Komplexitätstheorie

WiSe 2008/09 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, $\mathbf{P} = \mathbf{NP}$ 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:

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

z.B. binär interpretiert als $\mathtt{VAR} = \{x_1, x_2, \ldots\}$

Literale:

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

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

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

1. $\mathtt{VAR} \subset \mathtt{WFF}$
2. $F \in \mathtt{WFF} \Longrightarrow \neg F \in \mathtt{WFF}$
3. $F_1, \ldots F_n \in \mathtt{WFF} \Longrightarrow (F_1 \wedge \ldots \wedge F_n) \in \mathtt{WFF}$
4. $F_1, \ldots F_n \in \mathtt{WFF} \Longrightarrow (F_1 \vee \ldots \vee F_n) \in \mathtt{WFF}$
5. $F_1, F_2 \in \mathtt{WFF} \Longrightarrow (F_1 \rightarrow F_2) \in \mathtt{WFF}$
6. $F_1, F_2 \in \mathtt{WFF} \Longrightarrow (F_1 \leftrightarrow F_2) \in \mathtt{WFF}$

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

$$\phi : \mathtt{VAR} \to \{0, 1\}$$

$\Rightarrow$ jeder Variablen wird Wert 0 ('falsch') oder 1 ('wahr') zugeordnet.

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

$$\mathtt{WERT}_\phi(F) :=$$

$$\begin{cases} 1 - \mathtt{WERT}_\phi(F') & \text{falls } F = \neg F' \\ \min\{\mathtt{WERT}_\phi(F_1), \ldots, \mathtt{WERT}_\phi(F_n)\} & \text{falls } F = (F_1 \wedge \ldots \wedge F_n) \\ \max\{\mathtt{WERT}_\phi(F_1), \ldots, \mathtt{WERT}_\phi(F_n)\} & \text{falls } F = (F_1 \vee \ldots \vee F_n) \\ \max\{1 - \mathtt{WERT}_\phi(F_1), \mathtt{WERT}_\phi(F_2)\} & \text{falls } F = (F_1 \to F_2) \\ \min\{\mathtt{WERT}_\phi(F_1 \to F_2), \\ \qquad \mathtt{WERT}_\phi(F_2 \to F_1)\} & \text{falls } F = (F_1 \leftrightarrow F_2) \end{cases}$$

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

Lemma 1 $\texttt{WFF} \in \texttt{DSPACE}(\log n)$

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) $\forall w \in \texttt{WFF} : \#K(w) = 0$.

Ferner ist wahr: (2) $\forall w \in \texttt{WFF} \forall v : v$ 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 2-3 Zähler nötig.

Für die Korrektheit bliebe noch per Induktion zu zeigen, dass jedes Wort w, welches die Prozedur "übersteht", in WFF liegt.

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

(disjunktive) *Klauseln*:

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

CNF-Ausdruck (conjunctive normal form)

$$(c_1 \wedge \ldots \wedge c_k)$$

wobei die c_i Klauseln sind.

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:

$$A_0 = ((\mathtt{X1} \vee \neg \mathtt{X10}) \wedge (\mathtt{X1} \vee \mathtt{X11}))$$

oder in anderer Schreibweise

$$((x_1 \vee \overline{x_2}) \wedge (x_1 \vee x_3))$$

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 $\texttt{AGAPvP} \leq_{\log} \texttt{CNF-ERFÜLLBARKEIT}$

