Theorie der Informatik

21. einige NP-vollständige Probleme

Malte Helmert Gabriele Röger

Universität Basel

19. Mai 2014

Überblick: Vorlesung

Vorlesungsteile

I. Logik ✓
II. Automatentheorie und formale Sprachen ✓
III. Berechenbarkeitstheorie ✓
IV. Komplexitätstheorie

Überblick: Komplexitätstheorie

IV. Komplexitätstheorie

18. Motivation und Einführung ✓
19. P, NP und polynomielle Reduktionen ✓
20. Satz von Cook und Levin ✓
21. einige NP-vollständige Probleme

Nachlesen

Literatur zu diesem Vorlesungskapitel

Theoretische Informatik - kurz gefasst
von Uwe Schöning (5. Auflage)

- Kapitel 3.3

Übersicht

Weitere NP-vollständige Probleme

- Der Beweis der NP-Vollständigkeit von SAT war kompliziert.
- Aber: mit seiner Hilfe können wir sehr viel einfacher beweisen, dass weitere Probleme NP-vollständig sind.

Weitere NP-vollständige Probleme

- Der Beweis der NP-Vollständigkeit von SAT war kompliziert.
- Aber: mit seiner Hilfe können wir sehr viel einfacher beweisen, dass weitere Probleme NP-vollständig sind.

Satz (NP-Vollständigkeits-Beweise durch Reduktionen)

Seien A und B Probleme, für die gilt:

- *A ist NP-hart*
- *$A \leq_p B$*

Dann ist B ebenfalls NP-hart.
Wenn ausserdem $B \in NP$ gilt, dann ist B NP-vollständig.

Beweis.

Erster Teil: Hausaufgaben. Zweiter Teil: offensichtlich. □

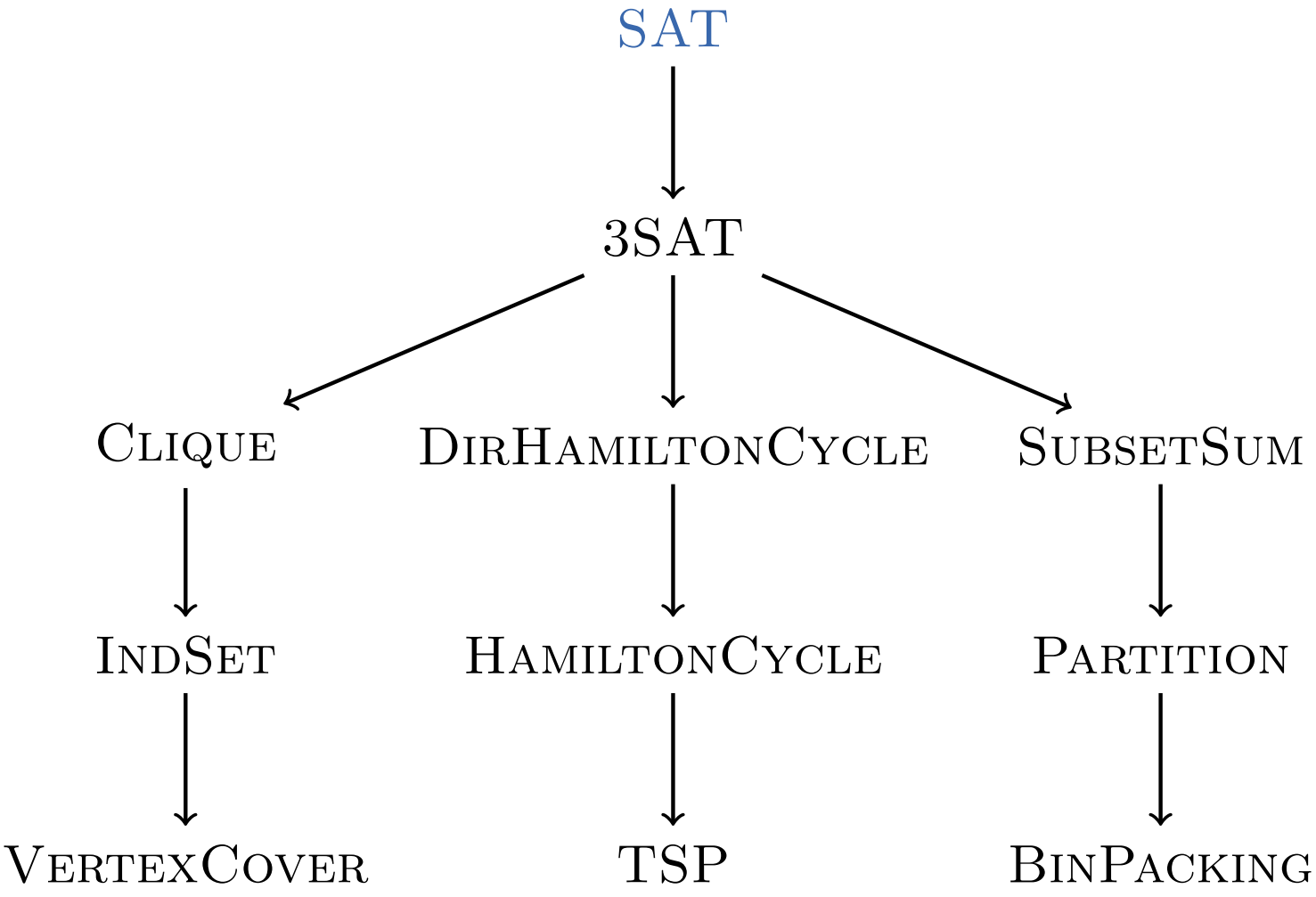
NP-vollständige Probleme

- Es gibt Tausende bekannte NP-vollständige Probleme.
- Ein umfangreicher Katalog NP-vollständiger Probleme aus vielen Bereichen der Informatik ist enthalten in:

  Michael R. Garey und David S. Johnson:
  Computers and Intractability —
  A Guide to the Theory of NP-Completeness
  W. H. Freeman, 1979.

- In diesem Kapitel lernen wir einige dieser Probleme kennen.

Übersicht über die Reduktionen

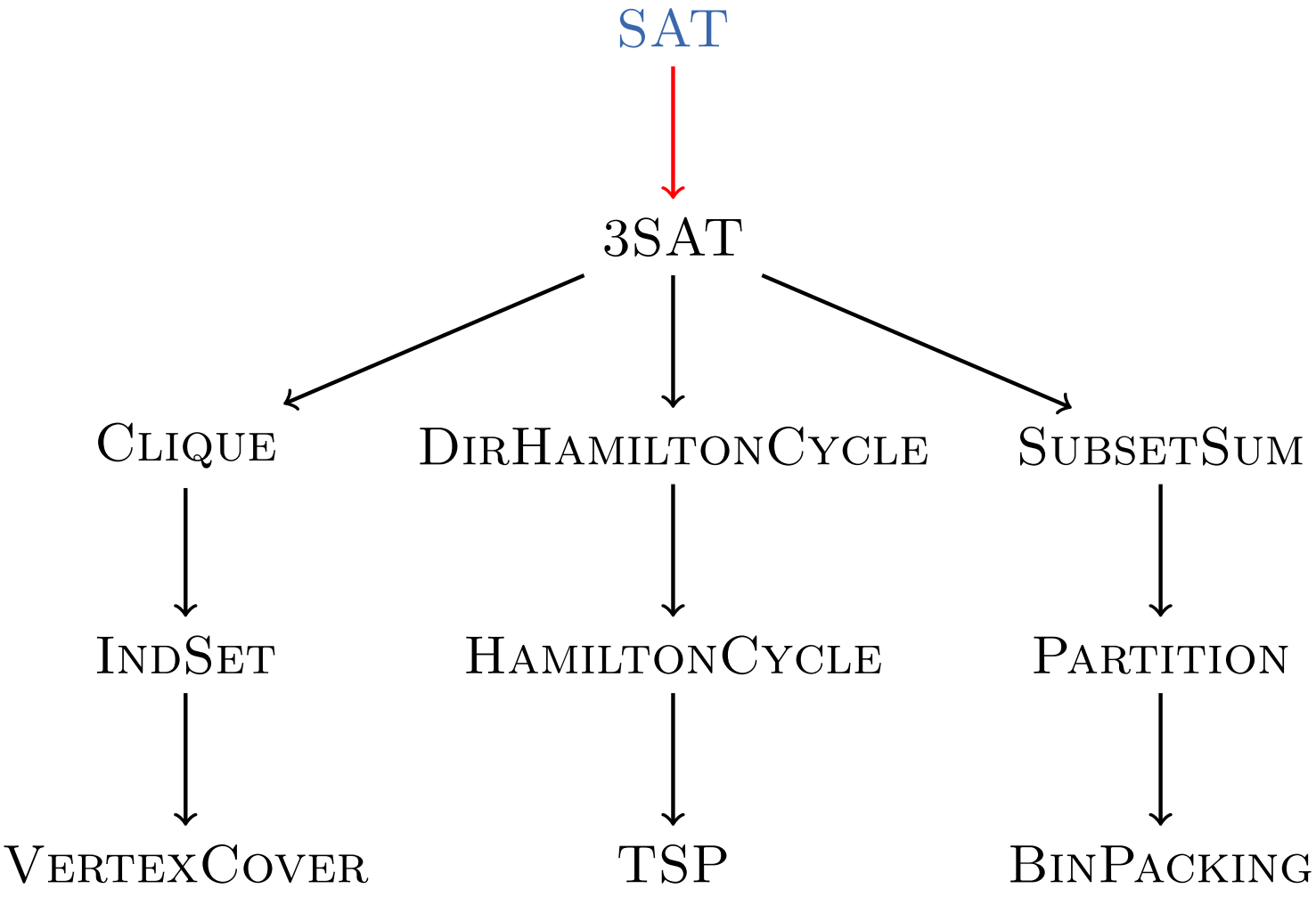


Was ist zu tun?

- Wir wollen die NP-Vollständigkeit dieser 11 Probleme zeigen.
- Wir wissen bereits, dass SAT NP-vollständig ist.
- Damit reicht es aus zu zeigen,
  - dass für alle Kanten der Abbildung polynomielle Reduktionen existieren (damit sind alle Probleme NP-hart)
  - und dass die Probleme alle in NP liegen.

(Es würde reichen, die Mitgliedschaft in NP nur für die Blätter der Abbildung zu zeigen. Aber die Mitgliedschaft ist so einfach zu zeigen, dass dies keine Arbeit spart.)

