$\mathcal{NP}$-Vollständigkeit von (UN-)DIRECTED-HAMILTON-CYCLE, TRAVELLING-SALESPERSON und GRAPH-COLORING

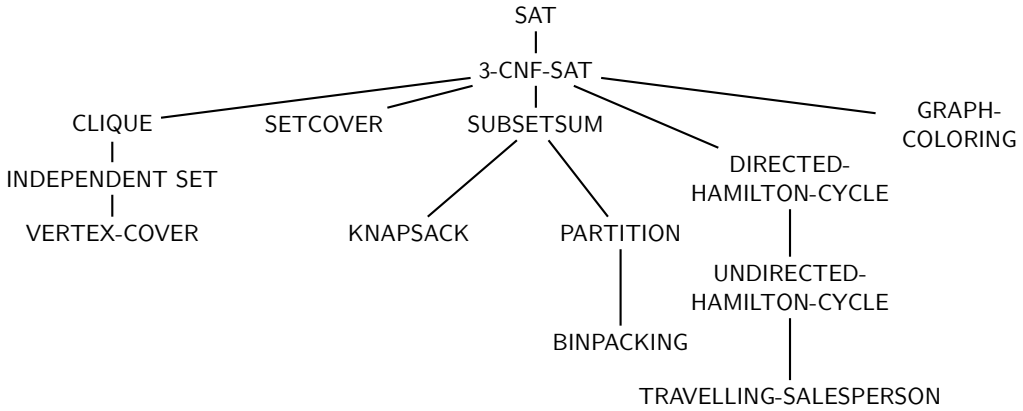
Prof. Dr. David Sabel

LFE Theoretische Informatik

Letzte Änderung der Folien: 24. Juli 2022

Inhalt der kommenden Vorlesungen

$\mathcal{NP}$-Vollständigkeitsbeweise für eine Auswahl an Problemen.



Heute: $\mathcal{NP}$-Vollständigkeit von (UN-)DIRECTED-HAMILTON-CYCLE, TRAVELLING-SALESPERSON und GRAPH-COLORING

Hamiltonische Kreise

Definition

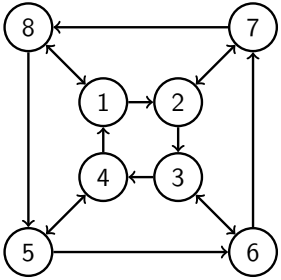
In einem gerichteten Graphen ist ein Hamilton-Kreis, ein Kreis, der genau alle Knoten einmal besucht.
Formal: Für $G = (V, E)$ mit $V = \{v_1, \ldots, v_n\}$ ist ein Hamilton-Kreis eine Permutation $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$, sodass $(v_{\pi(i)}, v_{\pi(i+1)}) \in E$ und $(v_{\pi(n)}, v_{\pi(1)}) \in E$.

Hamiltonische Kreise

Definition

In einem gerichteten Graphen ist ein Hamilton-Kreis, ein Kreis, der genau alle Knoten einmal besucht.
Formal: Für $G = (V, E)$ mit $V = \{v_1, \ldots, v_n\}$ ist ein Hamilton-Kreis eine Permutation $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$, sodass $(v_{\pi(i)}, v_{\pi(i+1)}) \in E$ und $(v_{\pi(n)}, v_{\pi(1)}) \in E$.

Beispiel:

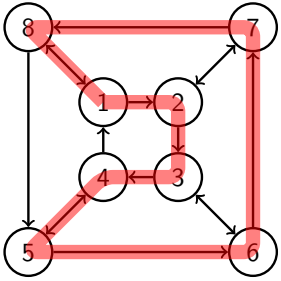


Hamiltonische Kreise

Definition

In einem gerichteten Graphen ist ein Hamilton-Kreis, ein Kreis, der genau alle Knoten einmal besucht.
Formal: Für $G = (V, E)$ mit $V = \{v_1, \ldots, v_n\}$ ist ein Hamilton-Kreis eine Permutation $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$, sodass $(v_{\pi(i)}, v_{\pi(i+1)}) \in E$ und $(v_{\pi(n)}, v_{\pi(1)}) \in E$.

Beispiel:



Das DIRECTED-HAMILTON-CYCLE-Problem

Definition (DIRECTED-HAMILTON-CYCLE-Problem)

Das DIRECTED-HAMILTON-CYCLE-Problem lässt sich in der gegeben/gefragt-Notation wie folgt formulieren:

gegeben: Ein gerichteter Graph $G = (V, E)$ mit $V = \{v_1, \ldots, v_n\}$

gefragt: Gibt es einen Hamilton-Kreis in G?

$\mathcal{NP}$-Vollständigkeit von DIRECTED-HAMILTON-CYCLE

Satz

DIRECTED-HAMILTON-CYCLE ist $\mathcal{NP}$-vollständig.

Beweis: DIRECTED-HAMILTON-CYCLE $\in \mathcal{NP}$:

- Rate nichtdeterministisch die Permutation π
- Verifiziere, ob $(v_{\pi(i)}, v_{\pi(i+1)}) \in E$ und $(v_{\pi(n)}, v_{\pi(1)}) \in E$ gilt.
- DIRECTED-HAMILTON-CYCLE kann daher kann auf einer NTM in Polynomialzeit entschieden werden.

$\mathcal{NP}$-Vollständigkeit von DIRECTED-HAMILTON-CYCLE (2)

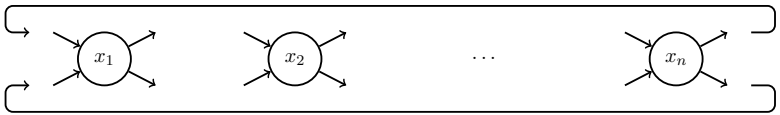
DIRECTED-HAMILTON-CYCLE ist $\mathcal{NP}$-schwer

- Ziel: 3-CNF-SAT $\leq_p$ DIRECTED-HAMILTON-CYCLE
- Sei $F = K_1 \wedge \ldots \wedge K_m$ eine 3-CNF, sodass jede Klausel K_i genau 3 Literale enthält. Sei $Var(F) = \{x_1, \ldots, x_n\}$

$\mathcal{NP}$-Vollständigkeit von DIRECTED-HAMILTON-CYCLE (2)

DIRECTED-HAMILTON-CYCLE ist $\mathcal{NP}$-schwer

- Ziel: 3-CNF-SAT $\leq_p$ DIRECTED-HAMILTON-CYCLE
- Sei $F = K_1 \wedge \ldots \wedge K_m$ eine 3-CNF, sodass jede Klausel K_i genau 3 Literale enthält. Sei $Var(F) = \{x_1, \ldots, x_n\}$
- Erzeuge zunächst

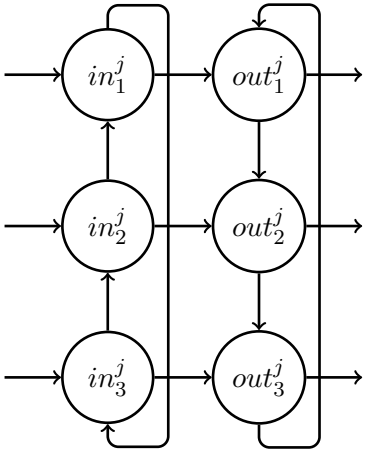


- Idee: Hamilton-Kreis läuft oben durch x_i, wenn $I(x_i) = 1$ und unten durch x_i, wenn $I(x_i) = 0$.