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

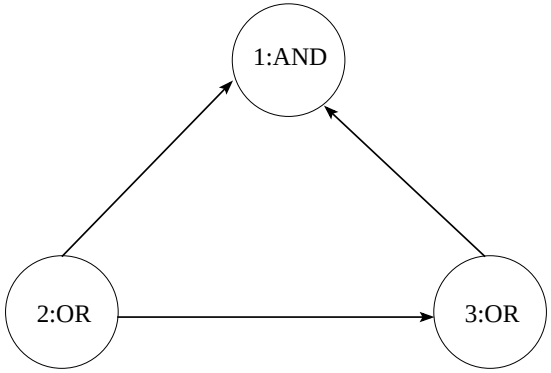
Sonderfall: In G gibt es keinen Knoten ohne Nachfolger.
Setze $w = ((\texttt{x1}) \wedge (\neg \texttt{X1}))$. w ist trivial unerfüllbar.
Im Folgenden daher Annahme: Es gibt einen Knoten ohne Nachfolger.
Führe *Variablen* $x[v, i]$ ein zu $v \in V$ und $1 \leq i \leq n$ mit der Bedeutung:
Für (insgesamt) 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. $(\bigvee(x[v,n] \mid v \text{ hat keinen Nachfolger}))$
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 $L(w) = \texttt{OR}$.
5. $(\overline{x[w,i+1]} \vee x[v,i])$, $1 \leq i < n$, für $w \in V$ mit $L(w) = \texttt{AND}$ 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 $\Rightarrow (G,L,P) \in \texttt{AGAPvP}$ sei Übungsaufgabe.

Beispiel:



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

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

$$(x[1,3]) \wedge$$
$$(\overline{x[3,1]} \vee x[2,0]) \wedge (\overline{x[3,2]} \vee x[2,1]) \wedge (\overline{x[3,3]} \vee x[2,2]) \wedge$$

$$(\overline{x[1,1]} \vee x[2,0]) \wedge (\overline{x[1,1]} \vee x[3,0]) \wedge$$
$$(\overline{x[1,2]} \vee x[2,1]) \wedge (\overline{x[1,2]} \vee x[3,1]) \wedge$$
$$(\overline{x[1,3]} \vee x[2,2]) \wedge (\overline{x[1,3]} \vee x[3,2])$$

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: $(G, L, P) \in \texttt{AGAPvP} \implies w$ erfüllbar.

$v_1, \ldots, v_s$ Knotenfolge, die nacheinander vom Pebble-Spiel markiert werden kann.
x_1 hat keinen Vorgänger und x_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.

1. $(\overline{x[v, 0]})$: Klar nach Def. von ϕ.
2. $(\overline{x[v, i]} \vee \overline{x[w, j]})$: 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 $L(w) = \texttt{OR}$:
   Ist $\phi(x[w, i+1]) = 1$, so $w = v_j$, $1 \leq j \leq s$ und $i + 1 \geq j$.
   Vorgängermenge nicht-leer (sonst Fall 1.) $\rightsquigarrow \exists 1 \leq k < j$: 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 $L(w) = \texttt{AND}$ 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

—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: $((x_1 \vee x_2 \vee \neg x_2) \wedge (x_1 \vee \neg x_2 \vee \neg x_3))$

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: $\texttt{SAT} \in \mathbf{NP}$.
Dabei muss auch überprüft werden, ob Eingabe in WFF liegt (s.o.).

Wir zeigen nun: $\texttt{CNF-ERFÜLLBARKEIT} \leq_{\log} \texttt{SAT}$.
Fixiere nicht erfüllbare Formel $w_0 \in \texttt{WFF}$.
Es kann in logarithmischem Platz (wie genau?) geprüft werden, ob ein Wort w ein CNF-Ausdruck ist.
Deshalb ist die folgende Abbildung eine Logspace many-one 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 $3-\text{SAT} \in \mathbf{NP}$.