$\text{SAT} \leq_p \text{3SAT}$



3SAT

SAT und 3SAT

Definition (SAT; Wiederholung)

Das Problem SAT (Erfüllbarkeit = satisfiability) ist wie folgt definiert:

Gegeben: eine aussagenlogische Formel φ

Frage: Ist φ erfüllbar?

Definition (3SAT)

Das Problem 3SAT ist wie folgt definiert:

Gegeben: eine aussagenlogische Formel φ in konjunktiver Normalform mit höchstens drei Literalen pro Klausel

Frage: Ist φ erfüllbar?

3SAT ist NP-vollständig

Satz (3SAT ist NP-vollständig)

3SAT *ist NP-vollständig.*

Beweis.

$3\mathrm{SAT} \in \mathrm{NP}$: Raten und prüfen.

$3\mathrm{SAT}$ ist NP-hart: $\mathrm{SAT} \leq_\mathsf{p} 3\mathrm{SAT}$

$\rightsquigarrow$ Tafel. □

Eingeschränktes 3SAT

Anmerkung: 3SAT bleibt NP-vollständig,
wenn wir zusätzlich fordern, dass

- jede Klausel genau drei Literale enthält und
- eine Klausel dasselbe Literal nicht doppelt enthalten darf

Eingeschränktes 3SAT

Anmerkung: 3SAT bleibt NP-vollständig,
wenn wir zusätzlich fordern, dass

- jede Klausel genau drei Literale enthält und
- eine Klausel dasselbe Literal nicht doppelt enthalten darf

Idee:

- Entferne duplizierte Literale aus jeder Klausel.

Eingeschränktes 3SAT

Anmerkung: 3SAT bleibt NP-vollständig,
wenn wir zusätzlich fordern, dass

- jede Klausel genau drei Literale enthält und
- eine Klausel dasselbe Literal nicht doppelt enthalten darf

Idee:

- Entferne duplizierte Literale aus jeder Klausel.
- füge neue Variablen hinzu: X, Y, Z
- füge neue Klauseln hinzu: $(X \vee Y \vee Z)$, $(X \vee Y \vee \neg Z)$, $(X \vee \neg Y \vee Z)$, $(\neg X \vee Y \vee Z)$, $(X \vee \neg Y \vee \neg Z)$, $(\neg X \vee Y \vee \neg Z)$, $(\neg X \vee \neg Y \vee Z)$

Eingeschränktes 3SAT

Anmerkung: 3SAT bleibt NP-vollständig,
wenn wir zusätzlich fordern, dass

- jede Klausel genau drei Literale enthält und
- eine Klausel dasselbe Literal nicht doppelt enthalten darf

Idee:

- Entferne duplizierte Literale aus jeder Klausel.
- füge neue Variablen hinzu: X, Y, Z
- füge neue Klauseln hinzu: $(X \vee Y \vee Z)$, $(X \vee Y \vee \neg Z)$, $(X \vee \neg Y \vee Z)$, $(\neg X \vee Y \vee Z)$, $(X \vee \neg Y \vee \neg Z)$, $(\neg X \vee Y \vee \neg Z)$, $(\neg X \vee \neg Y \vee Z)$

$\rightsquigarrow$ genau dann erfüllt, wenn X, Y, Z alle wahr

Eingeschränktes 3SAT

Anmerkung: 3SAT bleibt NP-vollständig,
wenn wir zusätzlich fordern, dass

- jede Klausel genau drei Literale enthält und
- eine Klausel dasselbe Literal nicht doppelt enthalten darf

Idee:

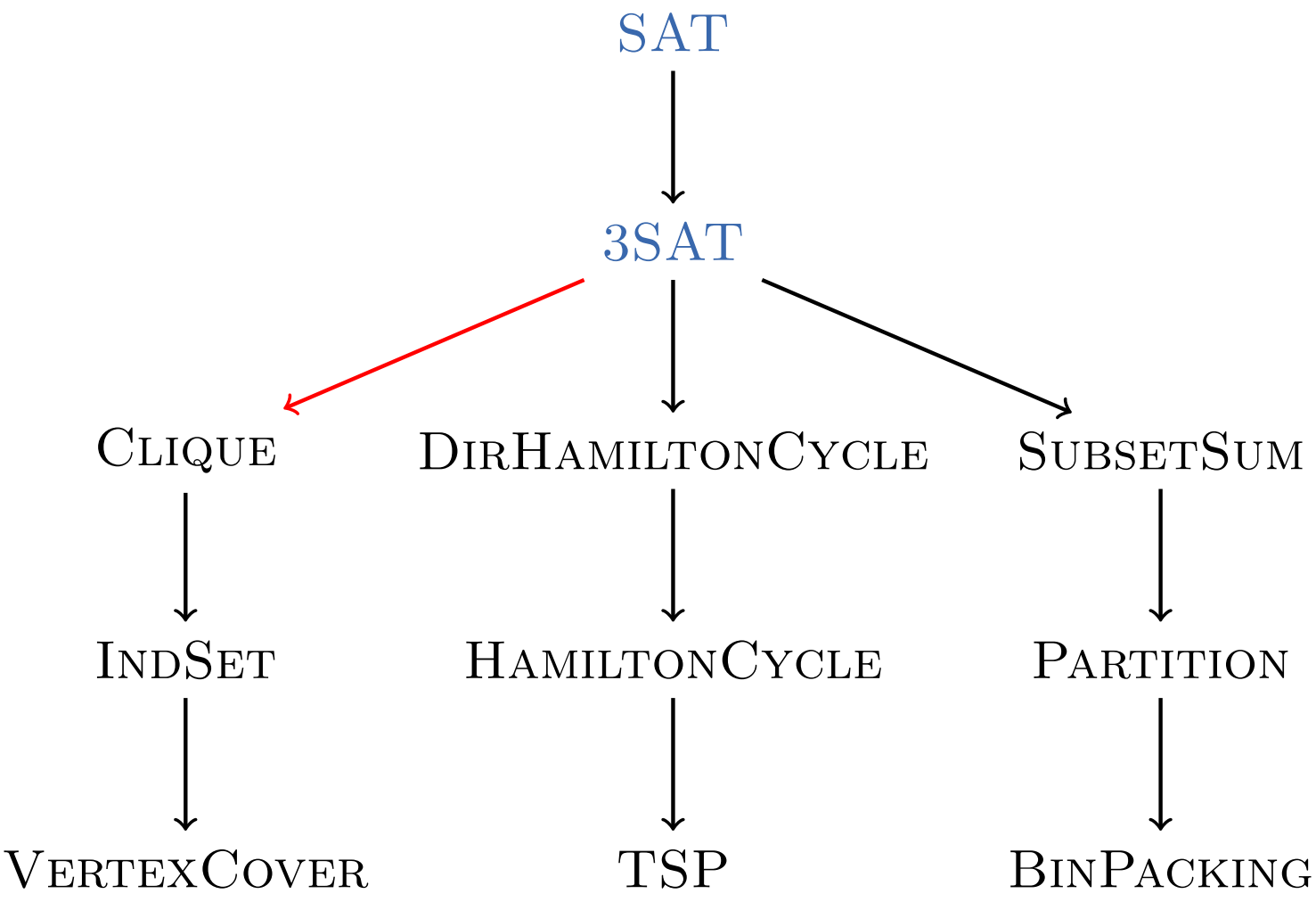
- Entferne duplizierte Literale aus jeder Klausel.
- füge neue Variablen hinzu: X, Y, Z
- füge neue Klauseln hinzu: $(X \vee Y \vee Z)$, $(X \vee Y \vee \neg Z)$, $(X \vee \neg Y \vee Z)$, $(\neg X \vee Y \vee Z)$, $(X \vee \neg Y \vee \neg Z)$, $(\neg X \vee Y \vee \neg Z)$, $(\neg X \vee \neg Y \vee Z)$

$\rightsquigarrow$ genau dann erfüllt, wenn X, Y, Z alle wahr

- fülle Klauseln mit weniger als drei Literalen mit $\neg X$ und gegebenenfalls $\neg Y$ auf

Graphenprobleme

3SAT $\leq_p$ Clique



Clique ist NP-vollständig (1)

Definition (Clique)

Das Problem Clique ist wie folgt definiert:

Gegeben: ungerichteter Graph $G = \langle V, E \rangle$, Zahl $K \in \mathbb{N}_0$

Frage: Enthält G eine Clique der Grösse K oder mehr,
d. h. eine Knotenmenge $C \subseteq V$ mit $|C| \geq K$
und $\{u, v\} \in E$ für alle $u, v \in C$ mit $u \neq v$?

Clique ist NP-vollständig (1)

Definition (Clique)

Das Problem Clique ist wie folgt definiert:

Gegeben: ungerichteter Graph $G = \langle V, E \rangle$, Zahl $K \in \mathbb{N}_0$

Frage: Enthält G eine Clique der Grösse K oder mehr,
d. h. eine Knotenmenge $C \subseteq V$ mit $|C| \geq K$
und $\{u, v\} \in E$ für alle $u, v \in C$ mit $u \neq v$?

Satz (Clique ist NP-vollständig)

Clique *ist NP-vollständig.*

Clique ist NP-vollständig (2)

Beweis.

Clique $\in$ NP: Raten und Prüfen.

Clique ist NP-vollständig (2)

Beweis.

Clique $\in$ NP: Raten und Prüfen.

Clique ist NP-hart: $3\mathrm{SAT} \leq_{\mathsf{p}} \textsc{Clique}$

Clique ist NP-vollständig (2)

Beweis.

Clique $\in$ NP: Raten und Prüfen.

Clique ist NP-hart: $3\mathrm{SAT} \leq_{\mathsf{p}} \textsc{Clique}$

- Sei 3-KNF-Formel φ gegeben,
  bei der jede Klausel genau drei Literale enthält.
- Wir müssen in polynomieller Zeit
  Graphen $G = \langle V, E \rangle$ und Zahl K konstruieren, so dass gilt:
  G hat Clique der Grösse K oder mehr gdw. φ erfüllbar.

Clique ist NP-vollständig (2)

Beweis.

Clique $\in$ NP: Raten und Prüfen.

Clique ist NP-hart: $3\mathrm{SAT} \leq_{\mathsf{p}} \textsc{Clique}$

- Sei 3-KNF-Formel φ gegeben,
  bei der jede Klausel genau drei Literale enthält.
- Wir müssen in polynomieller Zeit
  Graphen $G = \langle V, E \rangle$ und Zahl K konstruieren, so dass gilt:
  G hat Clique der Grösse K oder mehr gdw. φ erfüllbar.
- Konstruktion von V, E, K auf folgenden Folien.

...

Clique ist NP-vollständig (3)

Beweis (Fortsetzung).

Sei m Anzahl Klauseln in φ.

