Schwere der Approximation

Ignaz Rutter

Algorithmics Group
Faculty of Informatics
Universität Karlsruhe (TH)

18. Juli 2008

Wie beweist man Schwere der Approximation?

Wie beweist man Schwere der Approximation?

Reduktion von SAT auf VERTEXCOVER:

Zu Formel ϕ erzeuge in polynomieller Zeit Graph $G = (V, E)$ mit

- ϕ erfüllbar $\Rightarrow$ G besitzt VC $\leq \frac{2}{3}|V|$.
- ϕ nicht erfüllbar $\Rightarrow$ jedes VC von G hat Größe $> \alpha\frac{2}{3}|V|$

$\alpha > 1$ konstant

Wie beweist man Schwere der Approximation?

Reduktion von SAT auf VERTEXCOVER:

Zu Formel ϕ erzeuge in polynomieller Zeit Graph $G = (V, E)$ mit

- ϕ erfüllbar $\Rightarrow$ G besitzt VC $\leq \frac{2}{3}|V|$.
- ϕ nicht erfüllbar $\Rightarrow$ jedes VC von G hat Größe $> \alpha\frac{2}{3}|V|$

$\alpha > 1$ konstant

Dann α-Approximation von VERTEXCOVER nicht in polynomieller Zeit möglich wenn $\mathcal{P} \neq \mathcal{NP}$.

Reduktionen und Lücken

Definition (Lückeneinführende Reduktion)

Von SAT auf Minimierungsproblem Π:

- $\phi \mapsto x$, Paramter f, α
- ϕ erfüllbar $\Rightarrow \mathrm{OPT}(x) \leq f(x)$
- ϕ nicht erfüllbar $\Rightarrow \mathrm{OPT}(x) > \alpha(|x|)f(x)$

Reduktionen und Lücken

Definition (Lückeneinführende Reduktion)

Von Sat auf Minimierungsproblem Π:

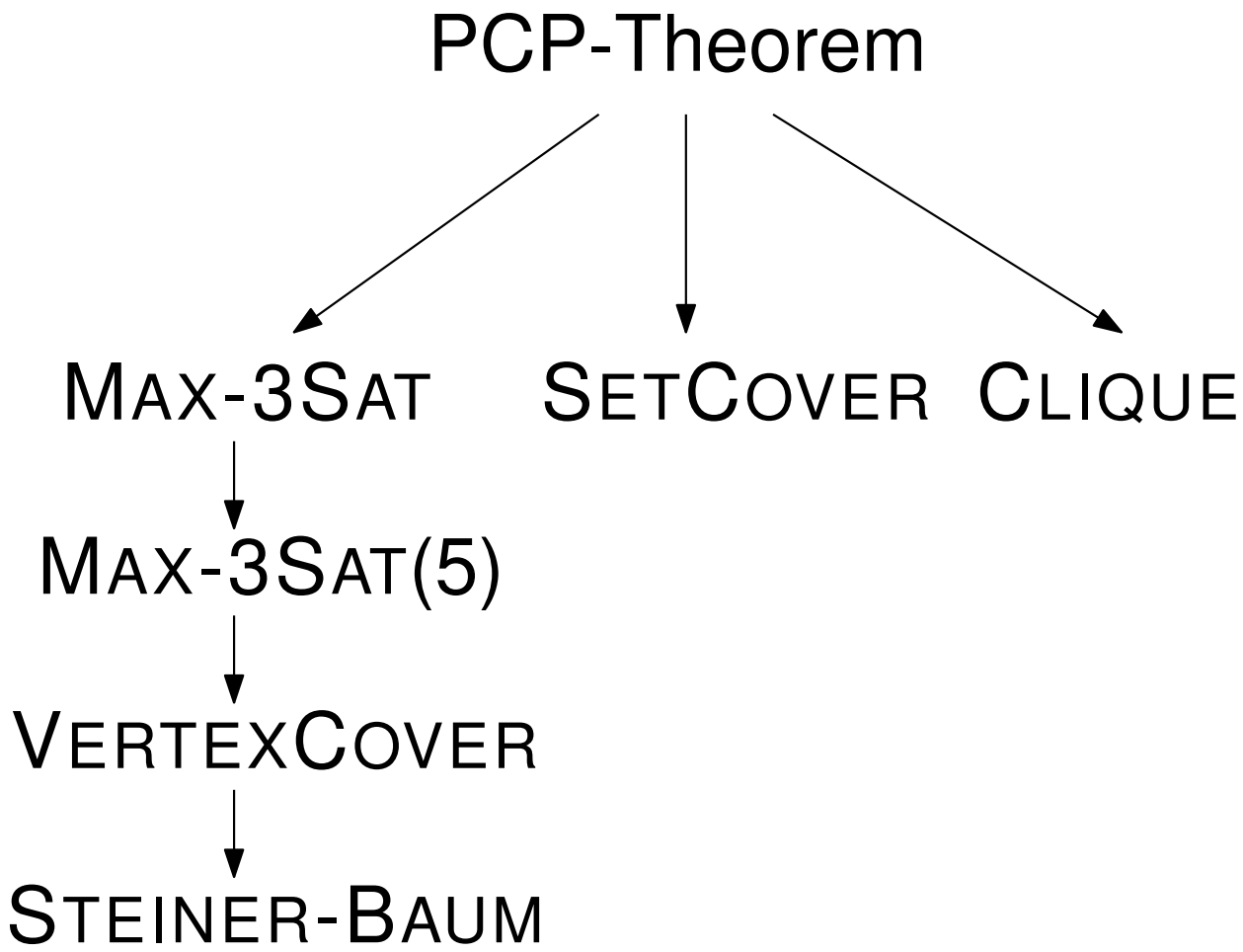
- $\phi \mapsto x$, Paramter f, α
- ϕ erfüllbar $\Rightarrow \mathrm{OPT}(x) \leq f(x)$
- ϕ nicht erfüllbar $\Rightarrow \mathrm{OPT}(x) > \alpha(|x|)f(x)$