Wir zeigen nun: 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).
Klauseln $(x \vee y)$, $x, y \in \text{LIT}$, werden mit einer neuen Variablen z "aufgefüllt":
$(x \vee y) \equiv ((x \vee y \vee z) \wedge (x \vee y \vee \neg z))$.
Klauseln (x), $x \in \text{LIT}$, werden mit zwei neuen Variablen y, z "aufgefüllt" (analog).
Klauseln $(x_1 \vee x_2 \vee \cdots \vee x_k)$, $k \geq 4$, $x_i \in \text{LIT}$, werden ersetzt durch:
$(x_1 \vee x_2 \vee y_1) \wedge (\neg y_1 \vee x_3 \vee y_2) \wedge \cdots \wedge (\neg y_{k-4} \vee x_{k-2} \vee y_{k-3}) \wedge (\neg y_{k-3} \vee x_{k-1} \vee x_k))$.
Hierbei sind $y_1, \ldots, y_{k-3}$ neue Variablen.
Beachte: der Zähler für die neuen Variablen belegt nur Logspace.

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

$\in$ **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.

$3-\text{SAT} \leq_{\log}$ 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 $\ell_{ij} = x \in$ VAR, so $(x_a \vee x_b \vee p_{ij})$, $(x_a \vee x_b \vee q_{ij})$.
—Gilt $\ell_{ij} = \bar{x}$, $x \in$ VAR, so $(x_a \vee \overline{x_b} \vee p_{ij})$, $(x_a \vee \overline{x_b} \vee q_{ij})$.
Drei weitere Klauseln sind: $(p_{i1} \vee q_{i1} \vee \overline{p_{i2}})$, $(p_{i2} \vee q_{i2} \vee \overline{p_{i3}})$, $(q_{i1} \vee q_{i2} \vee \overline{q_{i3}})$.

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 aufwändiger.

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 $\mathrm{co} - \mathbf{NL} = \mathbf{NL}$.

Probleme aus Graphentheorie:

Satz 4 $\mathbf{NP}$*-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

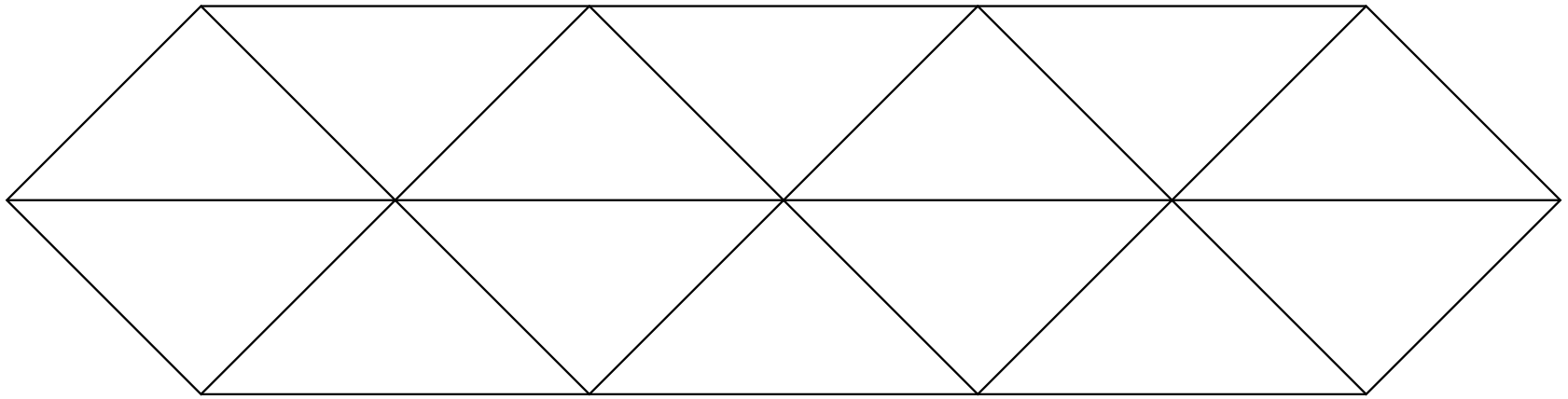
SIMPLE MAX CUT: Gegeben seien ein ungerichteter Graph $G = (V, E)$ und eine natürliche Zahl k.
Frage: Kann man die Knotenmenge V so in zwei disjunkte Teilmengen V_a, V_b aufteilen, dass die Zahl der Kanten, die ein Ende in V_a und ein Ende in V_b haben, mindestens k ist?

