zu beweisen, ist für keines der NP-vollständigen Probleme ein praktisch verwendbarer Algorithmus bekannt!

Im Folgenden **NP**-vollständige Probleme aus verschiedenen Problembereichen:

Logik, Graphentheorie, Mengen- und Zahlentheorie

Zunächst: Grundbegriffe aus der Aussagenlogik:

$$\Sigma_L := \{X, 0, 1, (,), \neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$$

Menge der Variablen:

$$\text{VAR} = \{X1w \mid w \in W(\{0, 1\})\}$$

z.B. binär interpretiert als $\text{VAR} = \{x_1, x_2, \dots\}$

Literale:

$$\text{LIT} := \text{VAR} \cup \{\neg v \mid v \in \text{VAR}\}$$

dabei $\neg v$ auch als \bar{v}

Die Menge WFF der *wohlgeformten Formeln* ist definiert als die kleinste Menge ($\subset W(\Sigma_L)$) mit folgenden Eigenschaften:

1. $VAR \subset WFF$
2. $F \in WFF \implies \neg F \in WFF$
3. $F_1, \dots, F_n \in WFF \implies (F_1 \wedge \dots \wedge F_n) \in WFF$
4. $F_1, \dots, F_n \in WFF \implies (F_1 \vee \dots \vee F_n) \in WFF$
5. $F_1, F_2 \in WFF \implies (F_1 \rightarrow F_2) \in WFF$
6. $F_1, F_2 \in WFF \implies (F_1 \leftrightarrow F_2) \in WFF$

Eine *Wahrheitswertezuweisung* (*Interpretation, Belegung*) ist eine Funktion

$$\phi: \text{VAR} \rightarrow \{0, 1\}$$

⇒ jeder Variablen wird Wert 0 ('falsch') oder 1 ('wahr') zugeordnet.

Zu gegebenem ϕ sei $\text{WERT}_\phi: \text{WFF} \rightarrow \{0, 1\}$ definiert durch

$$\text{WERT}_\phi(F) := \begin{cases} 1 - \text{WERT}_\phi(F') & \text{falls } F = \neg F' \\ \min\{\text{WERT}_\phi(F_1), \dots, \text{WERT}_\phi(F_n)\} & \text{falls } F = (F_1 \wedge \dots \wedge F_n) \\ \max\{\text{WERT}_\phi(F_1), \dots, \text{WERT}_\phi(F_n)\} & \text{falls } F = (F_1 \vee \dots \vee F_n) \\ \max\{1 - \text{WERT}_\phi(F_1), \text{WERT}_\phi(F_2)\} & \text{falls } F = (F_1 \rightarrow F_2) \\ \min\{\text{WERT}_\phi(F_1 \rightarrow F_2), \\ \quad \text{WERT}_\phi(F_2 \rightarrow F_1)\} & \text{falls } F = (F_1 \leftrightarrow F_2) \end{cases}$$

Eine Formel $F \in \text{WFF}$ heißt *erfüllbar*,
wenn es eine Interpretation ϕ mit $\text{WERT}_\phi(F) = 1$ gibt.

Zur Komplexität von WF_F

Lemma 1 $WF_F \in DSPACE(\log n)$

Aus dem Grundstudium dürfte bekannt sein, wie man (rekursiv) Ausdrücke auf ihre Korrektheit prüft; dabei erfolgt der rekursive Abstieg “entlang” der rekursiven Definition der Syntax solcher Ausdrücke.

Problem: Der Rekursionskeller kann zu groß werden, d.h., es wird eher linearer als logarithmischer Platz benötigt.

Beweis: Logspace genügt... **Grundidee:**

$\#K(w)$: # öffnender Klammern in w minus # schließender Klammern in w .

Offenbar gilt: (1) $\forall w \in WFF : \#K(w) = 0$.

Ferner ist wahr: (2) $\forall w \in WFF \forall v : v \text{ ist Präfix von } w \implies \#K(v) \geq 0$.

Betrachte konkretes $w = w_1 \dots w_n \in \Sigma^n$. Teste für alle $1 \leq k \leq \max\{\#K(w_1 \dots w_j) \mid j \leq n\}$, ob für alle $1 \leq i < n$ mit $\#K(w_1 \dots w_{i-1}) = k - 1$ und $\#K(w_1 \dots w_i) = k$ gilt:

Es gibt $i < j \leq n$ mit $\#K(w_1 \dots w_j) = k - 1$ und für $i \leq \ell < j$ gilt: $\#K(w_1 \dots w_\ell) \geq k$.

Für diese Tests sind nur wenige Zähler nötig.

Für die Korrektheit bliebe noch per Induktion zu zeigen, dass jedes Wort w , welches die Prozedur "übersteht", in WFF liegt, sofern auf der gleichen Klammerebene dieselben Operatoren verwendet werden; um das Letztere zu testen, benötigt man konstant viele Pointer in die Eingabe.