$\mathcal{NP}$-Vollständigkeit von DIRECTED-HAMILTON-CYCLE (3)

DIRECTED-HAMILTON-CYCLE ist $\mathcal{NP}$-schwer

Teilgraphen K_j für jede Klausel K_j:

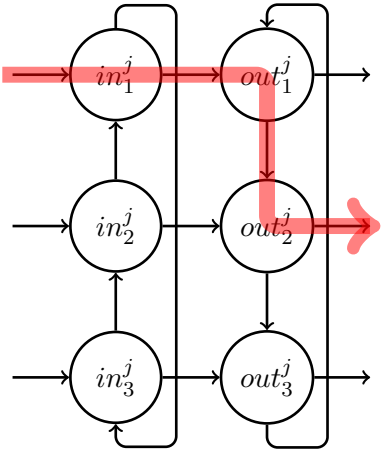


Eigenschaften:

$\mathcal{NP}$-Vollständigkeit von DIRECTED-HAMILTON-CYCLE (3)

DIRECTED-HAMILTON-CYCLE ist $\mathcal{NP}$-schwer

Teilgraphen K_j für jede Klausel K_j:



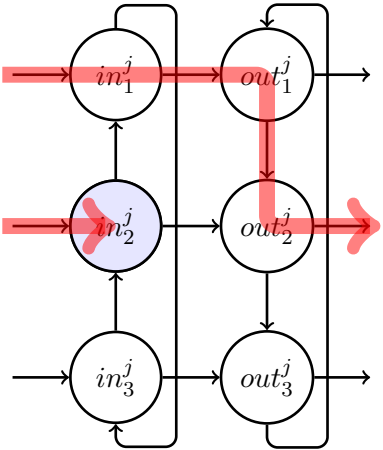
Eigenschaften:

- Jeder Hamilton-Kreis, der durch Eingang in_i^j in K_j hereingeht, muss K_j durch den Knoten out_i^j verlassen.

$\mathcal{NP}$-Vollständigkeit von DIRECTED-HAMILTON-CYCLE (3)

DIRECTED-HAMILTON-CYCLE ist $\mathcal{NP}$-schwer

Teilgraphen K_j für jede Klausel K_j:



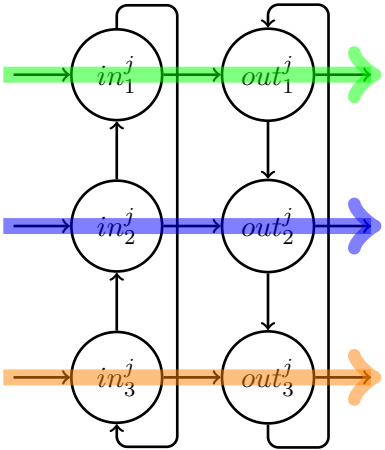
Eigenschaften:

- Jeder Hamilton-Kreis, der durch Eingang in_i^j in K_j hereingeht, muss K_j durch den Knoten out_i^j verlassen. (sonst bleibt man stecken)

$\mathcal{NP}$-Vollständigkeit von DIRECTED-HAMILTON-CYCLE (3)

DIRECTED-HAMILTON-CYCLE ist $\mathcal{NP}$-schwer

Teilgraphen K_j für jede Klausel K_j:



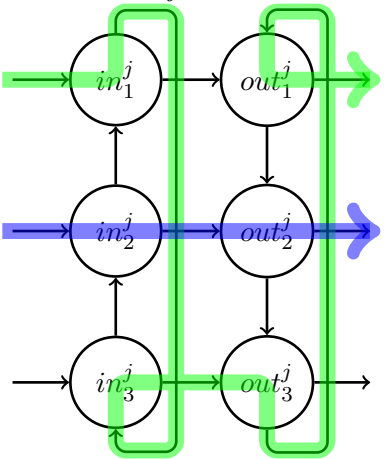
Eigenschaften:

- Jeder Hamilton-Kreis, der durch Eingang in_i^j in K_j hereingeht, muss K_j durch den Knoten out_i^j verlassen. (sonst bleibt man stecken)
- Der Graph kann einmal, zweimal oder dreimal in einem Hamilton-Kreis durchlaufen werden

$\mathcal{NP}$-Vollständigkeit von DIRECTED-HAMILTON-CYCLE (3)

DIRECTED-HAMILTON-CYCLE ist $\mathcal{NP}$-schwer

Teilgraphen K_j für jede Klausel K_j:



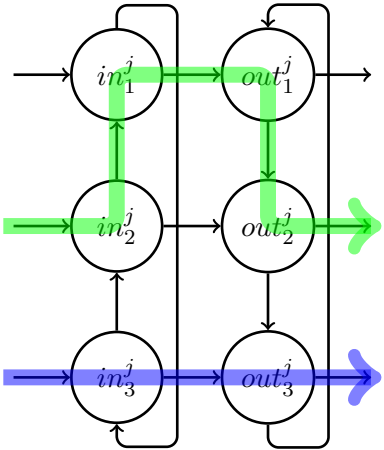
Eigenschaften:

- Jeder Hamilton-Kreis, der durch Eingang in_i^j in K_j hereingeht, muss K_j durch den Knoten out_i^j verlassen. (sonst bleibt man stecken)
- Der Graph kann einmal, zweimal oder dreimal in einem Hamilton-Kreis durchlaufen werden

$\mathcal{NP}$-Vollständigkeit von DIRECTED-HAMILTON-CYCLE (3)

DIRECTED-HAMILTON-CYCLE ist $\mathcal{NP}$-schwer

Teilgraphen K_j für jede Klausel K_j:



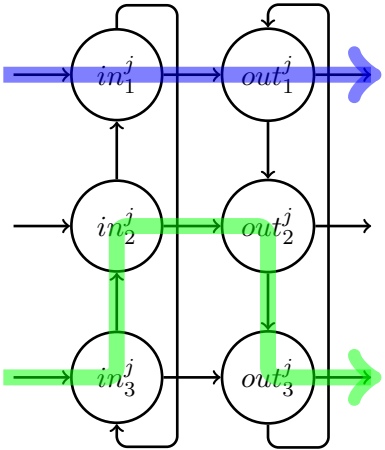
Eigenschaften:

- Jeder Hamilton-Kreis, der durch Eingang in_i^j in K_j hereingeht, muss K_j durch den Knoten out_i^j verlassen. (sonst bleibt man stecken)
- Der Graph kann einmal, zweimal oder dreimal in einem Hamilton-Kreis durchlaufen werden

$\mathcal{NP}$-Vollständigkeit von DIRECTED-HAMILTON-CYCLE (3)

DIRECTED-HAMILTON-CYCLE ist $\mathcal{NP}$-schwer

Teilgraphen K_j für jede Klausel K_j:



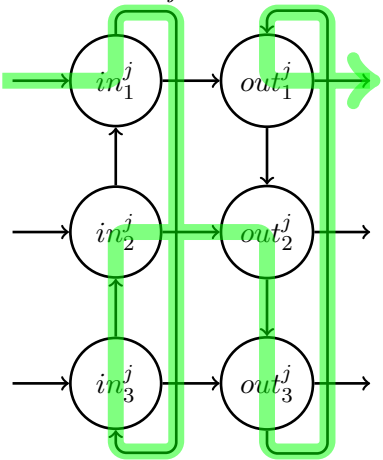
Eigenschaften:

- Jeder Hamilton-Kreis, der durch Eingang in_i^j in K_j hereingeht, muss K_j durch den Knoten out_i^j verlassen. (sonst bleibt man stecken)
- Der Graph kann einmal, zweimal oder dreimal in einem Hamilton-Kreis durchlaufen werden

$\mathcal{NP}$-Vollständigkeit von DIRECTED-HAMILTON-CYCLE (3)