Definition (Lückenerhaltende Reduktion)

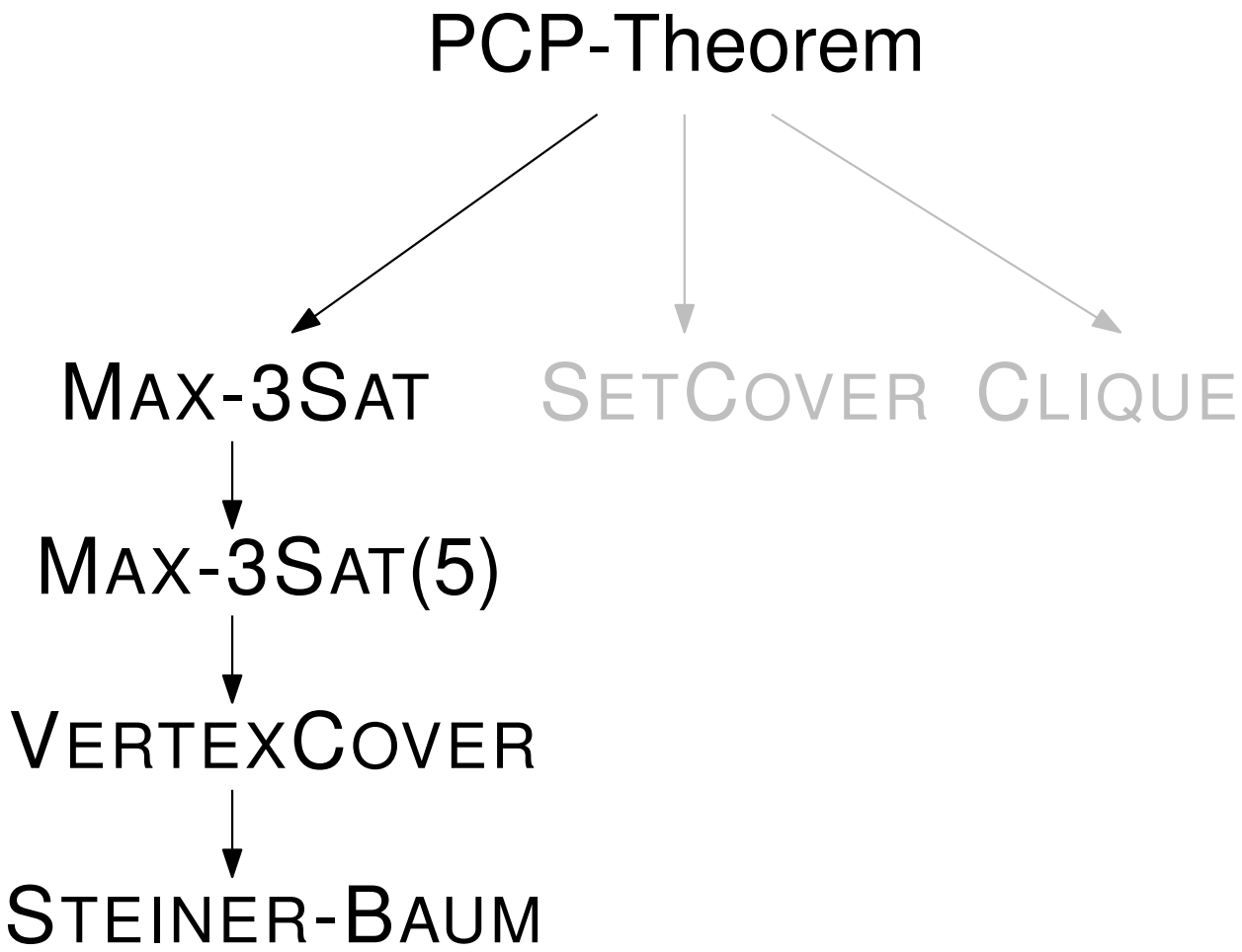
Von Minimierungsproblem Π_1 auf Maximierungsproblem Π_2:

- $x \mapsto y$, Parameter f_1, α, f_2, β
- $\mathrm{OPT}(x) \leq f_1(x) \Rightarrow \mathrm{OPT}(y) \geq f_2(y)$
- $\mathrm{OPT}(x) > \alpha(|x|)f_1(x) \Rightarrow \mathrm{OPT}(y) < \beta(|y|)f_2(y)$

