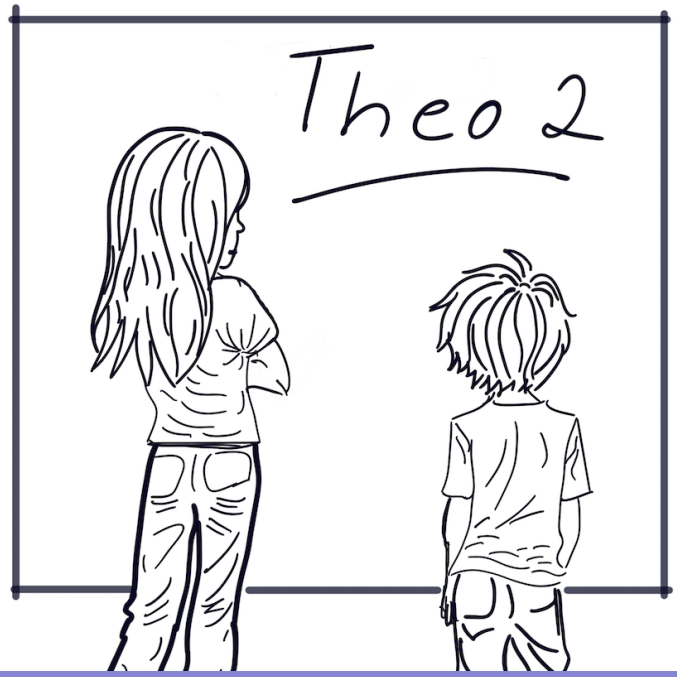
Theoretische Informatik 2

SoSe 2024

Berechenbarkeit und Komplexität

Prof. Dr. Sebastian Siebertz
AG Theoretische Informatik
MZH, Raum 3160
siebertz@uni-bremen.de

Universität Bremen



P vs NP

- Die Komplexitätsklasse PTime, kurz P, enthält alle Entscheidungsprobleme, die in Polynomialzeit von einer deterministischen Turingmaschine lösbar sind.

$$\mathsf{P} := \bigcup_{p \text{ Polynom in } n} \mathsf{DTime}(p(n)).$$

- Die Komplexitätsklasse NPTime, kurz NP, enthält alle Entscheidungsprobleme, die in Polynomialzeit von einer nicht-deterministischen Turingmaschine lösbar sind.

$$\mathsf{NP} := \bigcup_{p \text{ Polynom in } n} \mathsf{NTime}(p(n)).$$

Vorsicht: NP steht nicht für nicht-polynomiell.

NP = polynomialzeit-verifizierbare Lösungen

Satz

Ein Problem $L \subseteq \Sigma^*$ liegt in NP genau dann, wenn ein Polynom p und eine polynomialzeitbeschränkte deterministische Turingmaschine M existieren (genannt Verifizierer), so dass für jedes $v \in \Sigma^*$ gilt

$$v \in L \Leftrightarrow \text{es ex. } w \in \Sigma^{p(|v|)}, \text{ so dass } M \text{ das Wort } vw \text{ akzeptiert.}$$

Das Wort w der Länge $p(|v|)$ wird ein Zertifikat für v genannt.

Ein Problem L liegt in NP genau dann,

wenn Lösungen für L in Polynomialzeit verifiziert werden können.

Polynomialzeitreduktionen und NP-schwere Probleme

- Wir vermuten, dass P $\neq$ NP, haben aber noch keinen Beweis gefunden.
- Es gibt hunderte von wichtigen Problemen in NP, für die wir keinen Polyomialzeitalgorithmus kennen.
- Eine Reduktion f von $L_1 \subseteq \Sigma^*$ auf $L_2 \subseteq \Sigma^*$ heißt Polynomialzeitreduktion, wenn es ein Polynom p und eine $p(n)$-zeitbeschränkte DTM gibt, die f berechnet.
- Wenn eine Polynomialzeitreduktion von L_1 auf L_2 existiert, dann schreiben wir $L_1 \leq_p L_2$.
- Eine Sprache L heißt NP-schwer, wenn für alle $L' \in$ NP gilt: $L' \leq_p L$.
- L heißt NP-vollständig, wenn $L \in$ NP und L NP-schwer ist.

Der Satz von Cook und Levin

Satz von Cook und Levin

SAT ist NP-vollständig.

- Wir müssen zeigen: für alle $L' \in \text{NP}$ gilt $L' \leq_p \text{SAT}$.
- L' ist beliebige Sprache in NP!
- Wir wissen wir nur, dass eine nichtdeterministische $p(n)$-zeitbeschränkten Turingmaschine $M = (Q, \Sigma, \Gamma, \sqcup, \Delta, Q_s, F)$ existiert, für ein Polynom p, die L entscheidet.
- Wir suchen eine polynomialzeitberechenbare Funktion f, die ein Wort $w \in \Sigma^*$ auf eine (Kodierung einer) Formel φ_w der Aussagenlogik abbildet, so dass

$$M \text{ akzeptiert } w \Leftrightarrow \varphi_w \text{ erfüllbar.}$$

Der Satz von Cook und Levin

- Verwende drei Familien boolescher Variablen mit den folgenden Belegung:
  - $B_{a,i,t}$: „Zum Zeitpunkt t ist die Bandposition i mit a beschriftet.“
  - $K_{i,t}$: „Zum Zeitpunkt t ist der Kopf an Position i.“
  - $Z_{q,t}$: „Zum Zeitpunkt t ist der aktuelle Zustand q.“
- M ist $p(n)$-zeitbeschränkt
  - jede Berechnung von M auf w terminiert in Zeit höchstens $p(|w|)$.
  - In jeder Berechnung von M auf w werden höchstens die Bandzellen mit Indizes $-p(|w|), \ldots, p(|w|)$ beschrieben.
- Im Folgenden: $1 \leq t \leq p(n)$, $-p(n) \leq i \leq p(n)$, $a, b \in \Gamma$ und $p, q \in Q$.

Der Satz von Cook und Levin

- Konstruiere zu Eingabe w die Formel φ_w, so dass die erfüllenden Belegungen von φ_w genau den akzeptierenden Berechnungen von M auf w entsprechen.
- Konjunktion aus mehreren Teilformeln:
  - Die Variablen kodieren Konfigurationen von M:

$$\psi_{\text{config}} := \bigwedge_{t,p,q,p\neq q} \neg(Z_{p,t} \wedge Z_{q,t}) \wedge \bigwedge_{t,i,a,b,a\neq b} \neg(B_{a,i,t} \wedge B_{b,i,t}) \wedge$$

$$\bigwedge_{t,i,j,i\neq j} \neg(K_{i,t} \wedge K_{j,t}) \wedge$$

$$\bigwedge_{t} \bigvee_{q} Z_{q,t} \wedge \bigwedge_{t} \bigvee_{i} K_{i,t} \wedge \bigwedge_{t} \bigwedge_{i} \bigvee_{a} B_{a,i,t}$$

Der Satz von Cook und Levin

- Behauptung (folgt sofort aus der Konstruktion von ψ_{config}):

  Jede Belegung $\mathfrak{I}$ mit $\mathfrak{I} \models \psi_{\text{config}}$ erfüllt die folgenden Eigenschaften:

  - für jedes t existiert genau ein q, so dass $\mathfrak{I} \models Z_{q,t}$,
  - für jedes t und jedes i existiert genau ein a, so dass $\mathfrak{I} \models B_{a,i,t}$ und
  - für jedes t existiert genau ein i, so dass $\mathfrak{I} \models K_{i,t}$.

Der Satz von Cook und Levin

Lemma

Sei $\mathfrak{I}$ eine Belegung mit $\mathfrak{I} \models \psi_{\text{config}}$. Dann definiert $\mathfrak{I}$ eindeutig eine Folge $\alpha_1, \ldots, \alpha_{p(|w|)}$ von Konfigurationen von M, so dass $\alpha_t = (v, k, q)$ mit

- *$v_i = a$ für das eindeutige a, so dass $\mathfrak{I} \models B_{a,i,t}$,*
- *k ist die eindeutige Zahl, so dass $\mathfrak{I} \models K_{i,t}$ und*
- *q ist der eindeutige Zustand, so dass $\mathfrak{I} \models Z_{q,t}$.*

Der Satz von Cook und Levin

- Wir fügen weitere Teilformeln als Konjunkte hinzu um sicherzustellen, dass die Berechnung mit der Startkonfiguration für $w = a_1 \cdots a_n$ beginnt:

$$\psi_{\text{init}} := Z_{q_s,1} \wedge K_{1,1} \wedge \bigwedge_{1 \leq i \leq n} B_{a_i,i,1} \wedge \bigwedge_{n < i \leq p(n)} B_{\sqcup,i,1} \wedge \bigwedge_{-p(n) \leq i \leq 0} B_{\sqcup,i,1}$$

Der Satz von Cook und Levin

Lemma

Sei $\mathfrak{I}$ eine Belegung mit $\mathfrak{I} \vDash \psi_{\text{config}} \wedge \psi_{\text{init}}$. Sei $\alpha_1, \ldots, \alpha_{p(|w|)}$ die eindeutig von $\mathfrak{I}$ bestimmte Folge von Konfigurationen von M. Dann ist α_1 die Startkonfiguration von M auf w.

Der Satz von Cook und Levin

- Eine NTM akzeptiert, wenn sie sich in einer akzeptierenden Stoppkonfiguration befindet.
- Eine Stoppkonfiguration ist eine Konfiguration ohne Nachfolgekonfiguration.
- Die nächsten Teilformeln stellen sicher, dass solange α_t keine Stoppkonfiguration ist, so gilt $\alpha_t \vdash_M \alpha_{t+1}$.

Der Satz von Cook und Levin

$$\psi_{\text{compute}} := \bigwedge_{\substack{t,i,a,p \\ \text{es ex. } (p,a,q,b,m)\in\Delta}} \Big(\big(B_{a,i,t} \wedge K_{i,t} \wedge Z_{p,t}\big) \rightarrow \bigvee_{(p,a,q,b,m)\in\Delta} \big(B_{b,i+m,t+1} \wedge K_{i+m,t+1} \wedge Z_{q,t+1}\big) \Big),$$

wobei $m \in \{-1, 0, +1\}$ und

$$\psi_{\text{no-change}} := \bigwedge_{t,i,a} \Big(\big(\neg K_{i,t} \wedge B_{a,i,t}\big) \rightarrow B_{a,i,t+1} \Big).$$

Der Satz von Cook und Levin

Lemma

Sei $\mathfrak{I}$ eine Belegung mit $\mathfrak{I} \vDash \psi_{\text{config}} \wedge \psi_{\text{init}} \wedge \psi_{\text{compute}} \wedge \psi_{\text{no-change}}$.

Sei $\alpha_1, \ldots, \alpha_{p(|w|)}$ die eindeutig von $\mathfrak{I}$ bestimmte Folge von Konfigurationen von M, wobei die Konfiguration α_1 die Startkonfiguration von M auf w ist.

Dann existiert t, so dass $\alpha_1 \vdash_M \ldots \vdash_M \alpha_t$ und t ist Stoppkonfiguration von M.

