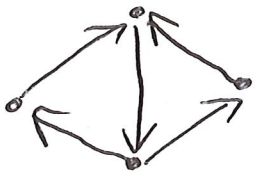
Weitere NP-schwere Probleme

D ≙ Digraph
A ≙ "Arcs", gerichtete Kanten

Problem: Directed Hamiltonian Cycle (DHC)

Gegeben: Ein gerichteter Graph $D=(V,A)$

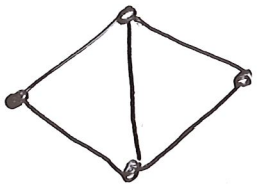
Frage: Existiert in D eine gerichtete Tour, sodass jeder Knoten genau einmal besucht wird und zum Startknoten zurückgekehrt wird?



Problem: (Undirected) Hamiltonian Cycle (HC)

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

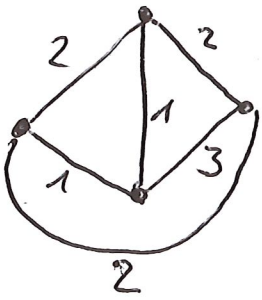
Frage: Existiert in G eine Tour, sodass jeder Knoten genau einmal besucht wird und zum Startknoten zurückgekehrt wird?



Problem: Traveling Salesman Problem (TSP)

Gegeben: Ein vollständiger Graph $G=(V,E)$ mit Kantenkosten $c: E \to \mathbb{R}^+$

Gesucht: Länge einer kürzesten Tour, die alle Knoten besucht und zum Start zurückkehrt.



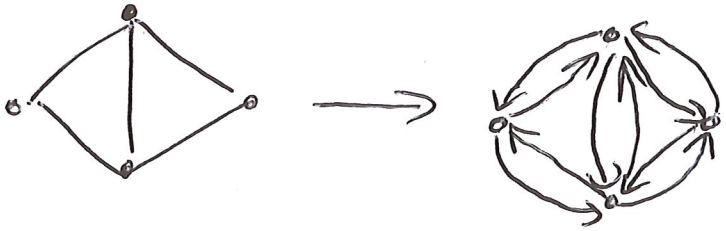
Wir zeigen nun folgendes:

$$3SAT \overset{(3)}{\leq_p} DHC \overset{(1)}{\leq_p} HC \overset{(2)}{\leq_p} TSP$$

Zum Aufwärmen:

$HC \leq_p DHC$

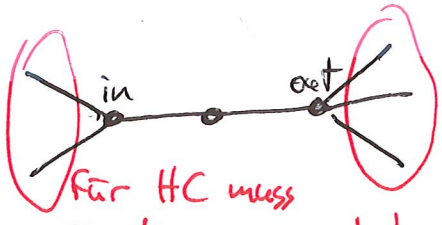
Was muss man tun? → Ersetze jede Kante $\{u,v\} \in E$ durch zwei gerichtete Kanten $(u,v), (v,u)$



① $DHC \leq_p HC$

Ideen für eine Reduktion?

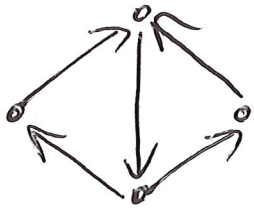
Baue Gadget für Knoten:

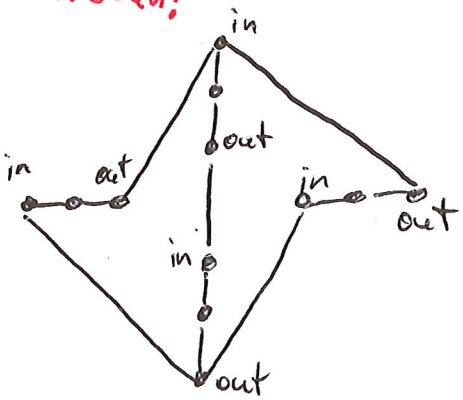


Für HC muss eine hiervon genutzt werden!

Für HC muss eine hiervon genutzt werden!

Im Beispiel:





Dass diese Reduktion

a) korrekt ist und

b) polynomielle Laufzeit besitzt,

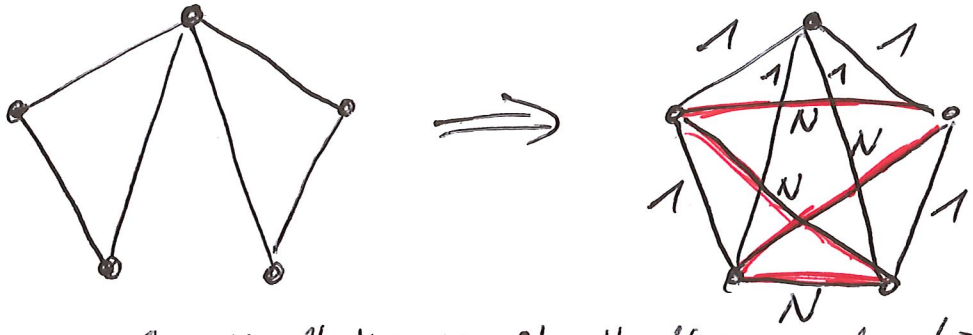
lassen wir an dieser Stelle aus.