Beispiel: Es werden zwei Bereiche für $k = 2$ und einer für $k = 1$ geprüft.

$$(\ x \ 1 \ \wedge \ \neg \ (\ x \ 1 \ 0 \ \vee \ x \ 1 \ 1 \) \ \wedge \ (\ x \ 1 \ \wedge \ x \ 1 \ 1 \) \)$$

$$\#K \ 0 \ 1 \ 1 \ 1 \ 1 \ 1 \ 2 \ 2 \ 2 \ 2 \ 2 \ 2 \ 2 \ 2 \ 1 \ 1 \ 2 \ 2 \ 2 \ 2 \ 2 \ 2 \ 2 \ 1 \ 0$$

Vereinbarungen und Sprechweisen

(disjunktive) *Klauseln*: WFF der Form:

$$\{(l_1 \vee \dots \vee l_n) \mid n > 0, l_i \in \text{LIT}\}$$

CNF-Ausdruck (conjunctive normal form): WFF der Form

$$(c_1 \wedge \dots \wedge c_k),$$

wobei die c_i Klauseln sind.

Beachte:

(1) Kurzschreibweise $\phi(F) = \text{WERT}_\phi(F)$, da WFF den Definitionsbereich von ϕ in natürlicher Weise erweitern.

(2) CNF-Ausdruck A ist wahr (d.h. erhält den Wert 1) bei Interpretation ϕ , wenn jede Klausel von A unter ϕ wahr wird (formal: $\phi(A) = 1$).

Beispiel für CNF-Ausdruck:

$$A_0 = ((x_1 \vee \neg x_2) \wedge (x_1 \vee x_3))$$

oder in anderer Schreibweise

$$((x_1 \vee \bar{x}_2) \wedge (x_1 \vee x_3))$$

Falls $\phi(x_1) = 1$ oder $\phi(x_2) = 0, \phi(x_3) = 1$, so ist $\phi(A_0) = 1$

A_0 ist also insbesondere auch erfüllbar.

CNF-ERFÜLLBARKEIT: Wichtiges NP-vollständiges Problem auf CNF-Ausdrücken.

CNF-ERFÜLLBARKEIT: Menge aller erfüllbaren CNF-Ausdrücke

Lemma 2 $AGAP_{\forall P} \leq_{\log} \text{CNF-ERFÜLLBARKEIT}$

Konstruktion: Sei ein AND/OR-Graph (G, L) , $G = (V, E)$, $V = \{1, \dots, n\}$ und eine Menge $P \subseteq V \times V$ verbotener Paare gegeben. Wir geben CNF-Ausdruck w dazu an mit: $(G, L, P) \in AGAP_{\forall P} \iff w$ ist erfüllbar.

Sonderfall: In G gibt es keinen Knoten ohne Nachfolger.

Setze $w = ((x_1) \wedge (\neg x_1))$. w ist trivial unerfüllbar.

Im Folgenden daher **Annahme: Es gibt einen Knoten ohne Nachfolger.**

Führe **Variablen** $x[v, i]$ ein zu $v \in V$ und $1 \leq i \leq n$ mit der Bedeutung:

Für (insgesamt) erfüllende Belegung ϕ gilt: $\phi(x[v, i]) = 1$ gdw.

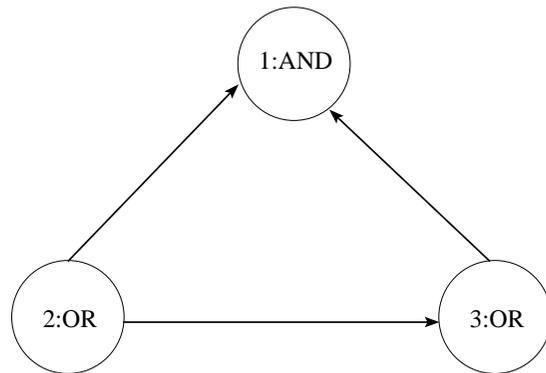
im vorgelegten Pebble-Spiel ist v nach i Schritten (bereits) markiert.

Klauseln für die Reduktion

1. $(\overline{x[v, 0]})$ für alle $v \in V$: Nach null Schritten ist kein Knoten markiert.
2. $(\overline{x[v, i]} \vee \overline{x[w, j]})$ für $(v, w) \in P$ und $1 \leq i, j \leq n$
3. $(\vee(x[v, n] \mid v \text{ hat keinen Nachfolger}))$
4. $(\overline{x[w, i+1]} \vee \vee(x[v, i] \mid (v, w) \in E))$, $1 \leq i < n$, für $w \in V$ mit $L(w) = \text{OR}$ und w hat einen Vorgänger in G .
5. $(\overline{x[w, i+1]} \vee x[v, i])$, $1 \leq i < n$, für $w \in V$ mit $L(w) = \text{AND}$ und $v \in V$ mit $(v, w) \in E$.

Die Konstruktion geht in Logspace, da Zähler zur Herstellung von w ausreichen.
Die Implikation w erfüllbar $\Rightarrow (G, L, P) \in \text{AGAP}_{\text{VP}}$ sei Übungsaufgabe.