Sei l_{ij} das j-te Literal in Klausel i.

Clique ist NP-vollständig (3)

Beweis (Fortsetzung).

Sei m Anzahl Klauseln in φ.

Sei l_{ij} das j-te Literal in Klausel i.

Definiere V, E, K wie folgt:

Clique ist NP-vollständig (3)

Beweis (Fortsetzung).

Sei m Anzahl Klauseln in φ.

Sei l_{ij} das j-te Literal in Klausel i.

Definiere V, E, K wie folgt:

- $V = \{\langle i, j\rangle \mid 1 \leq i \leq m, 1 \leq j \leq 3\}$
  $\rightsquigarrow$ ein Knoten für jedes Literal jeder Klausel

Clique ist NP-vollständig (3)

Beweis (Fortsetzung).

Sei m Anzahl Klauseln in φ.

Sei l_{ij} das j-te Literal in Klausel i.

Definiere V, E, K wie folgt:

- $V = \{\langle i, j\rangle \mid 1 \leq i \leq m, 1 \leq j \leq 3\}$
  $\rightsquigarrow$ ein Knoten für jedes Literal jeder Klausel
- E enthält Kante zwischen $\langle i, j\rangle$ und $\langle i', j'\rangle$ genau dann, wenn
  - $i \neq i' \rightsquigarrow$ gehören zu verschiedenen Klauseln
  - l_{ij} und $l_{i'j'}$ sind keine komplementären Literale

Clique ist NP-vollständig (3)

Beweis (Fortsetzung).

Sei m Anzahl Klauseln in φ.

Sei l_{ij} das j-te Literal in Klausel i.

Definiere V, E, K wie folgt:

- $V = \{\langle i, j\rangle \mid 1 \leq i \leq m, 1 \leq j \leq 3\}$
  $\rightsquigarrow$ ein Knoten für jedes Literal jeder Klausel
- E enthält Kante zwischen $\langle i, j\rangle$ und $\langle i', j'\rangle$ genau dann, wenn
  - $i \neq i' \rightsquigarrow$ gehören zu verschiedenen Klauseln
  - l_{ij} und $l_{i'j'}$ sind keine komplementären Literale
- $K = m$

Clique ist NP-vollständig (3)

Beweis (Fortsetzung).

Sei m Anzahl Klauseln in φ.

Sei l_{ij} das j-te Literal in Klausel i.

Definiere V, E, K wie folgt:

- $V = \{\langle i, j\rangle \mid 1 \leq i \leq m, 1 \leq j \leq 3\}$
  $\rightsquigarrow$ ein Knoten für jedes Literal jeder Klausel
- E enthält Kante zwischen $\langle i, j\rangle$ und $\langle i', j'\rangle$ genau dann, wenn
  - $i \neq i' \rightsquigarrow$ gehören zu verschiedenen Klauseln
  - l_{ij} und $l_{i'j'}$ sind keine komplementären Literale
- $K = m$

$\rightsquigarrow$ offensichtlich polynomiell berechenbar

Clique ist NP-vollständig (3)

Beweis (Fortsetzung).

Sei m Anzahl Klauseln in φ.

Sei l_{ij} das j-te Literal in Klausel i.

Definiere V, E, K wie folgt:

- $V = \{\langle i, j\rangle \mid 1 \leq i \leq m, 1 \leq j \leq 3\}$
  $\rightsquigarrow$ ein Knoten für jedes Literal jeder Klausel
- E enthält Kante zwischen $\langle i, j\rangle$ und $\langle i', j'\rangle$ genau dann, wenn
  - $i \neq i' \rightsquigarrow$ gehören zu verschiedenen Klauseln
  - l_{ij} und $l_{i'j'}$ sind keine komplementären Literale
- $K = m$

$\rightsquigarrow$ offensichtlich polynomiell berechenbar

zu zeigen: Reduktionseigenschaft ...

Clique ist NP-vollständig (4)

Beweis (Fortsetzung).

($\Rightarrow$): Wenn φ erfüllbar ist, enthält $\langle V, E \rangle$ Clique der Grösse K:

Clique ist NP-vollständig (4)

Beweis (Fortsetzung).

($\Rightarrow$): Wenn φ erfüllbar ist, enthält $\langle V, E \rangle$ Clique der Grösse K:

- Wähle zu erfüllender Belegung in jeder Klausel einen Knoten zu einem erfüllten Literal.
- Die gewählten K Knoten sind alle miteinander verbunden.

...

Clique ist NP-vollständig (5)

Beweis (Fortsetzung).

($\Leftarrow$): Wenn $\langle V, E \rangle$ Clique der Grösse K enthält, ist φ erfüllbar:

Clique ist NP-vollständig (5)

Beweis (Fortsetzung).

($\Leftarrow$): Wenn $\langle V, E \rangle$ Clique der Grösse K enthält, ist φ erfüllbar:

- Die gewählten Knoten müssen alle zu unterschiedlichen Klauseln gehören

$\rightsquigarrow$ genau ein Knoten pro Klausel wird ausgewählt

- Die gewählten Knoten können nicht komplementären Literalen X und $\neg X$ entsprechen.

Clique ist NP-vollständig (5)

Beweis (Fortsetzung).

($\Leftarrow$): Wenn $\langle V, E \rangle$ Clique der Grösse K enthält, ist φ erfüllbar:

- Die gewählten Knoten müssen alle zu unterschiedlichen Klauseln gehören

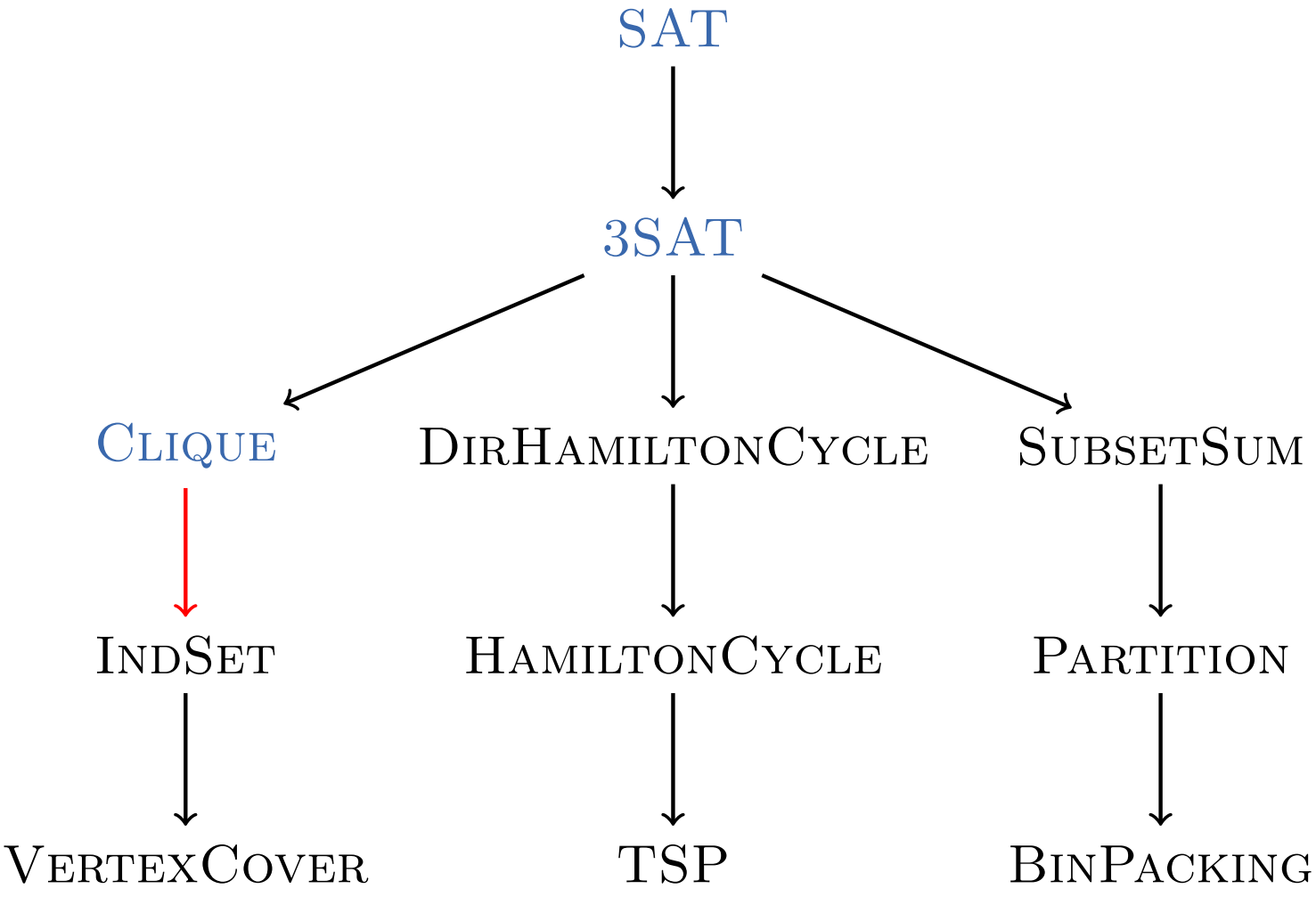
$\rightsquigarrow$ genau ein Knoten pro Klausel wird ausgewählt

- Die gewählten Knoten können nicht komplementären Literalen X und $\neg X$ entsprechen.
- Wenn X ausgewählt wurde, belege X mit **wahr**.
- Wenn $\neg X$ ausgewählt wurde, belege X mit **falsch**.
- Wenn keiner von beiden ausgewählt wurde, belege X beliebig.

$\rightsquigarrow$ erfüllende Belegung

□

Clique $\leq_p$ IndSet



IndSet ist NP-vollständig

Definition (IndSet)

Das Problem IndSet ist wie folgt definiert:

Gegeben: ungerichteter Graph $G = \langle V, E \rangle$, Zahl $K \in \mathbb{N}_0$

Frage: Enthält G eine unabhängige Menge der Grösse K oder mehr, d. h. eine Knotenmenge $I \subseteq V$ mit $|I| \geq K$ und $\{u, v\} \notin E$ für alle $u, v \in I$ mit $u \neq v$?

IndSet ist NP-vollständig

Definition (IndSet)

Das Problem IndSet ist wie folgt definiert:

Gegeben: ungerichteter Graph $G = \langle V, E \rangle$, Zahl $K \in \mathbb{N}_0$

Frage: Enthält G eine unabhängige Menge der Grösse K oder mehr, d. h. eine Knotenmenge $I \subseteq V$ mit $|I| \geq K$ und $\{u, v\} \notin E$ für alle $u, v \in I$ mit $u \neq v$?