Der Satz von Cook und Levin

- Falls wir eine Stoppkonfiguration erreichen, so propagieren wir diese Konfiguration bis zum Zeitpunkt $p(n)$. Dafür benötigen wir die Formel

$$\psi_{\text{propagate}} := \bigwedge_{\substack{t,i,a,p \\ \text{es ex. kein}(p,a,q,b,m)\in\Delta}} \Big(\big(B_{a,i,t} \wedge K_{i,t} \wedge Z_{p,t}\big) \rightarrow \bigwedge_{t<t'\leq p(n)} \big(B_{a,i,t'} \wedge K_{i,t'} \wedge Z_{p,t'}\big) \Big),$$

Der Satz von Cook und Levin

Lemma

Sei $\mathfrak{I}$ eine Belegung mit
$\mathfrak{I} \vDash \psi_{\text{config}} \wedge \psi_{\text{init}} \wedge \psi_{\text{compute}} \wedge \psi_{\text{no-change}} \wedge \psi_{\text{propagate}}$.

Sei $\alpha_1, \ldots, \alpha_{p(|w|)}$ die eindeutig von $\mathfrak{I}$ bestimmte Folge von Konfigurationen von M.

Sei t so dass $\alpha_1 \vdash_M \ldots \vdash_M \alpha_t$ und α_t ist Stoppkonfiguration von M.

Dann gilt $\alpha_t = \alpha_{t+1} = \ldots = \alpha_{p(|w|)}$.

Der Satz von Cook und Levin

- Schließlich stellen wir sicher, dass die Stoppkonfiguration von M auf w eine akzeptierende Stoppkonfiguration ist.

$$\psi_{\text{acc}} := \bigvee_{q \in F} Z_{q,p(|w|)}.$$

Lemma

Sei $\mathfrak{I}$ eine Belegung mit

$$\mathfrak{I} \vDash \psi_{\text{config}} \wedge \psi_{\text{init}} \wedge \psi_{\text{compute}} \wedge \psi_{\text{no-change}} \wedge \psi_{\textit{propagate}} \wedge \psi_{\text{acc}} =: \varphi_w.$$

Dann akzeptiert M das Wort w.

Der Satz von Cook und Levin

- M akzeptiert die Eingabe w genau dann, wenn φ_w erfüllbar ist:

  „$\Rightarrow$“ Wenn w von M akzeptiert wird, dann gibt es eine akzeptierende Berechnung

  $$\alpha_1 \vdash_M \alpha_2 \vdash_M \cdots \vdash_M \alpha_t$$

  von M auf w. Wenn $t < p(|w|)$, so betrachte die Folge $\alpha_1, \ldots, \alpha_t, \ldots, \alpha_t$.

  Daraus lesen wir eine erfüllende Belegung $\mathfrak{I}$ für die Variablen $B_{a,i,t}$, $K_{i,t}$ und $Z_{q,t}$ ab.

  „$\Leftarrow$“ Dies ist die Aussage des letzten Lemmas.

- φ_w kann in Polynomialzeit aus w konstruiert werden. □

NP-Schwere beweisen

Lemma

Ist L_1 NP-schwer und gilt $L_1 \leq_p L_2$, so ist auch L_2 NP-schwer.

Beweis.

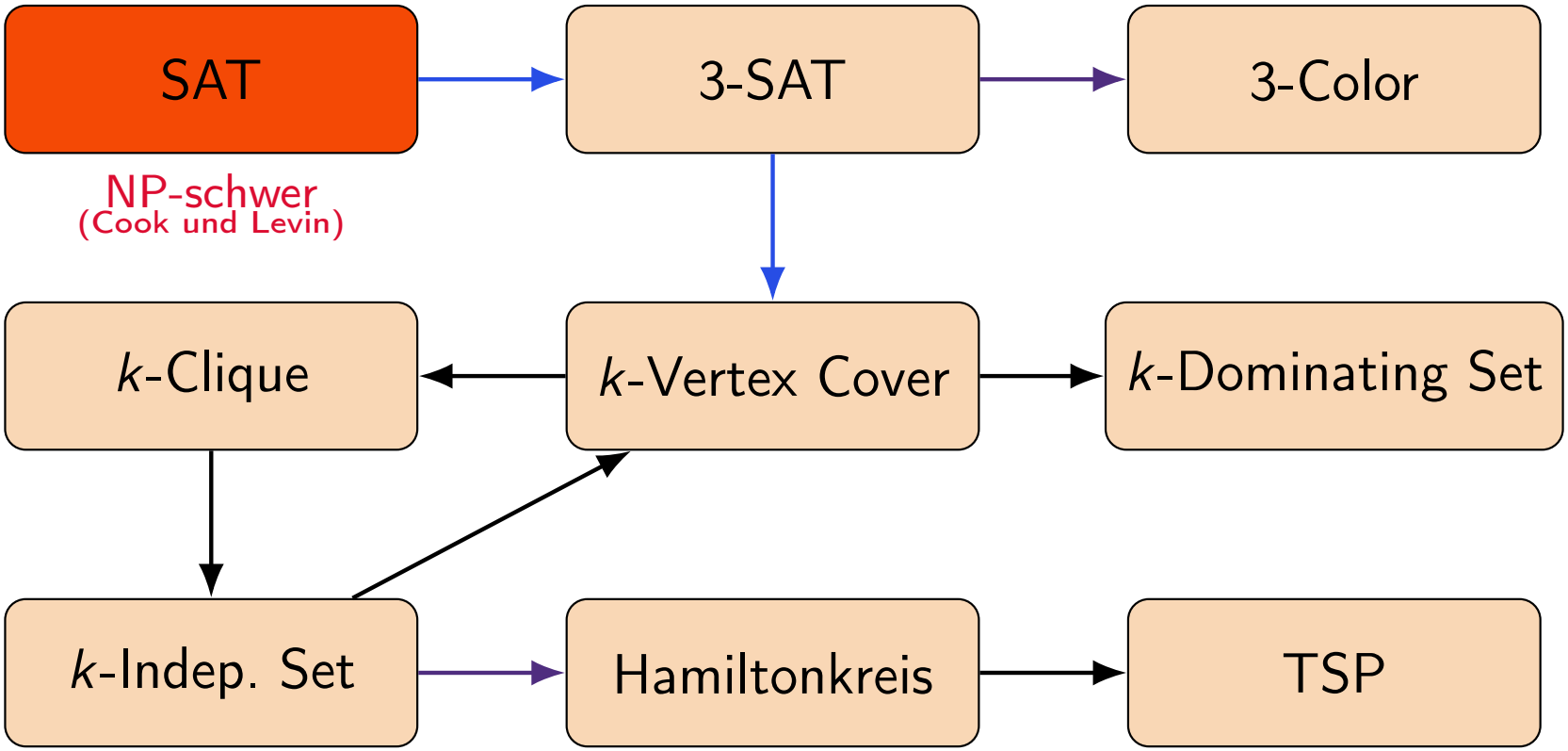
- Sei $L \in \text{NP}$.
- Da L_1 NP-schwer ist, gilt $L \leq_p L_1$.
- Mit $L_1 \leq_p L_2$ folgt $L \leq_p \leq L_1 \leq_p L_2$, also $L \leq_p L_2$.

Korollar

Um zu zeigen, dass L NP-schwer ist genügt es zu zeigen, dass

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

NP-vollständige Probleme



3-SAT – Konjunktive Normalform

- Literal L: Variable X oder negierte Variable $\neg X$.
- Disjunktion von Literalen:

$$\left(\bigvee_j L_{ij}\right)$$

- Eine Formel der Aussagenlogik ist in konjunktiver Normalform (KNF), wenn sie eine Konjunktion von Disjunktionen von Literalen ist:

$$\bigwedge_i \left(\bigvee_j L_{ij}\right),$$

  wobei jedes L_{ij} eine Variable oder negierte Variable ist.
- Die Disjunktionen $\bigvee_j L_{ij}$ werden Klauseln genannt.
- Eine Formel ist in q-KNF, wenn jede Klausel höchstens q Literale enthält.

3-SAT ist NP-vollständig

Satz

Das Erfüllbarkeitsproblem für aussagenlogische Formeln in 3-KNF, genannt 3-SAT, ist NP-vollständig.

Beweis.

- Wir zeigen SAT $\leq_p$ 3SAT.
- Dazu zeigen wir: jede aussagenlogische Formel φ kann in Polynomialzeit in eine erfüllbarkeitsäquivalente Formel ψ in 3-KNF umgewandelt werden.
  - φ und ψ erfüllbarkeitsäquivalent: φ erfüllbar $\Leftrightarrow$ ψ erfüllbar.
- Jede Formel ist sogar äquivalent zu einer Formel in KNF, aber diese Formel kann exponentielle Länge haben, insbesondere würde die Berechnung exponentielle Zeit benötigen.

Tseitin-Transformation

- Tseitin-Transformation: Beispiel $\varphi = \neg(x \wedge y) \vee \neg z$
- Schritt 1: Einführen von neuen Variablen für nicht-atomare Teilformeln
  - x_1 für $x \wedge y$
  - x_2 für $\neg(x \wedge y)$
  - x_3 für $\neg z$
  - x_4 für $\neg(x \wedge y) \vee \neg z$
- Schritt 2: Bedeutung der neuen Variablen festlegen und Teilformeln ersetzen
  - $x_1 \leftrightarrow x \wedge y$
  - $x_2 \leftrightarrow \neg x_1$
  - $x_3 \leftrightarrow \neg z$
  - $x_4 \leftrightarrow x_2 \vee x_3$

Tseitin-Transformation

- Schritt 3: Konjunktion bilden
  - $x_4 \wedge (x_1 \leftrightarrow x \wedge y) \wedge (x_2 \leftrightarrow \neg x_1) \wedge (x_3 \leftrightarrow \neg z) \wedge (x_4 \leftrightarrow x_2 \vee x_3)$
- Schritt 4: Konjunkte in KNF umformen (zum Beispiel mittels Wahrheitstafel).
- In jedem Konjunkt tauchen höchstens 3 Literale auf $\rightarrow$ die Formel ist in 3-KNF.
- Größe der konstruierten Formel ψ:
  - Anzahl der Konjunkte = Anzahl Teilformeln von $\varphi + 1$.
  - Jedes Konjunkt hat 3 Variablen, erzeugt also ≤ 8 Klauseln.
  - Insgesamt also Größe $\leq 8 \cdot (\varphi + 1) \in \mathcal{O}(n)$.

Tseitin-Transformation

Satz

Wenn ψ aus φ durch die Tseitin-Transformation hervorgeht, dann sind φ und ψ erfüllbarkeitsäquivalent.

Beweis.

- Bezeichne mit x_ϑ die für Teilformel ϑ eingeführte Variable.
- Angenommen ψ ist erfüllbar mit Modell $\mathfrak{I}$.
- Zeige per Induktion über die Teilformeln $\vartheta \in TF(\varphi)$:

$$\text{Wenn } \mathfrak{I}(x_\vartheta) = 1, \text{ dann } \mathfrak{I} \vDash \vartheta.$$

- Da x_φ Konjunkt von ψ, gilt $\mathfrak{I}(x_\varphi) = 1$, also $\mathfrak{I} \vDash \varphi$.

Tseitin-Transformation

Beweis fortgesetzt.