Beispiel:



Hierbei: $P = \{(2, 3)\}$

$$\begin{aligned}
 w = & \overline{x[1, 0]} \wedge \overline{x[2, 0]} \wedge \overline{x[3, 0]} \wedge \\
 & (\overline{x[2, 1]} \vee \overline{x[3, 1]}) \wedge (\overline{x[2, 1]} \vee \overline{x[3, 2]}) \wedge (\overline{x[2, 1]} \vee \overline{x[3, 3]}) \wedge \\
 & (\overline{x[2, 2]} \vee \overline{x[3, 1]}) \wedge \boxed{(\overline{x[2, 2]} \vee \overline{x[3, 2]})} \wedge (\overline{x[2, 2]} \vee \overline{x[3, 3]}) \wedge \\
 & (\overline{x[3, 2]} \vee \overline{x[3, 1]}) \wedge (\overline{x[3, 2]} \vee \overline{x[3, 2]}) \wedge (\overline{x[3, 2]} \vee \overline{x[3, 3]}) \wedge \\
 & (x[1, 3]) \wedge \\
 & (\overline{x[3, 1]} \vee x[2, 0]) \wedge (\overline{x[3, 2]} \vee x[2, 1]) \wedge (\overline{x[3, 3]} \vee x[2, 2]) \wedge \\
 & (\overline{x[1, 1]} \vee x[2, 0]) \wedge (\overline{x[1, 1]} \vee x[3, 0]) \wedge \\
 & (\overline{x[1, 2]} \vee x[2, 1]) \wedge (\overline{x[1, 2]} \vee x[3, 1]) \wedge \\
 & (\overline{x[1, 3]} \vee x[2, 2]) \wedge (\overline{x[1, 3]} \vee x[3, 2])
 \end{aligned}$$

Wäre w erfüllbar, so wegen **3**. $\phi(x[1, 3]) = 1$.

Wegen der **letzten Zeile** wäre daher $\phi(x[2, 2]) = \phi(x[3, 2]) = 1$.

Dann wäre die **umrahmte Klausel** aber falsch.

Klauseln für die Reduktion: $(G, L, P) \in \text{AGAP}_{\forall P} \implies w$ erfüllbar.

v_1, \dots, v_s Knotenfolge, die nacheinander vom Pebble-Spiel markiert werden kann.
 v_1 hat keinen Vorgänger und v_s hat keinen Nachfolger. P wird beachtet.

Belegung ϕ ggb. durch $\phi(x) = 1$ gdw. $\exists i, j : (i \geq j) \wedge x[v_j, i] = x$.

Beh.: Alle Klauseln in w werden durch ϕ wahr.

Beachte: $\neg x[v_1, 1]$ kommt in der Formel allenfalls aufgrund verbotener Paare vor.

1. $\overline{x[v, 0]}$: Klar nach Def. von ϕ .
2. $\overline{x[v, i] \vee x[w, j]}$: Knotenfolge enthält kein verbotenes Paar.
3. $(\bigvee (x[v, n] \mid v \text{ hat keinen Nachf.}))$: $s \leq n \implies \phi(x[v_s, n]) = 1$.
4. $\overline{x[w, i+1] \vee \bigvee (x[v, i] \mid (v, w) \in E)}$, $1 \leq i < n$, für $w \in V$ mit $L(w) = \text{OR}$:
 Ist $\phi(x[w, i+1]) = 1$, so $w = v_j$, $1 \leq j \leq s$ und $i+1 \geq j$.
 Vorgängermenge nicht-leer (nach Konstruktion) $\rightsquigarrow \exists 1 \leq k < j$: v_k ist Vorgänger von v_j .
 Da $i \geq k$, gilt $\phi(x[v_k, i]) = 1$, was diese Klausel erfüllt.
5. $\overline{x[w, i+1] \vee x[v, i]}$, $1 \leq i < n$, für $w \in V$ mit $L(w) = \text{AND}$ und $v \in V$, $(v, w) \in E$:
 Fall analog: alle Vorgänger von v_j müssen sich in der Knotenfolge befinden,
 sobald $\phi(x[v_j, i+1]) = 1$.

Spezialformen von Ausdrücken: 3-CNF-Ausdrücke

—in jeder Klausel drei Literale aus drei verschiedenen Variablen

Diese Einschränkung verursacht im Folgenden technische Probleme.

—erlaubt z.B.: $((x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3))$

—**nicht** erlaubt: $((x_1 \vee x_2 \vee \neg x_2) \wedge (x_1 \vee \neg x_2 \vee \neg x_3))$ In der Literatur oft zulässig!

Satz 3 NP-vollständig (bzgl. \leq_{\log}) sind:

- die Menge SAT aller erfüllbaren (engl.: satisfiable) Formeln aus WFF.
- die Menge 3-SAT aller erfüllbaren 3-CNF-Ausdrücke.
- die Menge NOT-ALL-EQUAL-3-SAT aller 3-CNF-Ausdrücke, bei denen eine Interpretation existiert, die in jeder Klausel mindestens ein Literal wahr und ein Literal falsch macht.

SAT ist NP-vollständig

“Guess and Check” liefert sofort: $SAT \in NP$.

