Einordnung von Problemen in P, NP, PSPACE

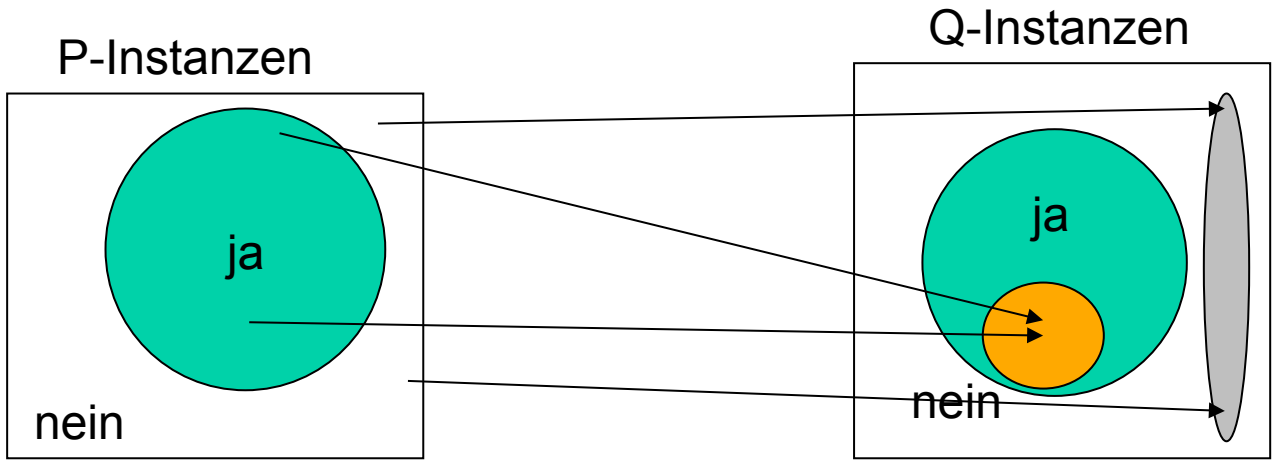
TECHNISCHE UNIVERSITÄT DARMSTADT

- **Angabe eines Algorithmus für Problem**
- **Reduktionstechnik steht im Zentrum**

Definition: Seien P,Q Probleme. Sei L_P (L_Q) die Menge der Instanzen des Problems P (Q), für die die Antwort „ja“ ist. P heißt auf Q polynomiell reduzierbar ($P \leq_p Q$), wenn es eine von einem deterministischen Algorithmus in Polynomzeit berechenbare Funktion f: $\Sigma^* \rightarrow \Sigma^*$ gibt, so dass

$$x \in L_P \Leftrightarrow f(x) \in L_Q$$

Bsp.:

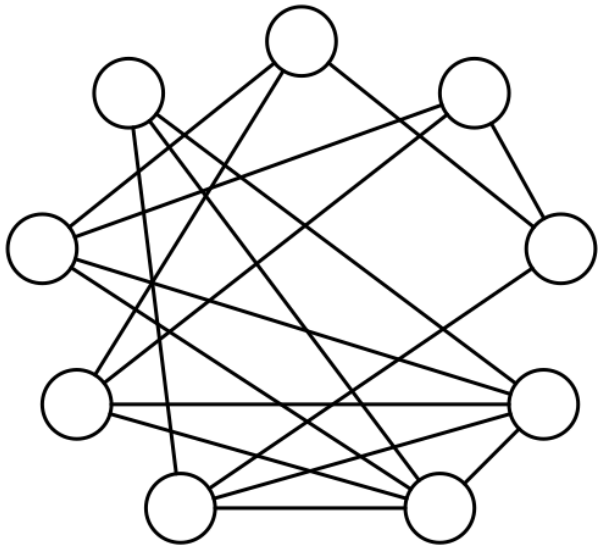


Das 3-SAT-Problem und das Clique-Problem

TECHNISCHE UNIVERSITÄT DARMSTADT

- 3-SAT:
  - **Gegeben**:
    - Eine Boolesche Formel in 3-CNF
  - **Gesucht**:
    - Gibt es eine erfüllende Belegung
- Definition k-Clique
  - Ein ungerichteter Graph G=(V,E) hat eine k-Clique,
    - falls es k verschiedene Knoten gibt,
    - so dass jeder mit jedem anderen eine Kante in G verbindet
- CLIQUE:
  - **Gegeben**:
    - Ein ungerichteter Graph G
    - Eine Zahl k
  - **Gesucht**:
    - Hat der Graph G eine Clique der Größe k?