Anwendung z.B.:
—G Netz von elektrischen Widerständen
—jede Kante aus E ist Widerstand (z.B. 200 Ohm)
—jeder Knoten mit Spannungsquelle ($0V$ oder $5V$) verbunden
—Zwischen benachbarten Knoten: 1 mA Strom, wenn sie mit dem verschiedenen Polen verbunden sind.
—Frage: Kann man die Knoten so anschließen, dass der Gesamtstrom k mA überschreitet?

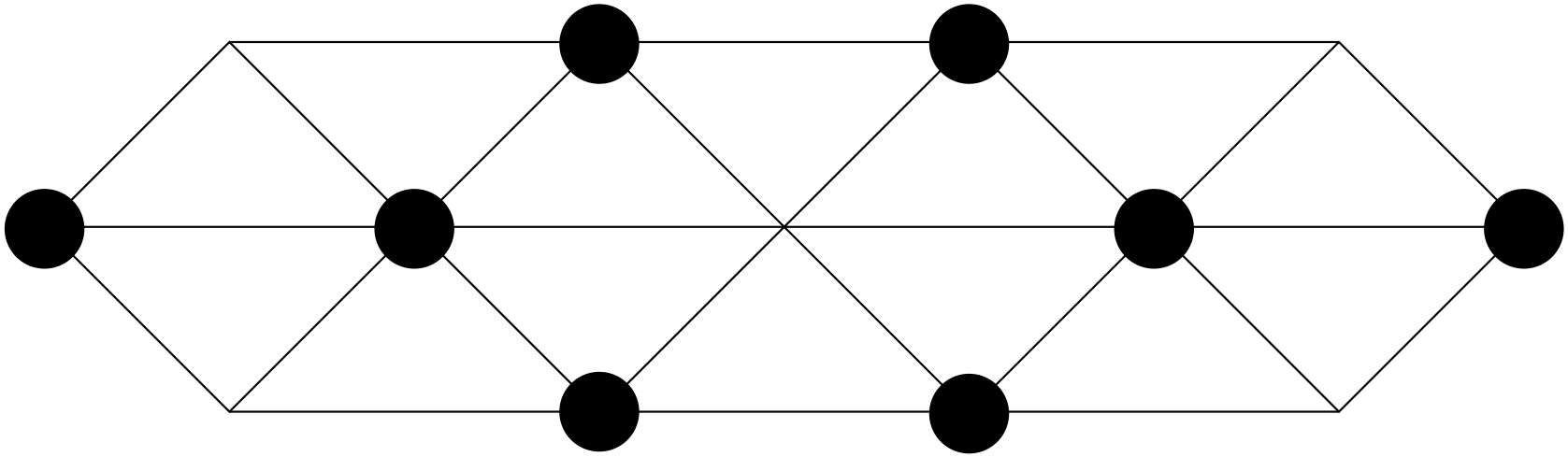
Sei $G = (V, E)$ ungerichteter Graph:
$V' \subseteq V$ ist *Knotenüberdeckung*, wenn jede Kante aus E mindestens einen Endpunkt in V' hat.

$\Rightarrow$ 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?