Dabei muss auch überprüft werden, ob Eingabe in WFF liegt (s.o.).

Wir zeigen nun: $CNF\text{-ERFÜLLBARKEIT} \leq_{\log} SAT$.

Was ist überhaupt zu zeigen? Reine Syntax !

Fixiere nicht erfüllbare Formel $w_0 \in WFF$.

Es kann in logarithmischem Platz (wie genau?) geprüft werden, ob ein Wort w ein CNF-Ausdruck ist.

Deshalb ist die folgende Abbildung eine Logspace Karp-Reduktion:

$$f(w) := \begin{cases} w, & w \text{ CNF-Ausdruck} \\ w_0, & \text{sonst} \end{cases}$$

3-SAT ist NP-vollständig

Als Spezialisierung von CNF-ERFÜLLBARKEIT folgt $3 - \text{SAT} \in \text{NP}$.

Wir zeigen nun: $\text{CNF-ERFÜLLBARKEIT} \leq_{\log} 3 - \text{SAT}$.

Die Reduktionsmaschine muss sich zunächst die Länge der längsten $\{0, 1\}$ -Folge in binärem Zähler merken. Wenn im Folgenden “neue Variablen” eingeführt werden, bedeutet das weitere Hochzählen dieses Zählers und Ausgabe von $X1^k$, wobei k der aktuelle Zählerstand ist.

Nun wird die Eingabe fortwährend in die Ausgabe kopiert, solange Klauseln mit drei Literalen gelesen werden (das muss zunächst überprüft werden).

1. Klauseln (x) , $x \in \text{LIT}$, werden mit zwei neuen Variablen y, z “aufgefüllt”:

(x) erfüllbar gdw. $((x \vee y \vee z) \wedge (x \vee \bar{y} \vee z) \wedge (x \vee y \vee \bar{z}) \wedge (x \vee \bar{y} \vee \bar{z}))$.

2. Klauseln $(x \vee y)$, $x, y \in \text{LIT}$, werden mit einer neuen Variablen z “aufgefüllt”:

$(x \vee y)$ gdw. $((x \vee y \vee z) \wedge (x \vee y \vee \bar{z}))$.

Achtung: Gilt $x = y$, so wie im Fall 1; ist y Negation von x , so lasse Klausel bei Übersetzung fort.

3. Klauseln $(x_1 \vee x_2 \vee \dots \vee x_k)$, $k \geq 3$, $x_i \in \text{LIT}$, werden ersetzt durch:

$(\neg y_k \vee x_1 \vee y_1) \wedge (\neg y_1 \vee x_2 \vee y_2) \wedge \dots \wedge (\neg y_{k-2} \vee x_{k-1} \vee y_{k-1}) \wedge (\neg y_{k-1} \vee y_k \vee x_k)$.

Hierbei sind y_1, \dots, y_k neue Variablen. $k = 3$ deckt z.B. $(x \vee y \vee x)$ ab.

Beachte: der Zähler für die neuen Variablen belegt nur Logspace.

NOT-ALL-EQUAL-3-SAT ist NP-vollständig: Einige Gedanken.

∈ NP: Guess and Check sowie WFF-Überprüfung

Bei NOT-ALL-EQUAL-3-SAT ist die Belegung insofern bedeutungslos, als dass die komplementäre Belegung ebenfalls genügt.

$3-SAT \leq_{\log} \text{NOT-ALL-EQUAL-3-SAT}$:

Zu jeder Variablen x (der 3-SAT-Instanz) führe zwei Variablen x_a, x_b ein.

Ist $C_i = (\ell_{i1} \vee \ell_{i2} \vee \ell_{i3})$ Klausel (der 3-SAT-Instanz), führe 6 neue Variablen p_{ij}, q_{ij} ein, $1 \leq j \leq 3$.

Für jedes Literal ℓ_{ij} gibt es zwei Klauseln:

—Gilt $\ell_{ij} = x \in \text{VAR}$, so $(x_a \vee x_b \vee p_{ij}), (x_a \vee x_b \vee q_{ij})$.

—Gilt $\ell_{ij} = \bar{x}, x \in \text{VAR}$, so $(x_a \vee \bar{x}_b \vee p_{ij}), (x_a \vee \bar{x}_b \vee q_{ij})$.

Drei weitere Klauseln sind: $(p_{i1} \vee q_{i1} \vee \bar{p}_{i2}), (p_{i2} \vee q_{i2} \vee \bar{p}_{i3}), (q_{i1} \vee q_{i2} \vee q_{i3})$.

Ist w eine 3-SAT-Instanz, so bezeichne w' die so konstruierte NOT-ALL-EQUAL-3-SAT-Instanz.

Ist w' eine JA-Instanz, so gibt es entspr. Belegung ϕ' . Def.: $\phi(x) = 1 \iff \phi'(x_a) \neq \phi'(x_b)$.