$$\psi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$

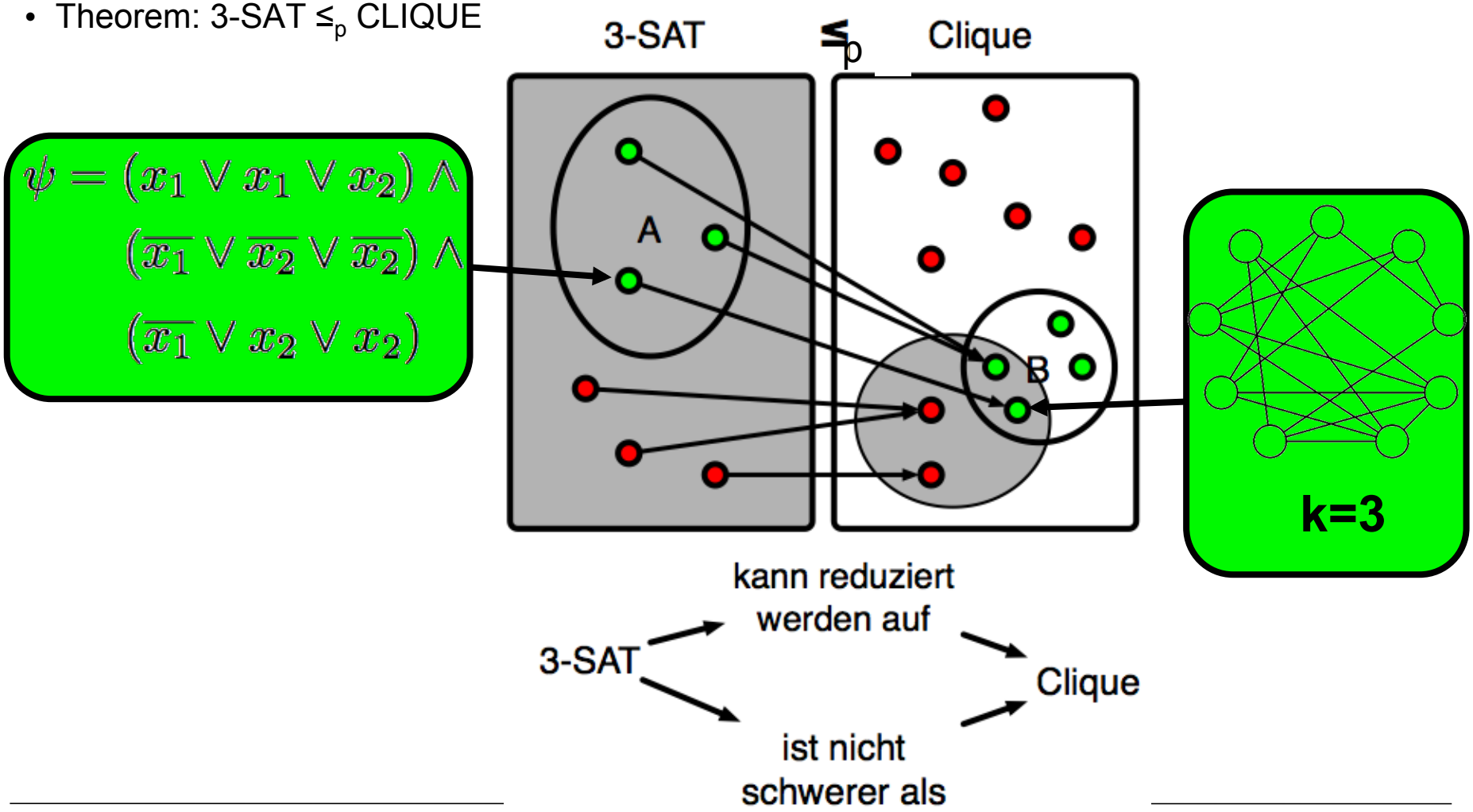


k=3

3-SAT läßt sich auf Clique reduzieren

TECHNISCHE UNIVERSITÄT DARMSTADT

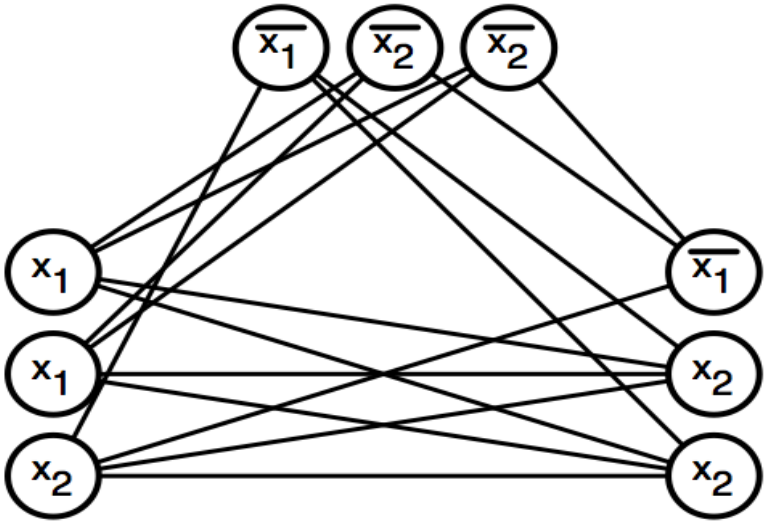
- Theorem: 3-SAT $\leq_p$ CLIQUE



3-SAT läßt sich auf Clique reduzieren

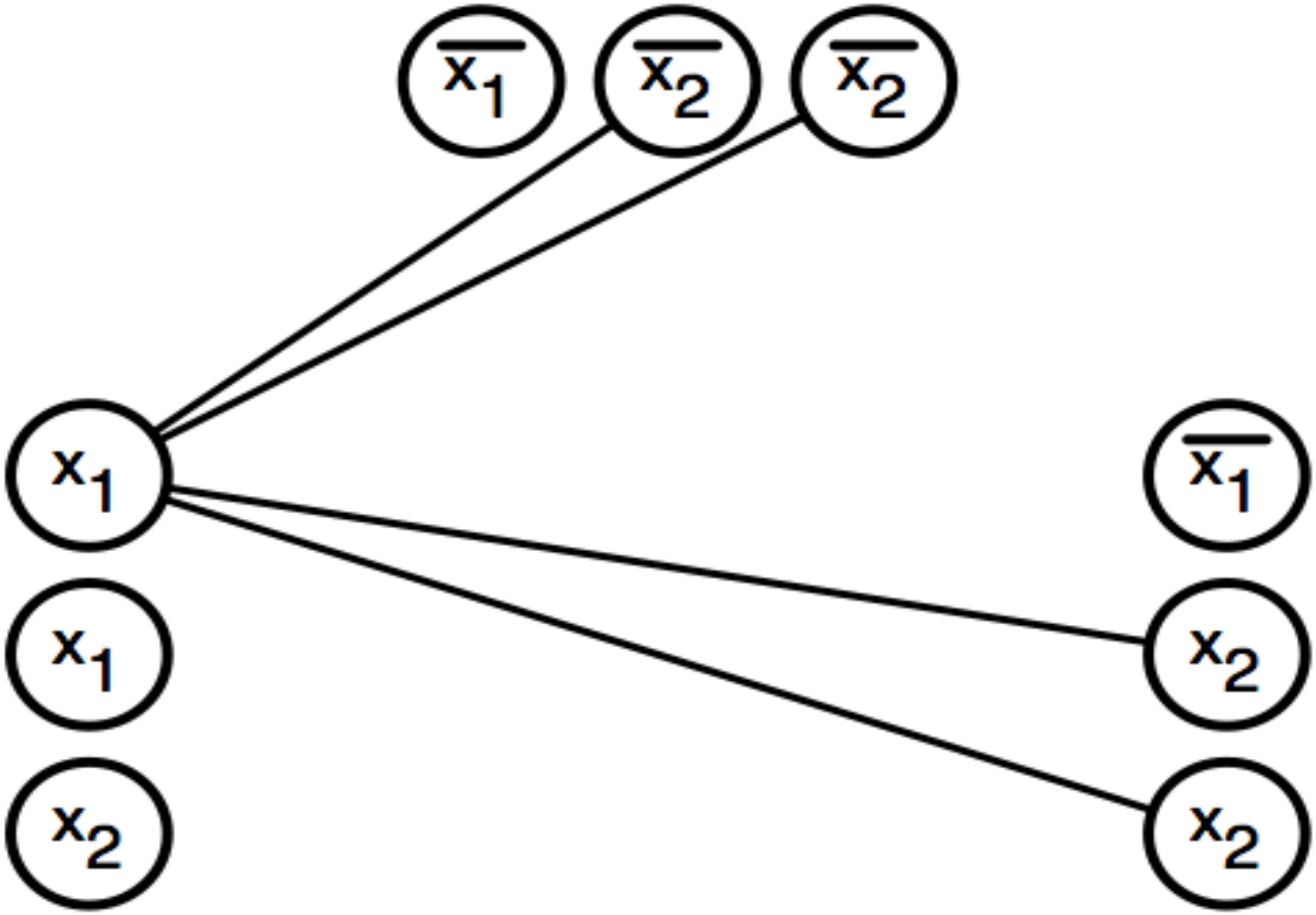
TECHNISCHE UNIVERSITÄT DARMSTADT

- Theorem: 3-SAT $\leq_p$ CLIQUE
- Beweis
  - Konstruiere Reduktionsfunktion f wie folgt:
  - f(ϕ) = <G,k>
  - k = Anzahl der Klauseln
  - Für jede Klausel C in ϕ werden drei Knoten angelegt, die mit den Literalen der Klausel bezeichnet werden
  - Füge Kante zwischen zwei Knoten ein, gdw.
    - die beiden Knoten nicht zur selben Klausel gehören und
    - die beiden Knoten nicht einer Variable und der selben negierten Variable entsprechen.