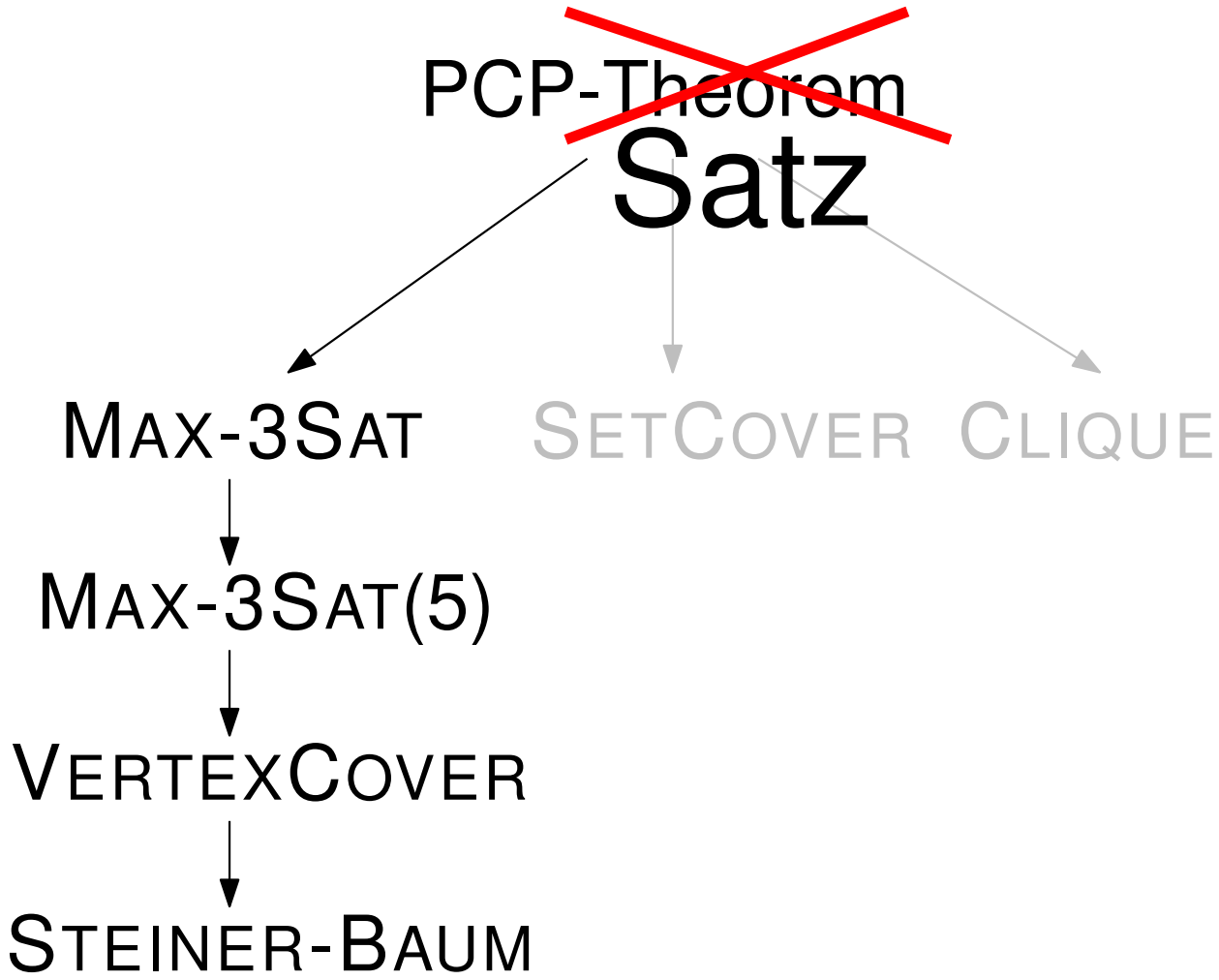
Übersicht


PCP-Theorem
Max-3Sat
SetCover
Clique
Max-3Sat(5)
VertexCover
Steiner-Baum

Übersicht

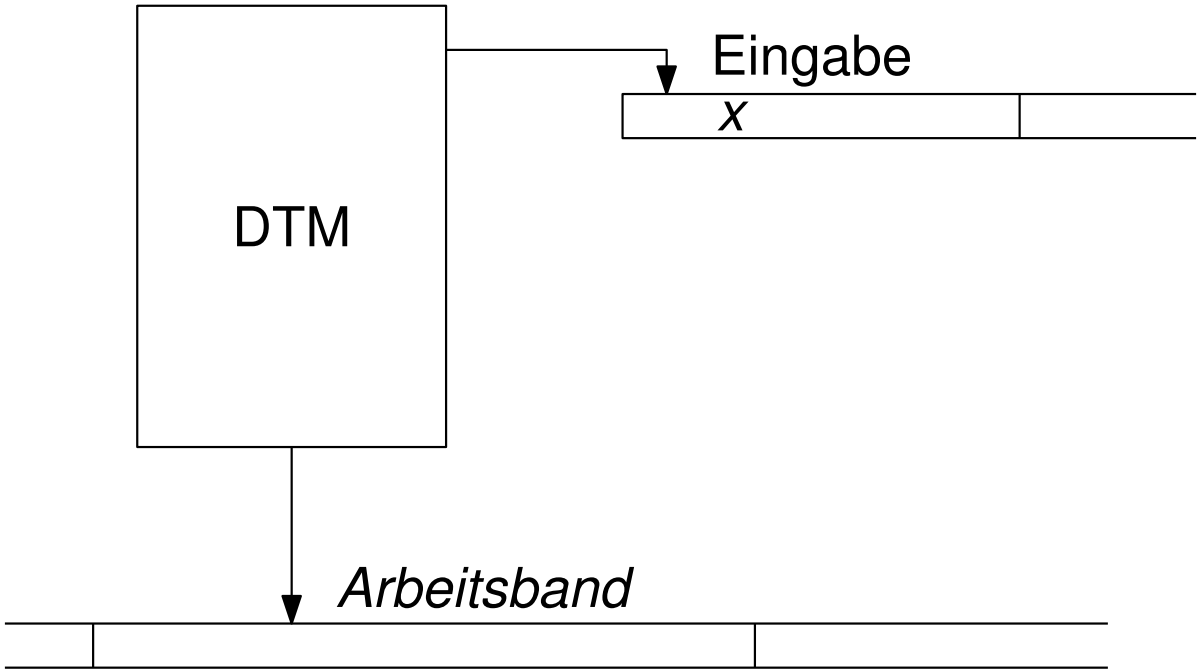

PCP-Theorem
Max-3Sat
SetCover
Clique
Max-3Sat(5)
VertexCover
Steiner-Baum

Übersicht

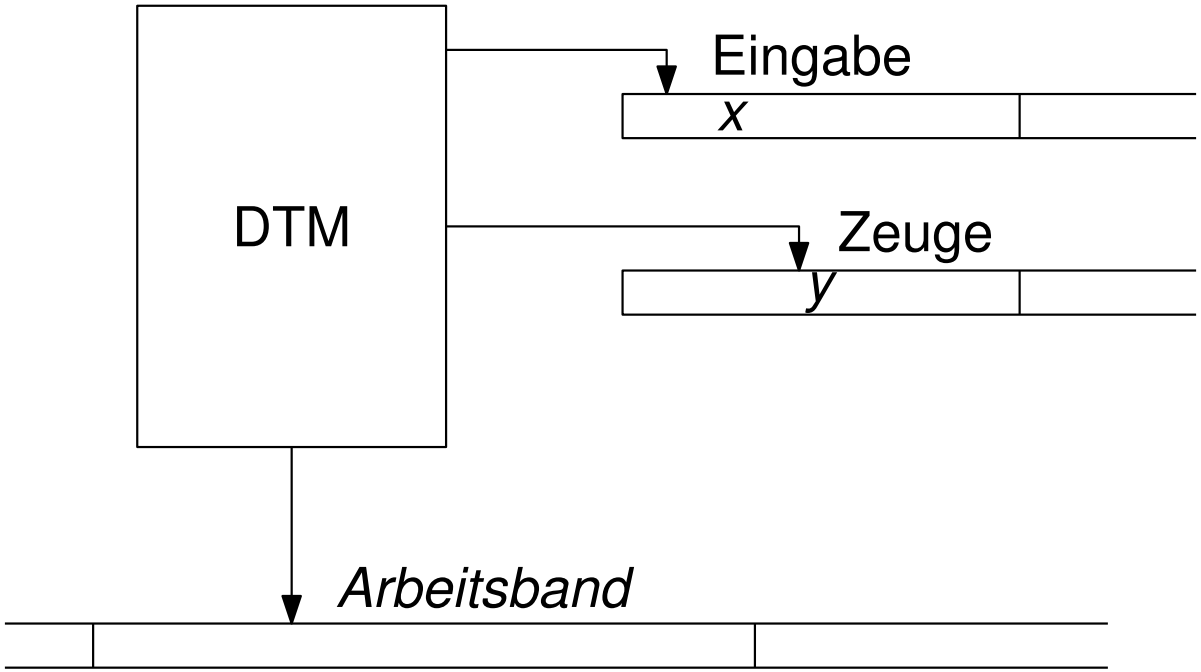