Widerspruchsbeweis: Ist ϕ nicht erfüllend, so gibt es Klausel $C_i = (\ell_{i1} \vee \ell_{i2} \vee \ell_{i3})$, in der kein Literal erfüllt wird. Betrachte den Fall $\ell_{ij} = x. \rightsquigarrow \phi(x) = 0$, also: $\phi'(x_a) = \phi'(x_b). \rightsquigarrow \phi'(p_{ij}) \neq \phi'(x_a)$

und $\phi'(q_{ij}) \neq \phi'(x_a)$, also $\phi'(p_{ij}) = \phi'(q_{ij})$. Dieselbe Schlussfolgerung ist für $\ell_{ij} = \bar{x}$ möglich.

Rote Klauseln $\rightsquigarrow \phi'(p_{i1}) = \phi'(q_{i1}) = \phi'(p_{i2}) = \phi'(q_{i2}) = \phi'(p_{i3}) = \phi'(q_{i3}) \rightsquigarrow$ **letzte Klausel** !?!

Die umgekehrte Richtung ist aufwendiger.

Grundgedanken der entgegengesetzten Richtung

Ist w eine JA-Instanz, so gibt es entspr. Belegung ϕ .

Def.: $\phi'(x_a) = 1$ sowie $\phi'(x_b) = \phi(x)$ für jede Variable x von w .

Damit: $\phi(x) = 1 \iff \phi'(x_a) \neq \phi'(x_b)$.

Die Belegung der Variablen q_{ij} und p_{ij} ergibt sich aus der Struktur von C_i und wird durch etliche Fallunterscheidungen festgelegt.

Wir betrachten exemplarisch:

$C_i = (x \vee \neg y \vee z)$ sowie eine Belegung mit $\phi(x) = \phi(y) = 1$ und $\phi(z) = 0$.

Wegen $\phi'(x_a) = \phi'(x_b) = 1$ folgt aus $(x_a \vee x_b \vee p/q_{i1})$: $\phi'(q_{i1}) = \phi'(p_{i1}) = 0$.

Wegen $\phi'(y_a) = 1, \phi'(y_b) = 1$ folgt aus $(y_a \vee \overline{y_b} \vee p/q_{i2})$ nichts.

Wegen $\phi'(z_a) = 1, \phi'(z_b) = 0$ folgt aus $(z_a \vee z_b \vee p/q_{i3})$ nichts.

$(p_{i1} \vee q_{i1} \vee \overline{p_{i2}}) \rightsquigarrow \phi'(p_{i2}) = 0$.

$(p_{i2} \vee q_{i2} \vee \overline{p_{i3}})$ sowie $(q_{i1} \vee q_{i2} \vee q_{i3})$ sind o.k. z.B. mit der Belegung:

$\phi'(q_{i2}) = \phi'(p_{i2}) = 0, \phi'(p_{i3}) = 0, \phi'(q_{i3}) = 1$.

Anmerkungen:

—SAT war das erste bekannte **NP**-vollständige Problem (1971, S. A. Cook; zeitgleich: Levin).

—2-SAT: erfüllbare CNF-Ausdrücke, bei denen jede Klausel aus 2 Literalen zusammengesetzt ist, liegt in $co - \mathbf{NL} = \mathbf{NL}$.

Probleme aus Graphentheorie:

Satz 4 NP-vollständig (bzgl. \leq_{\log}) sind:

- SIMPLE MAX CUT
- VERTEX COVER
- CLIQUE
- INDEPENDENT SET
- 3-FÄRBBARKEIT
- GERICHTETER HAMILTON-KREIS
- UNGERICHTETER HAMILTON-KREIS
- TRAVELING SALESMAN
- SUBGRAPH ISOMORPHISM

SIMPLE MAX CUT:

Gegeben seien ein ungerichteter Graph $G = (V, E)$ und eine natürliche Zahl k .

Frage: Kann man die Knotenmenge V so in zwei disjunkte Teilmengen V_a, V_b aufteilen, dass die Zahl der Kanten, die ein Ende in V_a und ein Ende in V_b haben, mindestens k ist?

Anwendung z.B.:

—G Netz von elektrischen Widerständen

—jede Kante aus E ist Widerstand (z.B. 500 Ohm)

—jeder Knoten mit Spannungsquelle (0V oder 5V) verbunden

—Zwischen benachbarten Knoten: 10 mA Strom, wenn sie mit verschiedenen Polen verbunden sind. Erinnerung: Ohmsches Gesetz $U = R \times I$.

—**Frage:** Kann man die Knoten so anschließen, dass der Gesamtstrom $k \times 10$ mA überschreitet?

Sei $G = (V, E)$ ungerichteter Graph:

$V' \subseteq V$ ist *Knotenüberdeckung*, wenn jede Kante aus E mindestens einen Endpunkt in V' hat.