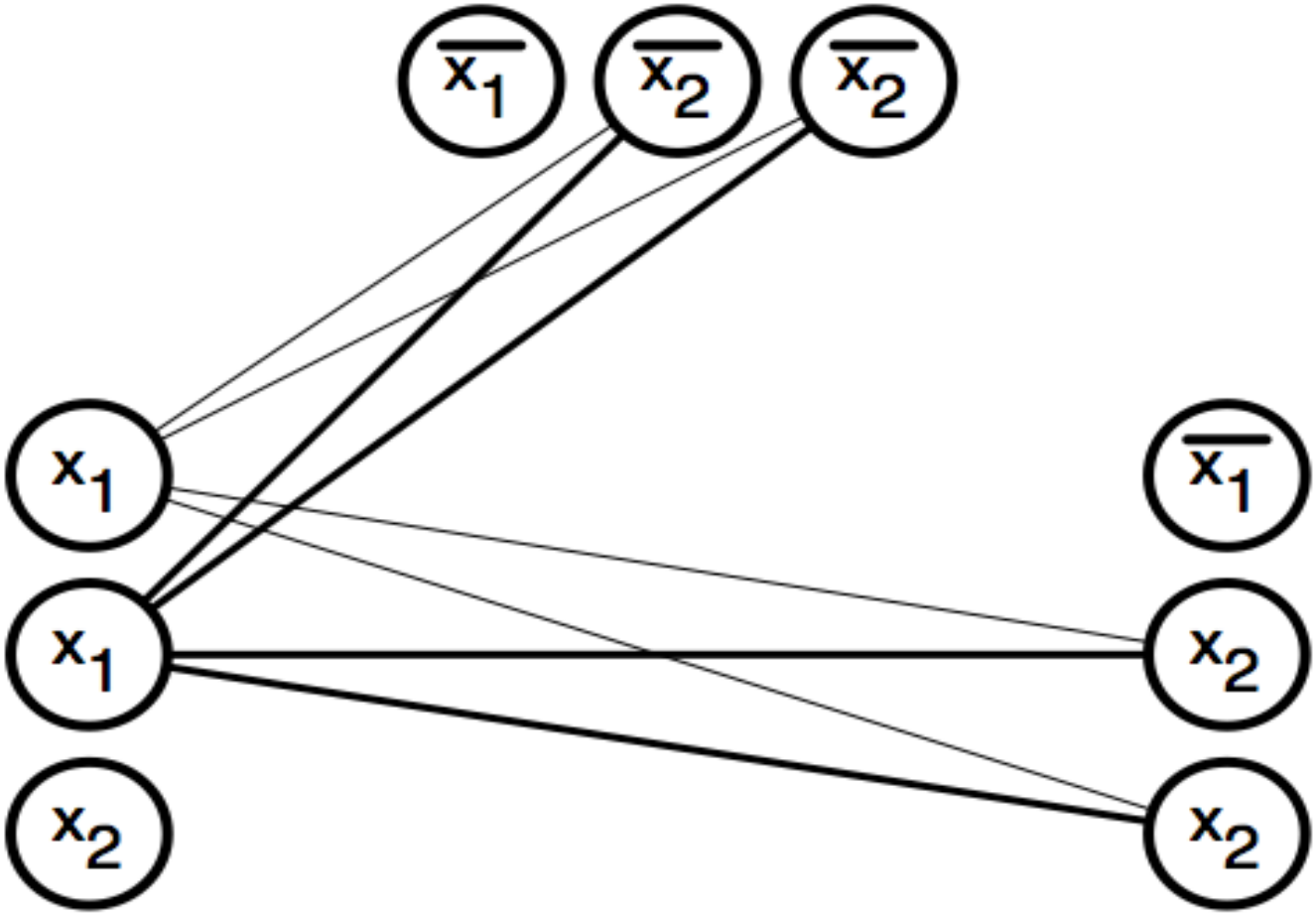


$$\psi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$

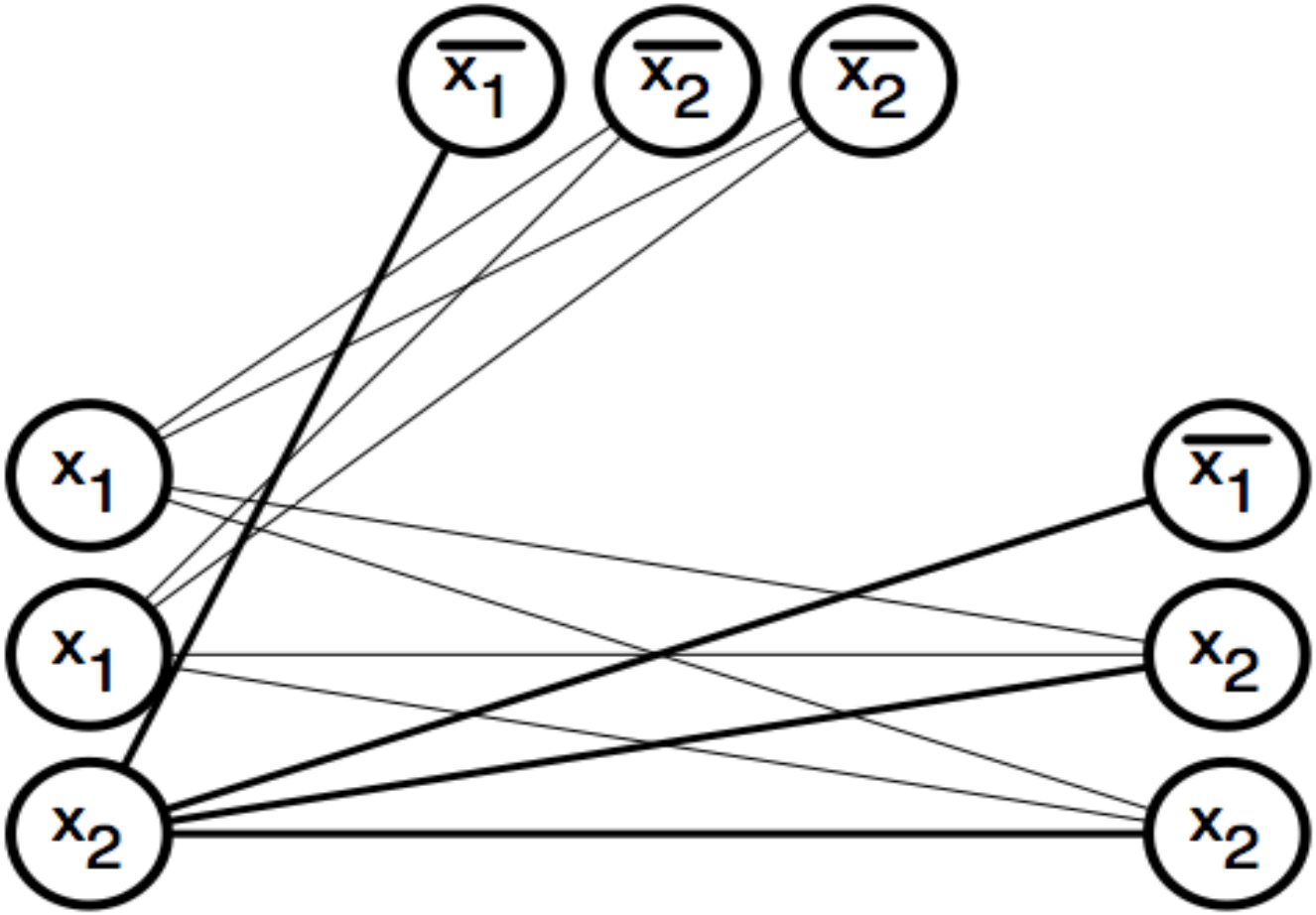
$$\psi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$