(2) $HC \leq_p TSP$

Wir haben einen Graphen gegeben, der ist allerdings
a) nicht vollständig
b) ungewichtet !

Also: füge Kanten ein und gib allen Kanten Gewichte.
Aber welche?

Idee: Gib neuen Kanten Gewicht $N \gg 1$ und allen anderen Gewicht 1. Nennen wir diesen Graphen G'



Nun: G enthält HC $\Leftrightarrow$ G' enthält Tour der Länge n.

Korrektheit:
Gibt es in G einen HC, dann gibt es in G' eine Tour der Länge n. ($\hat{=}$ Anzahl Knoten)
Gibt es in G keinen HC, dann muss in G' eine Kante der Länge N benutzt werden, wodurch die Tour eine Gesamtlänge von mindestens $n-1+N$ besitzt.

Laufzeit:
Es müssen maximal $O(n^2)$ Kanten hinzugefügt werden.
Allen $O(n^2)$ Kanten müssen Werte gegeben werden
$\Rightarrow$ Insgesamt eine Laufzeit $O(n^2)$.

Jetzt wird es schwieriger:

③ 3SAT $\leq_p$ DHC

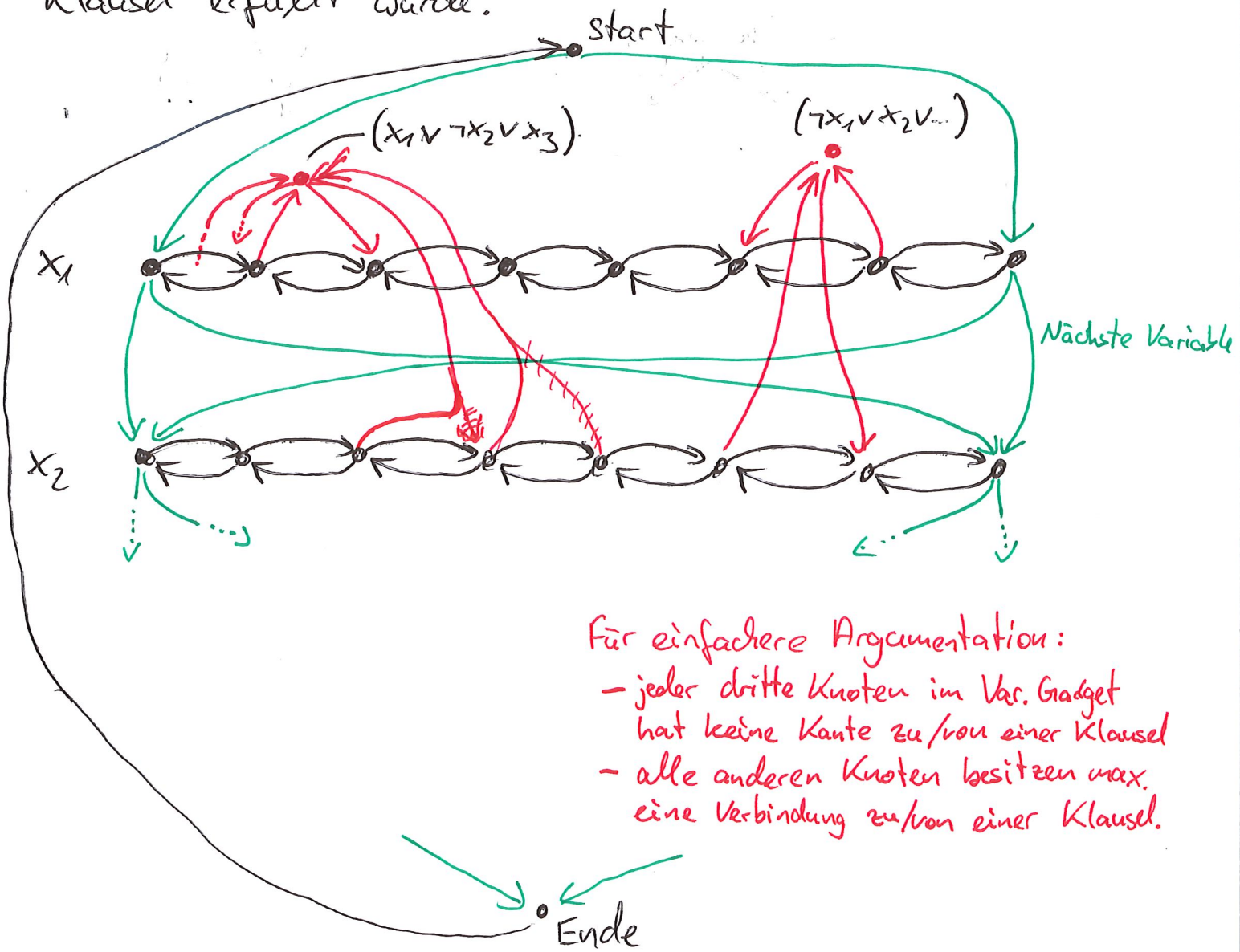
Wir haben gegeben:

Variablen $x_1, \dots, x_n$ und Klauseln

$$(l_{1,1} \vee l_{1,2} \vee l_{1,3}) \wedge \dots \wedge (l_{m,1} \vee l_{m,2} \vee l_{m,3}) =: \varphi$$

Gesucht ist ein Graph $D = (V, A)$, sodass D genau dann einen ger. HC besitzt, wenn φ erfüllbar ist.

Wie codiert man nun eine Variablenbelegung und wie kann man das nutzen um zu erkennen, ob eine Klausel erfüllt wurde?



Für einfachere Argumentation:

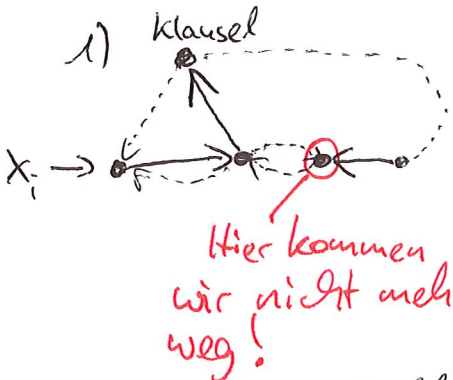
- jeder dritte Knoten im Var. Gadget hat keine Kante zu/von einer Klausel
- alle anderen Knoten besitzen max. eine Verbindung zu/von einer Klausel.

z.Z.: φ erfüllbar $\Leftrightarrow$ D enthält Hamiltonkreis

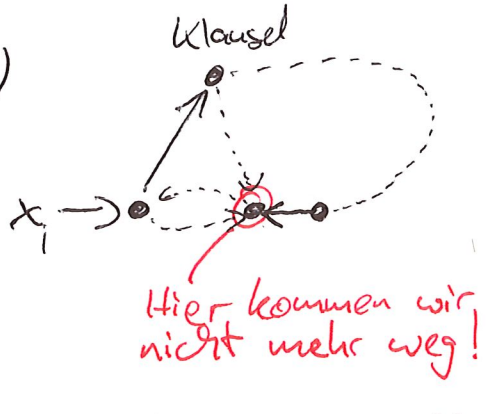
"$\Rightarrow$": Sei x^* eine erfüllende Variablenbelegung für φ.
Dann laufen wir im Var Gadget für Variable x_i nach rechts, falls $x_i^* = 1$, oder nach links, falls $x_i^* = 0$.
Da x^* φ erfüllt, können wir in D alle Klauseln besuchen.

"$\Leftarrow$": Angenommen D ~~besitzt~~ einen Hamiltonkreis.
Wenn wir einen Klauselknoten besuchen, müssen wir zu dem Var. Gadget zurückkehren, von dem wir gekommen sind. Betrachte andernfalls folgende Situationen:

1)



2)



In beiden Fällen können wir also keinen Hamiltonkreis konstruieren, obwohl D einen besitzt.

Da also die Variablen und Klauseln ordnungsgemäß abgelaufen werden, können wir darüber eine Lösung für die 3SAT-Instanz ablesen, die φ erfüllt.
Wir setzen $x_i = 1$, falls wir im Var. Gadget rechts gelaufen sind und $x_i = 0$, sonst.

□

Satz: DHC, HC und TSP sind NP-schwer Welche davon liegen in NP?

Satz: TSP lässt sich nicht approximieren.

Beweis:

Sei $f(n)$ eine Funktion mit $f: \mathbb{N} \to \mathbb{R}^+$.

Angenommen, es gäbe eine $f(n)$-Approximation für TSP, dann ~~wäre~~ hätte jede Tour durch die Approximation höchsten die Länge $f(n) \cdot OPT$.

Wählen wir in der Reduktion von HC auf TSP $N = f(n) \cdot n$. ~~dann gäbe uns~~ der Algorithmus gäbe uns einen Wert $\leq n \cdot f(n)$, falls ein HC existierte.

Da $\underbrace{n-1+N}_{n-1+f(n)\cdot n} > n \cdot f(n)$, hätte der Approximationsalgorithmus den HC gefunden. Wir könnten also HC in polyzeit lösen ↯ □