PCP-Theorem
Satz
MAX-3SAT
SETCOVER
CLIQUE
MAX-3SAT(5)
VERTEXCOVER
STEINER-BAUM

Probabilistically Checkable Proofs

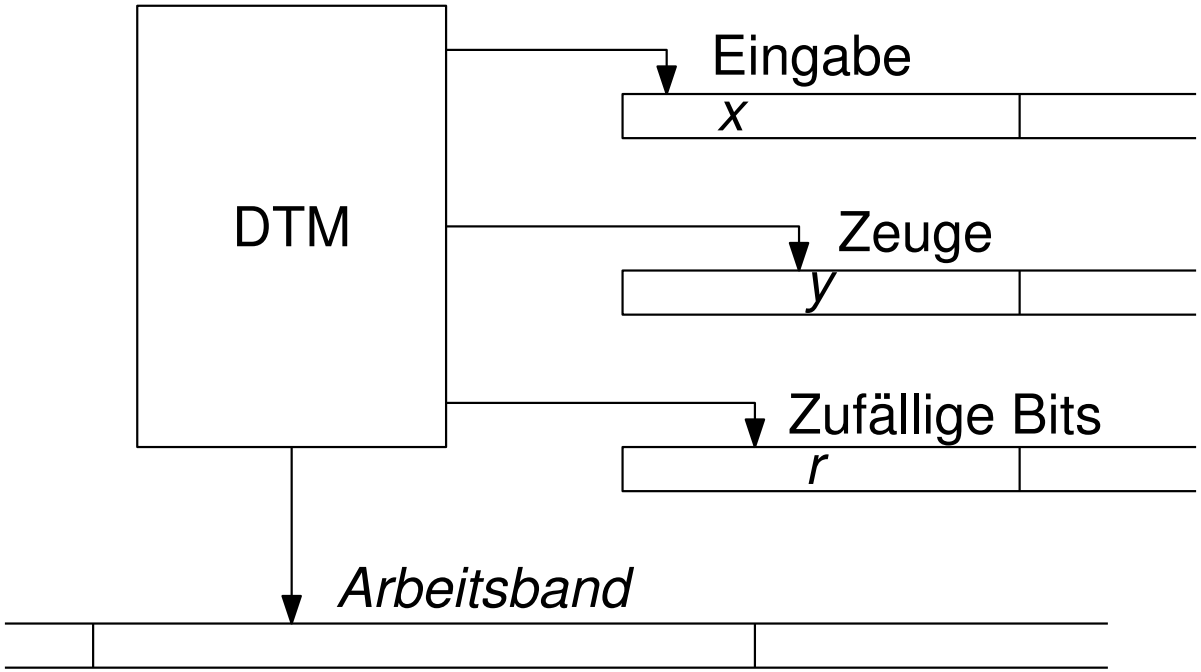
Verifier



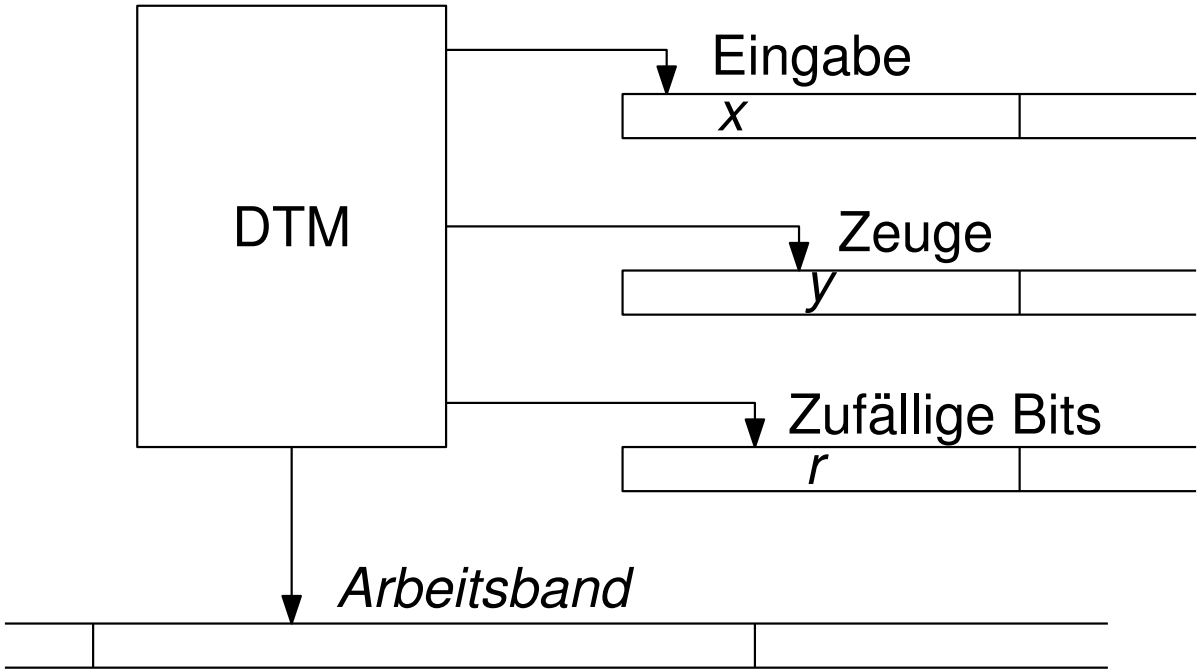
Verifier



Verifier



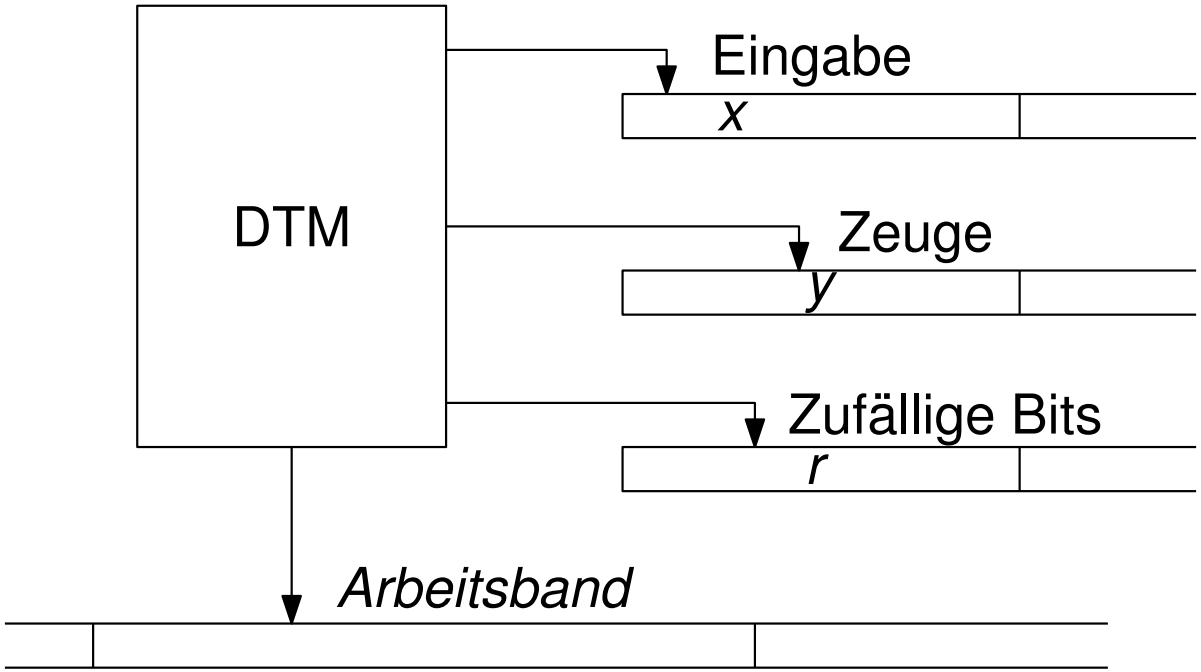
Verifier



Beschränkungen bei Eingabe der Länge n:

- Laufzeit polynomiell in n
- $O(r(n))$ Zufallsbits
- $O(q(n))$ Bits aus dem Beweis lesen

Verifier



Beschränkungen bei Eingabe der Länge n:

- Laufzeit polynomiell in n
- $O(r(n))$ Zufallsbits
- $O(q(n))$ Bits aus dem Beweis lesen

Definiert Problemklassen $\mathrm{PCP}(r(n), q(n))$.