DIRECTED-HAMILTON-CYCLE ist $\mathcal{NP}$-schwer

Teilgraphen K_j für jede Klausel K_j:



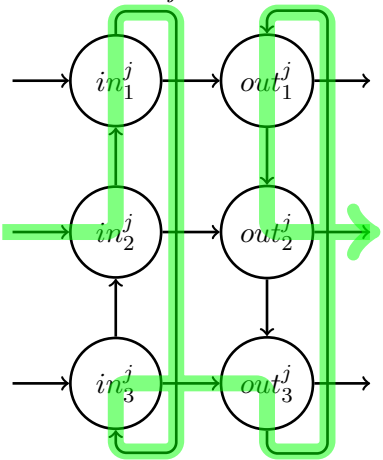
Eigenschaften:

- Jeder Hamilton-Kreis, der durch Eingang in_i^j in K_j hereingeht, muss K_j durch den Knoten out_i^j verlassen. (sonst bleibt man stecken)
- Der Graph kann einmal, zweimal oder dreimal in einem Hamilton-Kreis durchlaufen werden

$\mathcal{NP}$-Vollständigkeit von DIRECTED-HAMILTON-CYCLE (3)

DIRECTED-HAMILTON-CYCLE ist $\mathcal{NP}$-schwer

Teilgraphen K_j für jede Klausel K_j:



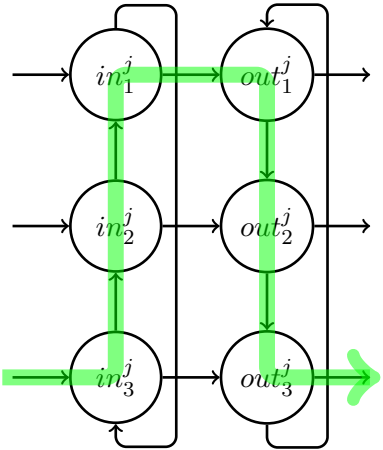
Eigenschaften:

- Jeder Hamilton-Kreis, der durch Eingang in_i^j in K_j hereingeht, muss K_j durch den Knoten out_i^j verlassen. (sonst bleibt man stecken)
- Der Graph kann einmal, zweimal oder dreimal in einem Hamilton-Kreis durchlaufen werden

$\mathcal{NP}$-Vollständigkeit von DIRECTED-HAMILTON-CYCLE (3)

DIRECTED-HAMILTON-CYCLE ist $\mathcal{NP}$-schwer

Teilgraphen K_j für jede Klausel K_j:



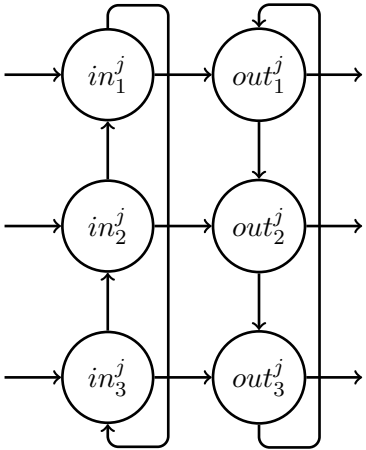
Eigenschaften:

- Jeder Hamilton-Kreis, der durch Eingang in_i^j in K_j hereingeht, muss K_j durch den Knoten out_i^j verlassen. (sonst bleibt man stecken)
- Der Graph kann einmal, zweimal oder dreimal in einem Hamilton-Kreis durchlaufen werden

$\mathcal{NP}$-Vollständigkeit von DIRECTED-HAMILTON-CYCLE (3)

DIRECTED-HAMILTON-CYCLE ist $\mathcal{NP}$-schwer

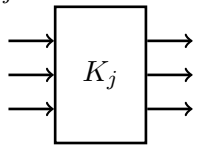
Teilgraphen K_j für jede Klausel K_j:



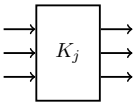
Eigenschaften:

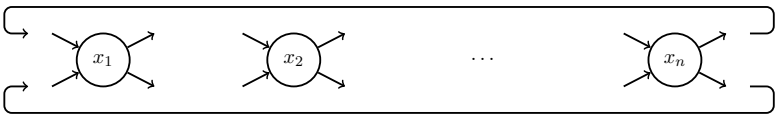
- Jeder Hamilton-Kreis, der durch Eingang in_i^j in K_j hereingeht, muss K_j durch den Knoten out_i^j verlassen. (sonst bleibt man stecken)
- Der Graph kann einmal, zweimal oder dreimal in einem Hamilton-Kreis durchlaufen werden

Teilgraph K_j abstrakt als Bauteil:



$\mathcal{NP}$-Vollständigkeit von DIRECTED-HAMILTON-CYCLE (4)

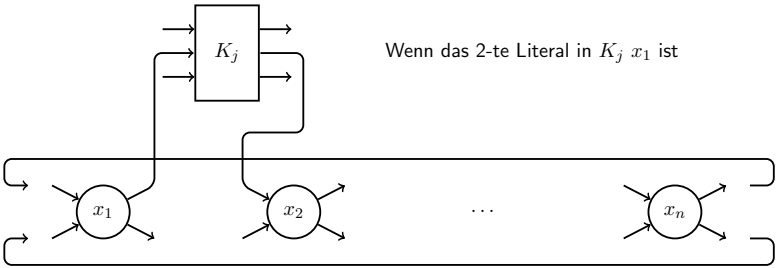




Verbindungen:

- i-ter Ein-/Ausgang von K_j wird zwischen x_k und x_{k+1} verbunden, wenn i-tes Literal in Klausel K_j ist x_k oder $\neg x_k$
- obere Verbindung, wenn Literal x_k ist
- untere Verbindung, wenn Literal $\neg x_k$ ist
- mehrere K_j hintereinander erlaubt.

$\mathcal{NP}$-Vollständigkeit von DIRECTED-HAMILTON-CYCLE (4)



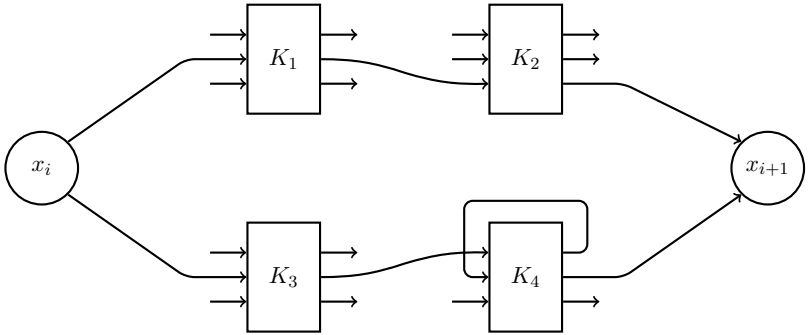
Verbindungen:

- i-ter Ein-/Ausgang von K_j wird zwischen x_k und x_{k+1} verbunden, wenn i-tes Literal in Klausel K_j ist x_k oder $\neg x_k$
- obere Verbindung, wenn Literal x_k ist
- untere Verbindung, wenn Literal $\neg x_k$ ist
- mehrere K_j hintereinander erlaubt.

