Approximationsschema für Knapsack

Lemma a) Für $0 \le v_j$ gilt: $\mathrm{MaxKnap}(\underline{g},\underline{w}) \le \mathrm{MaxKnap}(\underline{g},\underline{w}+\underline{v})$

b) und für $v_j \le \ell$: $\mathrm{MaxKnap}(\underline{g},\underline{w}+\underline{v}) \le \mathrm{MaxKnap}(\underline{g},\underline{w}) + n\cdot\ell$

c) sowie $\mathrm{MaxKnap}(\underline{g},k\cdot\underline{w}) = k \cdot \mathrm{MaxKnap}(\underline{g},\underline{w})$

$$\lfloor \underline{w}/k \rfloor \cdot k \le \underline{w} < \lfloor \underline{w}/k \rfloor \cdot k + k$$

Skalierungsmethode: Fixiere k und setze $w_j' := \lfloor w_j / k \rfloor$

Berechne $\mathrm{opt}' := k \cdot \mathrm{MaxKnap}(g,g_1,\ldots,g_n,w_1',\ldots,w_n')$ und J'

mittels **`ExactKnapsack`** in Zeit $\mathrm{poly}(n)\cdot W/k$. Dann

$$\mathrm{opt} \ge \mathrm{opt}' = \mathrm{MaxKnap}(g, \lfloor \underline{w}/k \rfloor \cdot k) > \mathrm{MaxKnap}(g,\underline{w}-k)$$

$$\ge \mathrm{opt} - n\cdot k \ge \mathrm{opt}\cdot(1 - n\cdot k/W)$$

Setze nun $k := \varepsilon \cdot W/n$.

oBdA $g_j \le g \Rightarrow$
$\max_j w_j =: W \le \mathrm{opt} \le n\cdot W$

$$\mathrm{MaxKnap}(\underline{g},\underline{w}) = \max\{ \textstyle\sum_{i\in J} w_i : J \subseteq \{1..n\}, \sum_{i\in J} g_i \le g \}$$

Nicht-/Approximierbarkeit

- **VC** kann in $\mathcal{P}$ mit Güte 2 approximiert werden,
- ebenso **MTSP**: mit Güte 2
- **Knapsack** $\in \mathcal{NP}$ kann in $\mathcal{P}$ mit Güte $(1-\varepsilon)$ approximiert werden für beliebiges $\varepsilon>0$.
- Ebenso **SubsetSum** als Spezialfall mit $w_i=g_i$; Reduktion **3SAT**$\leq_p$**SubsetSum** benutzte große Zahlen!
- **TSP** erlaubt in $\mathcal{P}$ keine Approx. mit konst. Güte
- Kann **CLIQUE** trivial mit Güte n approximieren;
- aber in $\mathcal{P} \neq \mathcal{NP}$ nicht mit Güte $O(n^{1-\varepsilon})$ für beliebiges $\varepsilon>0$ **(Johan Håstad 1996)**

PSPACE *und* QBF

Bsp: $\varphi(x,y,z) = (x \vee y \vee z) \wedge (\neg x \vee \neg y \vee \neg z)$

SAT = alle erfüllbaren booleschen Formeln

⇔ alle <u>existenziell</u> quantifizierten wahren BF

z.B. Φ = „$\exists x\, \exists y\, \exists z\;\; \varphi(x,y,z)$"

QBF := { alle (beliebig) <u>vollständig</u> quantifizierten wahren boolesche Formeln Φ }

Bsp: Φ = „ $\exists x\, \forall y\, \exists z\;\; \varphi(x,y,z)$"

Allgemein:

$$Q_1 x_1\; Q_2 x_2\; Q_3 x_3\; \ldots\; Q_n x_n : \varphi(x_1,x_2,x_3,\ldots,x_n)$$

entscheidbar in polynom. Platz

2.6.7 QBF ist PSPACE-schwer

Gezeigt: Für jedes $L \in NP$ gilt: $L \leqslant_p SAT$

Thm: Für jedes $L \in PSPACE$ gilt: $L \leqslant_p QBF$

d.h. QBF ist PSPACE-schwer

(und damit PSPACE-vollständig)

Warum zeigt Cook-Levin nicht:

"SAT ist PSPACE-schwer?"

M akzeptiert w $\Leftrightarrow$ Φ ist erfüllbar (nur $\exists$-Quantoren)

Φ hat Länge $O(S \cdot T)$, wobei S,T=Platz/Zeit von M

QBF ist PSPACE-schwer: Beweis

$\mathrm{succ}_\ell(V_1,V_2)$:= „V_2 ist Nachfolgekonfig. von V_1, die nach höchstens 2^ℓ Rechenschritten erreicht werden kann“

Für $t(n)$-zeitbeschr. & $s(n)$-platzbeschr. Maschine M:

$\underline{w} \in L(M) \iff \exists V,V' \in \{0,1\}^{O(s(|\underline{w}|))}$:

$\mathrm{start}(V,\underline{w}) \wedge \mathrm{accept}(V') \wedge \mathrm{succ}_{\log t(|\underline{w}|)}(V,V') \wedge \mathrm{legal}(V) \wedge \mathrm{legal}(V')$

Erinnerung an den Beweis von Cook-Levin:

$\mathrm{legal}(V)$:= “V beschreibt legale Konfiguration (von M)”

$\mathrm{start}(V,\underline{w})$:= “V beschreibt Startkonfigur. von M auf w”

$\mathrm{accept}(V)$:= “V beschreibt eine akzept. Konfiguration”

$\mathrm{succ}(V_1,V_2)$:= “V_2 ist *direkte* Nachfolgekonfig. von V_1”

QBF ist PSPACE-schwer: Beweis

$\mathrm{succ}_\ell(V_1,V_2)$:= „V_2 ist Nachfolgekonfig. von V_1, die nach höchstens 2^ℓ Rechenschritten erreicht werden kann"

Für $t(n)$-zeitbeschr. & $s(n)$-platzbeschr. Maschine M:

$\underline{w}\in L(M) \Leftrightarrow \exists V,V'\in\{0,1\}^{O(s(|\underline{w}|))}$: $\boxed{s(n)=\mathrm{poly}(n),\ t(n)=2^{O(s(n))}}$

$\mathrm{start}(V,\underline{w}) \wedge \mathrm{accept}(V') \wedge \mathrm{succ}_{\log t(|\underline{w}|)}(V,V') \wedge \mathrm{legal}(V) \wedge \mathrm{legal}(V')$

Beachte: $\mathrm{succ}_\ell(V,V'') \Leftrightarrow \exists V'$: $\mathrm{succ}_{\ell-1}(V,V') \wedge \mathrm{succ}_{\ell-1}(V',V'')$

Aber succ_ℓ so aufzuschreiben, erfordert zu viel Platz&Zeit andererseits noch kein „$\forall$" benutzt… Also weiter:

$$\mathrm{succ}_{\ell'}(V_1,V_2) \wedge \mathrm{succ}_{\ell'}(V_2,V_3) \Leftrightarrow$$
$$\big(\forall U,U': ((U=V_1 \wedge U'=V_2) \vee (U=V_2 \wedge U'=V_3)) \Rightarrow \mathrm{succ}_{\ell'}(U,U')\big)$$

jetzt Rekursion ab $\ell:=\log t(|\underline{w}|)$: succ_ℓ, $\mathrm{succ}_{\ell-1}$, $\mathrm{succ}_{\ell-2}$

3QBF ebenfalls PSPACE-schwer

- Ergibt quantifizierte Formel welcher Länge? / berechenbar in welcher Laufzeit?
- Wie viele Quantoren welcher Sorte?

QBF := vollständig quantifizierte wahre boolesche Formeln Φ; z.B. „$\exists x\ \forall y\ \exists z\ \ \varphi(x,y,z)$“

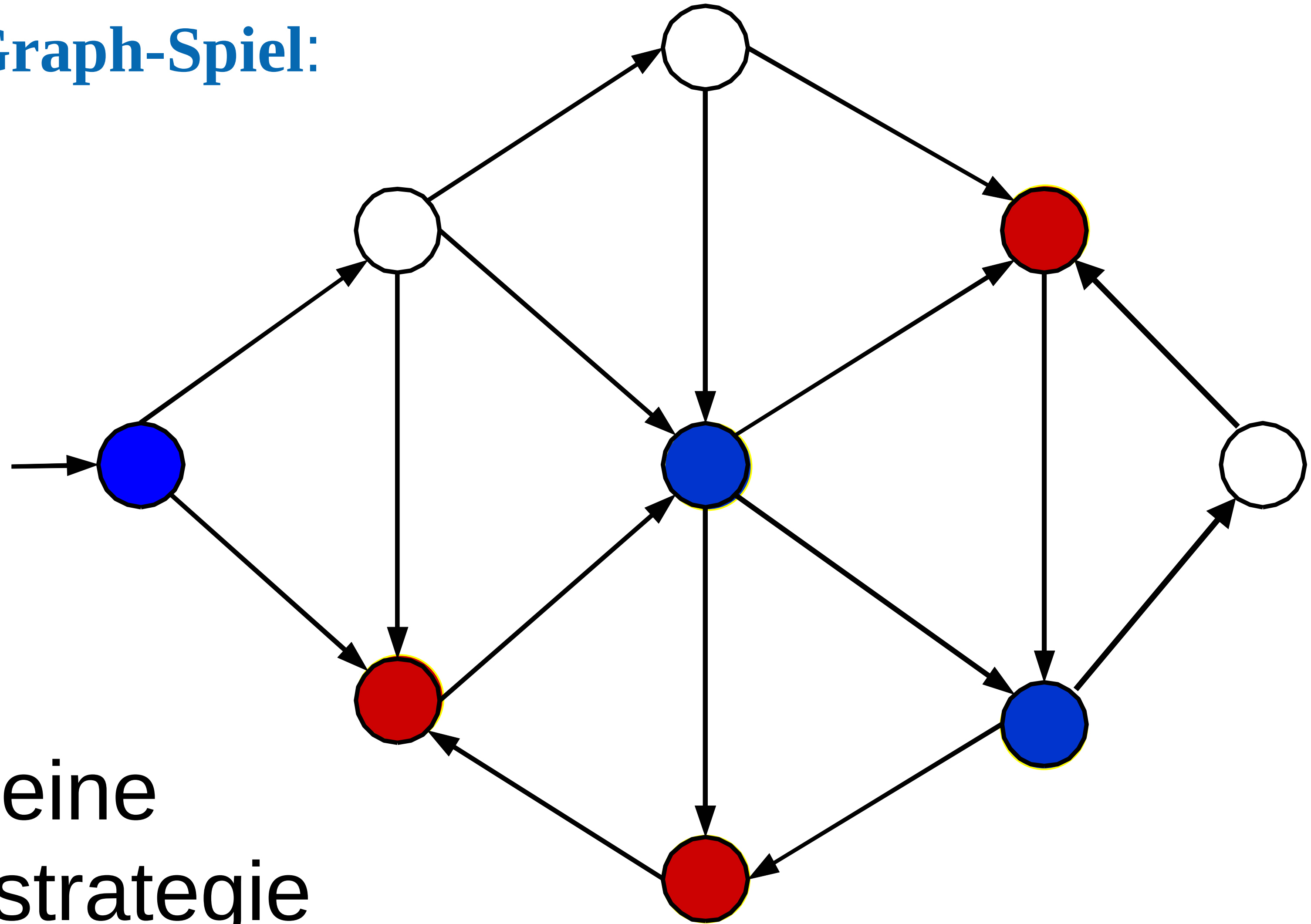
- o.B.d.A. Quantoren „$\exists x_1\ \forall x_2\ \exists x_3\ \forall x_4 \ldots \exists x_n$“
- o.B.d.A. φ in 3KNF