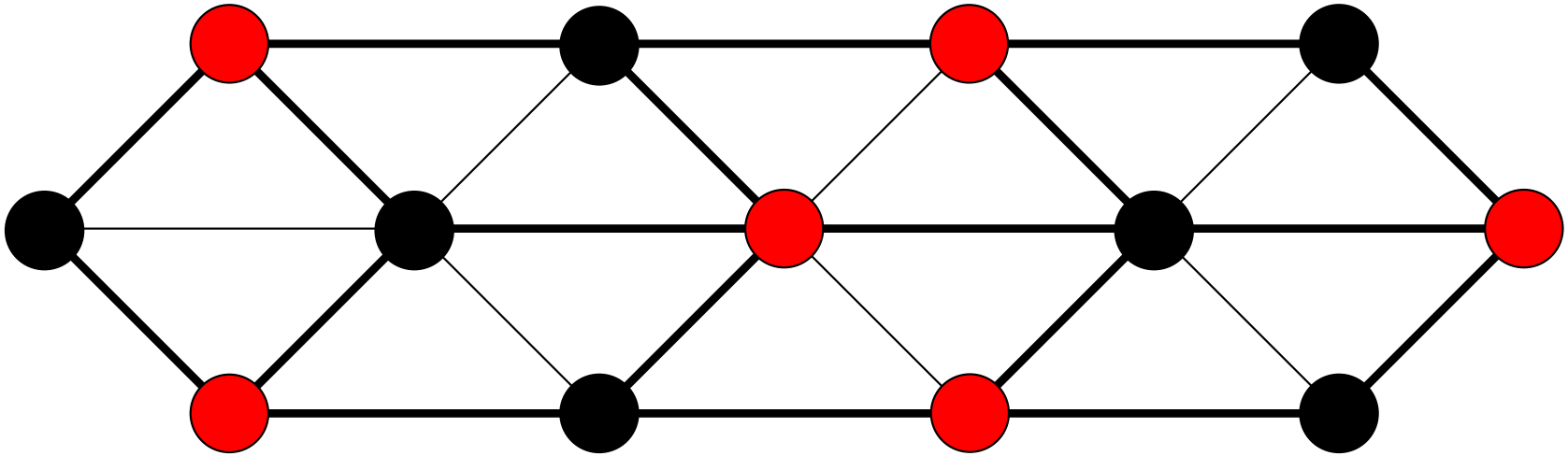
Beispiel für die Begriffe: Betrachte folgenden Graphen:



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



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

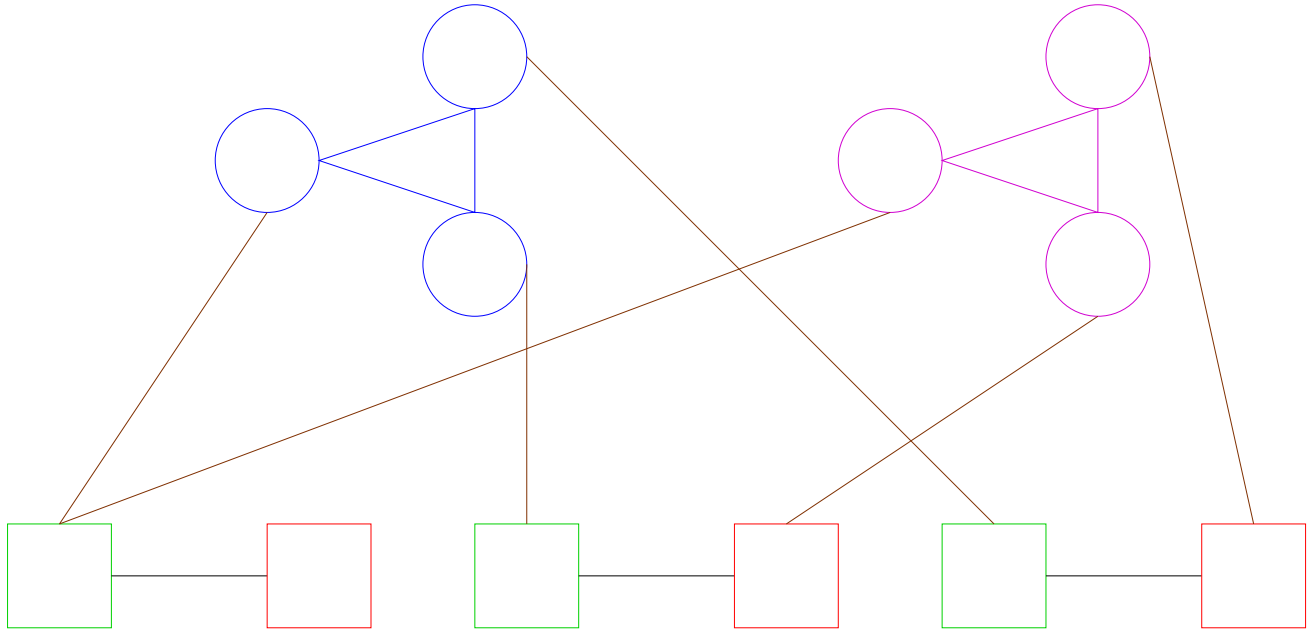


Graph enthält 7 kantendisjunkte Dreiecke

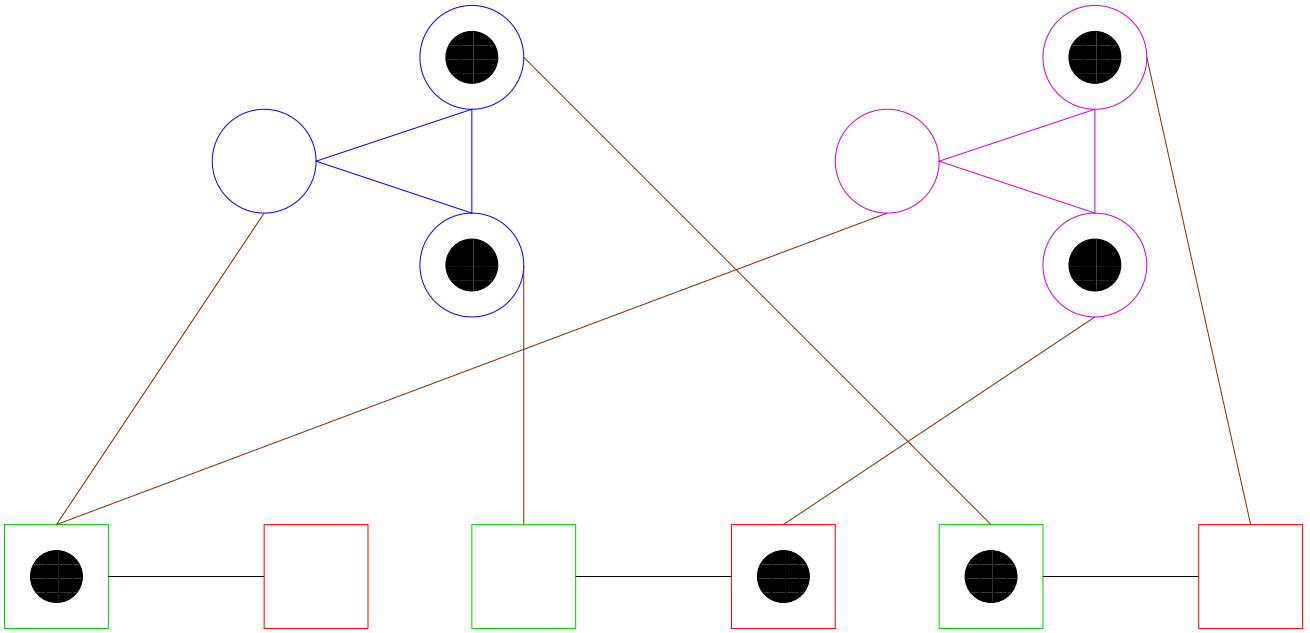
$\rightsquigarrow$ höchstens zwei Kanten jedes dieser Dreiecke kommt in den Schnitt

$\rightsquigarrow$ $k = 19$ 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 $w \mapsto G(w)$.