Die Klassen $\mathrm{PCP}(r(n), q(n))$

Definition ($\mathrm{PCP}(r(n), q(n))$)

$L \in \mathrm{PCP}(r(n), q(n)) :\Leftrightarrow$

$\exists$ Verifier V, der

- $\gg$ $O(r(n))$ Zufallsbits liest
- $\gg$ $O(q(n))$ Bits aus Beweis liest

Verifier V verhält sich wie folgt:

- $\gg$ $x \in L \Rightarrow \exists$ Zeuge y, so dass V mit WK 1 akzeptiert.
- $\gg$ $x \notin L \Rightarrow \forall$ Zeugen y akzeptiert V mit WK $< 1/2$

Die Klassen $\mathrm{PCP}(r(n), q(n))$

Definition ($\mathrm{PCP}(r(n), q(n))$)

$L \in \mathrm{PCP}(r(n), q(n)) :\Leftrightarrow$ $\exists$ Verifier V, der

- $O(r(n))$ Zufallsbits liest
- $O(q(n))$ Bits aus Beweis liest

Verifier V verhält sich wie folgt:

- $x \in L \Rightarrow \exists$ Zeuge y, so dass V mit WK 1 akzeptiert.
- $x \notin L \Rightarrow \forall$ Zeugen y akzeptiert V mit WK $< 1/2$

$\mathrm{PCP}(0, 0) = ?$

Die Klassen $\mathrm{PCP}(r(n), q(n))$

Definition ($\mathrm{PCP}(r(n), q(n))$)

$L \in \mathrm{PCP}(r(n), q(n)) :\Leftrightarrow$

$\exists$ Verifier V, der

- $\gg$ $O(r(n))$ Zufallsbits liest
- $\gg$ $O(q(n))$ Bits aus Beweis liest

Verifier V verhält sich wie folgt:

- $\gg$ $x \in L \Rightarrow \exists$ Zeuge y, so dass V mit WK 1 akzeptiert.
- $\gg$ $x \notin L \Rightarrow \forall$ Zeugen y akzeptiert V mit WK $< 1/2$

$\mathrm{PCP}(0, 0) = \mathcal{P}$

Die Klassen $\mathrm{PCP}(r(n), q(n))$

Definition ($\mathrm{PCP}(r(n), q(n))$)

$L \in \mathrm{PCP}(r(n), q(n)) :\Leftrightarrow$

$\exists$ Verifier V, der

- $O(r(n))$ Zufallsbits liest
- $O(q(n))$ Bits aus Beweis liest

Verifier V verhält sich wie folgt:

- $x \in L \Rightarrow \exists$ Zeuge y, so dass V mit WK 1 akzeptiert.
- $x \notin L \Rightarrow \forall$ Zeugen y akzeptiert V mit WK $< 1/2$

$\mathrm{PCP}(0, 0) = \mathcal{P}$

$\mathrm{PCP}(0, \mathrm{poly}(n)) = ?$

Die Klassen $\mathrm{PCP}(r(n), q(n))$

Definition ($\mathrm{PCP}(r(n), q(n))$)

$L \in \mathrm{PCP}(r(n), q(n)) :\Leftrightarrow$

$\exists$ Verifier V, der

- $O(r(n))$ Zufallsbits liest
- $O(q(n))$ Bits aus Beweis liest

Verifier V verhält sich wie folgt:

- $x \in L \Rightarrow \exists$ Zeuge y, so dass V mit WK 1 akzeptiert.
- $x \notin L \Rightarrow \forall$ Zeugen y akzeptiert V mit WK $< 1/2$

$\mathrm{PCP}(0, 0) = \mathcal{P}$

$\mathrm{PCP}(0, \mathrm{poly}(n)) = \mathcal{NP}$

Die Klassen $\mathrm{PCP}(r(n), q(n))$

Definition ($\mathrm{PCP}(r(n), q(n))$)

$L \in \mathrm{PCP}(r(n), q(n)) :\Leftrightarrow$ $\exists$ Verifier V, der

- $\gg$ $O(r(n))$ Zufallsbits liest
- $\gg$ $O(q(n))$ Bits aus Beweis liest

Verifier V verhält sich wie folgt:

- $\gg$ $x \in L \Rightarrow \exists$ Zeuge y, so dass V mit WK 1 akzeptiert.
- $\gg$ $x \notin L \Rightarrow \forall$ Zeugen y akzeptiert V mit WK $< 1/2$

$\mathrm{PCP}(0, 0) = \mathcal{P}$

$\mathrm{PCP}(0, \mathrm{poly}(n)) = \mathcal{NP}$

$\mathrm{PCP}(\mathrm{poly}(n), 0) = ?$

Die Klassen $\mathrm{PCP}(r(n), q(n))$

Definition ($\mathrm{PCP}(r(n), q(n))$)

$L \in \mathrm{PCP}(r(n), q(n)) :\Leftrightarrow$

$\exists$ Verifier V, der

- $\gg$ $O(r(n))$ Zufallsbits liest
- $\gg$ $O(q(n))$ Bits aus Beweis liest

Verifier V verhält sich wie folgt:

- $\gg$ $x \in L \Rightarrow \exists$ Zeuge y, so dass V mit WK 1 akzeptiert.
- $\gg$ $x \notin L \Rightarrow \forall$ Zeugen y akzeptiert V mit WK $< 1/2$

$\mathrm{PCP}(0, 0) = \mathcal{P}$

$\mathrm{PCP}(0, \mathrm{poly}(n)) = \mathcal{NP}$

$\mathrm{PCP}(\mathrm{poly}(n), 0) = \text{co-}\mathcal{RP}$

Das PCP-Theorem

Satz

$$\mathrm{PCP}(\log n, 1) = \mathcal{NP}$$

Das PCP-Theorem

Satz

$$\mathrm{PCP}(\log n, 1) = \mathcal{NP}$$

Beweisidee:

"$\subseteq$:" $L \in \mathrm{PCP}(\log n, 1) \Rightarrow \exists$ geeigneter Verifier V.

Das PCP-Theorem

Satz

$$\mathrm{PCP}(\log n, 1) = \mathcal{NP}$$

Beweisidee:
"$\subseteq$:" $L \in \mathrm{PCP}(\log n, 1) \Rightarrow \exists$ geeigneter Verifier V.
Konstruiere NP-Maschine für L:

1. Rate Zeugen y zu Eingabe x.
2. Simuliere V für alle Folgen von Zufallsbits ($O(n)$ Folgen).
3. Akzeptiere x wenn V immer akzeptiert hat.