- Angenommen φ ist erfüllbar mit Modell $\mathfrak{I}$.
- Sei $\mathfrak{I}'$ die Erweiterung von $\mathfrak{I}$ mit

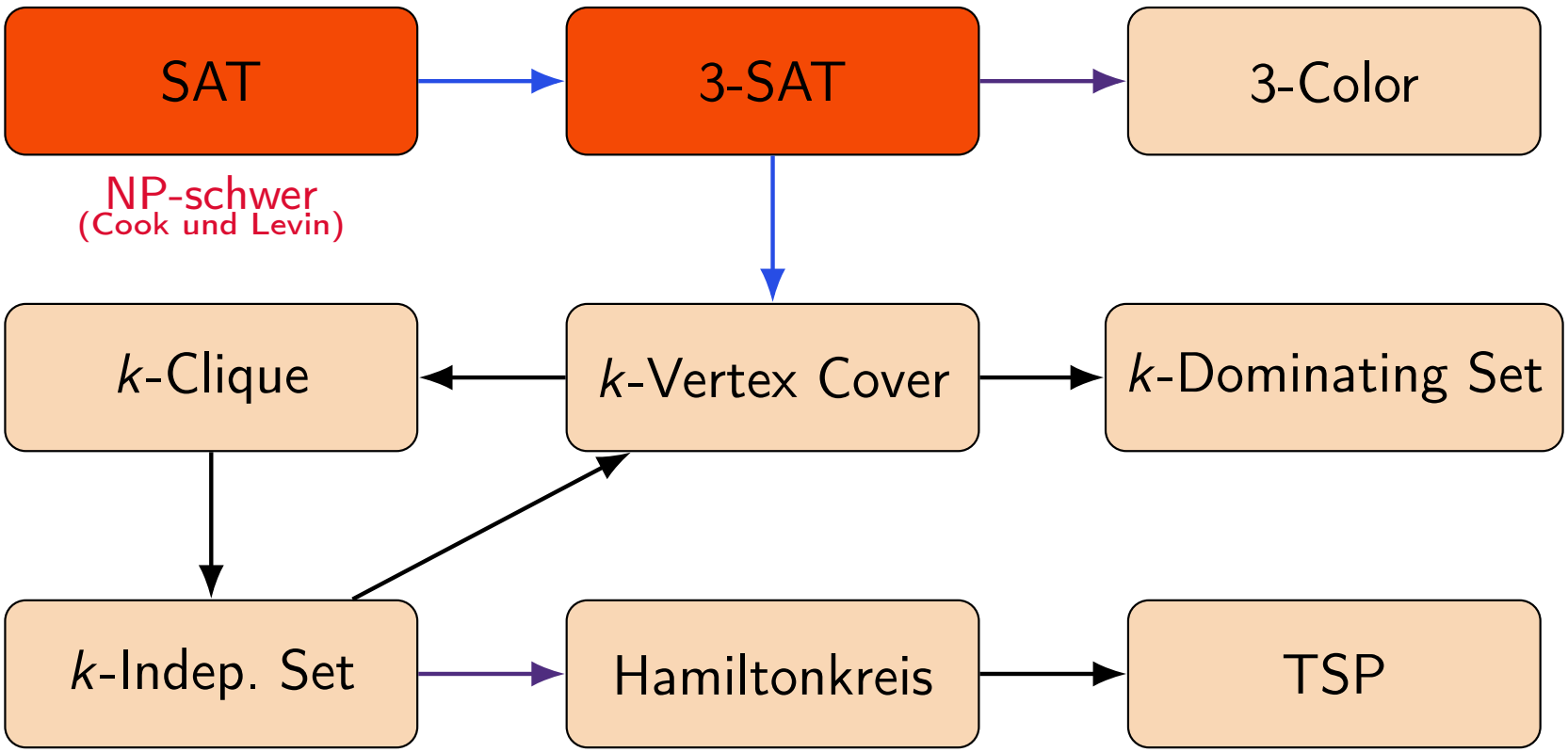
$$\mathfrak{I}'(x_\vartheta) = \begin{cases} 1 & \text{wenn } \mathfrak{I} \vDash \vartheta \\ 0 & \text{sonst.} \end{cases}$$

- Zeige per Induktion, dass $\mathfrak{I}' \vDash \psi$.

3-SAT ist NP-vollständig

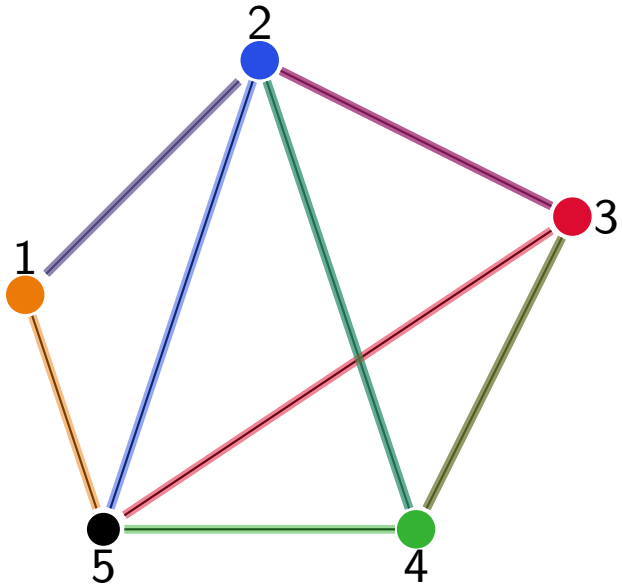
- Gezeigt: jede aussagenlogische Formel φ kann in Polynomialzeit in eine erfüllbarkeitsäquivalente Formel ψ in 3-KNF umgewandelt werden.
- Dies ist eine Polynomialzeitreduktion, also SAT $\leq_p$ 3SAT.
- Da SAT NP-schwer ist, ist auch 3-SAT NP-schwer.
- Außerdem können wir Lösungen in Polynomialzeit verifizieren, also liegt 3-SAT in NP.
- Also ist 3-SAT NP-vollständig.

NP-vollständige Probleme

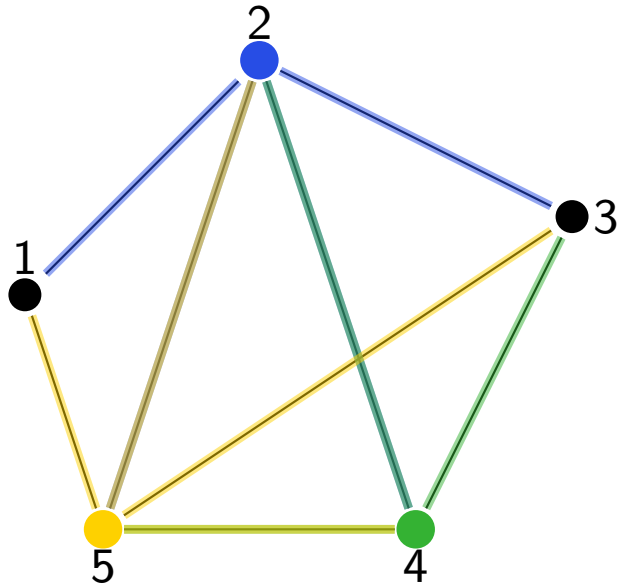


Vertex Cover

Gegeben: Graph $G = (V, E)$, natürliche Zahl k
Frage: Gibt es eine Teilmenge $U \subseteq V$ mit $|U| \leq k$ und $\{u, v\} \cap U \neq \emptyset$ für alle $\{u, v\} \in E$?



$\{1, 2, 3, 4\}$ ist VC der Größe 4



$\{2, 4, 5\}$ ist VC der Größe 3

Vertex Cover

Vertex Cover ist in NP:

- Rate Vertex Cover $C \subseteq V$ der Größe höchstens k
- Verifiziere C: Iteriere über alle Kanten $\{u, v\} \in E$ und teste, ob u oder v in C. Möglich in Zeit $\mathcal{O}(|E| \cdot k)$.

Vertex Cover ist NP-schwer: Wir zeigen 3-SAT $\leq_p$ Vertex Cover.

Vertex Cover

- Sei φ eine aussagenlogische Formel in 3-KNF

$$\varphi = \bigwedge_i (L_{i1} \vee L_{i2} \vee L_{i3}),$$

  wobei die L_{ijk} Literale sind.

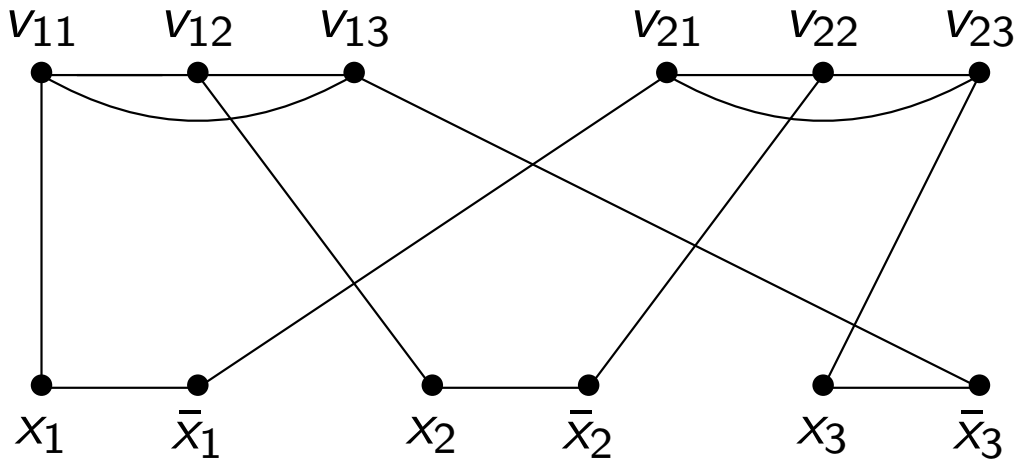
  - n Variablen,
  - c Klauseln.

- Konstruiere (in Polynomialzeit) Graphen G_φ, so dass

  G_φ hat Vertex Cover mit $n + 2c$ Knoten $\Leftrightarrow$ φ erfüllbar.

Vertex Cover

$$(X_1 \vee X_2 \vee \neg X_3) \wedge (\neg X_1 \vee \neg X_2 \vee X_3)$$



- Für jede Variable X_i aus φ füge zwei Knoten x_i und $\bar{x}_i$ hinzu und verbinde diese mit einer Kante.
- Für jede Klausel $(L_{i1} \vee L_{i2} \vee L_{i3})$ füge 3 Knoten v_{i1}, v_{i2}, v_{i3} hinzu und verbinde diese 3 Knoten miteinander.
- Verbinde v_{ik} für $1 \leq k \leq 3$ mit dem Knoten x_j, wenn $L_{ik} = X_j$ und mit dem Knoten $\bar{x}_j$, wenn $L_{ik} = \neg X_j$.

.