Satz (IndSet ist NP-vollständig)

IndSet *ist NP-vollständig.*

IndSet ist NP-vollständig

Definition (IndSet)

Das Problem IndSet ist wie folgt definiert:

Gegeben: ungerichteter Graph $G = \langle V, E \rangle$, Zahl $K \in \mathbb{N}_0$

Frage: Enthält G eine unabhängige Menge der Grösse K oder mehr, d. h. eine Knotenmenge $I \subseteq V$ mit $|I| \geq K$ und $\{u, v\} \notin E$ für alle $u, v \in I$ mit $u \neq v$?

Satz (IndSet ist NP-vollständig)

IndSet *ist NP-vollständig.*

Beweis.

IndSet $\in$ NP: Raten und Prüfen.

IndSet ist NP-vollständig

Definition (IndSet)

Das Problem IndSet ist wie folgt definiert:

Gegeben: ungerichteter Graph $G = \langle V, E \rangle$, Zahl $K \in \mathbb{N}_0$

Frage: Enthält G eine unabhängige Menge der Grösse K oder mehr, d. h. eine Knotenmenge $I \subseteq V$ mit $|I| \geq K$ und $\{u, v\} \notin E$ für alle $u, v \in I$ mit $u \neq v$?

Satz (IndSet ist NP-vollständig)

IndSet *ist NP-vollständig.*

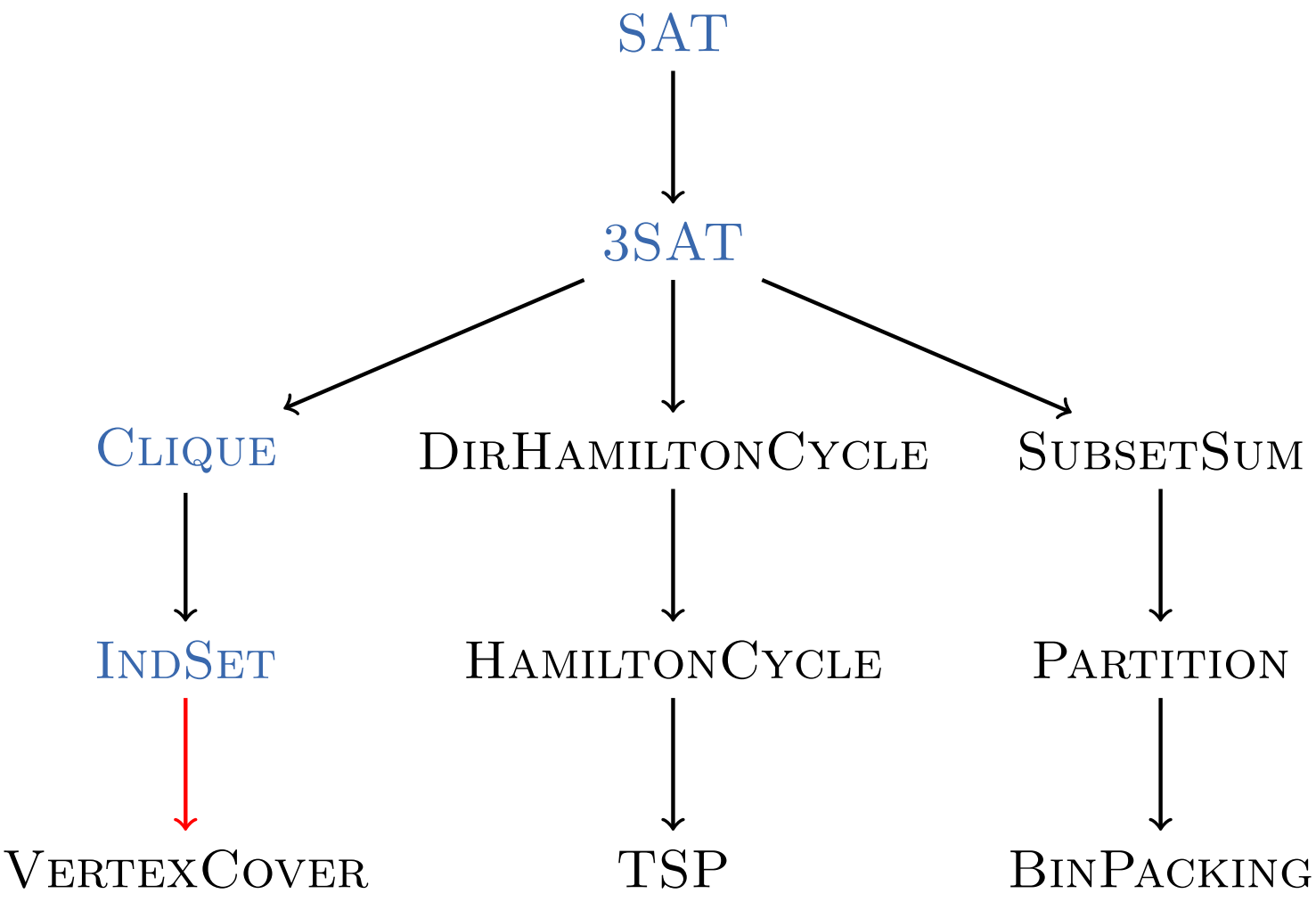
Beweis.

IndSet $\in$ NP: Raten und Prüfen.

IndSet ist NP-hart: Clique $\leq_p$ IndSet

$\rightsquigarrow$ Tafel $\square$

IndSet $\leq_p$ VertexCover



VertexCover ist NP-vollständig

Definition (VertexCover)

Das Problem VertexCover ist wie folgt definiert:

Gegeben: ungerichteter Graph $G = \langle V, E \rangle$, Zahl $K \in \mathbb{N}_0$

Frage: Enthält G eine Knotenüberdeckung der Grösse K oder weniger, d. h. eine Knotenmenge $C \subseteq V$ mit $|C| \leq K$ und $\{u, v\} \cap C \neq \emptyset$ für alle $\{u, v\} \in E$?

VertexCover ist NP-vollständig

Definition (VertexCover)

Das Problem VertexCover ist wie folgt definiert:

Gegeben: ungerichteter Graph $G = \langle V, E \rangle$, Zahl $K \in \mathbb{N}_0$

Frage: Enthält G eine Knotenüberdeckung der Grösse K oder weniger, d. h. eine Knotenmenge $C \subseteq V$ mit $|C| \leq K$ und $\{u, v\} \cap C \neq \emptyset$ für alle $\{u, v\} \in E$?

Satz (VertexCover ist NP-vollständig)

VertexCover *ist NP-vollständig.*

VertexCover ist NP-vollständig

Definition (VertexCover)

Das Problem VertexCover ist wie folgt definiert:

Gegeben: ungerichteter Graph $G = \langle V, E \rangle$, Zahl $K \in \mathbb{N}_0$

Frage: Enthält G eine Knotenüberdeckung der Grösse K oder weniger, d. h. eine Knotenmenge $C \subseteq V$ mit $|C| \leq K$ und $\{u, v\} \cap C \neq \emptyset$ für alle $\{u, v\} \in E$?

Satz (VertexCover ist NP-vollständig)

VertexCover *ist NP-vollständig.*

Beweis.

VertexCover $\in$ NP: Raten und Prüfen.

VertexCover ist NP-vollständig

Definition (VertexCover)

Das Problem VertexCover ist wie folgt definiert:

Gegeben: ungerichteter Graph $G = \langle V, E \rangle$, Zahl $K \in \mathbb{N}_0$

Frage: Enthält G eine Knotenüberdeckung der Grösse K oder weniger, d. h. eine Knotenmenge $C \subseteq V$ mit $|C| \leq K$ und $\{u, v\} \cap C \neq \emptyset$ für alle $\{u, v\} \in E$?

Satz (VertexCover ist NP-vollständig)

VertexCover *ist NP-vollständig.*

Beweis.

VertexCover $\in$ NP: Raten und Prüfen.

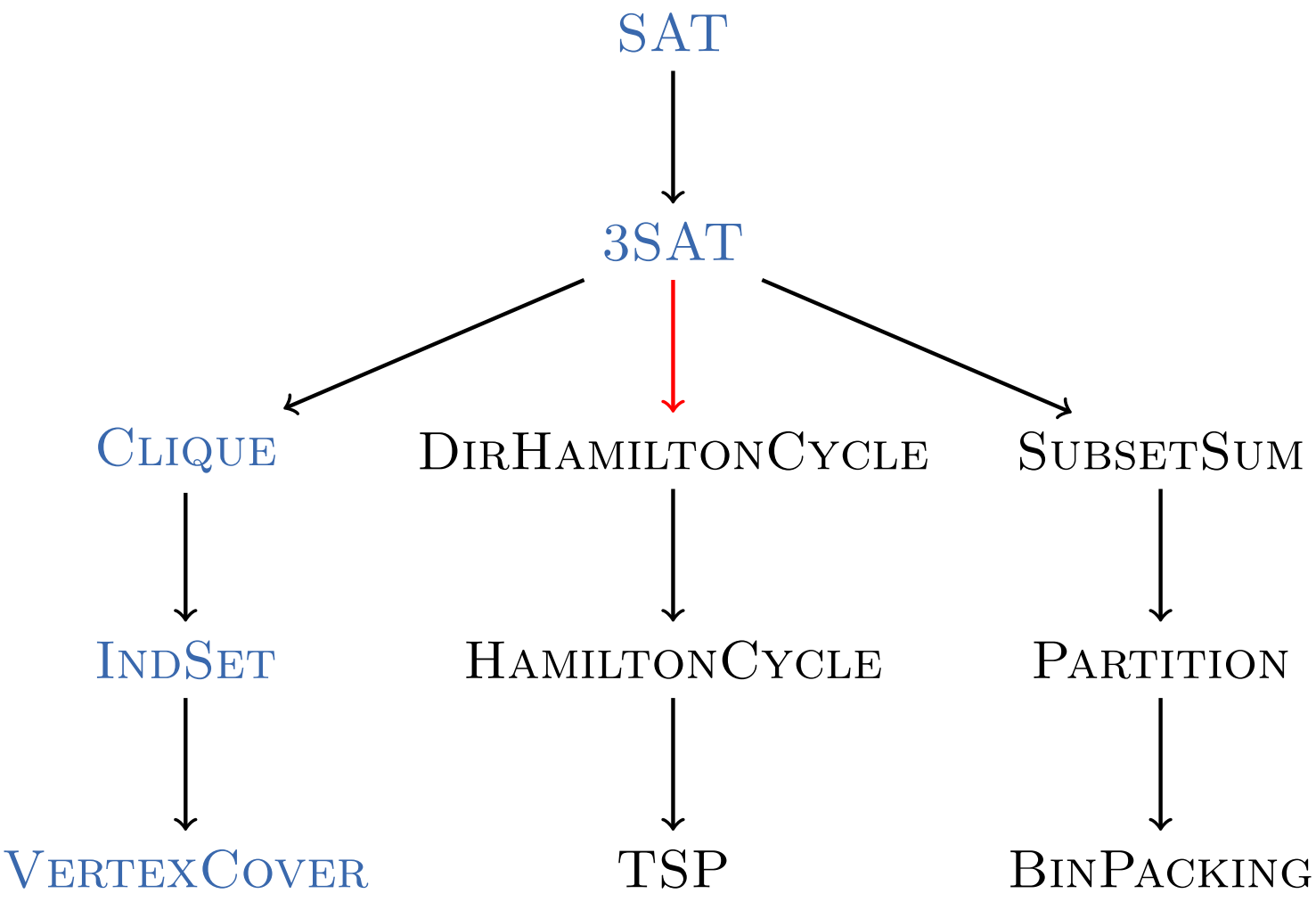
VertexCover ist NP-hart: IndSet $\leq_p$ VertexCover