$\mathcal{NP}$-Vollständigkeit von DIRECTED-HAMILTON-CYCLE (5)

Beispiel: $F = K_1 \wedge K_2 \wedge K_3 \wedge K_4$ eine 3-CNF, sodass

- x_i kommt an 2. Position in K_1 vor
- x_i kommt an 3. Position in K_2 vor
- $\neg x_i$ kommt an 2. Position in K_3 vor
- $\neg x_i$ kommt an 1. und 2. Position in K_4 vor



$\mathcal{NP}$-Vollständigkeit von DIRECTED-HAMILTON-CYCLE (6)

- Konstruktion des Graph in Polynomialzeit möglich:
  Füge iterativ die K_j-Graphen ein.
- Alle Anschlüsse werden angeschlossen: der Graph ist wohl geformt.
- Wenn der Graph einen Hamilton-Kreis hat, dann läuft der Kreis entweder oben durch x_i oder unten durch x_i.
  - Das liefert Belegung $I(x_i) = 1$ bzw. $I(x_i) = 0$
  - Da jedes K_j mindestens einmal von Hamilton-Kreis durchlaufen wird: Jede Klausel wird durch I erfüllt.
- Wenn I Belegung mit $I(F) = 1$, dann gibt es einen Hamilton-Kreis:
  - Durchlaufe Knoten x_i oben, wenn $I(x_i) = 1$ und unten, wenn $I(x_i) = 0$.
  - Durchlaufe K_j ein bis drei Mal, je nachdem welche und wie viele Literale in der Klausel K_j durch I wahr gemacht werden.
- Daher 3-CNF-SAT $\leq_p$ DIRECTED-HAMILTON-CYCLE

Das UNDIRECTED-HAMILTON-CYCLE-Problem

Hamilton-Kreis im ungerichteten Graph

Sei $G = (V, E)$ ein ungerichteter Graph. Ein Hamilton-Kreis ist eine Permutation $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$, sodass $\{v_{\pi(i)}, v_{\pi(i+1)}\} \in E$ und $\{v_{\pi(n)}, v_{\pi(1)}\} \in E$?

Definition (UNDIRECTED-HAMILTON-CYCLE-Problem)

Das UNDIRECTED-HAMILTON-CYCLE-Problem lässt sich in der gegeben/gefragt-Notation wie folgt formulieren:

gegeben: Ein ungerichteter Graph $G = (V, E)$.

gefragt: Gibt es einen Hamilton-Kreis in G?

$\mathcal{NP}$-Vollständigkeit von UNDIRECTED-HAMILTON-CYCLE

Satz

UNDIRECTED-HAMILTON-CYCLE ist $\mathcal{NP}$-vollständig.

Beweis: UNDIRECTED-HAMILTON-CYCLE $\in \mathcal{NP}$:

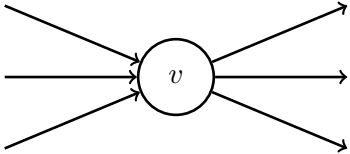
- Rate die Permutation π nichtdeterministisch
- Verifiziere, ob $\{v_{\pi(i)}, v_{\pi(i+1)}\} \in E$ und $\{v_{\pi(n)}, v_{\pi(1)}\} \in E$ gilt.
- Dies kann auf einer NTM in Polynomialzeit durchgeführt werden.

$\mathcal{NP}$-Vollständigkeit von UNDIRECTED-HAMILTON-CYCLE

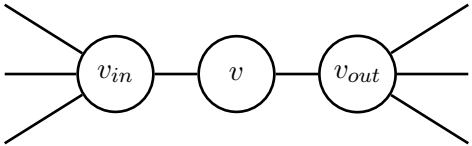
UNDIRECTED-HAMILTON-CYCLE ist $\mathcal{NP}$-schwer:

- Sei f die Funktion, die aus einem gerichteten Graphen (V, E) einen ungerichteten Graphen macht, sodass $f(V) = \bigcup\{\{v_{in}, v, v_{out}\} \mid v \in V\}$ und $f(E) = \{\{u_{out}, v_{in}\} \mid (u, v) \in E\} \cup \bigcup\{\{\{v_{in}, v\}, \{v, v_{out}\}\} \mid v \in V\}$.

  D.h. jeder Knoten v mit Ein- und Ausgängen:



wird ersetzt durch:



$\mathcal{NP}$-Vollständigkeit von UNDIRECTED-HAMILTON-CYCLE

UNDIRECTED-HAMILTON-CYCLE ist $\mathcal{NP}$-schwer:

- In $f(G)$ kann der Knoten v durch einen Hamilton-Kreis nur in einer Richtung durchlaufen werden
  (von v_{in} durch v durch v_{out} oder umgekehrt).
- Falls $v_{in,\pi(1)}, v_{\pi(1)}, v_{out,\pi(1)}, \ldots v_{in,\pi(n)}, v_{\pi(n)}, v_{out,\pi(n)}, v_{in,\pi(1)}$
  oder $v_{out,\pi(1)}, v_{\pi(1)}, v_{in,\pi(1)}, \ldots v_{out,\pi(n)}, v_{\pi(n)}, v_{in,\pi(n)}, v_{out,\pi(1)}$
  ein ungerichteter Hamilton-Kreis,
  dann ist $v_{\pi(1)}, \ldots, v_{\pi(n)}, v_{\pi(1)}$ gerichteter Hamilton Kreis.
- Umgekehrte Richtung: Trivial
- Da f in Polynomialzeit berechenbar ist, folgt
  DIRECTED-HAMILTON-CYCLE $\leq_p$ UNDIRECTED-HAMILTON-CYCLE.

Das TRAVELLING-SALESPERSON-Problem

Definition (TRAVELLING-SALESPERSON-Problem)

Das TRAVELLING-SALESPERSON-Problem lässt sich in der gegeben/gefragt-Notation wie folgt formulieren:

gegeben: Ein Menge von Knoten (Städten) $V = \{v_1, \ldots, v_n\}$, eine $(n \times n)$-Matrix $(M_{i,j})$ mit Entfernungen $M_{i,j} \in \mathbb{N}$ zwischen den Städten v_i und v_j, sowie eine Zahl $k \in \mathbb{N}$.

gefragt: Gibt es eine Rundreise, die alle Städte besucht, der Startort gleich dem Zielort ist, und nicht länger als k ist.Formal: Gibt es eine Permutation $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$, sodass $(\sum_{i=1}^{n-1} M_{\pi(i),\pi(i+1)}) + M_{\pi(n)+\pi(1)} \leq k$.

$\mathcal{NP}$-Vollständigkeit von TRAVELLING-SALESPERSON

Satz

Das TRAVELLING-SALESPERSON-Problem ist $\mathcal{NP}$-vollständig.

Beweis: TRAVELLING-SALESPERSON $\in \mathcal{NP}$:

- Rate die Permutation π nichtdeterministisch
- Prüfe, ob $(\sum_{i=1}^{n-1} M_{\pi(i),\pi(i+1)}) + M_{\pi(n)+\pi(1)} \leq k$ gilt.
- Dies kann auf einer NTM in Polynomialzeit durchgeführt werden.

$\mathcal{NP}$-Vollständigkeit von TRAVELLING-SALESPERSON

TRAVELLING-SALESPERSON ist $\mathcal{NP}$-schwer:
Sei $G = (V, E)$ mit $V = \{1, \ldots, n\}$. Dann sei $f(G) = (V, (M_{i,j}, n))$ mit

$$Mi, j = \begin{cases} 1, & \text{wenn } \{i, j\} \in E \\ 2, & \text{wenn } \{i, j\} \notin E \end{cases}$$

- Wenn G einen ungerichteten Hamilton-Kreis hat, dann ist das eine Rundreise der Länge n.
- Wenn $f(G)$ eine Rundreise der Länge $\leq n$ hat, dann muss die Länge genau n sein, und es können nur Kanten mit Entfernung 1 verwendet werden. Daher hat G einen ungerichteten Hamilton-Kreis.

Da f Polynomialzeit von einer DTM berechnet werden kann:
UNDIRECTED-HAMILTON-CYCLE $\leq_p$ TRAVELLING-SALESPERSON.