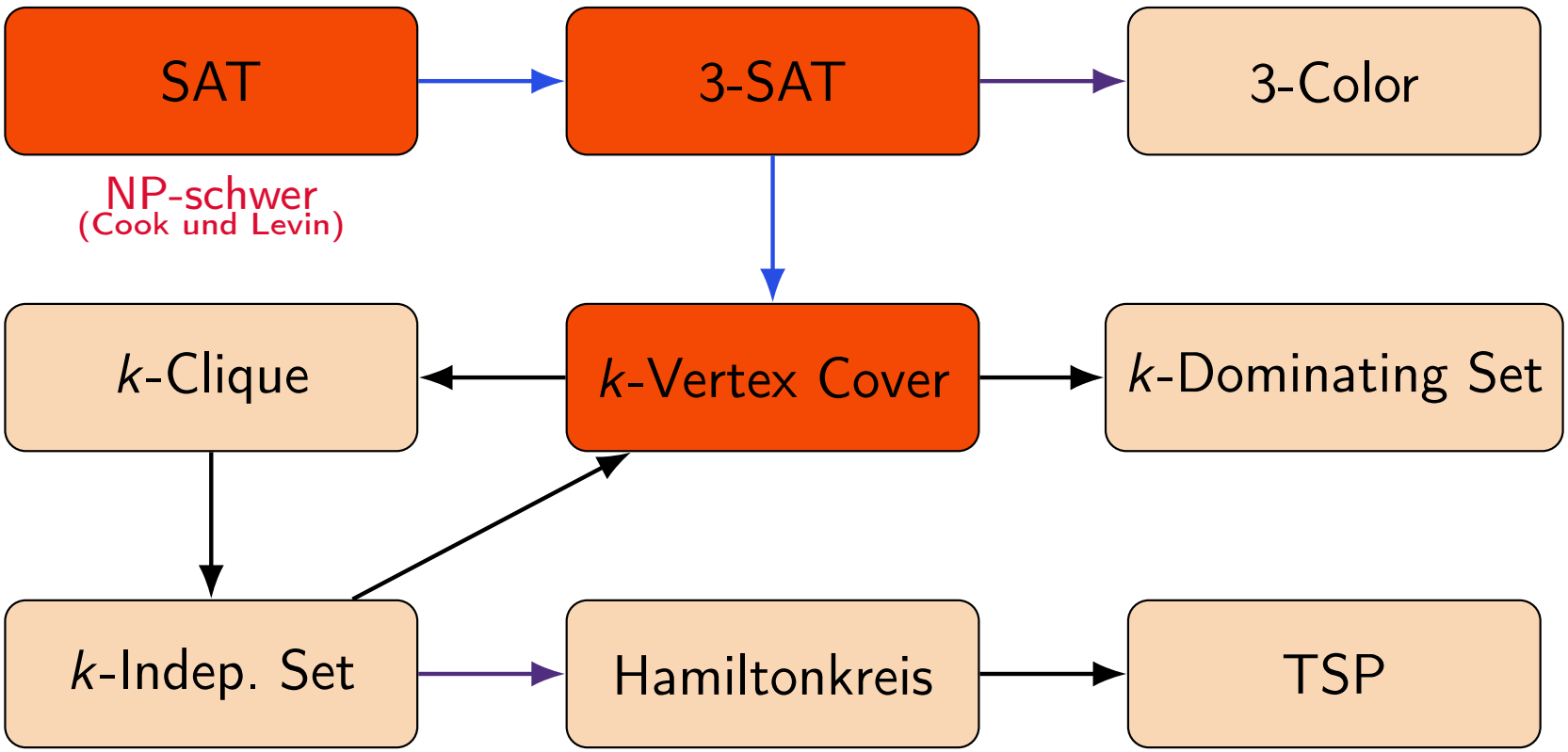
Vertex Cover

- Wir zeigen, dass φ erfüllbar ist genau dann, wenn G_φ ein Vertex Cover mit $n + 2c$ Knoten hat.
- Sei $\mathfrak{I}$ erfüllende Belegung von φ.
- Konstruiere das folgende Vertex Cover C von G_φ.
  - Falls $\mathfrak{I}(X_i) = 1$, so füge x_i zu C hinzu, sonst füge $\bar{x}_i$ hinzu (n Knoten).
  - Für jede Klausel $(L_{i1} \vee L_{i2} \vee L_{i3})$ gilt $\mathfrak{I} \vDash L_{ik}$ für mindestens ein k.
  - Falls $k = 1$, so füge v_{i2} und v_{i3} zu C hinzu, für $k = 2, 3$ entsprechend die beiden anderen Knoten (weitere $2c$ Knoten).
  - C ist ein Vertex Cover:
    - Da x_i oder $\bar{x}_i$ in C wird die Kante zwischen x_i und $\bar{x}_i$ gecovert.
    - Da v_{i2} und v_{i3} in C liegen, werden alle Kanten innerhalb der Klausel abgedeckt und außerdem die Kanten, die inzident mit v_{i2} und v_{i3} sind.
    - Die weitere Kante, die mit v_{i1} inzident ist wird von x_i oder $\bar{x}_i$ abgedeckt.

Vertex Cover

- Sei umgekehrt C eine Vertex Cover der Größe $n + 2c$ von G_φ.
- Für jedes j muss die Kante zwischen x_j und $\bar{x}_j$ abgedeckt werden, also muss x_j oder $\bar{x}_j$ in C liegen.
- Außerdem müssen für jede Klausel $(L_{i1} \vee L_{i2} \vee L_{i3})$ mindestens 2 Knoten v_{ik} und $v_{i\ell}$ in C liegen, um die Kanten im Dreieck dieser Knoten abzudecken.
- Da $|C| = n + 2c$ folgt, das für jede Klausel $(L_{i1} \vee L_{i2} \vee L_{i3})$ genau 2 Knoten v_{ik} und $v_{i\ell}$ in C liegen und für jedes j genau einer der Knoten x_j oder $\bar{x}_j$.
  - Angenommen es liegen v_{i2} und v_{i3} in C und $\{v_{i1}, x_j\} \in E(G_\varphi)$. Dann wird $\{v_{i1}, x_j\}$ von x_j abgedeckt.
- Also ist $\mathfrak{I}$ mit $\mathfrak{I}(X_j) = 1 \Leftrightarrow x_j \in C$ eine erfüllende Belegung von φ ist.

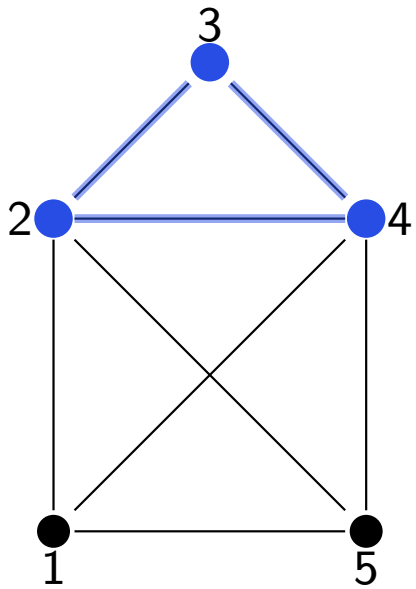
NP-vollständige Probleme



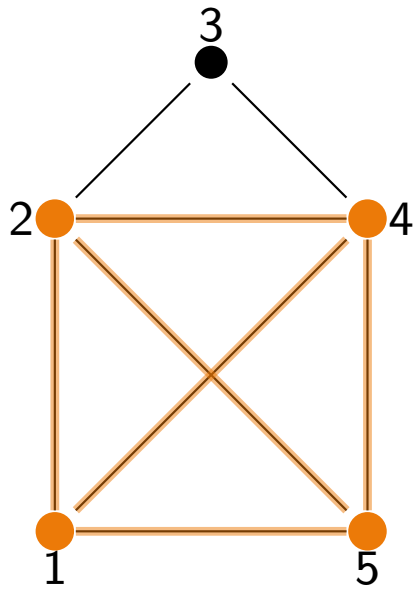
Definition: Clique

Gegeben: Graph $G = (V, E)$, natürliche Zahl k

Frage: Gibt es eine Teilmenge $C \subseteq V$ mit $|C| \geq k$ und für alle $u, v \in U$ gilt $\{u, v\} \in E$?



$\{2, 3, 4\}$ ist Clique der Größe 3



$\{1, 2, 4, 5\}$ ist Clique der Größe 4

- Wollen zeigen: Clique ist NP-vollständig.

Clique ist NP-vollständig

Clique ist in NP:

- Rate Clique $C \subseteq V$ der Größe mindestens k
- Verifiziere C: Iteriere über alle Knotenpaare $u, v \in C$ und teste, ob $\{u, v\} \in E$. Möglich in Zeit $\mathcal{O}(|C|^2) \subseteq \mathcal{O}(|V|^2)$

Clique ist NP-schwer: Vertex Cover $\leq_p$ Clique

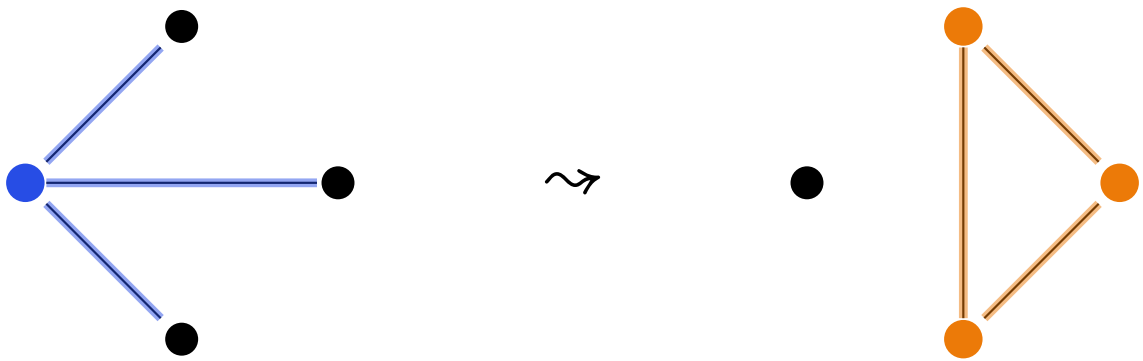
- Sei $G = (V, E)$ ein Graph. Wir definieren das Komplement von G als $\bar{G} = (V, \binom{V}{2} \setminus E)$; d. h., wir flippen alle Kanten
- Folgende Funktion f ist die gesuchte Reduktion:

$$f(G, k) = (\bar{G}, |V| - k)$$

Korrektheitsbeweis der Reduktion

Wollen zeigen: $f(G,k) = (\bar{G}, |V| - k)$ ist eine Polynomialzeitreduktion von Vertex Cover auf Clique.