$\rightsquigarrow$ Tafel

□

Routing-Probleme

3SAT $\leq_p$ DirHamiltonCycle



DirHamiltonCycle ist NP-vollständig (1)

Definition (DirHamiltonCycle; Wiederholung)

Das Problem DirHamiltonCycle ist wie folgt definiert:

Gegeben: gerichteter Graph $G = \langle V, E \rangle$

Frage: Enthält G einen Hamiltonkreis?

DirHamiltonCycle ist NP-vollständig (1)

Definition (DirHamiltonCycle; Wiederholung)

Das Problem DirHamiltonCycle ist wie folgt definiert:

Gegeben: gerichteter Graph $G = \langle V, E \rangle$

Frage: Enthält G einen Hamiltonkreis?

Satz (DirHamiltonCycle ist NP-vollständig)

DirHamiltonCycle *ist NP-vollständig.*

DirHamiltonCycle ist NP-vollständig (2)

Beweis.

DirHamiltonCycle $\in$ NP: Raten und Prüfen.

DirHamiltonCycle ist NP-vollständig (2)

Beweis.

DirHamiltonCycle $\in$ NP: Raten und Prüfen.

DirHamiltonCycle ist NP-hart:

$3\text{SAT} \leq_p \text{DirHamiltonCycle}$

DirHamiltonCycle ist NP-vollständig (2)

Beweis.

DirHamiltonCycle $\in$ NP: Raten und Prüfen.

DirHamiltonCycle ist NP-hart:
$3\text{SAT} \leq_p \text{DirHamiltonCycle}$

- Sei 3-KNF-Formel φ gegeben, bei der jede Klauseln genau drei Literale enthält und keine Klausel duplizierte Literale enthält.
- Wir müssen in polynomieller Zeit gerichteten Graphen $G = \langle V, E \rangle$ konstruieren, so dass gilt:
  G enthält einen Hamiltonkreis gdw. φ ist erfüllbar.
- Konstruktion von $\langle V, E \rangle$ auf folgenden Folien.

...

DirHamiltonCycle ist NP-vollständig (3)

Beweis (Fortsetzung).

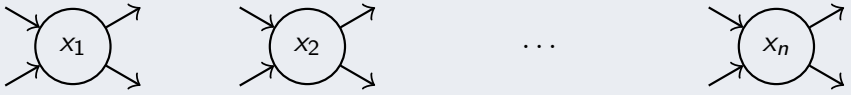
- Seien $v_1, \ldots, v_n$ die Aussagevariablen in φ.
- Seien $c_1, \ldots, c_m$ die Klauseln von φ mit $c_i = l_{i1} \vee l_{i2} \vee l_{i3}$.
- Erzeuge Graph mit $6m + n$ Knoten
  (auf folgenden Folien beschrieben).

...

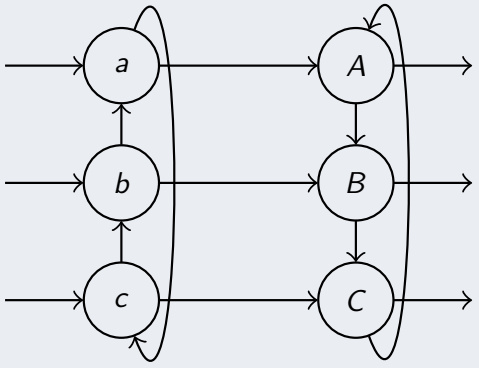
DirHamiltonCycle ist NP-vollständig (4)

Beweis (Fortsetzung).

- Für jede Variable v_i, führe Knoten x_i mit 2 eingehenden und 2 ausgehenden Kanten ein:



- Für jede Klausel c_j führe Teilgraph C_j mit 6 Knoten ein:



...

DirHamiltonCycle ist NP-vollständig (5)

Beweis (Fortsetzung).

Sei π ein Hamiltonkreis des Gesamtgraphen.

- Wann immer π Teilgraphen C_j durchläuft, muss der „Ausgang“ zum „Eingang“ korrespondieren ($a \longrightarrow A$, $b \longrightarrow B$, $c \longrightarrow C$).
  Sonst kann π kein Hamiltonkreis sein.
- Hamiltonkreise können sich bezüglich C_j wie folgt verhalten:
  - π durchläuft C_j einmal, durch einen beliebigen Eingang
  - π durchläuft C_j zweimal, durch zwei beliebige Eingänge
  - π durchläuft C_j dreimal, durch jeden Eingang einmal

...

DirHamiltonCycle ist NP-vollständig (6)

Beweis (Fortsetzung).

Verbinde „offene Enden“ des Graphen wie folgt:

- Identifiziere Eingänge/Ausgänge der C_j-Graphen mit den drei Literalen in Klausel c_j.
- Ein Ausgang von x_i ist positiv, der andere negativ.
- Für den positiven Ausgang, bestimme die Klauseln, in denen das positive Literal v_i auftritt:
  - Verbinde den positiven Ausgang von x_i mit dem v_i-Eingang des ersten solchen Klauselgraphen.
  - Verbinde den v_i-Ausgang dieses Klauselgraphen mit dem v_i-Eingang des zweiten solchen Klauselgraphen und so weiter.
  - Verbinde den v_i-Ausgang der letzten solchen Klausel mit dem positiven Eingang von x_{i+1} (bzw. x_1 falls $i = n$)
- analog für den negativen Ausgang von x_i und das Literal $\neg v_i$

...

DirHamiltonCycle ist NP-vollständig (7)

Beweis (Fortsetzung).

Die Konstruktion ist polynomiell und ist eine Reduktion:

($\Rightarrow$): Konstruiere Hamiltonkreis zu erfüllender Belegung

- Gegeben erfüllende Belegung α konstruiere Hamiltonkreis, der x_i durch positiven Ausgang verlässt, falls $\alpha(v_i)$ wahr ist und durch den negativen Ausgang, wenn $\alpha(v_i)$ falsch ist.
- Anschliessend besuchen wir alle C_j-Graphen für die Klauseln, die durch dieses Literal erfüllt werden.
- Insgesamt besuchen wir jeden C_j-Graphen 1–3 mal.

...

DirHamiltonCycle ist NP-vollständig (8)

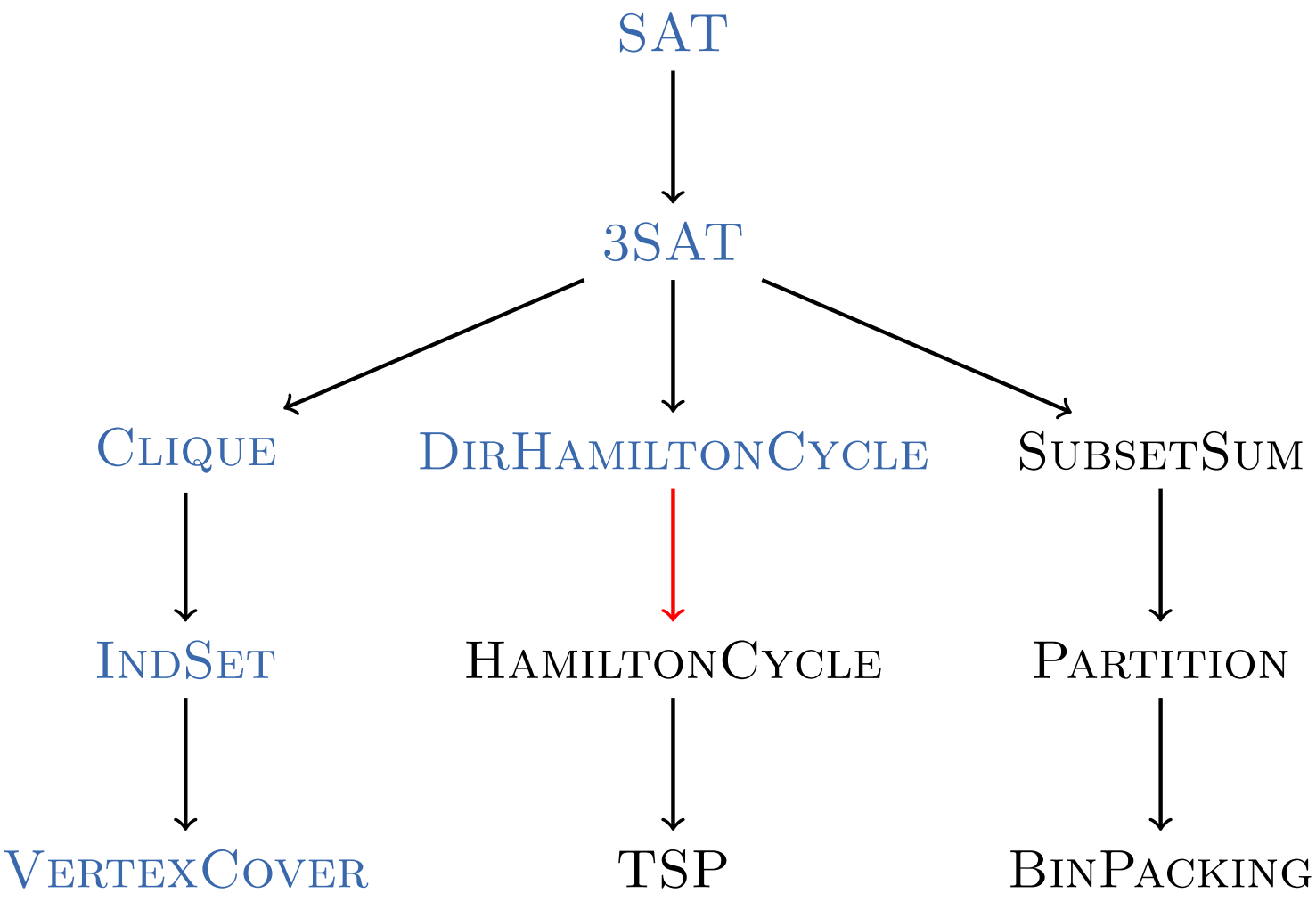
Beweis (Fortsetzung).