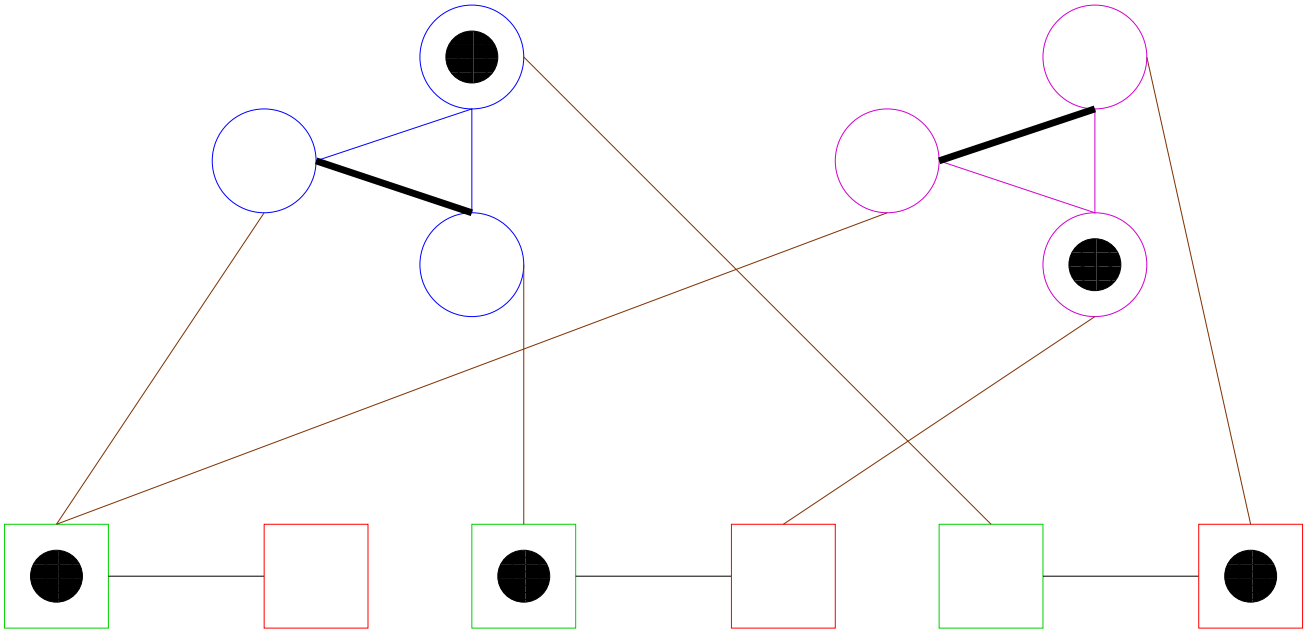


1. Konstruktion: $w \in$ 3-SAT gdw.
$(G(w), 2m + n) \in$ VERTEX COVER.
Im Beispiel: $(x \vee y \vee z) \wedge (x \vee \bar{y} \vee \bar{z})$.



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

2. Konstruktion: $w \in$ NOT-ALL-EQUAL-3-SAT gdw.
$(G(w), 5m + n) \in$ SIMPLE MAX CUT.
Im Beispiel: $(x \vee y \vee z) \wedge (x \vee \bar{y} \vee \bar{z})$.



Punkte markieren Knoten aus V_a.
Die “dicken” Kanten liegen NICHT im Schnitt.

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 $\mathtt{f}$?
——(a) Ist $\mathtt{I}$ eine JA-Instanz des Ausgangsproblems (z.B. $\mathtt{3-SAT}$) so ist $\mathtt{f(I)}$ eine JA-Instanz des Bildproblems (z.B. $\mathtt{VC}$).
——(b) Ist $\mathtt{f(I)}$ eine JA-Instanz des Bildproblems (z.B. $\mathtt{VC}$), so ist $\mathtt{I}$ eine JA-Instanz des Ausgangsproblems (z.B. $\mathtt{3-SAT}$).