- Sei $G = (V,E)$ ein Graph und $U \subseteq V$ ein VC mit $|U| \leq k$.
- Dann verläuft keine Kante zwischen zwei Knoten in $\bar{U} = V \setminus U$ (ansonsten wäre eine solche Kante nicht abgedeckt)
- Also ist $\bar{U}$ im Komplement $\bar{G}$ vollständig vernetzt
- Da $|\bar{U}| = |V| - |U| \geq |V| - k$, ist $C = \bar{U}$ eine Clique der Größe mindestens $|V| - k$ in $\bar{G}$



Korrektheitsbeweis der Reduktion

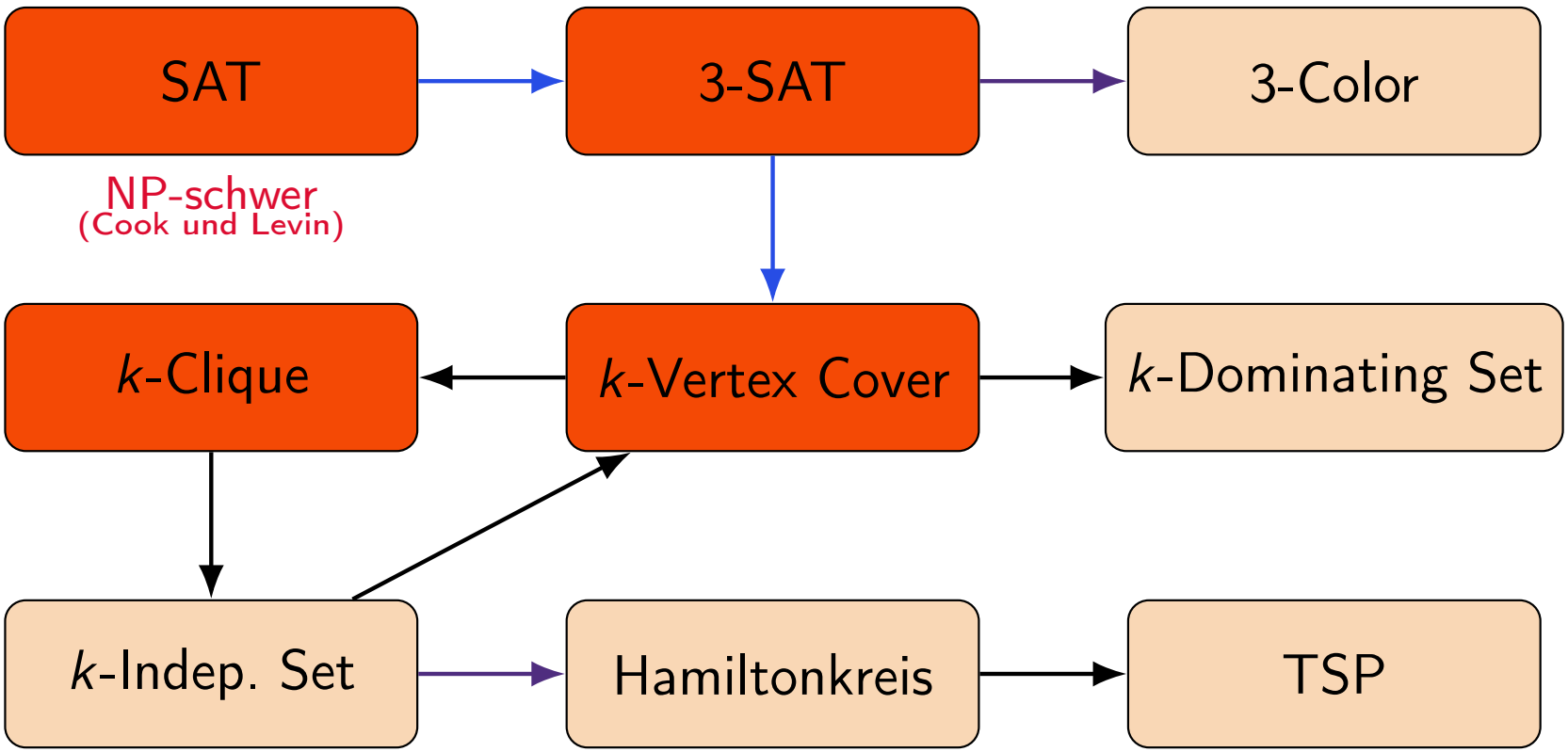
- Sei nun C eine Clique in $\bar{G}$ mit $|C| \geq |V| - k$.
- Dann verläuft in G keine Kante zwischen zwei Knoten in C.
- Kein Knoten aus C ist notwendig, um alle Kanten in G abzudecken (jede hat min. einen Endpunkt außerhalb von C).
- Alle anderen Knoten bilden also ein Vertex Cover. Die Menge $U = V \setminus C$ ist ein VC in G mit $|U| \leq |V| - (|V| - k) = k$.

Ferner ist f in Polynomialzeit berechenbar:

- Berechne $\bar{G}$: Iteration über alle $\mathcal{O}(|V|^2)$ Knotenpaare
- Berechne $|V| - k$: in konstanter Zeit möglich

▸ Clique ist NP-vollständig

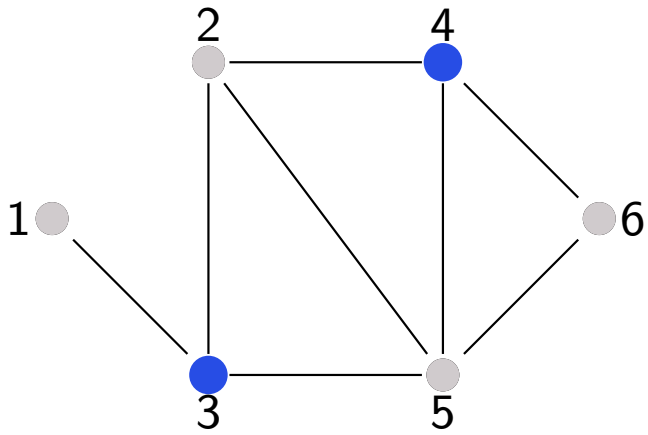
NP-vollständige Probleme



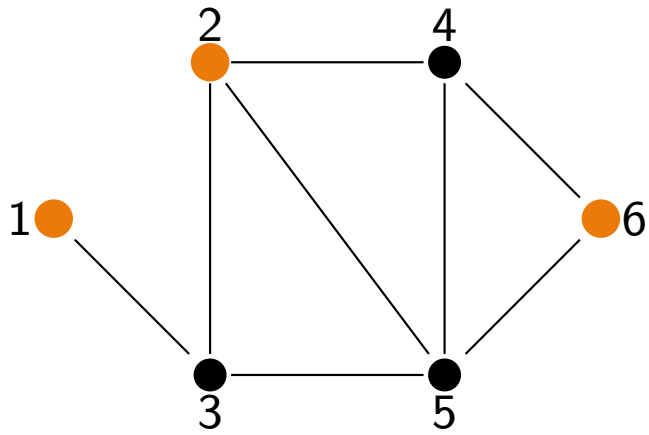
Definition: Independent Set

Gegeben: Graph $G = (V, E)$, natürliche Zahl k

Frage: Gibt es eine Teilmenge $I \subseteq V$ mit $|I| \geq k$ und für alle $u, v \in I$ gilt $\{u, v\} \notin E$?



$\{3, 4\}$ ist
IS der Größe 2



$\{1, 2, 6\}$ ist
IS der Größe 3

- Wollen zeigen: Independent Set ist NP-vollständig.

Independent Set ist NP-vollständig

Independent Set ist in NP (sehr ähnlich zu Clique):

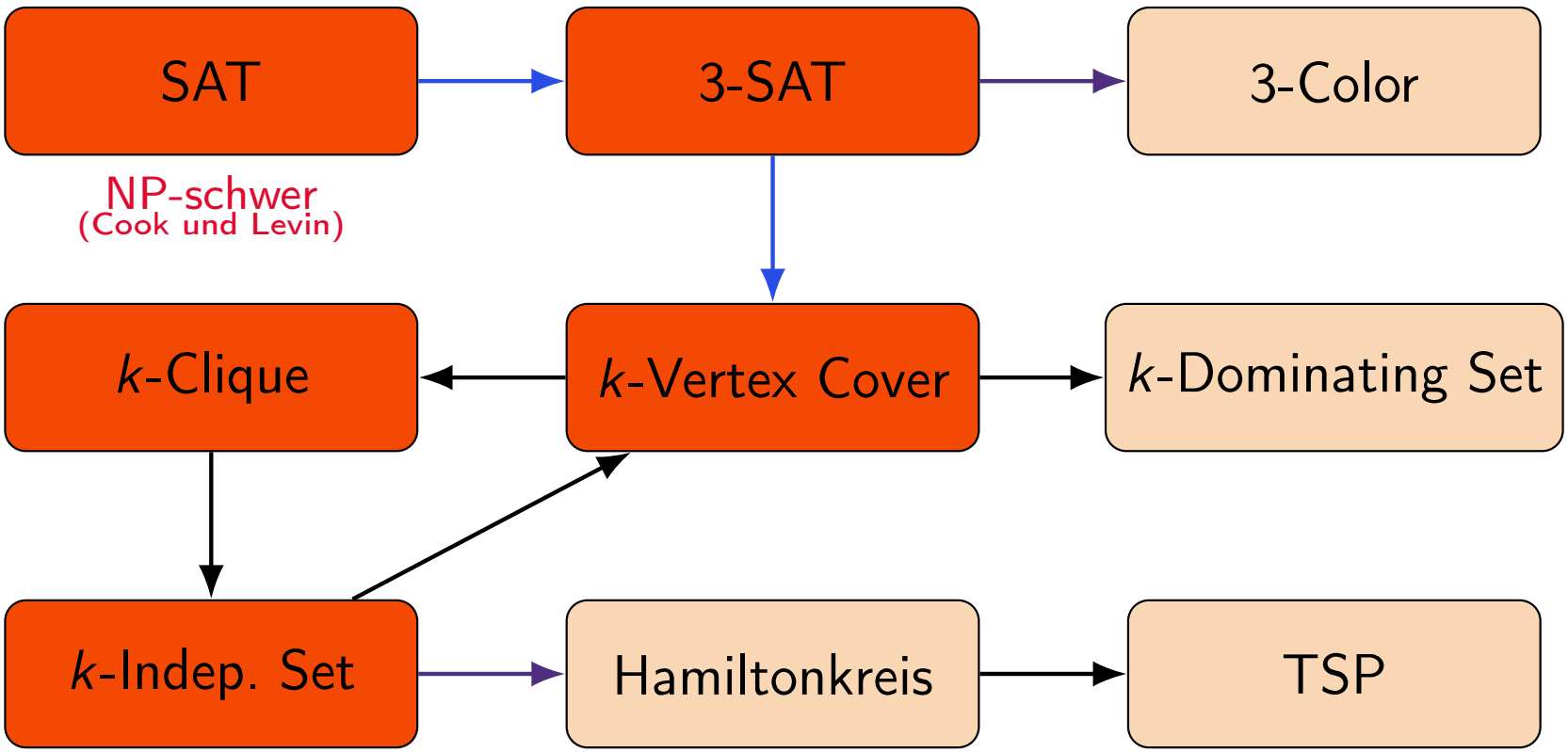
- Rate Independent Set $I \subseteq V$ der Größe mindestens k
- Verifiziere I: Iteriere über alle Knotenpaare $u, v \in I$ und teste, ob $\{u, v\} \notin E$. Dies ist auch in $\mathcal{O}(|I|^2) \subseteq \mathcal{O}(|V|^2)$ machbar