($\Leftarrow$): Konstruiere erfüllende Belegung zu Hamiltonkreis

- Ein Hamiltonkreis besucht jeden Knoten x_i und verlässt ihn durch den positiven oder negativen Ausgang.
- Setze v_i auf wahr oder falsch je nach dem, durch welchen Ausgang x_i verlassen wird.
- Dies ergibt eine erfüllende Belegung. (Details ausgelassen.)

□

DirHamiltonCycle $\leq_p$ HamiltonCycle



HamiltonCycle ist NP-vollständig (1)

Definition (HamiltonCycle; Wiederholung)

Das Problem HamiltonCycle ist wie folgt definiert:

Gegeben: ungerichteter Graph $G = \langle V, E \rangle$

Frage: Enthält G einen Hamiltonkreis?

HamiltonCycle ist NP-vollständig (1)

Definition (HamiltonCycle; Wiederholung)

Das Problem HamiltonCycle ist wie folgt definiert:

Gegeben: ungerichteter Graph $G = \langle V, E \rangle$

Frage: Enthält G einen Hamiltonkreis?

Satz (HamiltonCycle ist NP-vollständig)

HamiltonCycle *ist NP-vollständig.*

HamiltonCycle ist NP-vollständig (2)

Beweisskizze.

HamiltonCycle $\in$ NP: Raten und Prüfen.

HamiltonCycle ist NP-vollständig (2)

Beweisskizze.

HamiltonCycle $\in$ NP: Raten und Prüfen.

HamiltonCycle ist NP-hart:

DirHamiltonCycle $\leq_p$ HamiltonCycle

HamiltonCycle ist NP-vollständig (2)

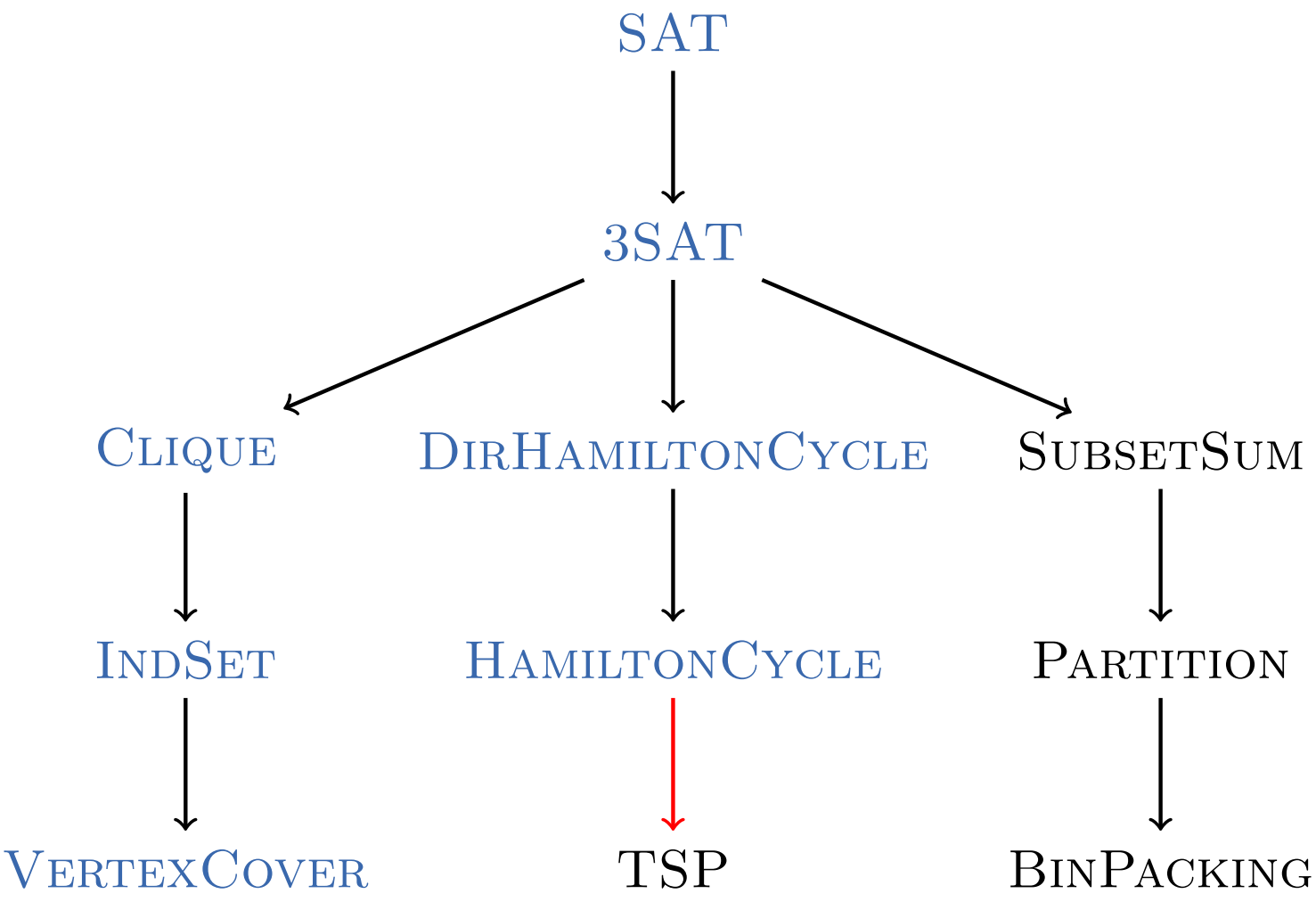
Beweisskizze.

HamiltonCycle $\in$ NP: Raten und Prüfen.

HamiltonCycle ist NP-hart:
DirHamiltonCycle $\leq_p$ HamiltonCycle

Grundbaustein der Reduktion:

HamiltonCycle $\leq_p$ TSP



TSP ist NP-vollständig

Definition (TSP; Wiederholung)

Das Problem TSP ist wie folgt definiert:

Gegeben: Menge S von Städten (endlich, nicht-leer), symmetrische Kostenfunktion $cost : S \times S \to \mathbb{N}_0$, Kostenschranke $K \in \mathbb{N}_0$

Frage: Gibt es eine Rundreise mit Gesamtkosten höchstens K, d. h. eine Permutation $\langle s_1, \ldots, s_n \rangle$ der Städte, so dass $\sum_{i=1}^{n-1} cost(s_i, s_{i+1}) + cost(s_n, s_1) \leq K$?

TSP ist NP-vollständig

Definition (TSP; Wiederholung)

Das Problem TSP ist wie folgt definiert:

Gegeben: Menge S von Städten (endlich, nicht-leer), symmetrische Kostenfunktion $cost : S \times S \to \mathbb{N}_0$, Kostenschranke $K \in \mathbb{N}_0$

Frage: Gibt es eine Rundreise mit Gesamtkosten höchstens K, d. h. eine Permutation $\langle s_1, \ldots, s_n \rangle$ der Städte, so dass $\sum_{i=1}^{n-1} cost(s_i, s_{i+1}) + cost(s_n, s_1) \leq K$?

Satz (TSP ist NP-vollständig)

TSP *ist NP-vollständig.*

TSP ist NP-vollständig

Definition (TSP; Wiederholung)

Das Problem TSP ist wie folgt definiert:

Gegeben: Menge S von Städten (endlich, nicht-leer), symmetrische Kostenfunktion $cost : S \times S \to \mathbb{N}_0$, Kostenschranke $K \in \mathbb{N}_0$

Frage: Gibt es eine Rundreise mit Gesamtkosten höchstens K, d. h. eine Permutation $\langle s_1, \ldots, s_n \rangle$ der Städte, so dass $\sum_{i=1}^{n-1} cost(s_i, s_{i+1}) + cost(s_n, s_1) \leq K$?

Satz (TSP ist NP-vollständig)

TSP *ist NP-vollständig.*

Beweis.

$\mathrm{TSP} \in \mathrm{NP}$: Raten und Prüfen.

TSP ist NP-vollständig

Definition (TSP; Wiederholung)

Das Problem TSP ist wie folgt definiert:

Gegeben: Menge S von Städten (endlich, nicht-leer), symmetrische Kostenfunktion $cost : S \times S \to \mathbb{N}_0$, Kostenschranke $K \in \mathbb{N}_0$

Frage: Gibt es eine Rundreise mit Gesamtkosten höchstens K, d. h. eine Permutation $\langle s_1, \ldots, s_n \rangle$ der Städte, so dass $\sum_{i=1}^{n-1} cost(s_i, s_{i+1}) + cost(s_n, s_1) \leq K$?