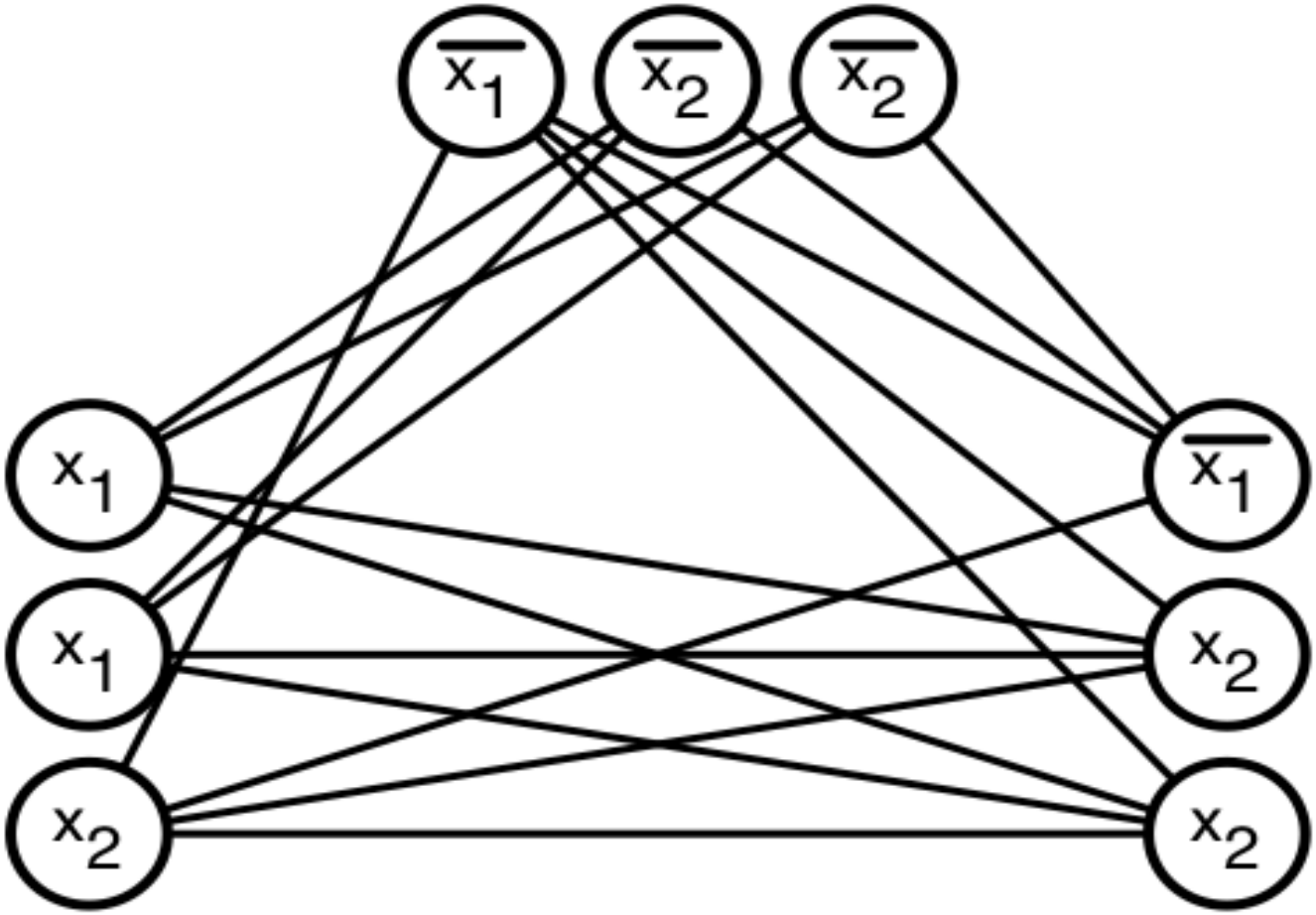
$$\psi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$



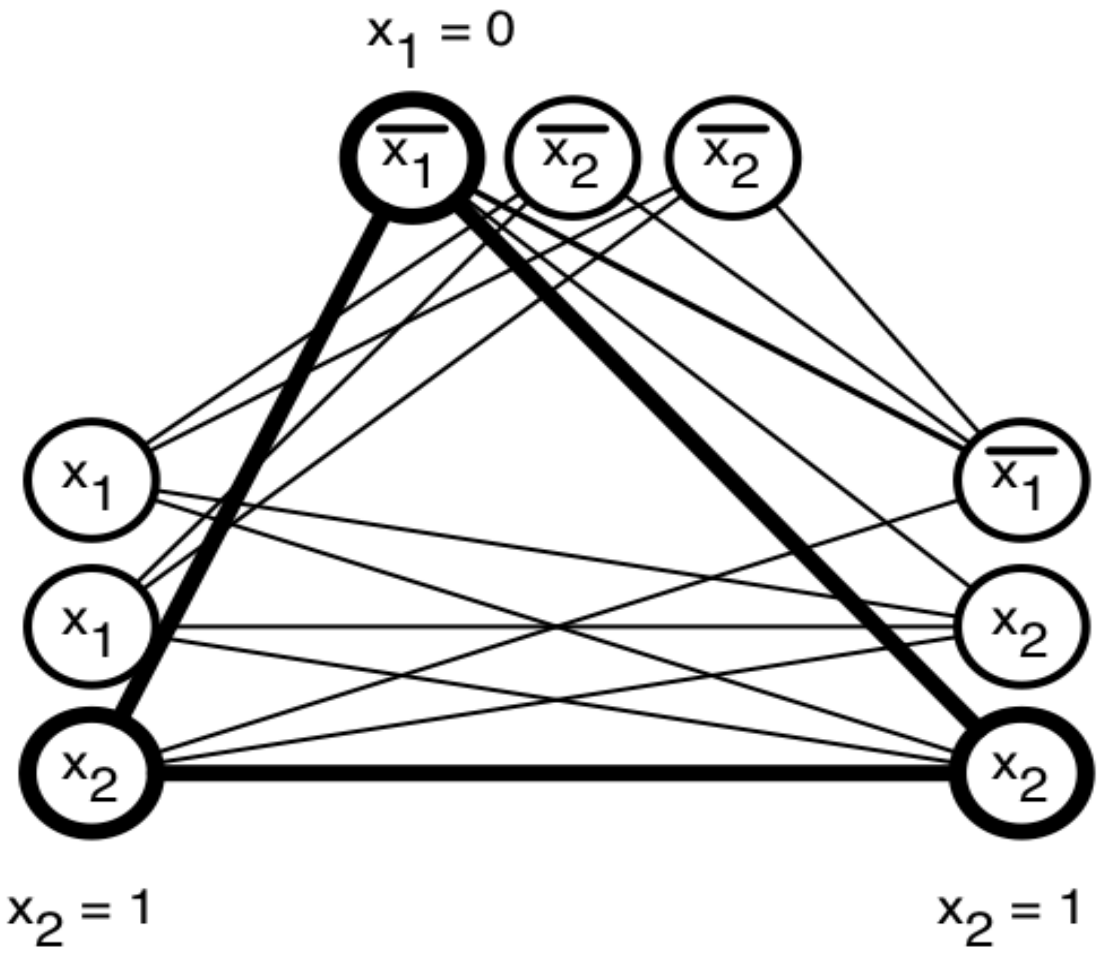
$$\psi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$