Independent Set ist NP-schwer: Clique $\leq_p$ Independent Set

- Jede Clique in G ist ein Independent Set in $\bar{G}$
- Folgende Funktion f ist also die gesuchte Reduktion:

$$f(G, k) = (\bar{G}, k)$$

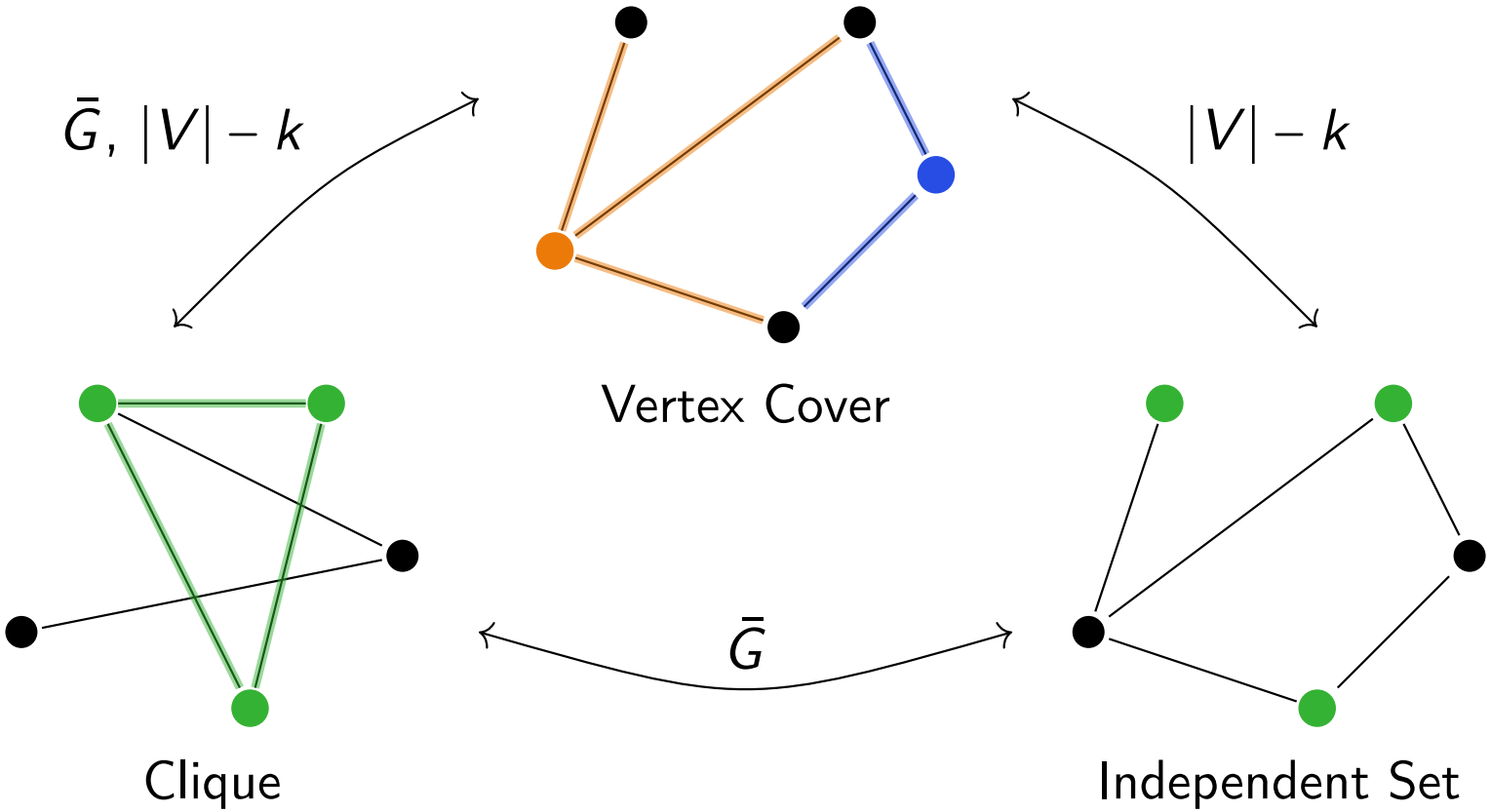
NP-vollständige Probleme



Vertex Cover, Clique und Independent Set

Der Korrektheitsbeweis ist analog zu Vertex Cover $\leq_p$ Clique

- All diese Probleme sind wechselseitig aufeinander reduzierbar

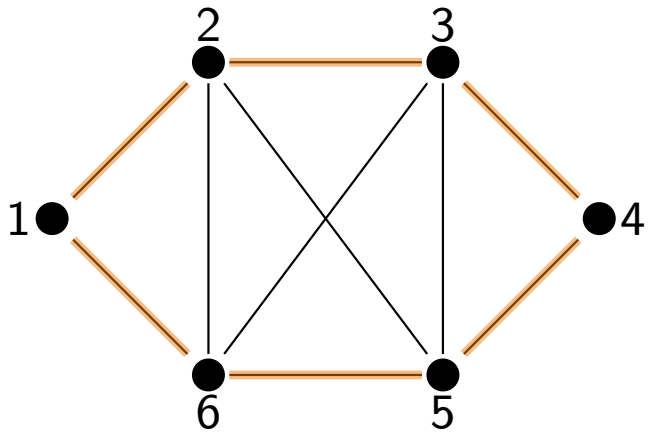


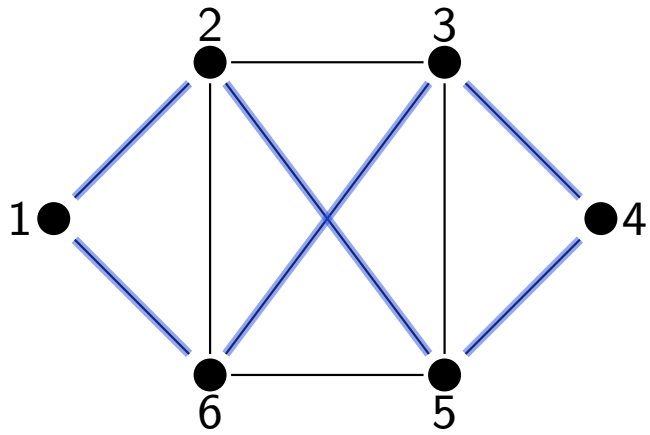
- All diese Probleme sind NP-vollständig

Definition: Hamiltonkreis

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

Frage: Gibt es in G einen Hamiltonkreis, also einen Rundweg, der jeden Knoten genau einmal besucht?





Wollen zeigen: Hamiltonkreis ist NP-vollständig

- Nicht zu verwechseln mit dem Eulerkreisproblem (in P)

Hamiltonkreis ist NP-vollständig

Hamiltonkreis ist in NP:

- Rate Hamiltonkreis $v_1 v_2 \dots v_n v_1$ in G.
- Verifikationsschritte:
  - Existieren Kanten $\{v_n, v_1\}$ und $\{v_i, v_{i+1}\}$ für alle $i < n$ in G?
  - Wurde jeder Knoten in G genau einmal besucht?

Verifikation ist (deterministisch) in Zeit $\mathcal{O}(n+m)$ möglich

Der NP-Schwere-Beweis ist deutlich schwieriger

- Wir zeigen Independent Set $\leq_p$ Hamiltonkreis
  (eigentlich Vertex Cover $\leq_p$ Hamiltonkreis)

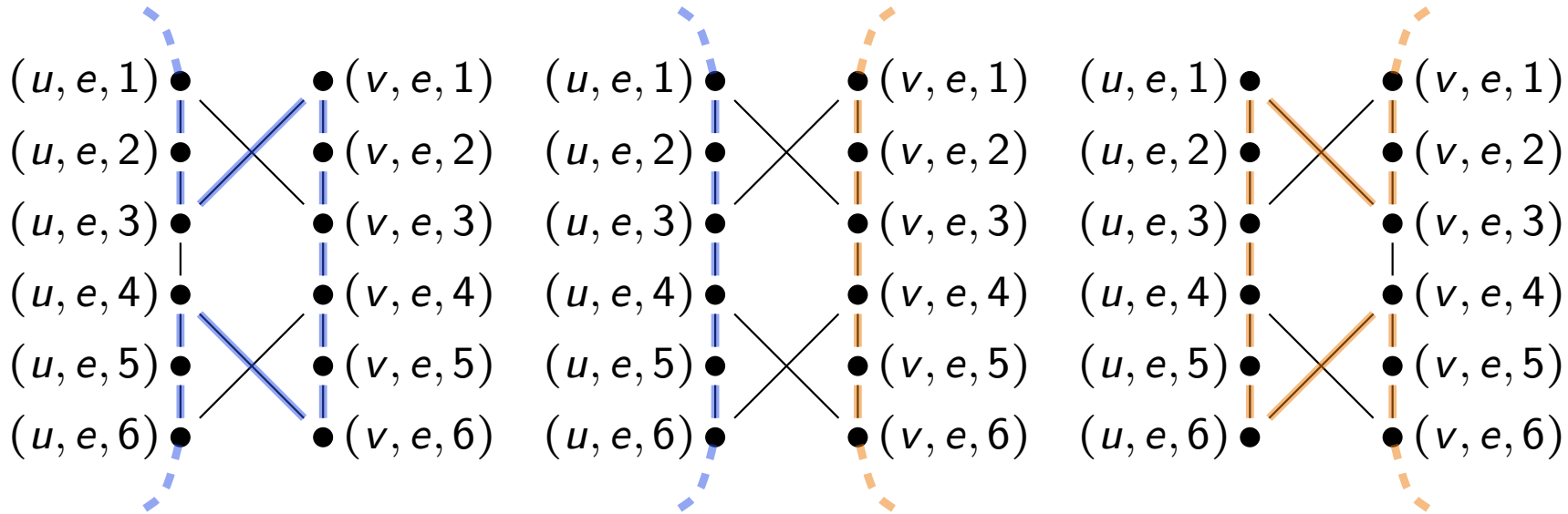
Schritte der Reduktion

Sei $(G = (V, E), k)$ eine Independent Set-Instanz

- $(G, |V| - k)$ ist äquivalente Vertex Cover-Instanz

Schritt 1: Erstelle $k' = |V| - k$ „Selektor-Knoten“ $s_1, \ldots, s_{k'}$

Schritt 2: Erstelle folgendes Gadget für jede Kante $e = \{u, v\} \in E$



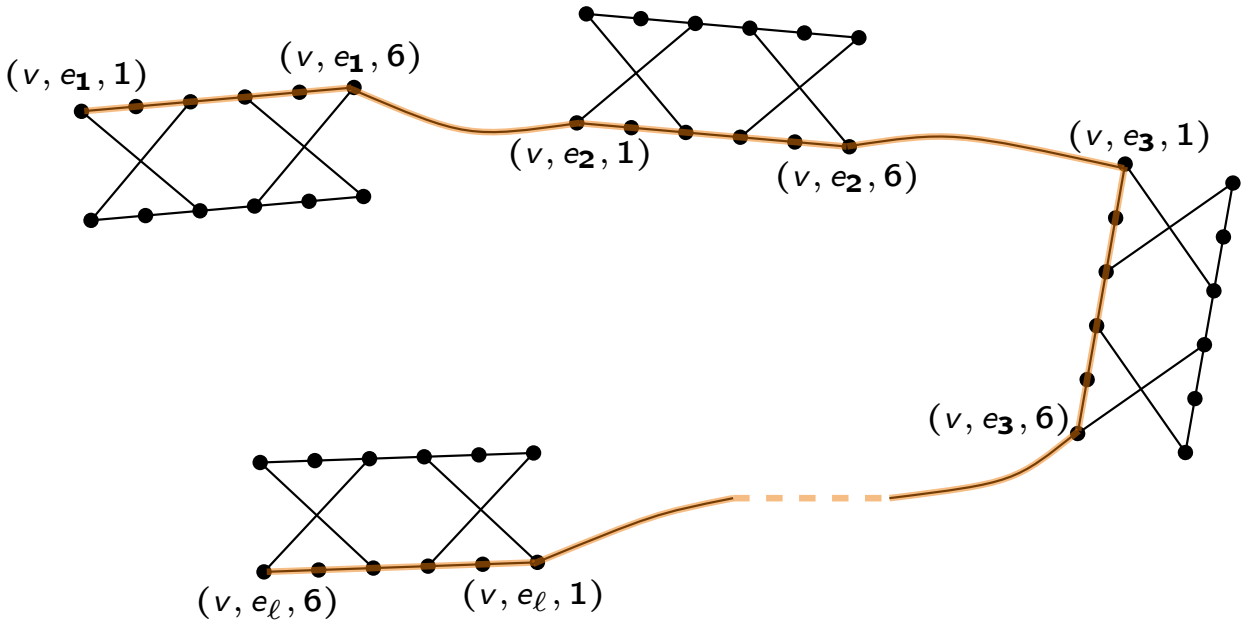
Es gibt nur drei Möglichkeiten, dieses Gadget zu traversieren