$$\mathrm{succ}_{\ell'}(V_1,V_2) \wedge \mathrm{succ}_{\ell'}(V_2,V_3) \Leftrightarrow$$
$$\Big(\forall U,U'\!:\ \big((U=V_1 \wedge U'=V_2) \vee (U=V_2 \wedge U'=V_3)\big) \Rightarrow \mathrm{succ}_{\ell'}(U,U')\Big)$$

jetzt Rekursion ab $\ell:=\log t(|\underline{w}|)$: succ_ℓ, $\mathrm{succ}_{\ell-1}$, $\mathrm{succ}_{\ell-2}$

Bewiesen: Es gibt eine PSPACE-vollständige Sprache

Jetzt ein **Graph-Spiel**:

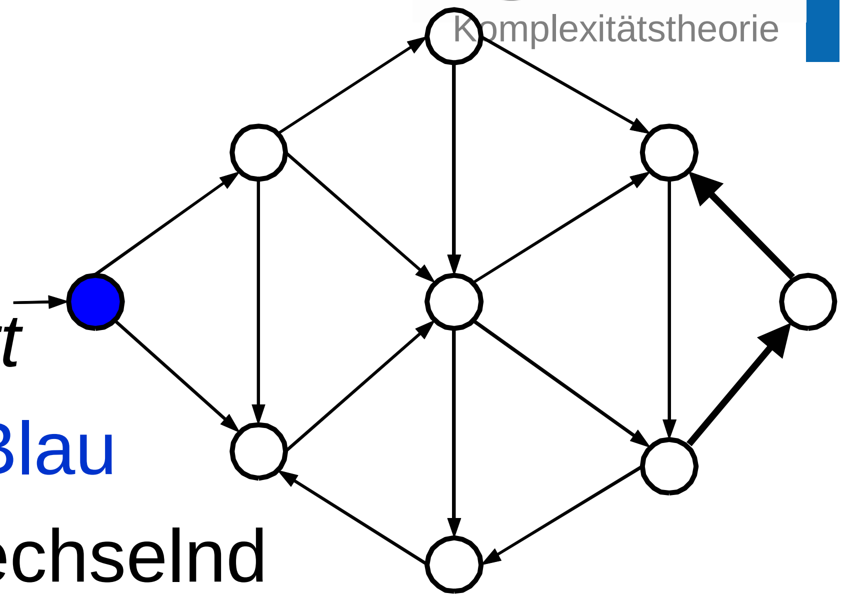


Rot hat eine Gewinnstrategie

Graph-Spiel

Spielregeln:

- gerichteter Graph *G*
- Startknoten *s, aktiv, markiert*
- zwei SpielerInnen Rot und Blau
- wählen und markieren abwechselnd
- einen neuen *aktiven* Knoten
  - Ziel einer Kante vom bisherigen
  - noch *nicht markiert*.
- Wer nicht mehr ziehen kann, verliert; der andere gewinnt.



Frage: *Gibt es eine Gewinnstrategie für Rot?*