Das GRAPH-COLORING-Problem

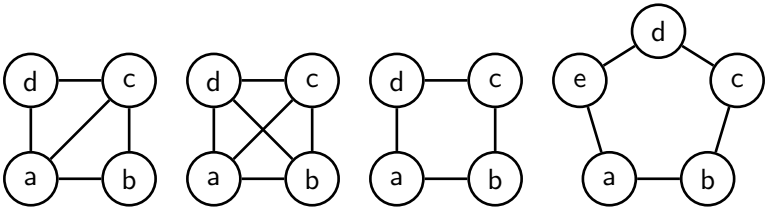
Definition (GRAPH-COLORING-Problem)

Das GRAPH-COLORING-Problem lässt sich in der gegeben/gefragt-Notation wie folgt formulieren:

gegeben: Ein ungerichteter Graph $G = (V, E)$ und eine Zahl $k \in \mathbb{N}$.

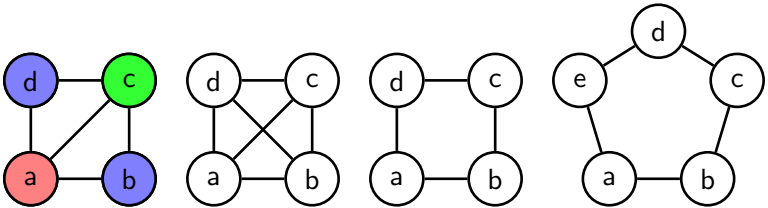
gefragt: Gibt es Färbung der Knoten in V mit $\leq k$ Farben, sodass keine zwei benachbarten Knoten in G, die gleiche Farbe erhalten.

Beispiel



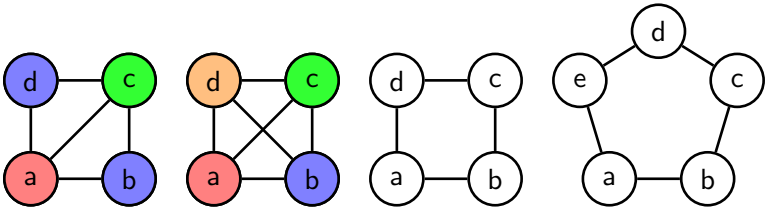
Wie viele Farben sind jeweils minimal notwendig?

Beispiel



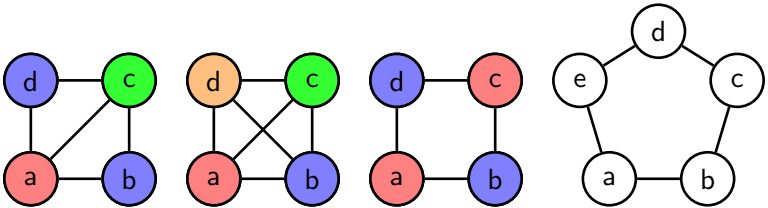
Wie viele Farben sind jeweils minimal notwendig?

Beispiel



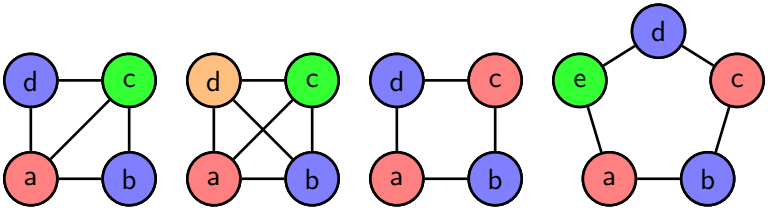
Wie viele Farben sind jeweils minimal notwendig?

Beispiel



Wie viele Farben sind jeweils minimal notwendig?

Beispiel



Wie viele Farben sind jeweils minimal notwendig?

$\mathcal{NP}$-Vollständigkeit von GRAPH-COLORING

Satz

GRAPH-COLORING ist $\mathcal{NP}$-vollständig.

Beweis: GRAPH-COLORING $\in \mathcal{NP}$

- Rate nichtdeterministisch die Farbe aus $\{1, \ldots, k\}$ für jeden Knoten $v \in V$.
- Prüfe, dass die Farben von u und v stets verschieden sind, für alle $\{u, v\} \in E$.
  Das geht in (deterministischer) Polynomialzeit.
- Daher kann GRAPH-COLORING auf einer NTM in polynomieller Zeit entschieden werden.

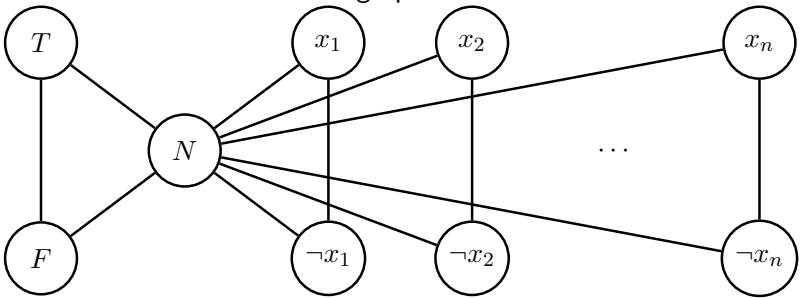
$\mathcal{NP}$-Vollständigkeit von GRAPH-COLORING (2)

GRAPH-COLORING ist $\mathcal{NP}$-schwer.

- Ziel: 3-CNF-SAT $\leq_p$ GRAPH-COLORING
- Sei $F = K_1 \wedge \ldots \wedge K_m$ eine 3-CNF, sodass jede Klausel K_i genau 3 Literale enthält
- Sei $Var(F) = \{x_1, \ldots, x_n\}$.
- Wir erzeugen ein GRAPH-COLORING Problem mit $k = 3$

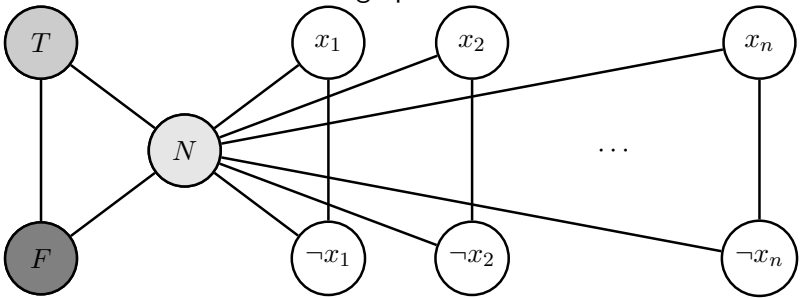
$\mathcal{NP}$-Vollständigkeit von GRAPH-COLORING (3)

Konstruiere zunächst den Teilgraphen



$\mathcal{NP}$-Vollständigkeit von GRAPH-COLORING (3)

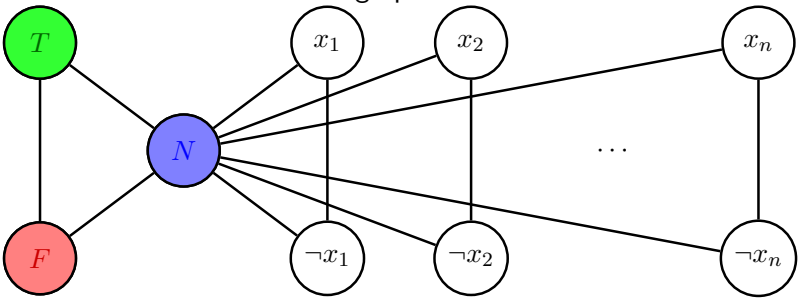
Konstruiere zunächst den Teilgraphen



- Wenn Graph 3-färbbar ist, dann müssen T, N, F mit allen drei Farben gefärbt werden.