Das PCP-Theorem

Satz

$$\mathrm{PCP}(\log n, 1) = \mathcal{NP}$$

Beweisidee:

"$\subseteq$:" $L \in \mathrm{PCP}(\log n, 1) \Rightarrow \exists$ geeigneter Verifier V.

Konstruiere NP-Maschine für L:

1. Rate Zeugen y zu Eingabe x.
2. Simuliere V für alle Folgen von Zufallsbits ($O(n)$ Folgen).
3. Akzeptiere x wenn V immer akzeptiert hat.

"$\supseteq$:" schwer!

Das PCP-Theorem

Satz

$$\mathrm{PCP}(\log n, 1) = \mathcal{NP}$$

Beweisidee:

"$\subseteq$:" $L \in \mathrm{PCP}(\log n, 1) \Rightarrow \exists$ geeigneter Verifier V.

Konstruiere NP-Maschine für L:

1. Rate Zeugen y zu Eingabe x.
2. Simuliere V für alle Folgen von Zufallsbits ($O(n)$ Folgen).
3. Akzeptiere x wenn V immer akzeptiert hat.

"$\supseteq$:" schwer!

Intuition: $3\mathrm{SAT}$, Zeuge ist Variablenbelegung

- $\gg$ wähle zufällige Klausel C, prüfe ob C erfüllt
- $\gg$ WK für Akzeptanz nicht erfüllbarer Formel $\leq 1 - 1/m$

PCP-Theorem und Nichtapproximierbarkeit

Problem (Maximiere Akzeptanzwahrscheinlichkeit (MAWK))

Sei V $\mathrm{PCP}(\log n, 1)$-Verifier für SAT.
Finde zu Eingabe ϕ Zeugen, der Akzeptanz-WK maximiert.

PCP-Theorem und Nichtapproximierbarkeit

Problem (Maximiere Akzeptanzwahrscheinlichkeit (MAWK))

Sei V $\mathrm{PCP}(\log n, 1)$-Verifier für SAT.
Finde zu Eingabe ϕ Zeugen, der Akzeptanz-WK maximiert.

Behauptung

$\mathcal{P} \neq \mathcal{NP} \Rightarrow \nexists$ *1/2-Approximations-Algorithmus A für* MAWK.

PCP-Theorem und Nichtapproximierbarkeit

Problem (Maximiere Akzeptanzwahrscheinlichkeit (MAWK))

Sei V $\mathrm{PCP}(\log n, 1)$-Verifier für SAT.
Finde zu Eingabe ϕ Zeugen, der Akzeptanz-WK maximiert.

Behauptung

$\mathcal{P} \neq \mathcal{NP} \Rightarrow \nexists$ *1/2-Approximations-Algorithmus A für* MAWK.

Beweis:

- ϕ erfüllbar $\Rightarrow \exists$ Zeuge, so dass V immer akzeptiert.
- ϕ nicht erfüllbar $\Rightarrow$ V akzeptiert mit WK $< 1/2$

PCP-Theorem und Nichtapproximierbarkeit

Problem (Maximiere Akzeptanzwahrscheinlichkeit (MAWK))

Sei V $\mathrm{PCP}(\log n, 1)$-Verifier für SAT.
Finde zu Eingabe ϕ Zeugen, der Akzeptanz-WK maximiert.

Behauptung

$\mathcal{P} \neq \mathcal{NP} \Rightarrow \nexists$ *1/2-Approximations-Algorithmus A für* MAWK.

Beweis:

- ϕ erfüllbar $\Rightarrow \exists$ Zeuge, so dass V immer akzeptiert.
- ϕ nicht erfüllbar $\Rightarrow$ V akzeptiert mit WK $< 1/2$

ϕ erfüllbar $\Rightarrow$ A findet Zeugen mit Akzeptanz-WK $\geq 1/2$.

PCP-Theorem und Nichtapproximierbarkeit

Problem (Maximiere Akzeptanzwahrscheinlichkeit (MAWK))

Sei V $\mathrm{PCP}(\log n, 1)$-Verifier für SAT.
Finde zu Eingabe ϕ Zeugen, der Akzeptanz-WK maximiert.

Behauptung

$\mathcal{P} \neq \mathcal{NP} \Rightarrow \nexists$ *1/2-Approximations-Algorithmus A für* MAWK.

Beweis:

- ϕ erfüllbar $\Rightarrow \exists$ Zeuge, so dass V immer akzeptiert.
- ϕ nicht erfüllbar $\Rightarrow$ V akzeptiert mit WK $< 1/2$

ϕ erfüllbar $\Rightarrow$ A findet Zeugen mit Akzeptanz-WK $\geq 1/2$.

Akzeptanz-WK kann in poly-Zeit berechnet werden. (Simulation!)

PCP-Theorem und Nichtapproximierbarkeit

Problem (Maximiere Akzeptanzwahrscheinlichkeit (MAWK))

Sei V $\mathrm{PCP}(\log n, 1)$-Verifier für SAT.
Finde zu Eingabe ϕ Zeugen, der Akzeptanz-WK maximiert.

Behauptung

$\mathcal{P} \neq \mathcal{NP} \Rightarrow \nexists$ *1/2-Approximations-Algorithmus A für* MAWK.

Beweis:

- ϕ erfüllbar $\Rightarrow \exists$ Zeuge, so dass V immer akzeptiert.
- ϕ nicht erfüllbar $\Rightarrow$ V akzeptiert mit WK $< 1/2$

ϕ erfüllbar $\Rightarrow$ A findet Zeugen mit Akzeptanz-WK $\geq 1/2$.

Akzeptanz-WK kann in poly-Zeit berechnet werden. (Simulation!)