- Möglichkeiten $\{u, v\}$ zu covern: nur u, beide, oder nur v

Schritte der Reduktion

Schritt 3: Gadgets verbinden: für jeden Knoten $v \in V$

- betrachte alle zu v inzidenten Kanten $e_1, \ldots, e_\ell$
- verbinde Knoten $(v, e_i, 6)$ und $(v, e_{i+1}, 1)$ für alle $i < \ell$
- Pfad über Gadgets aller Kanten, die v überdeckt



Schritte der Reduktion

Schritt 4: Selektor-Knoten mit Gadgets verbinden

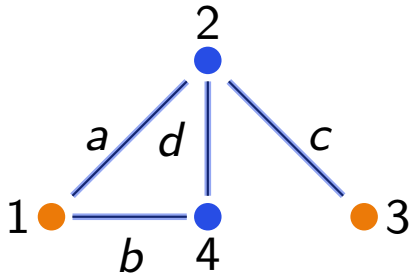
- Verbinde jeden Selektor-Knoten s_i...
- ...mit jedem Startknoten $(v, e_1, 1)$ aller Pfade
- ...mit jedem Endknoten $(v, e_\ell, 6)$ aller Pfade

Insgesamt haben wir also

- $|V| - k$ viele Selektorknoten s_i und ein Gadget pro Kante
- Kanten $\{(v, e_i, 6), (v, e_{i+1}, 1)\}$ für alle $i < \ell$
- Kanten $\{s_i, (v, e_1, 1)\}, \{s_i, (v, e_\ell, 6)\}$ für alle $i \leq k', v \in V$

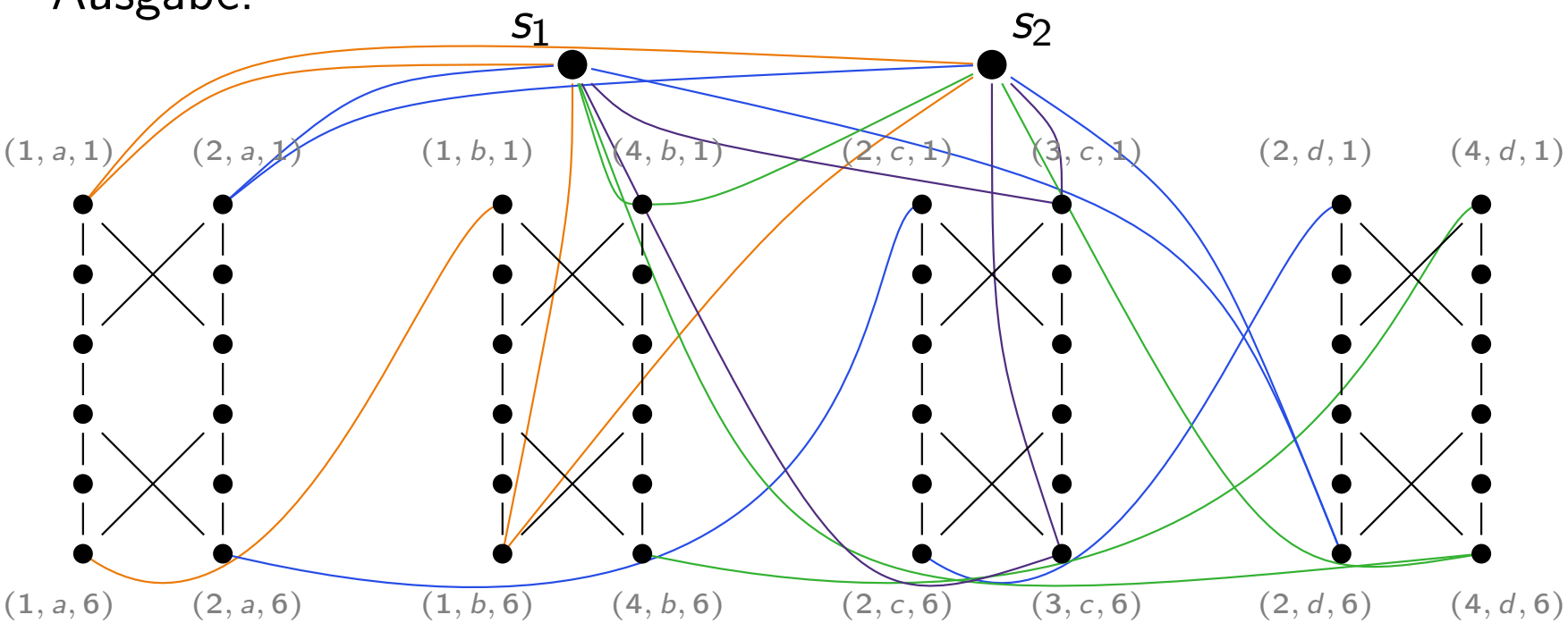
Beispiel

Eingabe:
$G, k = 2$



(Orange: IS der Größe 2)
(Blau: VC der Größe 4 – 2)

Ausgabe:



Korrektheit der Reduktion

Sei G ein Graph mit IS der Größe k (VC der Größe $k' = |V| - k$)

- Jede Kante $e = \{u, v\}$ ist gecovert. Wähle für den Hamiltonkreis Kanten aus dem Gadget für e abhängig davon, ob e nur durch u, nur durch v, oder u und v abgedeckt wird
- Wähle auch alle Kanten $\{(v, e_i, 6), (v, e_{i+1}, 1)\}$ für alle $i < \ell$
- Schließen den Hamiltonkreis:
  - Sei $U = \{u_1, \ldots, u_{k'}\}$ ein VC von G der Größe k'
  - Wählen schließlich Kanten $\{s_i, (u_i, e_1, 1)\}$ für alle $i \leq k'$ und $\{(u_{k'}, e_\ell, 6), s_1\}, \{(u_i, e_\ell, 6), s_{i+1}\}$ für alle $i < k'$
- Ausgabe enthält Hamiltonkreis

Korrektheit der Reduktion

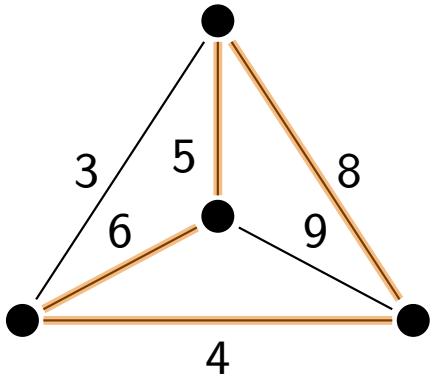
Sei $v_1 v_2 \dots v_n v_1$ ein Hamiltonkreis in der Ausgabe

- Sei $s_i \dots s_j$ Teilpfad mit genau zwei Selektorknoten s_i, s_j
- Betrachten alle besuchten Kanten-Gadgets in diesem Teilpfad
  - Entsprechen Kanten, die inzident zu einem Knoten v sind
  - Kein anderes Gadget liegt auf dem Pfad
  - Es gibt genau drei Möglichkeiten, jedes Gadget zu durchlaufen
- Es gibt genau k'-viele Teilpfade dieser Form
- Jeder Teilpfad korrespondiert zu allen Kanten, die durch einen Knoten v abgedeckt werden; alle Kanten werden abgedeckt
- Jene Knoten bilden ein VC der Größe k'
- Hamiltonkreis ist NP-vollständig

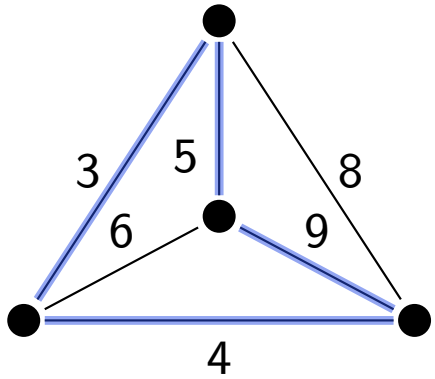
Definition: Travelling Salesperson Problem (TSP)

Gegeben: Graph $G = (V, E)$ mit Kantengewichten $w : E \to \mathbb{N}$, natürliche Zahl k

Frage: Gibt es in G einen Hamiltonkreis $v_1 v_2 \dots v_n v_1$ mit $w(v_n, v_1) + \sum_{i<n} w(v_i, v_{i+1}) \leq k$?



Summe: 23



Summe: 21

- Wollen zeigen: TSP ist NP-vollständig
  - Sogar unter massiven Einschränkungen...

TSP ist NP-vollständig

TSP ist in NP:

- Rate Hamiltonkreis $v_1 v_2 \dots v_n v_1$ in G
- Verifikation wie beim Hamiltonkreis
- Prüfe zusätzlich, ob $w(v_n, v_1) + \sum_{i<n} w(v_i, v_{i+1}) \leq k$

NP-Schwere folgt direkt per Reduktion von Hamiltonkreis

Zeigen stärkeres Resultat. TSP ist sogar NP-schwer, wenn...

- G ein vollständiger Graph ist (enthält alle möglichen Kanten)
- Nur die Gewichte 1 und 2 verwendet werden

Reduktion von Hamiltonkreis

Betrachten folgende Reduktion von Hamiltonkreis auf TSP.

- Sei $G = (V, E)$ ein Graph. Konstruiere TSP-Instanz (G', k):
- G' ist vollständiger Graph auf derselben Knotenmenge wie G
- Setzen Gewichte wie folgt: $w(e) = 1$, falls e in E vorhanden, sonst $w(e) = 2$. Wir wählen $k = |V|$

Nun gilt:

- G' ist vollständig und nutzt nur Gewichte 1 und 2.
- G hat Hamiltonkreis gdw. G' Hamiltonkreis mit Gesamtgewicht höchstens k hat

Korrektheit der Reduktion

Sei $G = (V, E)$ ein Graph mit Hamiltonkreis $v_1 v_2 \dots v_n v_1$

- Insbesondere gibt es in E Kanten v_1, v_2, $v_2, v_3 \dots$
- In G' haben all diese Kanten Gewicht 1. Also ist $v_1 v_2 \dots v_n v_1$ ein Kreis mit Gesamtgewicht $|V|$.