$\mathcal{NP}$-Vollständigkeit von GRAPH-COLORING (3)

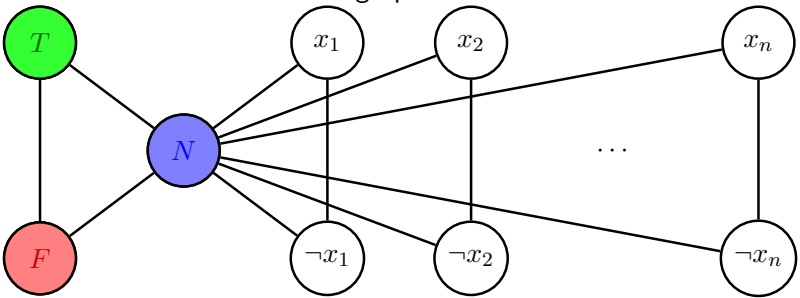
Konstruiere zunächst den Teilgraphen



- Wenn Graph 3-färbbar ist, dann müssen T, N, F mit allen drei Farben gefärbt werden.
- O.B.d.A. seien dies genau die Farben T, N, F mit genau den Knoten

$\mathcal{NP}$-Vollständigkeit von GRAPH-COLORING (3)

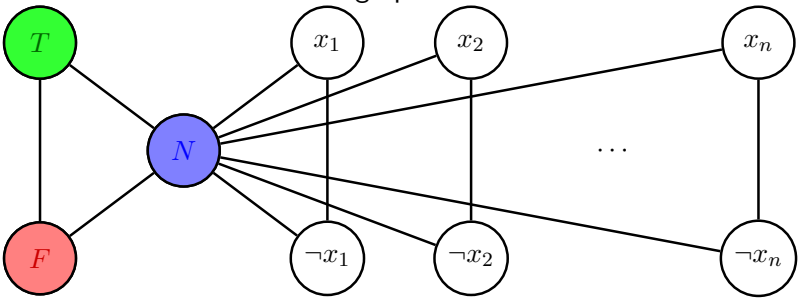
Konstruiere zunächst den Teilgraphen



- Wenn Graph 3-färbbar ist, dann müssen T, N, F mit allen drei Farben gefärbt werden.
- O.B.d.A. seien dies genau die Farben T, N, F mit genau den Knoten
- Für 3-Färbung muss $(x_i, \neg x_i)$ mit (T,F) oder (F,T) gefärbt werden.

$\mathcal{NP}$-Vollständigkeit von GRAPH-COLORING (3)

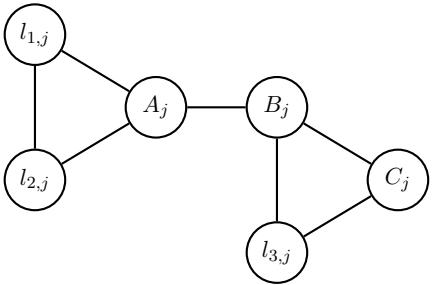
Konstruiere zunächst den Teilgraphen



- Wenn Graph 3-färbbar ist, dann müssen T, N, F mit allen drei Farben gefärbt werden.
- O.B.d.A. seien dies genau die Farben T, N, F mit genau den Knoten
- Für 3-Färbung muss $(x_i, \neg x_i)$ mit (T,F) oder (F,T) gefärbt werden.
- Daher: 3-Färbung erzeugt Belegung $I(x_i) = 1$, wenn x_i mit T gefärbt, $I(x_i)$=0, wenn x_i mit F gefärbt.

$\mathcal{NP}$-Vollständigkeit von GRAPH-COLORING (3)

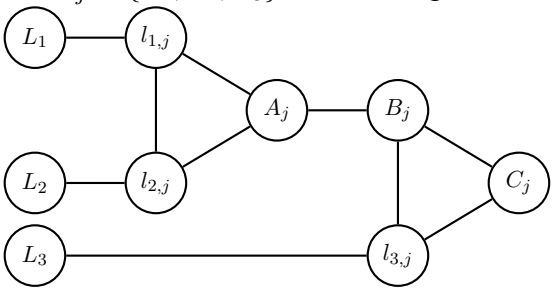
Für jede Klausel $K_j = \{L_1, L_2, L_3\}$ wird der folgende Teilgraph G_j erzeugt:



- Idee: G_j soll das logische Oder der drei Literale in K_j „berechnen“.

$\mathcal{NP}$-Vollständigkeit von GRAPH-COLORING (3)

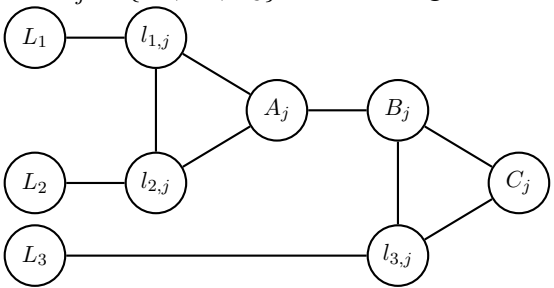
Für jede Klausel $K_j = \{L_1, L_2, L_3\}$ wird der folgende Teilgraph G_j erzeugt:



- Idee: G_j soll das logische Oder der drei Literale in K_j „berechnen“.
- Verbinde $l_{i,j}$ mit x_k oder $\neg x_k$ aus dem anderen Graphen.

$\mathcal{NP}$-Vollständigkeit von GRAPH-COLORING (3)

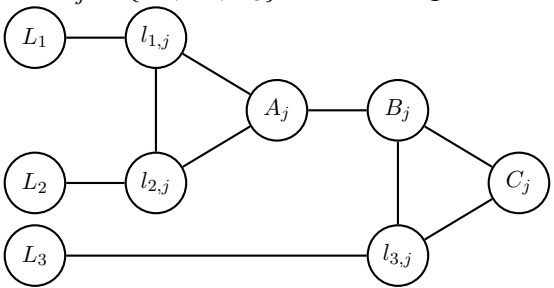
Für jede Klausel $K_j = \{L_1, L_2, L_3\}$ wird der folgende Teilgraph G_j erzeugt:



- Idee: G_j soll das logische Oder der drei Literale in K_j „berechnen“.
- Verbinde $l_{i,j}$ mit x_k oder $\neg x_k$ aus dem anderen Graphen.
- Farben der L_i daher F oder T.

$\mathcal{NP}$-Vollständigkeit von GRAPH-COLORING (3)

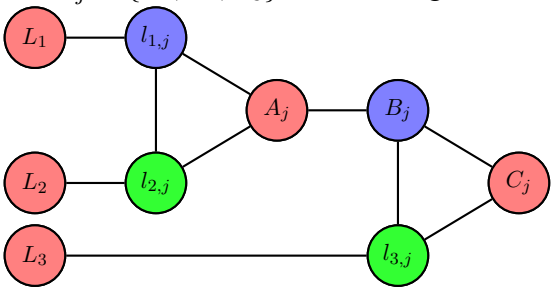
Für jede Klausel $K_j = \{L_1, L_2, L_3\}$ wird der folgende Teilgraph G_j erzeugt:



- Idee: G_j soll das logische Oder der drei Literale in K_j „berechnen“.
- Verbinde $l_{i,j}$ mit x_k oder $\neg x_k$ aus dem anderen Graphen.
- Farben der L_i daher F oder T.
- Wenn alle L_i mit F gefärbt, kann C_j nicht mit T (sondern nur mit F) gefärbt werden.