Satz (TSP ist NP-vollständig)

TSP *ist NP-vollständig.*

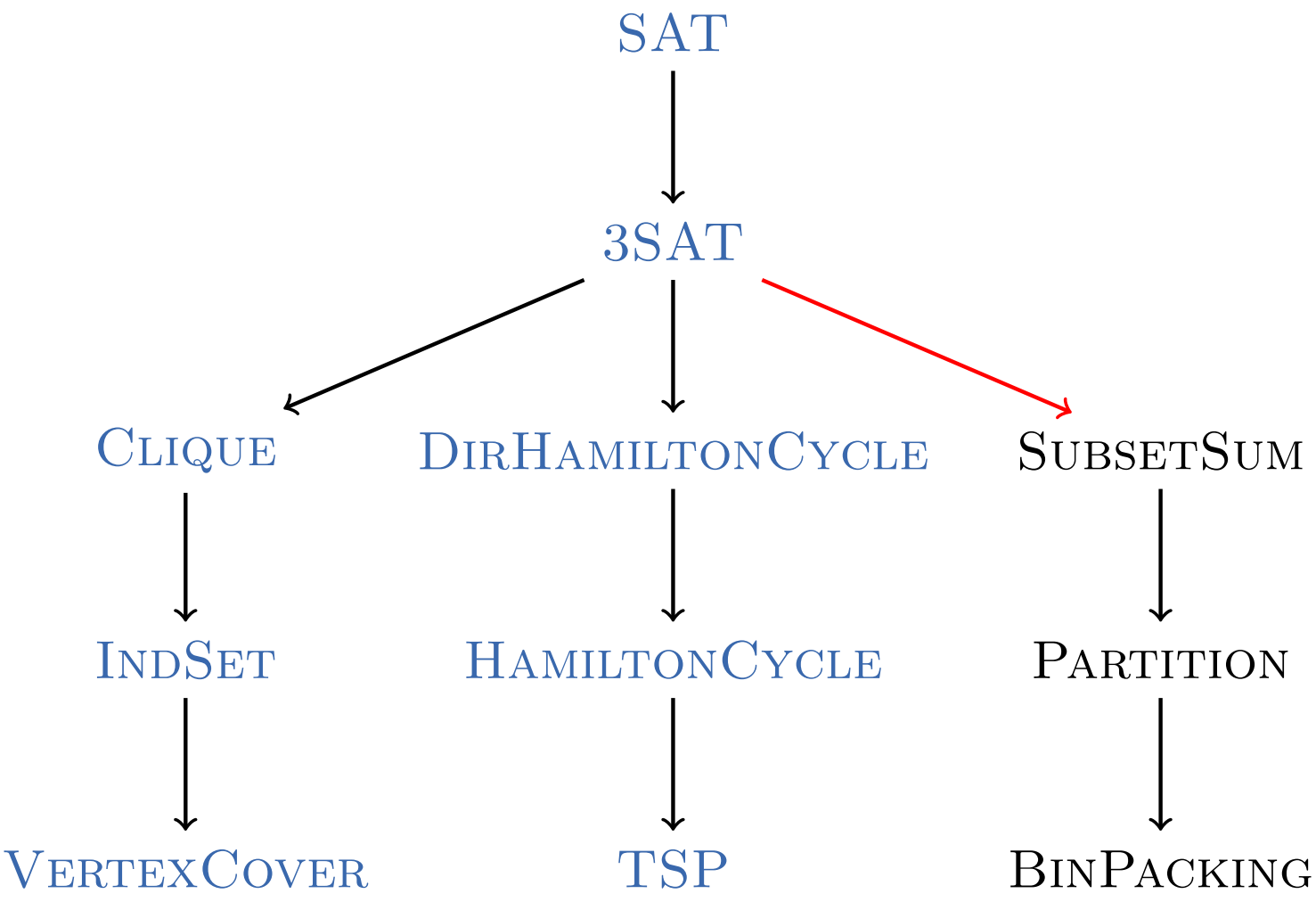
Beweis.

TSP $\in$ NP: Raten und Prüfen.

TSP ist NP-hart: HamiltonCycle $\leq_p$ TSP in Kapitel 19. $\square$

Packungsprobleme

3SAT $\leq_p$ SubsetSum



SubsetSum ist NP-vollständig

Definition (SubsetSum)

Das Problem SubsetSum ist wie folgt definiert:

Gegeben: Zahlen $a_1, \ldots, a_k \in \mathbb{N}_0$ und $b \in \mathbb{N}_0$

Frage: Gibt es eine Teilmenge $J \subseteq \{1, \ldots, k\}$ mit $\sum_{i \in J} a_i = b$?

SubsetSum ist NP-vollständig

Definition (SubsetSum)

Das Problem SubsetSum ist wie folgt definiert:

Gegeben: Zahlen $a_1, \ldots, a_k \in \mathbb{N}_0$ und $b \in \mathbb{N}_0$

Frage: Gibt es eine Teilmenge $J \subseteq \{1, \ldots, k\}$ mit $\sum_{i \in J} a_i = b$?

Satz (SubsetSum ist NP-vollständig)

SubsetSum *ist NP-vollständig.*

Beweis.

SubsetSum $\in$ NP: Raten und Prüfen.

SubsetSum ist NP-vollständig

Definition (SubsetSum)

Das Problem SubsetSum ist wie folgt definiert:

Gegeben: Zahlen $a_1, \ldots, a_k \in \mathbb{N}_0$ und $b \in \mathbb{N}_0$

Frage: Gibt es eine Teilmenge $J \subseteq \{1, \ldots, k\}$ mit $\sum_{i \in J} a_i = b$?

Satz (SubsetSum ist NP-vollständig)

SubsetSum *ist NP-vollständig.*

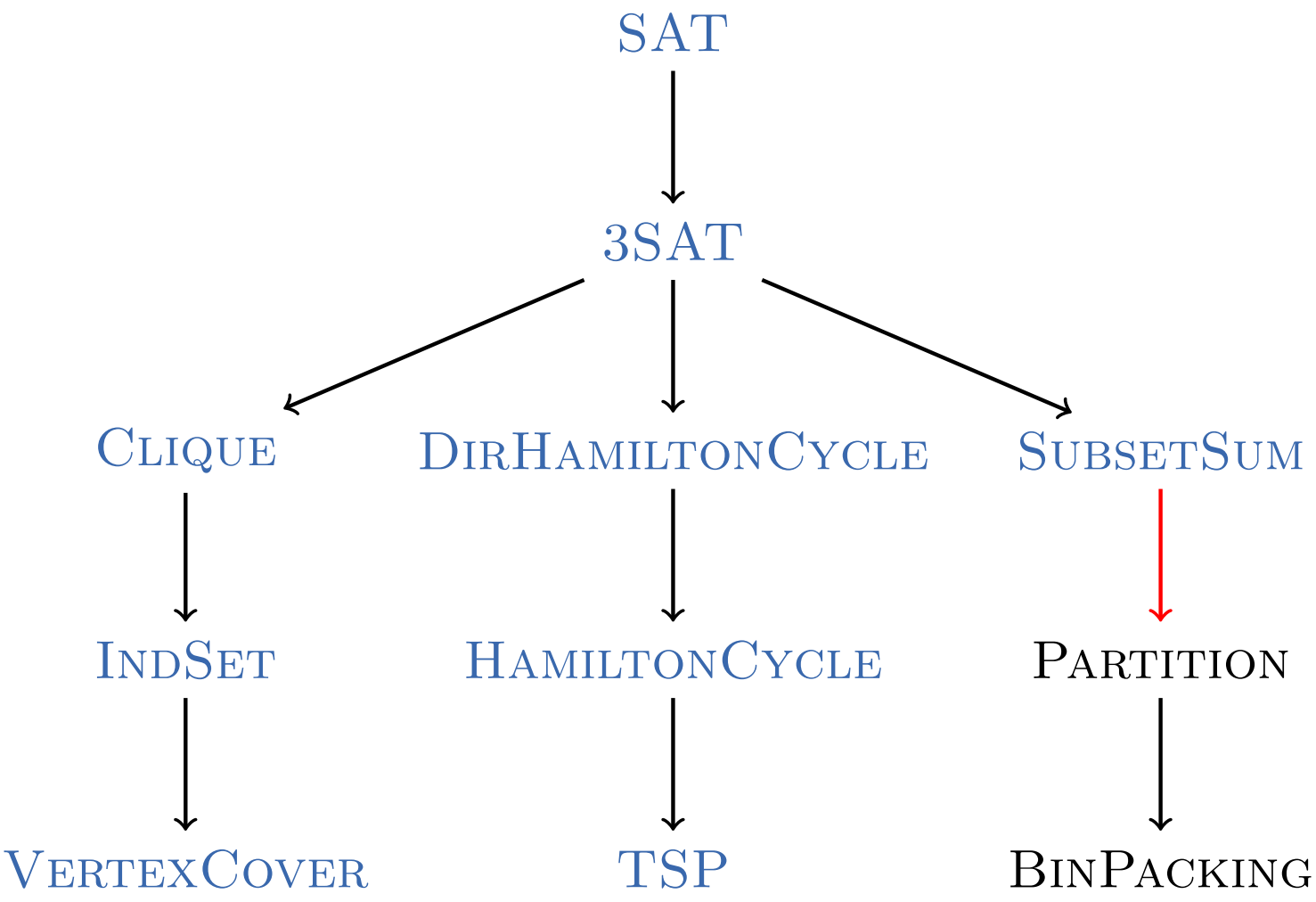
Beweis.

SubsetSum $\in$ NP: Raten und Prüfen.

SubsetSum ist NP-hart: $3\mathrm{SAT} \leq_p \mathrm{SubsetSum}$

(Beweis ausgelassen)

SubsetSum $\leq_p$ Partition



Partition ist NP-vollständig (1)

Definition (Partition)

Das Problem Partition ist wie folgt definiert:

Gegeben: Zahlen $a_1, \ldots, a_k \in \mathbb{N}_0$

Frage: Gibt es eine Teilmenge $J \subseteq \{1, \ldots, k\}$ mit $\sum_{i \in J} a_i = \sum_{i \in \{1,\ldots,k\} \setminus J} a_i$?

Partition ist NP-vollständig (1)

Definition (Partition)

Das Problem Partition ist wie folgt definiert:

Gegeben: Zahlen $a_1, \ldots, a_k \in \mathbb{N}_0$

Frage: Gibt es eine Teilmenge $J \subseteq \{1, \ldots, k\}$ mit $\sum_{i \in J} a_i = \sum_{i \in \{1,\ldots,k\} \setminus J} a_i$?

Satz (Partition ist NP-vollständig)

Partition *ist NP-vollständig.*

Partition ist NP-vollständig (2)

Beweis.

Partition $\in$ NP: Raten und Prüfen.

Partition ist NP-vollständig (2)

Beweis.

Partition $\in$ NP: Raten und Prüfen.

Partition ist NP-hart: SubsetSum $\leq_p$ Partition
Gegeben sei eine SubsetSum-Instanz mit Zahlen $a_1, \ldots, a_k$ und Zielgrösse b. Sei $M := \sum_{i=1}^{k} a_i$.

Partition ist NP-vollständig (2)

Beweis.

Partition $\in$ NP: Raten und Prüfen.

Partition ist NP-hart: SubsetSum $\leq_p$ Partition

Gegeben sei eine SubsetSum-Instanz mit Zahlen $a_1, \ldots, a_k$ und Zielgrösse b. Sei $M := \sum_{i=1}^{k} a_i$.

Erzeuge die Partition-Instanz $a_1, \ldots, a_k, M - b + 1, b + 1$ (offenbar in polynomieller Zeit berechenbar).

Partition ist NP-vollständig (2)

Beweis.

Partition $\in$ NP: Raten und Prüfen.

Partition ist NP-hart: SubsetSum $\leq_p$ Partition
Gegeben sei eine SubsetSum-Instanz mit Zahlen $a_1, \ldots, a_k$ und Zielgrösse b. Sei $M := \sum_{i=1}^{k} a_i$.