$$\psi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$



$$\psi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$



$$\psi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$

0	0	1	1	0	0	1	1	1

Beweis der Korrektheit der Reduktionsfunktion

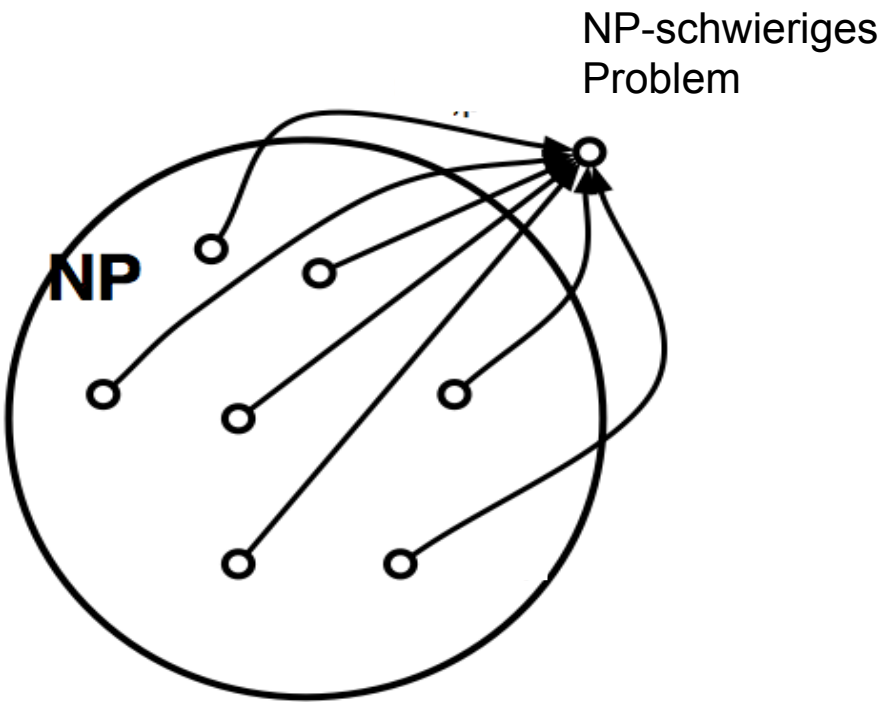
TECHNISCHE UNIVERSITÄT DARMSTADT

- Die Reduktionsfunktion ist korrekt:
- Behauptung;
  - Eine erfüllende Belegung in ϕ existert gdw. eine k-Clique in G existiert
- 1. Fall: eine erfüllende Belegung existiert in ϕ
  - Dann liefert die Belegung in jeder Klausel mindestens ein Literal mit Wert 1
  - Wähle aus der Knotenmenge einer Klausel ein beliebiges solches Literal
  - Die gewählte Knotenmenge besteht dann aus k Knoten
  - Zwischen allen Knoten existiert eine Kante, da Variable und negierte Variable nicht gleichzeitig 1 sein können
- 2. Fall: eine k-Clique existiert in G
  - Jeder der Knoten der Clique gehört zu einer anderen Klausel
  - Setze die entsprechenden Literale auf 1
  - Bestimmte daraus die Variablen-Belegung
  - Das führt zu keinem Widerspruch, da keine Kanten zwischen einem Literal und seiner negierten Version existieren
- Laufzeit:
  - Konstruktion des Graphens und der Kanten benötigt höchstens quadratische Zeit.

NP-Schwierig

TECHNISCHE UNIVERSITÄT DARMSTADT

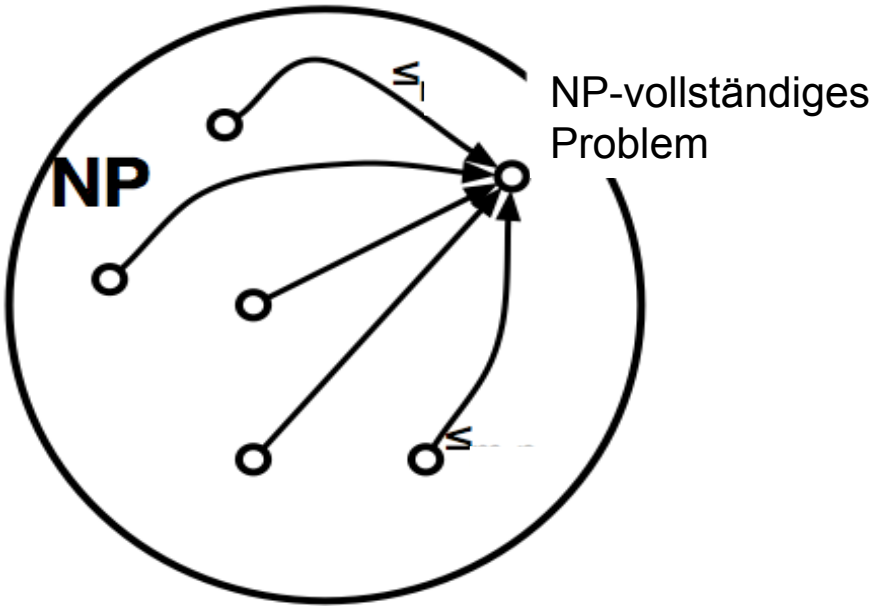
- Definition:
  - Ein Problem S ist **NP-schwierig** (NP-hard) wenn:
    - jedes Problem aus NP mit einer Polynom-Zeit-Abbildungsreduktion auf S reduziert werden kann, d.h.
    - für alle $L \in NP$: $L \leq_p S$
- Theorem
  - Falls ein NP-schwieriges Problem in P ist, ist P=NP
- Beweis
  - Falls $S \in P$ und $L \leq_p S$ gilt $L \in P$.



NP-Vollständigkeit

TECHNISCHE UNIVERSITÄT DARMSTADT

- Definition:
  - Ein Problem S ist **NP-vollständig** (NP-complete) wenn:
    - S ∈ NP
    - S ist NP-schwierig
- Korollar:
  - Ist eine NP-vollständiges Problem in P, dann ist P=NP
- Beweis:
  - folgt aus der NP-Schwierigkeit der NP-vollständigen Probleme.



SAT ist NP-Vollständigkeit

TECHNISCHE UNIVERSITÄT DARMSTADT

- Satz von Cook, Idee
  - Für jedes NP-vollständige Problem gibt es eine NTM, die das Problem entscheidet
  - Sei nun das Problem P, eine NTM M und eine Eingabe x gegeben.
  - Dann lässt sich eine boolsche Formel in KNF angeben, deren Variablen-Belegungen die Maschine M auf x simuliert.

  - Also: Die Entscheidung “akzeptiert M x” kann rückgeführt werden auf die Erfüllbarkeit einer SAT-Formel.

Nichtdeterministische Maschinen und Verifizierer

TECHNISCHE UNIVERSITÄT DARMSTADT

Def. Es sei L eine Sprache. Ein Verifizierer für L ist ein deterministischer Algorithmus A, mit L = {w | es gibt ein c mit A akzeptiert wc}

Der Zeitaufwand für einen Verifizierer wird abhängig von der Länge von w gemessen. L ist polynomiell prüfbar, wenn es einen Verifizierer mit polynomiellem Zeitaufwand gibt.

Satz: NP ist die Menge aller Probleme, für die es einen Verifzierer mit polynomiellem Zeitaufwand gibt.