$\mathcal{NP}$-Vollständigkeit von GRAPH-COLORING (3)

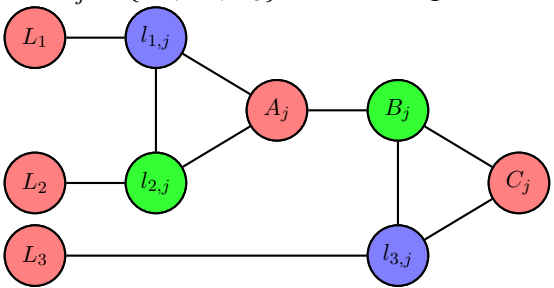
Für jede Klausel $K_j = \{L_1, L_2, L_3\}$ wird der folgende Teilgraph G_j erzeugt:



- Idee: G_j soll das logische Oder der drei Literale in K_j „berechnen“.
- Verbinde $l_{i,j}$ mit x_k oder $\neg x_k$ aus dem anderen Graphen.
- Farben der L_i daher F oder T.
- Wenn alle L_i mit F gefärbt, kann C_j nicht mit T (sondern nur mit F) gefärbt werden.

$\mathcal{NP}$-Vollständigkeit von GRAPH-COLORING (3)

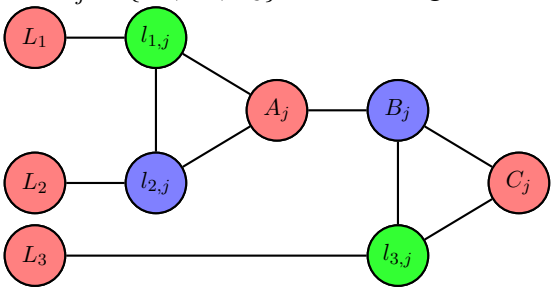
Für jede Klausel $K_j = \{L_1, L_2, L_3\}$ wird der folgende Teilgraph G_j erzeugt:



- Idee: G_j soll das logische Oder der drei Literale in K_j „berechnen“.
- Verbinde $l_{i,j}$ mit x_k oder $\neg x_k$ aus dem anderen Graphen.
- Farben der L_i daher F oder T.
- Wenn alle L_i mit F gefärbt, kann C_j nicht mit T (sondern nur mit F) gefärbt werden.

$\mathcal{NP}$-Vollständigkeit von GRAPH-COLORING (3)

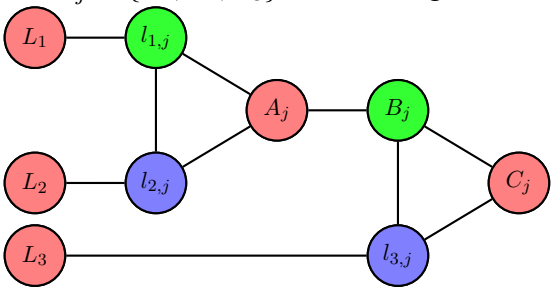
Für jede Klausel $K_j = \{L_1, L_2, L_3\}$ wird der folgende Teilgraph G_j erzeugt:



- Idee: G_j soll das logische Oder der drei Literale in K_j „berechnen“.
- Verbinde $l_{i,j}$ mit x_k oder $\neg x_k$ aus dem anderen Graphen.
- Farben der L_i daher F oder T.
- Wenn alle L_i mit F gefärbt, kann C_j nicht mit T (sondern nur mit F) gefärbt werden.

$\mathcal{NP}$-Vollständigkeit von GRAPH-COLORING (3)

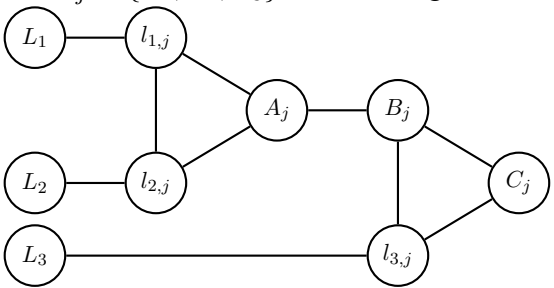
Für jede Klausel $K_j = \{L_1, L_2, L_3\}$ wird der folgende Teilgraph G_j erzeugt:



- Idee: G_j soll das logische Oder der drei Literale in K_j „berechnen“.
- Verbinde $l_{i,j}$ mit x_k oder $\neg x_k$ aus dem anderen Graphen.
- Farben der L_i daher F oder T.
- Wenn alle L_i mit F gefärbt, kann C_j nicht mit T (sondern nur mit F) gefärbt werden.

$\mathcal{NP}$-Vollständigkeit von GRAPH-COLORING (3)

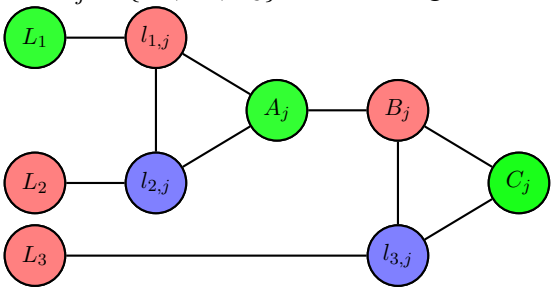
Für jede Klausel $K_j = \{L_1, L_2, L_3\}$ wird der folgende Teilgraph G_j erzeugt:



- Idee: G_j soll das logische Oder der drei Literale in K_j „berechnen“.
- Verbinde $l_{i,j}$ mit x_k oder $\neg x_k$ aus dem anderen Graphen.
- Farben der L_i daher F oder T.
- Wenn alle L_i mit F gefärbt, kann C_j nicht mit T (sondern nur mit F) gefärbt werden.
- Wenn mind. ein L_i mit T, kann C_j mit T gefärbt werden.

$\mathcal{NP}$-Vollständigkeit von GRAPH-COLORING (3)

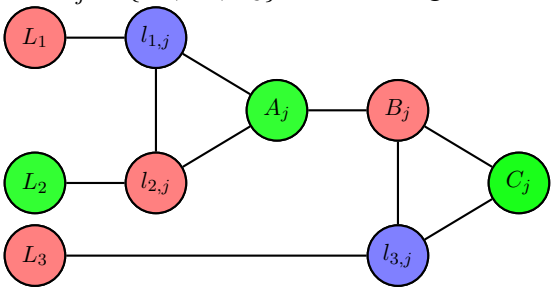
Für jede Klausel $K_j = \{L_1, L_2, L_3\}$ wird der folgende Teilgraph G_j erzeugt:



- Idee: G_j soll das logische Oder der drei Literale in K_j „berechnen“.
- Verbinde $l_{i,j}$ mit x_k oder $\neg x_k$ aus dem anderen Graphen.
- Farben der L_i daher F oder T.
- Wenn alle L_i mit F gefärbt, kann C_j nicht mit T (sondern nur mit F) gefärbt werden.
- Wenn mind. ein L_i mit T, kann C_j mit T gefärbt werden.

$\mathcal{NP}$-Vollständigkeit von GRAPH-COLORING (3)

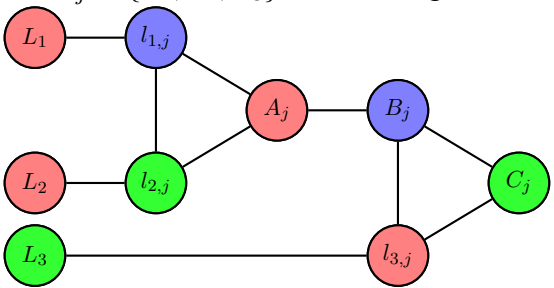
Für jede Klausel $K_j = \{L_1, L_2, L_3\}$ wird der folgende Teilgraph G_j erzeugt:



- Idee: G_j soll das logische Oder der drei Literale in K_j „berechnen“.
- Verbinde $l_{i,j}$ mit x_k oder $\neg x_k$ aus dem anderen Graphen.
- Farben der L_i daher F oder T.
- Wenn alle L_i mit F gefärbt, kann C_j nicht mit T (sondern nur mit F) gefärbt werden.
- Wenn mind. ein L_i mit T, kann C_j mit T gefärbt werden.