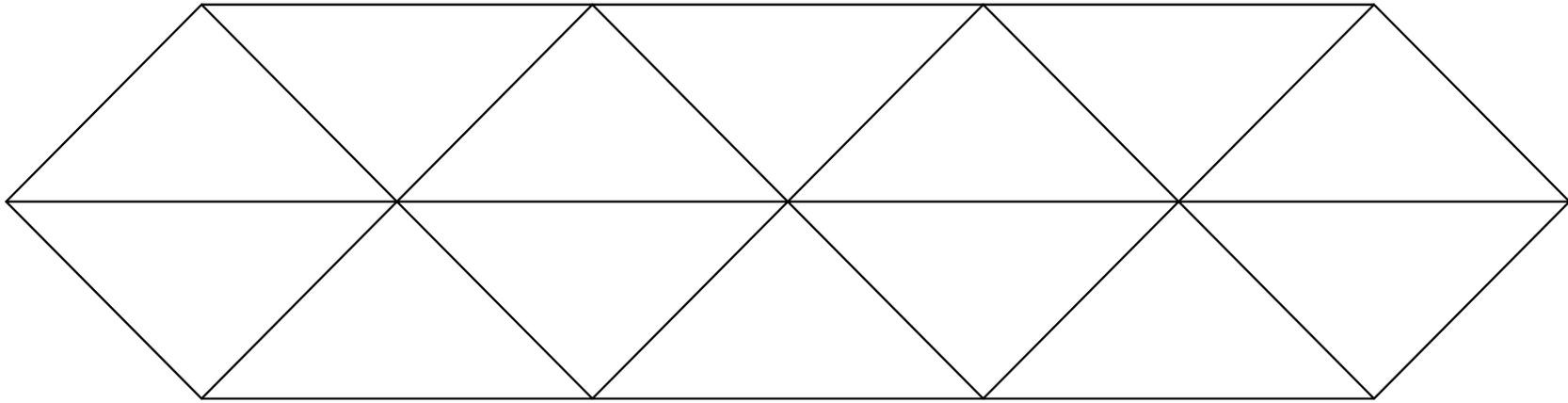
\Rightarrow VERTEX COVER als graphentheoretisches Überdeckungsproblem:

VERTEX COVER:

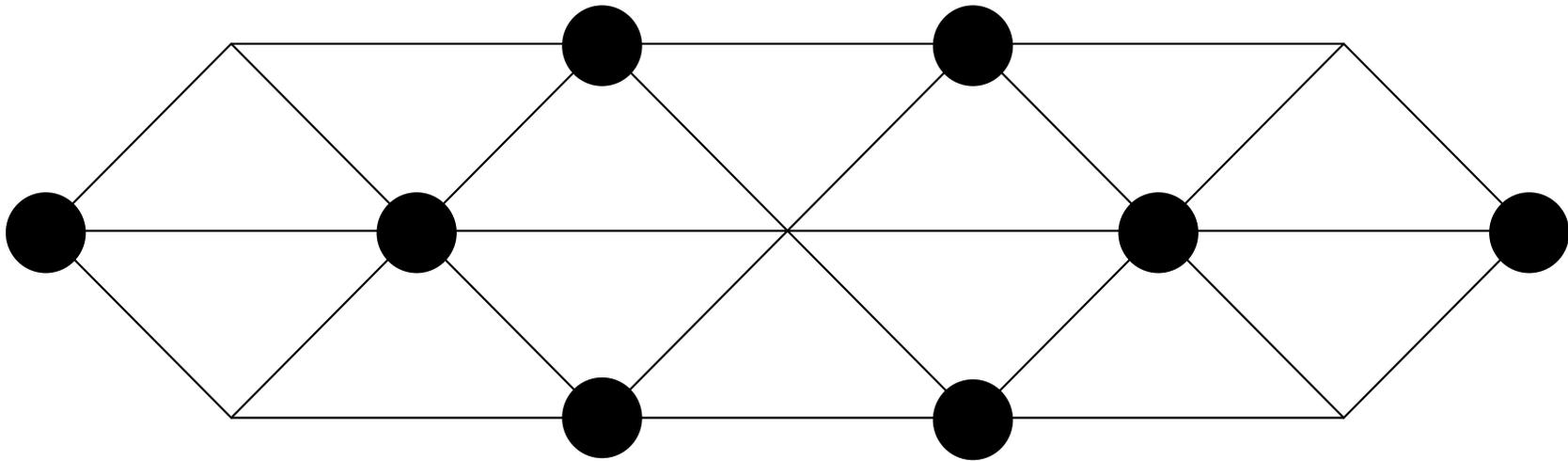
Gegeben seien ein ungerichteter Graph $G = (V, E)$ und eine natürliche Zahl k .

Frage: Gibt es eine Menge $V' \subseteq V$ aus höchstens k Knoten, die eine Knotenüberdeckung bildet?