(ohne Beweis)

Limits

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Komplexität wird gemessen mit Hilfe des worst-case,**
  - In der realen Welt treten aber oft „gutmütige“ Instanzen auf
  - andererseits: Haben wir dann das richtige Problem formuliert?

- **Die Eingabegröße**

  Sei ein Problem p gegeben. Wenn wir nun die Kodierung der Eingabe signifikant kleiner bekommen, wird das resultierende aus p abgeleitete Problem p‘ möglicherweise schwerer, obwohl die Laufzeit möglicherweise sinkt.

Fazit: Die Lücke zwischen der Komplexitätstheorie und der Wirklichkeit ist größer als wir es von der Physik gewohnt sind. Trotzdem ist die Komplexitätstheorie eine großartige Theorie.

Die PSPACE Welt

TECHNISCHE UNIVERSITÄT DARMSTADT

- Quantifizierte ganzzahlige lineare Programme (QIP)
- Go
- Schach

- Stochastic Satisfibility (SSAT)
- Dynamic Graph Reliability (DGR)
- Multi-Stage Stochastic Programming

Quantifiziertes Lineares Programm (QLP)

- Vektor mit Variablen $x = (x_1, .., x_n) \in \mathbb{Q}^n$
- Obere und untere Schranken $l_i \in \mathbb{Q}^n$ und $u_j \in \mathbb{Q}^n$ mit $l_i \leq x_i \leq u_i$
- Matrix $A \in \mathbb{Q}^{m \times n}$
- Vektor $b \in \mathbb{Q}^m$
- Quantifiziererstring $Q = (q_1, .., q_n) \in \{\forall, \exists\}^n$ mit $q_i x_i$ für alle $1 \leq i \leq n$

Qlp $G := [Q : Ax \leq b]$ oder $G := \left[Q(x, y) : A(x, y)^T \leq b \right]$

Quantifizierte Lineare Programme:

TECHNISCHE UNIVERSITÄT DARMSTADT

Beispiel: 3-dimensionales QLP

$$\forall x_1 \forall x_2 \exists x_3 : \begin{pmatrix} 10 & -4 & 2 \\ 10 & 4 & -2 \\ -10 & 4 & 1 \\ -10 & -4 & -1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \leq \begin{pmatrix} 0 \\ 4 \\ 12 \\ 8 \end{pmatrix}, \textit{with} \quad x_1 \in [-1,0], x_2 \in [0,1], x_3 \in [-2,2]$$

QLP $\left[Q(x,y) : A(x,y)^T \leq b \right]$ als Zwei-Personen-Nullsummenspiel:

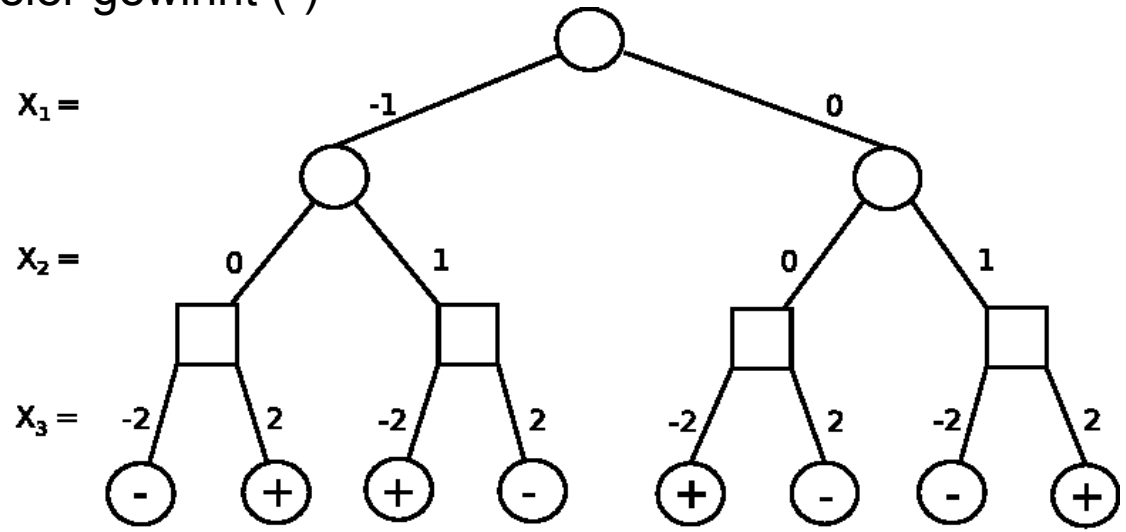
- Ein Spieler belegt die existenzquantifizierten Variablen (Existenzspieler)
- Ein Spieler belegt die allquantifizierten Variablen (Allspieler)
- Die Spieler belegen die Variablenblöcke abwechselnd, wie durch den Quantifiziererstring $Q(x,y)$ vorgegeben
- Ist ein Spieler am Zug bei Variable x_i, so kennt er die Belegung von $x_1, \ldots, x_{i-1}$
- Gilt am Ende $A(x,y)^T \leq b$ gewinnt der Existenzspieler, sonst der Allspieler

Ganzzahlige Quantifizierte Lineare Programme (QIP):

TECHNISCHE UNIVERSITÄT DARMSTADT

Beispiel mit Beschränkung auf Ganzzahligkeit / Variablengrenzen

- Knotentypen (Existenzknoten, Allknoten)
- Kantenbeschriftungen (Variablenbelegung)
- Spielausgänge an Blättern
  - Existenzspieler gewinnt (+)
  - Allspieler gewinnt (-)

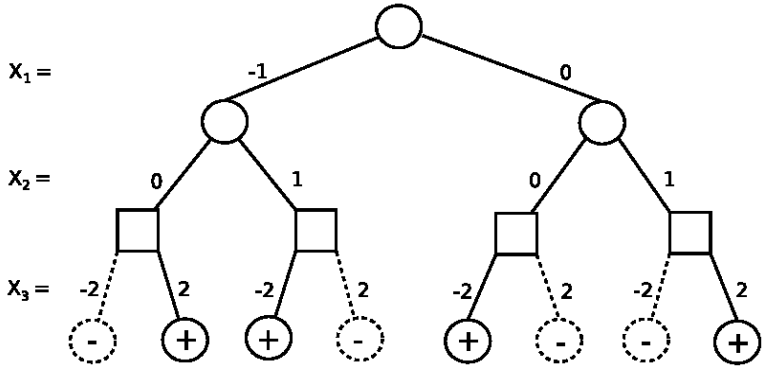


DISCRETE OPTIMIZATION

Ganzzahlige Quantifizierte Lineare Programme:

TECHNISCHE UNIVERSITÄT DARMSTADT

Definition1 (Strategie):



- Regelwerk für den Existenzspieler
- Teilbaum $\left(V_x \cup V_y, E, L\right)$ der Tiefe n
- Jede Kante $e_i \in E$ repräsentiert eine Belegung von Variable x_i auf ebene i des Baumes
- $l_i \in L$ repräsentiert den Wert von Variable x_i an Kante e_i
- Existenzknoten haben eine ausgehende Kante
- Allknoten haben zwei ausgehende Kanten

Eine Strategie ist eine **Gewinnstrategie** für den Existenzspieler, wenn alle Pfade von der Wurzel zu den Blättern einen Vektor x repräsentieren, so dass $Ax \leq b$ gilt.