$\mathcal{NP}$-Vollständigkeit von GRAPH-COLORING (3)

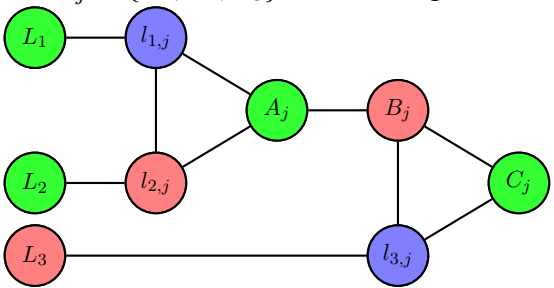
Für jede Klausel $K_j = \{L_1, L_2, L_3\}$ wird der folgende Teilgraph G_j erzeugt:



- Idee: G_j soll das logische Oder der drei Literale in K_j „berechnen“.
- Verbinde $l_{i,j}$ mit x_k oder $\neg x_k$ aus dem anderen Graphen.
- Farben der L_i daher F oder T.
- Wenn alle L_i mit F gefärbt, kann C_j nicht mit T (sondern nur mit F) gefärbt werden.
- Wenn mind. ein L_i mit T, kann C_j mit T gefärbt werden.

$\mathcal{NP}$-Vollständigkeit von GRAPH-COLORING (3)

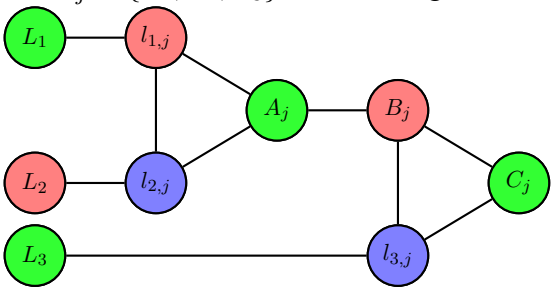
Für jede Klausel $K_j = \{L_1, L_2, L_3\}$ wird der folgende Teilgraph G_j erzeugt:



- Idee: G_j soll das logische Oder der drei Literale in K_j „berechnen“.
- Verbinde $l_{i,j}$ mit x_k oder $\neg x_k$ aus dem anderen Graphen.
- Farben der L_i daher F oder T.
- Wenn alle L_i mit F gefärbt, kann C_j nicht mit T (sondern nur mit F) gefärbt werden.
- Wenn mind. ein L_i mit T, kann C_j mit T gefärbt werden.

$\mathcal{NP}$-Vollständigkeit von GRAPH-COLORING (3)

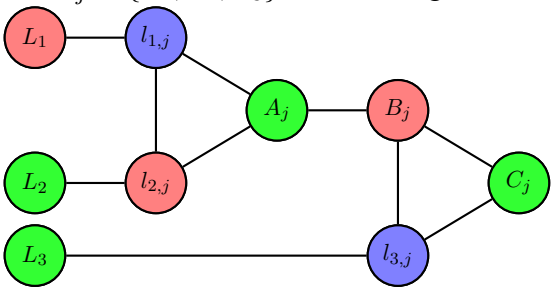
Für jede Klausel $K_j = \{L_1, L_2, L_3\}$ wird der folgende Teilgraph G_j erzeugt:



- Idee: G_j soll das logische Oder der drei Literale in K_j „berechnen“.
- Verbinde $l_{i,j}$ mit x_k oder $\neg x_k$ aus dem anderen Graphen.
- Farben der L_i daher F oder T.
- Wenn alle L_i mit F gefärbt, kann C_j nicht mit T (sondern nur mit F) gefärbt werden.
- Wenn mind. ein L_i mit T, kann C_j mit T gefärbt werden.

$\mathcal{NP}$-Vollständigkeit von GRAPH-COLORING (3)

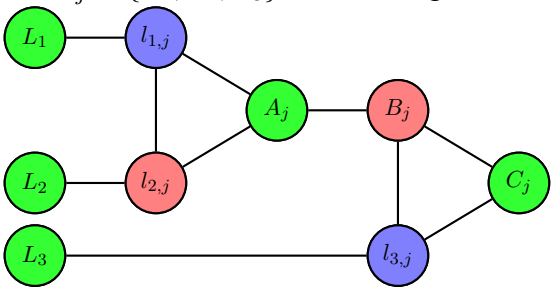
Für jede Klausel $K_j = \{L_1, L_2, L_3\}$ wird der folgende Teilgraph G_j erzeugt:



- Idee: G_j soll das logische Oder der drei Literale in K_j „berechnen“.
- Verbinde $l_{i,j}$ mit x_k oder $\neg x_k$ aus dem anderen Graphen.
- Farben der L_i daher F oder T.
- Wenn alle L_i mit F gefärbt, kann C_j nicht mit T (sondern nur mit F) gefärbt werden.
- Wenn mind. ein L_i mit T, kann C_j mit T gefärbt werden.

$\mathcal{NP}$-Vollständigkeit von GRAPH-COLORING (3)

Für jede Klausel $K_j = \{L_1, L_2, L_3\}$ wird der folgende Teilgraph G_j erzeugt:



- Idee: G_j soll das logische Oder der drei Literale in K_j „berechnen“.
- Verbinde $l_{i,j}$ mit x_k oder $\neg x_k$ aus dem anderen Graphen.
- Farben der L_i daher F oder T.
- Wenn alle L_i mit F gefärbt, kann C_j nicht mit T (sondern nur mit F) gefärbt werden.
- Wenn mind. ein L_i mit T, kann C_j mit T gefärbt werden.

$\mathcal{NP}$-Vollständigkeit von GRAPH-COLORING (4)

- Erzeuge alle Teilgraphen G_j für $K_1, \ldots, K_m$
- Verbinde $l_{i,j}$ mit x_k wenn $L_{i,j} = x_k$ und mit $\neg x_k$ wenn $L_{i,j} = \neg x_k$
- Verbinde jeweils C_j mit N und C_j mit F.
  (Zulässige Färbung muss C_j auf T setzen)

$\mathcal{NP}$-Vollständigkeit von GRAPH-COLORING (4)

- Erzeuge alle Teilgraphen G_j für $K_1, \ldots, K_m$
- Verbinde $l_{i,j}$ mit x_k wenn $L_{i,j} = x_k$ und mit $\neg x_k$ wenn $L_{i,j} = \neg x_k$
- Verbinde jeweils C_j mit N und C_j mit F.
  (Zulässige Färbung muss C_j auf T setzen)

Korrektheit:

- Sei Graph 3-färbbar
- Dann sind alle C_j mit T gefärbt
- x_i und $\neg x_i$ sind mit T und F oder umgekehrt gefärbt
- Belegung $I(x_i) = 1$ wenn x_i mit T und $I(x_i) = 0$ sonst
- I macht Formel F wahr, da in jeder Klausel ein Literal durch I wahr gemacht wird.

$\mathcal{NP}$-Vollständigkeit von GRAPH-COLORING (5)

- Analog kann aus erfüllendem I eine 3-Färbung erzeugt werden.
- Berechnung des Graph geht in Polynomialzeit.
- Daher 3-CNF-SAT $\leq_p$ GRAPH-COLORING.
- Da 3-CNF-SAT $\mathcal{NP}$-schwer, folgt:

  GRAPH-COLORING ist $\mathcal{NP}$-schwer

Zusammenfassung: Polynomielle Reduktionen