Sei nun $v_1 v_2 \dots v_n v_1$ ein Kreis in G' mit Gewicht höchstens $|V|$

- Der Kreis nutzt nur Kanten mit Gewicht 1
- Jene entsprechen genau denen, die auch in E
- Kreis $v_1 v_2 \dots v_n v_1$ ist auch Hamiltonkreis in G

- TSP ist NP-vollständig

Nachwort zum TSP

- Das TSP modelliert viele natürliche Probleme
- Leider selbst sehr eingeschränkt schon NP-schwer
- Noch schlimmer: im Allgemeinen auch nicht approximierbar

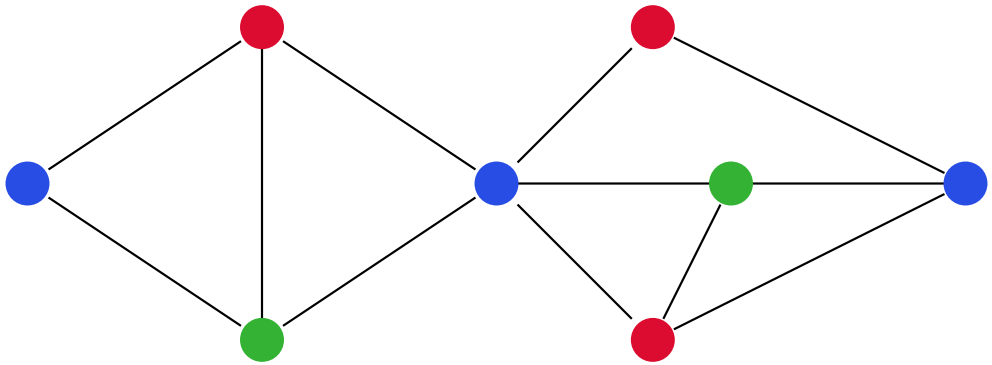
- In der Praxis: Einschränkungen hilfreich
  - Zum Beispiel Einschränkung auf Euklidische Distanzen
  - allgemeiner: metrisches TSP (erlaubt $\frac{3}{2}$-Approximation)

- Auch andere Zielfunktionen werden betrachtet
  - MaxTSP, Bottleneck TSP, Maximum Scatter TSP, ...
  - Im Allgemeinen natürlich auch NP-schwer

Definition: 3-Color

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

Frage: Existiert eine Färbung $c : V \to [3]$, sodass für alle Kanten $\{u, v\} \in E$ gilt: $c(u) \neq c(v)$?



Wollen zeigen: 3-Color ist NP-schwer

- Es folgt dann direkt die NP-Schwere von k-Color (ist wie 3-Color, aber mit Färbung $c : V \to [k]$)

3-Color ist NP-vollständig

3-Color ist in NP:

- Rate Färbung $c : V \to [3]$
- Verifiziere c: Iteriere über alle Kanten $\{u, v\} \in E$ und teste, ob $c(u) \neq c(v)$ gilt. Dies ist in Zeit $\mathcal{O}(|E|)$ möglich

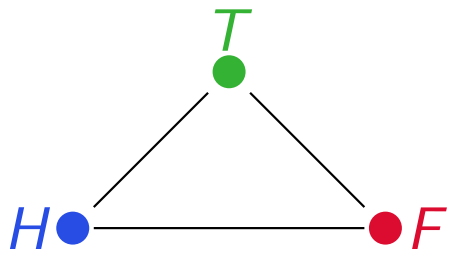
Der NP-Schwere-Beweis ist etwas schwieriger

- Wir zeigen 3-SAT $\leq_p$ 3-Color
- Dann ist 3-Color $\leq_p$ k-Color trivial

Schritte der Reduktion

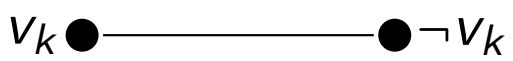
Sei $\varphi = (L_{11} \vee L_{12} \vee L_{13}) \wedge \cdots \wedge (L_{n1} \vee L_{n2} \vee L_{n3})$

Schritt 1: Erstelle eine Palette mit Knoten T, F, H:



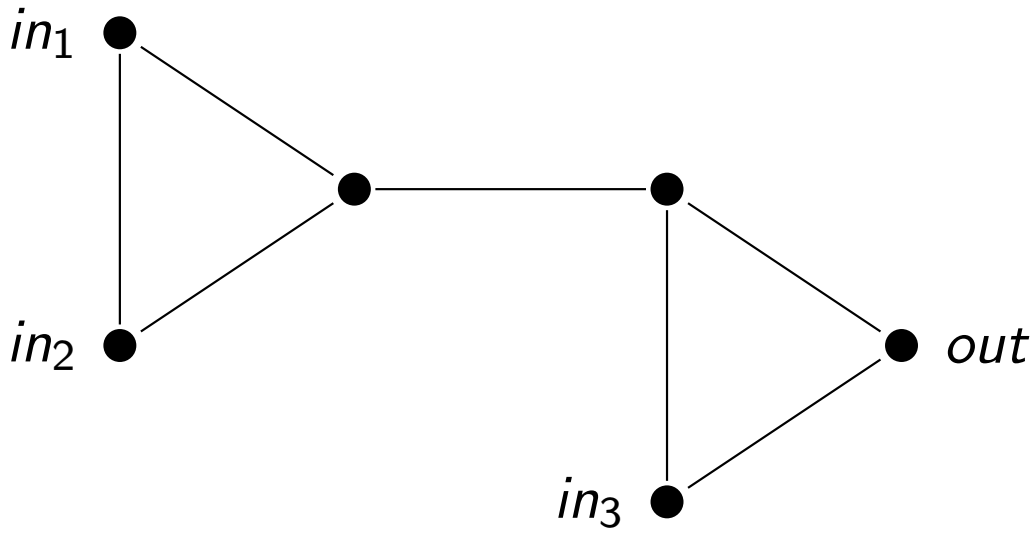
- Stellen Farben für True, False, und Help bereit

Schritt 2: Verbinde für jede Variable x_k neue Knoten v_k und $\neg v_k$:



Schritte der Reduktion

Schritt 3: Für jede Klausel $(L_{i1} \vee L_{i2} \vee L_{i3})$ ein „Oder-Gadget“:



- Verbinde in_j mit v_k, falls $L_{ij} = x_k$, und mit $\neg v_k$ sonst
- „Output“ *out* kann mit T gefärbt werden gdw. mindestens ein „Input“ (ein v_k bzw. $\neg v_k$) mit T gefärbt kann

Schritte der Reduktion

Schritt 4: Komponenten verbinden

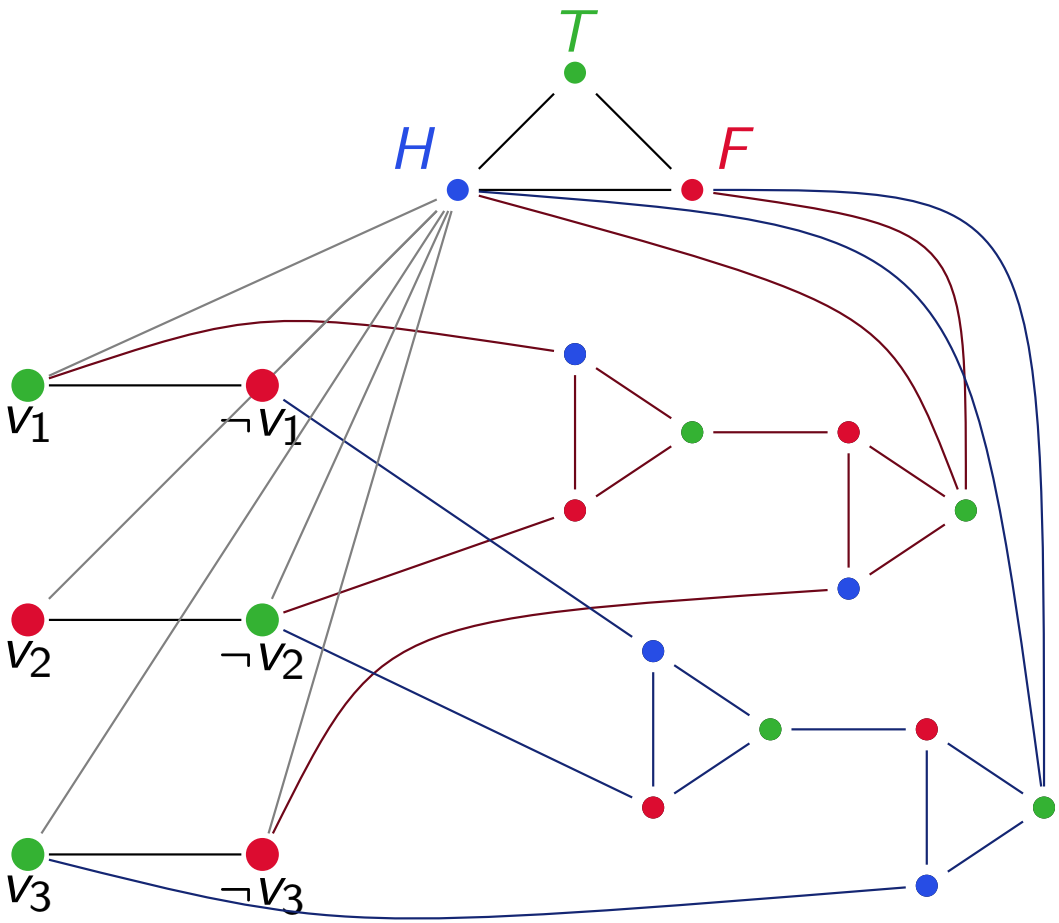
- Verbinde *in*-Knoten der Oder-Gadgets wie eben beschrieben
- Verbinde *out*-Knoten sowohl mit F als auch mit H
  - Alle Klauseln sollen wahr gemacht werden
- Verbinde alle v_k mit H und alle $\neg v_k$ mit H
  - Entweder v_k oder $\neg v_k$ ist wahr

Insgesamt haben wir also

- Palette mit Knoten T, F, H und Knoten zwei $v_k, \neg v_k$ pro x_k
- Oder-Gadgets für alle Klauseln $(L_{i1} \vee L_{i2} \vee L_{i3})$
  - Kanten $\{in_j, v_k\}$, falls $L_{ij} = x_k$, und $\{in_j, \neg v_k\}$, falls $L_{ij} = x_k$
  - Kanten $\{out, T\}, \{out, H\}$ für alle Ausgänge *out*
  - Kanten $\{v_k, H\}, \{\neg v_k, H\}$ für alle Variablen x_k

Beispiel

Eingabe: $\varphi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$



Korrektheit der Reduktion

Sei $\varphi = (L_{11} \vee L_{12} \vee L_{13}) \wedge \cdots \wedge (L_{n1} \vee L_{n2} \vee L_{n3})$ erfüllbar

- Sei $\mathfrak{I}$ eine Belegung, die φ wahr macht
- Wenn $\mathfrak{I}(x_i) = 1$, färbe v_i mit T und $\neg v_i$ mit F
- Ansonsten färbe v_i mit F und $\neg v_i$ mit T
- Nun „liegt“ an jedem Oder-Gadget ein Knoten mit Farbe T an
- Die Gadgets können so gefärbt werden, dass der Output T ist

Diese Färbung ist eine valide 3-Färbung,

- Der Ausgabegraph ist also 3-färbbar.

Korrektheit der Reduktion

Sei $c : V \to [3]$ eine Färbung des Ausgabegraphen

- Es gilt: $c(v_k) = T$ und $c(\neg v_k) = F$, oder andersherum
  - Wähle $\mathfrak{I}(x_k) = 1$, falls $c(v_k) = T$; sonst $\mathfrak{I}(x_k) = 0$
- Außerdem: alle Outputknoten der Gadgets sind mit T gefärbt, also auch mindestens ein Input pro Klausel.

Somit gilt: $\mathfrak{I}$ ist wohldefiniert und erfüllt alle Klauseln

- Also ist φ erfüllbar

- 3-Color und k-Color sind NP-vollständig

NP-vollständige Probleme