Fragestellung:

Existiert ein Algorithmus, der die Belegung von Variable x_i, bei Kennntis der vorherigen Belegungen von $x_1, \ldots, x_{i-1}$, berechnet, so dass der Existenzspieler das Spiel gewinnt, unabhängig davon wie der Allspieler agiert wenn er am Zug ist.

Satz: QIP ist PSPACE-vollständig.

Bew: Reduktion von QSAT auf QIP

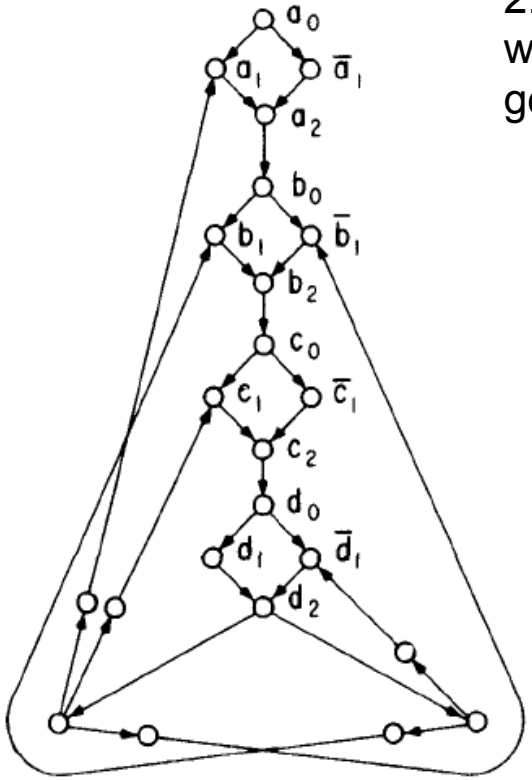
Seien die Variablen einer SAT Instanz $x_1,\ldots,x_n$. Seien $C_1,\ldots,C_m$ die Klauseln der SAT Instanz. Bilde 0/1-Variablen $y_1,\ldots,y_n$ für das QIP so, dass $y_i = 0$ genau dann, wenn x_i=false.

Das QIP, welches das QSAT Problem zu entscheiden hilft ist dies:

$$\sum_{j:y_j \in C_i} y_j + \sum_{j:\bar{y} \in C_i} (1 - y_j) \geq 1, \quad \forall C_i$$

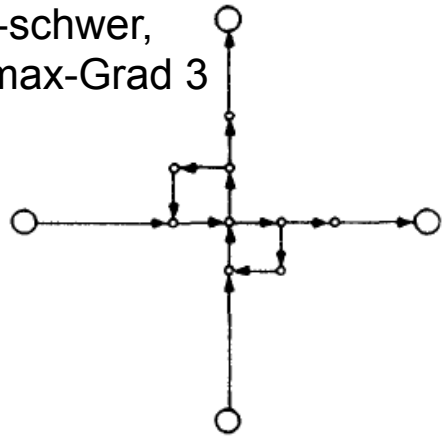
$$y_j \in \{0,1\}$$

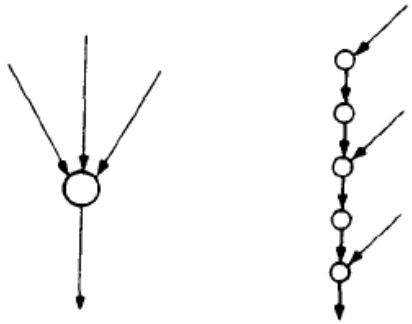
TECHNISCHE UNIVERSITÄT DARMSTADT



1. GEO is PSPACE-schwer

2. GEO ist selbst dann PSPACE-schwer, wenn auf planaren Graphen mit max-Grad 3 gespielt wird.





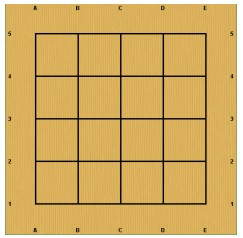
Eingangsgrad umbauen

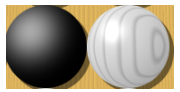
Das Go Spiel

TECHNISCHE UNIVERSITÄT DARMSTADT

- Go ist ein sehr altes Spiel
- über 3000 Jahre lange Geschichte in China
- einige Historiker behaupten, es sei vor über 4000 Jahren erfunden worden
- Name kommt von japanischem Igo, "surrounding boardgame"

Regeln:





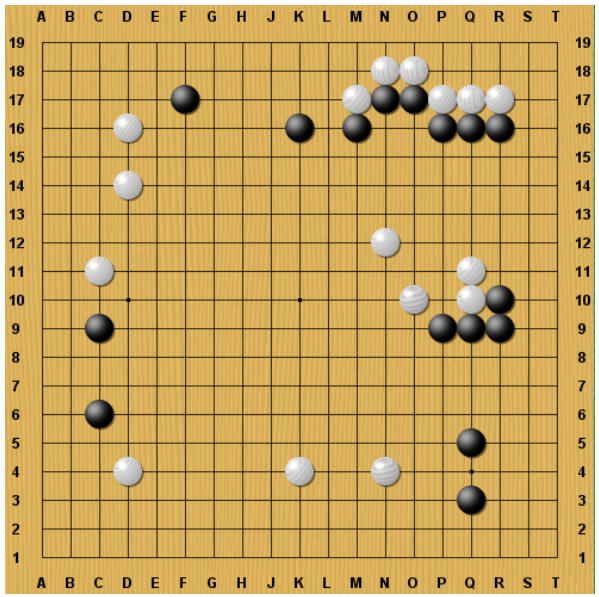
- zwei Spieler: Schwarz, Weiss
- Spielbrett: n x n Gitter (typisch: 9x9, 19x19)
- gespielt wird auf Schnittpunkten des Gitters.
- Steine: schwarze und weiße Steine
- Spieler ziehen, indem sie Steine ihrer Farbe auf leere Schnittpunkte plazieren. (auch passen ist erlaubt)
- Spieler ziehen abwechselnd

Das Spiel GO

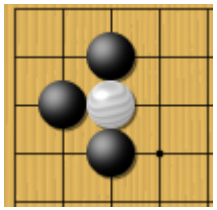
TECHNISCHE UNIVERSITÄT DARMSTADT

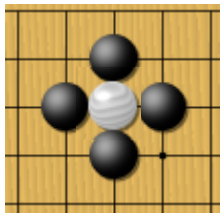
Regeln:

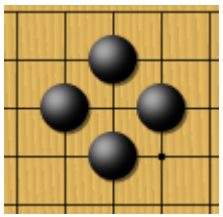
- connected component, Wurm, Block, String: eine gemäß der Gittertopologie verbundene Menge von gleichfarbigen Steinen
- Freiheiten: Anzahl angrenzender freier Schnitt-punkte an einen Block
- Steine schlagen: Blöcke mit 0 Freiheiten werden vom Brett genommen



- Selbstmord nicht erlaubt







Das Spiel GO

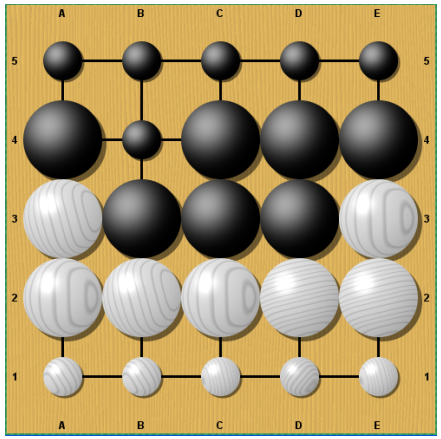
TECHNISCHE UNIVERSITÄT DARMSTADT

Spielende: beide Spieler passen

Gewinner ermitteln

Area Scoring (meistens im Computer Go), lt. senseis.net:

Man zähle für Weiß und für Schwarz getrennt die Steine auf dem Brett, sowie die Anzahl der umzingelten Schnittpunkte (Territorium).