Erzeuge die Partition-Instanz $a_1, \ldots, a_k, M - b + 1, b + 1$ (offenbar in polynomieller Zeit berechenbar).

Beobachtung: die Summe dieser Zahlen ist
$M + (M - b + 1) + (b + 1) = 2M + 2$
$\rightsquigarrow$ eine Lösung teilt die Zahlen in zwei Teile mit Summe $M + 1$

...

Partition ist NP-vollständig (3)

Beweis (Fortsetzung).

Reduktionseigenschaft:

($\Rightarrow$): Konstruiere Partition-Lösung aus SubsetSum-Lösung

Partition ist NP-vollständig (3)

Beweis (Fortsetzung).

Reduktionseigenschaft:

($\Rightarrow$): Konstruiere Partition-Lösung aus SubsetSum-Lösung

- Sei $J \subseteq \{1, \ldots, k\}$ eine SubsetSum-Lösung, d. h. $\sum_{i \in J} a_i = b$.

Partition ist NP-vollständig (3)

Beweis (Fortsetzung).

Reduktionseigenschaft:

($\Rightarrow$): Konstruiere Partition-Lösung aus SubsetSum-Lösung

- Sei $J \subseteq \{1, \ldots, k\}$ eine SubsetSum-Lösung, d. h. $\sum_{i \in J} a_i = b$.
- Dann ist J zusammen mit (dem Index zu) $M - b + 1$ eine Partition-Lösung, denn
  $\sum_{i \in J} a_i + (M - b + 1) = b + M - b + 1 = M + 1$
  (und somit addieren sich auch die restlichen Zahlen zu $M + 1$)

...

Partition ist NP-vollständig (4)

Beweis (Fortsetzung).

($\Leftarrow$): Konstruiere SubsetSum-Lösung aus Partition-Lösung

Partition ist NP-vollständig (4)

Beweis (Fortsetzung).

($\Leftarrow$): Konstruiere SubsetSum-Lösung aus Partition-Lösung

- einer der beiden Teile der Partition enthält die Zahl $M - b + 1$

Partition ist NP-vollständig (4)

Beweis (Fortsetzung).

($\Leftarrow$): Konstruiere SubsetSum-Lösung aus Partition-Lösung

- einer der beiden Teile der Partition enthält die Zahl $M - b + 1$
- Dann summieren sich die restlichen Zahlen in diesem Teil zu $(M + 1) - (M - b + 1) = b$.

Partition ist NP-vollständig (4)

Beweis (Fortsetzung).

($\Leftarrow$): Konstruiere SubsetSum-Lösung aus Partition-Lösung

- einer der beiden Teile der Partition enthält die Zahl $M - b + 1$
- Dann summieren sich die restlichen Zahlen in diesem Teil zu $(M + 1) - (M - b + 1) = b$.

$\rightsquigarrow$ Diese restlichen Zahlen müssen Indizes aus $\{1, \ldots, k\}$ haben, denn $M - b + 1$ gehört nicht dazu und $b + 1$ ist zu gross.

Partition ist NP-vollständig (4)

Beweis (Fortsetzung).

($\Leftarrow$): Konstruiere SubsetSum-Lösung aus Partition-Lösung

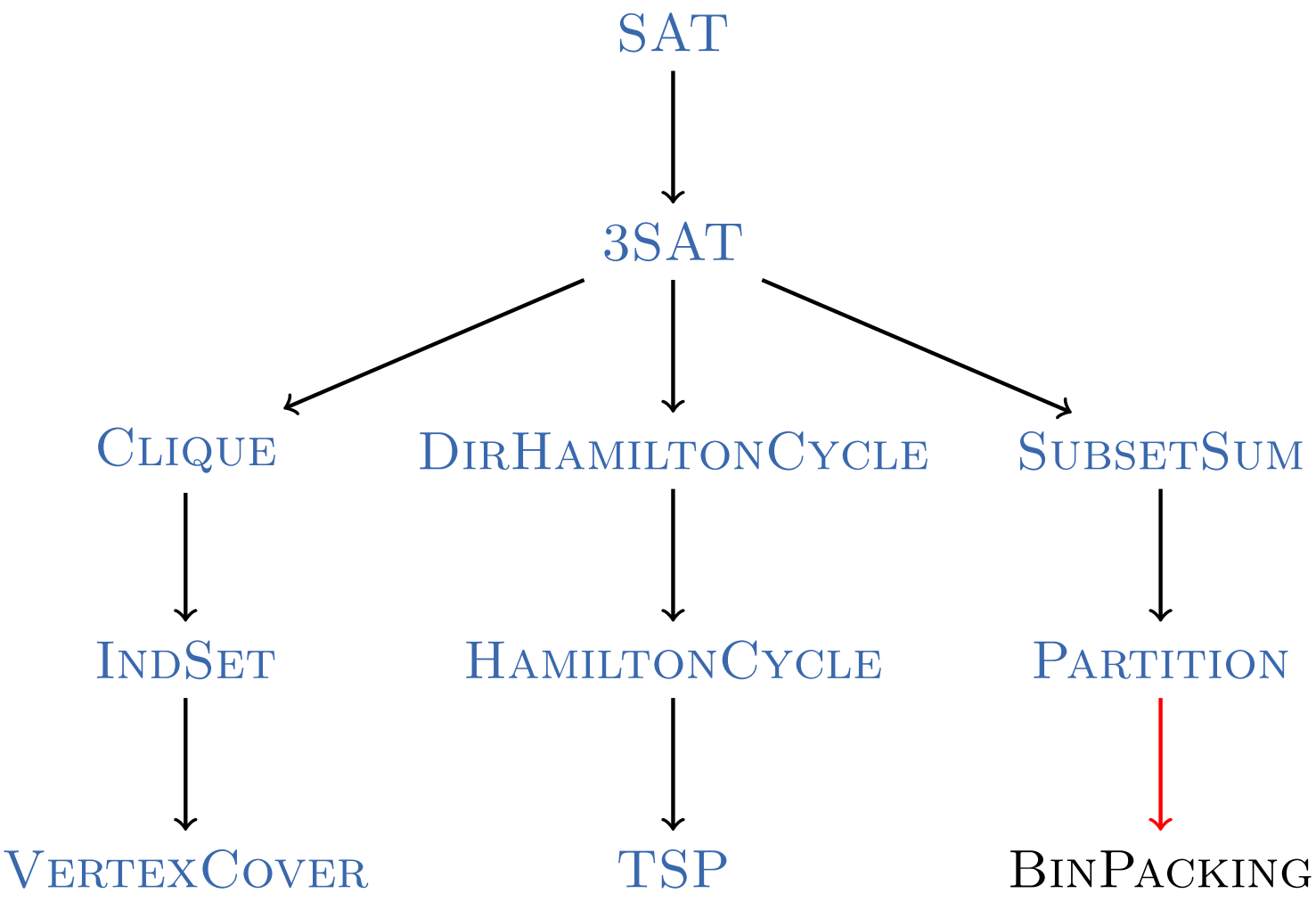
- einer der beiden Teile der Partition enthält die Zahl $M - b + 1$
- Dann summieren sich die restlichen Zahlen in diesem Teil zu $(M + 1) - (M - b + 1) = b$.

$\rightsquigarrow$ Diese restlichen Zahlen müssen Indizes aus $\{1, \ldots, k\}$ haben, denn $M - b + 1$ gehört nicht dazu und $b + 1$ ist zu gross.

$\rightsquigarrow$ Diese Zahlen bilden eine SubsetSum-Lösung.

□

Partition $\leq_p$ BinPacking



BinPacking ist NP-vollständig (1)

Definition (BinPacking)

Das Problem BinPacking ist wie folgt definiert:

Gegeben: Behältergrösse $b \in \mathbb{N}_0$, Anzahl Behälter $k \in \mathbb{N}_0$, Objekte $a_1, \ldots, a_n \in \mathbb{N}_0$

Frage: Passen die Objekte in die Behälter?
Formal: gibt es eine Abbildung $f : \{1, \ldots, n\} \to \{1, \ldots, k\}$ mit $\sum_{i \in \{1,\ldots,n\} \text{ mit } f(i)=j} a_i \leq b$ für alle $1 \leq j \leq k$?

BinPacking ist NP-vollständig (1)

Definition (BinPacking)

Das Problem BinPacking ist wie folgt definiert:

Gegeben: Behältergrösse $b \in \mathbb{N}_0$, Anzahl Behälter $k \in \mathbb{N}_0$, Objekte $a_1, \ldots, a_n \in \mathbb{N}_0$

Frage: Passen die Objekte in die Behälter?
Formal: gibt es eine Abbildung $f : \{1, \ldots, n\} \to \{1, \ldots, k\}$ mit $\sum_{i \in \{1,\ldots,n\} \text{ mit } f(i)=j} a_i \leq b$ für alle $1 \leq j \leq k$?

Satz (BinPacking ist NP-vollständig)

BinPacking *ist NP-vollständig.*

BinPacking ist NP-vollständig (2)

Beweis.

BinPacking $\in$ NP: Raten und Prüfen.

BinPacking ist NP-vollständig (2)

Beweis.

BinPacking $\in$ NP: Raten und Prüfen.

BinPacking ist NP-hart: Partition $\leq_p$ BinPacking

$\rightsquigarrow$ Tafel

□

Fazit

. . . und noch viel mehr

Weitere Beispiele NP-vollständiger Probleme:

- 3-Coloring: können die Knoten eines Graphen so mit drei Farben eingefärbt werden, dass benachbarte Knoten nie dieselbe Farbe haben?
- MinesweeperConsistency: Ist in einer gegebenen Minesweeper-Konfiguration ein gegebenes Feld sicher?
- GeneralizedFreeCell: Ist ein gegebenes verallgemeinertes FreeCell-Tableau (d. h. eines mit potenziell mehr als 52 Karten) lösbar?
- . . . und viele, viele mehr

Zusammenfassung

- Komplexitätstheorie untersucht, welche Probleme „einfach“ zu lösen sind und welche „schwierig“.
- Zwei wichtige Problemklassen:
  - P: Probleme, die mit üblichen Berechnungsmechanismen in polynomieller Zeit lösbar sind
  - NP: Probleme, die mit Hilfe von Nichtdeterminismus in polynomieller Zeit lösbar sind
- Wir wissen, dass $P \subseteq NP$, aber wir wissen nicht, ob $P = NP$.
- Viele praktisch wichtige Probleme sind NP-vollständig:
  - Sie gehören zu NP.
  - Alle Probleme in NP können auf sie reduziert werden.
- Falls es einen effizienten Algorithmus für ein NP-vollständiges Problem gibt, dann gibt es effiziente Algorithmen für alle Probleme in NP.