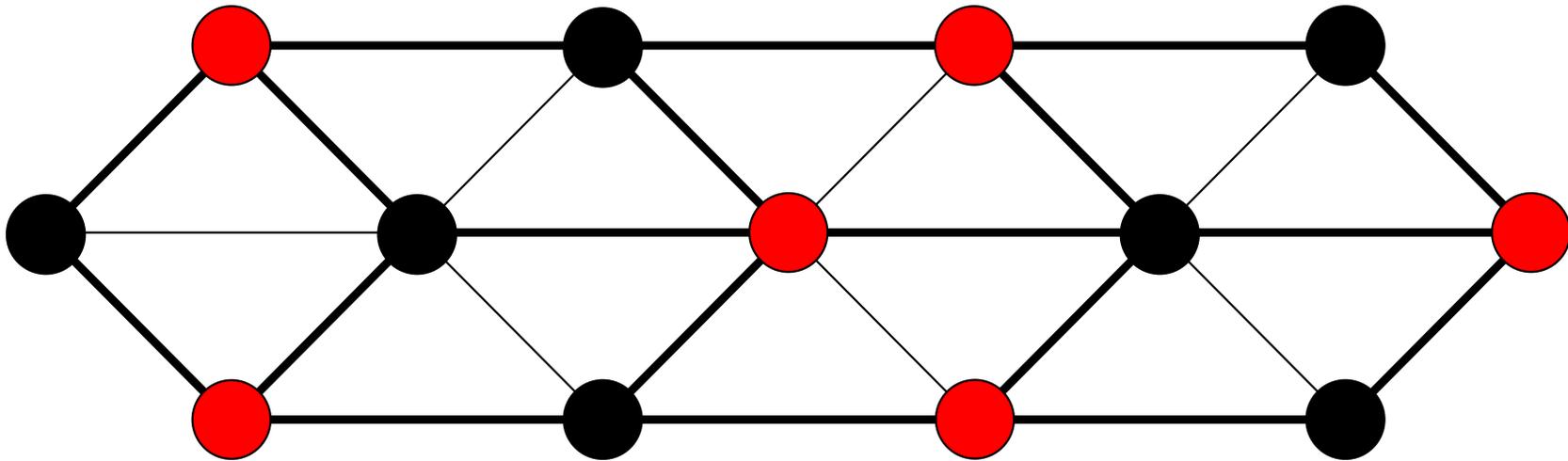
Beispiel für die Begriffe: Betrachte folgenden Graphen:



Beispiel: Lösung zu VERTEX COVER mit $k \geq 8$:



Beispiel: Lösung zu SIMPLE MAX CUT mit $k = 19$:

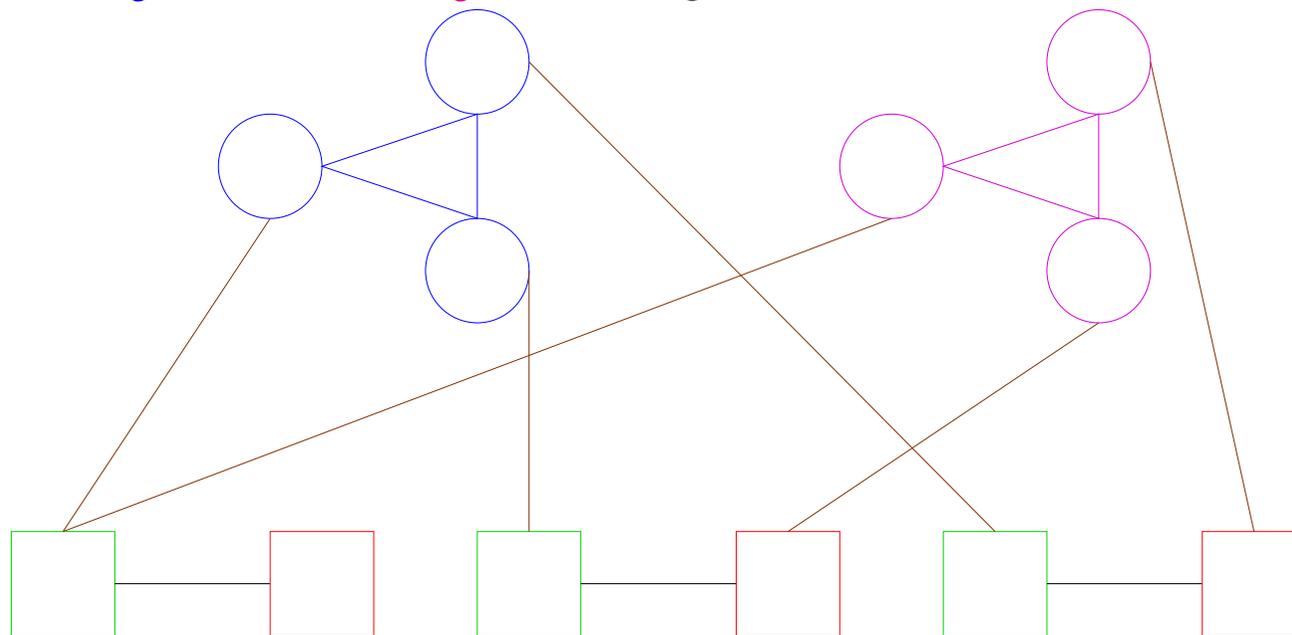


Graph enthält 7 kantendisjunkte Dreiecke

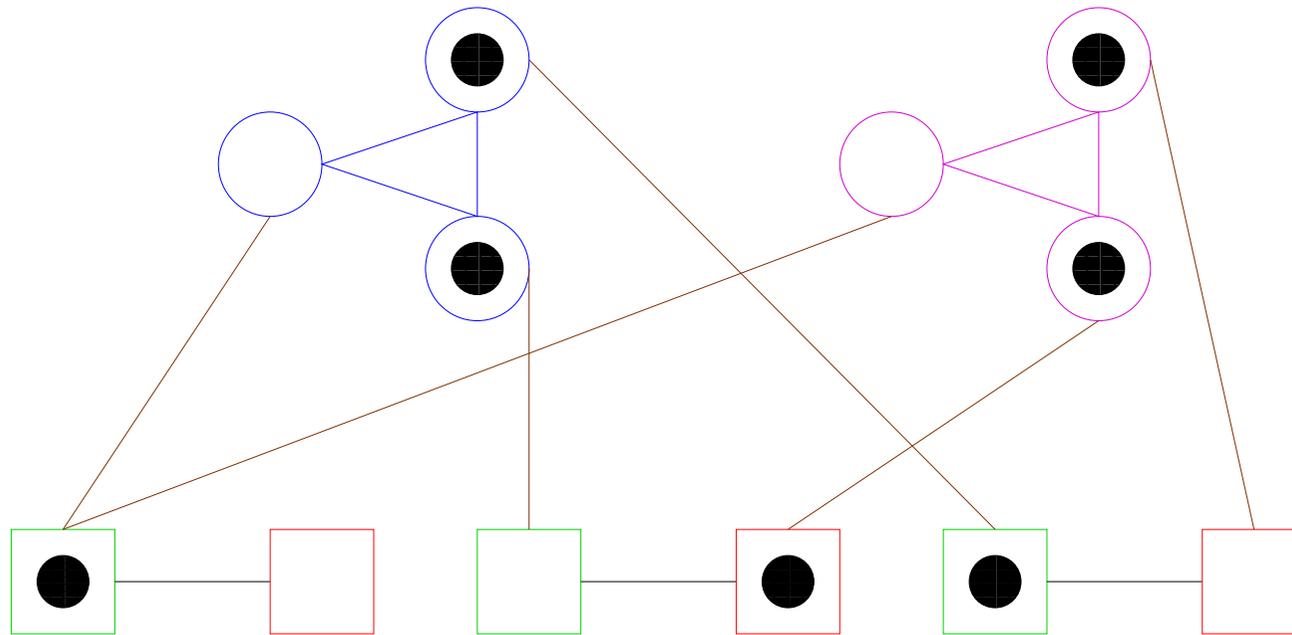
~> höchstens zwei Kanten jedes dieser Dreiecke kommt in den Schnitt

~> $k = 19$ ist maximal, da 26 Kanten insgesamt

Grundidee: Codiere 3-SAT (o.ä.) durch “Gadgets” für Variablen und Klauseln.
 “Oben”: $3m$ Knoten für die m Klauseln, “unten” $2n$ Knoten für die n Variablen.
 Kanten zwischen Klauselknoten und Literalknoten kennzeichnen Vorkommen.
 Im Beispiel: $(x \vee y \vee z) \wedge (x \vee \bar{y} \vee \bar{z})$. Allg.: Formel $w \mapsto G(w)$.



1. Konstruktion: $w \in 3\text{-SAT}$ gdw.
 $(G(w), 2m + n) \in \text{VERTEX COVER}$.
 Im Beispiel: $(x \vee y \vee z) \wedge (x \vee \bar{y} \vee \bar{z})$.



Punkte markieren die Knotenüberdeckung.
 Wegen der Dreiecke oben und Kanten unten folgt Minimalität.

Wie wird daraus ein Beweis?

- Gib Konstruktionen stets auch formal an.
- Überprüfe die Formalisierung an Beispielen.
- Lässt sich die Konstruktion mit den vorhandenen Ressourcen durchführen?
 - Konkret: Gibt es eine deterministische Logspace-TM, die die Konstruktion durchführen kann?
- Ist die Konstruktion wirklich eine (many-one) Reduktion f ?
 - (a) Ist I eine JA-Instanz des Ausgangsproblems (z.B. 3-SAT) so ist $f(I)$ eine JA-Instanz des Bildproblems (z.B. VC).
 - (b) Ist $f(I)$ eine JA-Instanz des Bildproblems (z.B. VC), so ist I eine JA-Instanz des Ausgangsproblems (z.B. 3-SAT).

Mögliche Übungsaufgaben

—Das Knotenüberdeckungsproblem ist NP-vollständig für Graphen von Maximalgrad drei.

—Ein Hypergraph $G = (V, E)$ besteht aus einer endlichen Knotenmenge V und einer Kantenmenge $E \subseteq 2^V$. $r(G) = \max\{|e| : e \in E\}$ heißt auch *Rang* von G . Eine Färbung $V \rightarrow \{0, 1\}$ heißt *farbenfroh*, falls jede Kante einen Knoten enthält, der mit 0, und einen Knoten enthält, der mit 1 gefärbt wurde.

Untersuchen Sie die Komplexität der folgenden Frage für verschiedene r :

Eingabe: Ein Hypergraph $G = (V, E)$ vom Rang r .

Frage: Gibt es eine farbenfrohe Färbung $c : V \rightarrow \{0, 1\}$ von G ?