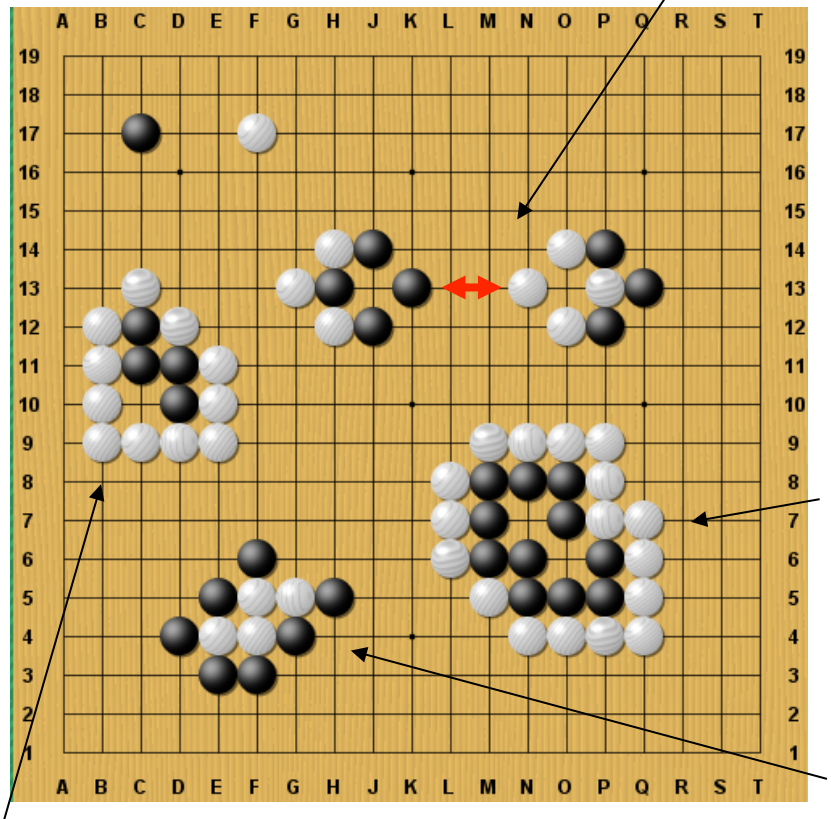
Schwarz: 7 Steine, 6 Schnittpunkte
Weiss: 7 Steine, 5 Schnittpunkte

Schwarz gewinnt

Das Spiel GO

TECHNISCHE UNIVERSITÄT DARMSTADT

ko, direkte Zugwiederholung ist nicht erlaubt

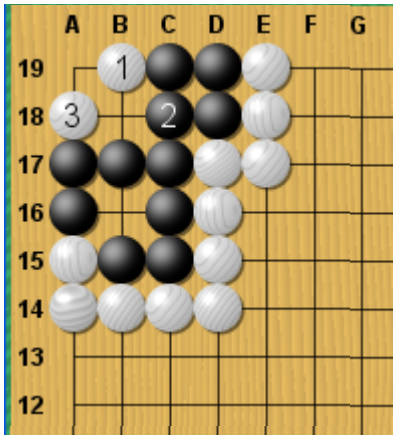


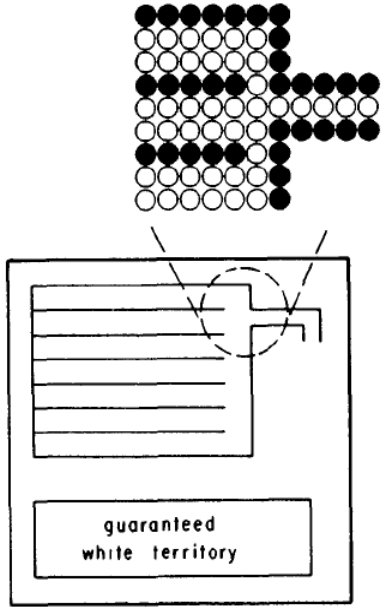
Schwarzer Block hat 2 Augen

Leiter

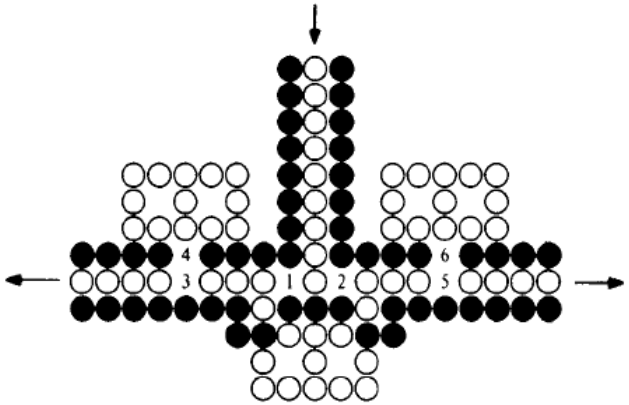
Weiss zieht und schlägt schwarzen Block

sichere schwarze Steine, aber unsicheres Territorium

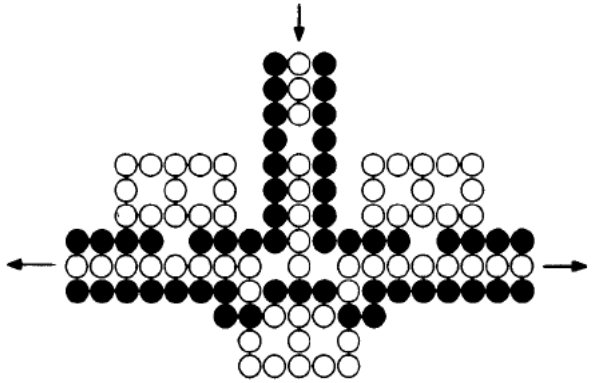




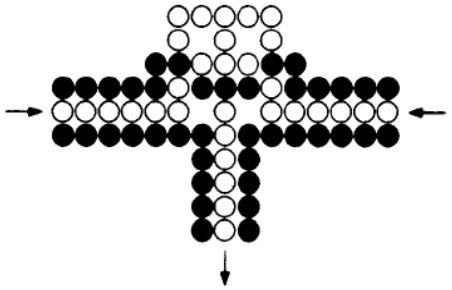
Globale Situation



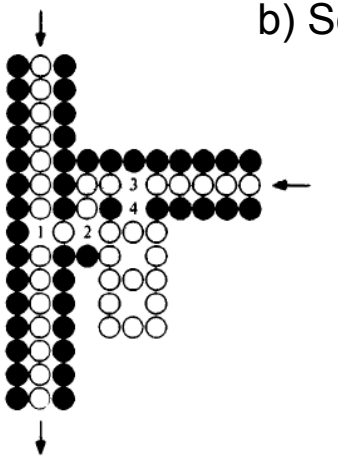
a) Weiss wählt



b) Schwarz wählt



c) Zusammenführung



d) Test