$\Rightarrow$ Mit A könnte SAT in polynomieller Zeit entschieden werden.

Max3Sat

Satz

Es gibt eine Konstante $\varepsilon_M > 0$, mit lückeneinführender Reduktion von Sat *auf* Max3Sat, *die ϕ zu ψ transformiert mit*

- ϕ *erfüllbar* $\Rightarrow \mathrm{OPT}(\psi) = m$
- ϕ *nicht erfüllbar* $\Rightarrow \mathrm{OPT}(\psi) < (1 - \varepsilon_M)m$

$m = \#$ *Klauseln von* ψ

Max3Sat

Satz

Es gibt eine Konstante $\varepsilon_M > 0$, mit lückeneinführender Reduktion von Sat *auf* Max3Sat, *die ϕ zu ψ transformiert mit*

- ϕ *erfüllbar* $\Rightarrow \mathrm{OPT}(\psi) = m$
- ϕ *nicht erfüllbar* $\Rightarrow \mathrm{OPT}(\psi) < (1 - \varepsilon_M)m$

$m = \#$ *Klauseln von* ψ

Korollar

Keine $(1 - \varepsilon_M)$-Approximation für Max3Sat, *wenn $\mathcal{P} \neq \mathcal{NP}$.*

Max k-Function Sat

Problem (Max k-Function Sat)

Gegeben:

- *boolesche Variablen $x_1, \ldots, x_n$, Funktionen $f_1, \ldots, f_m$,*
- *jedes f_i verwendet nur k Variablen.*

Gesucht:

Belegung, die Anzahl der erfüllten f_i maximiert.

Max k-Function Sat

Problem (Max k-Function Sat)

Gegeben:

- *boolesche Variablen $x_1, \ldots, x_n$, Funktionen $f_1, \ldots, f_m$,*
- *jedes f_i verwendet nur k Variablen.*

Gesucht:
Belegung, die Anzahl der erfüllten f_i maximiert.

Lemma

Es gibt Konstante k und lückeneinführende Reduktion von Sat *auf* Max k-Function Sat*: ϕ wird transformiert zu Instanz I mit*

- *ϕ erfüllbar $\Rightarrow \mathrm{OPT}(I) = m$*
- *ϕ nicht erfüllbar $\Rightarrow \mathrm{OPT}(I) < \frac{1}{2}m$*

Beweis des Lemmas

Sei V $\mathrm{PCP}(\log n, 1)$-Verifier, ϕ Instanz von Sat.

$\Rightarrow$ V verwendet $c \log n$ Zufallsbits und liest q Zeugenbits.

V liest höchstens qn^c Bits des Zeugen

$\Rightarrow$ Eine Variable für jedes dieser Bit – Variablenmenge B

Beweis des Lemmas

Sei V $\mathrm{PCP}(\log n, 1)$-Verifier, ϕ Instanz von Sat.

$\Rightarrow$ V verwendet $c \log n$ Zufallsbits und liest q Zeugenbits.

V liest höchstens qn^c Bits des Zeugen

$\Rightarrow$ Eine Variable für jedes dieser Bit – Variablenmenge B

Für jedes Zufallswort r definiere boolesche Funktion f_r:
Akzeptanz/Rückweisung durch V ist Funktion von ϕ, r und q Bits.
Für festes ϕ, r nur von q festen Bits des Beweises abhängig!

Menge von n^c Funktionen auf B, jedes f_r verwendet q Variablen.

Beweis des Lemmas

Sei V $\mathrm{PCP}(\log n, 1)$-Verifier, ϕ Instanz von Sat.

$\Rightarrow$ V verwendet $c \log n$ Zufallsbits und liest q Zeugenbits.

V liest höchstens qn^c Bits des Zeugen

$\Rightarrow$ Eine Variable für jedes dieser Bit – Variablenmenge B

Für jedes Zufallswort r definiere boolesche Funktion f_r:
Akzeptanz/Rückweisung durch V ist Funktion von ϕ, r und q Bits.
Für festes ϕ, r nur von q festen Bits des Beweises abhängig!

Menge von n^c Funktionen auf B, jedes f_r verwendet q Variablen.

- ϕ erfüllbar $\Rightarrow \exists$ Zeuge, so dass V mit WK 1 akzeptiert.
  $\Rightarrow$ Entsprechende Variablenbelegung erfüllt alle f_r.
- ϕ nicht erfüllbar $\Rightarrow$ V akzeptiert $\forall$ Zeugen mit WK $< 1/2$.
  $\Rightarrow$Jede Wahrheitsbelegung erfüllt $< \frac{1}{2} n^c$ der Funktionen

Zwischenstand

Lemma

Es gibt Konstante k und lückeneinführende Reduktion von Sat *auf* Max k-Function Sat*: ϕ wird transformiert zu Instanz I mit*

- ϕ *erfüllbar* $\Rightarrow \mathrm{OPT}(I) = m$
- ϕ *nicht erfüllbar* $\Rightarrow \mathrm{OPT}(I) < \frac{1}{2}m$

Zwischenstand

Lemma

Es gibt Konstante k und lückeneinführende Reduktion von Sat *auf* Max k-Function Sat*: ϕ wird transformiert zu Instanz I mit*

- ϕ *erfüllbar* $\Rightarrow \mathrm{OPT}(I) = m$
- ϕ *nicht erfüllbar* $\Rightarrow \mathrm{OPT}(I) < \frac{1}{2}m$

Satz

Es gibt eine Konstante $\varepsilon_M > 0$, mit lückeneinführender Reduktion von Sat *auf* Max3Sat*, die ϕ zu ψ transformiert mit*

- ϕ *erfüllbar* $\Rightarrow \mathrm{OPT}(\psi) = m$
- ϕ *nicht erfüllbar* $\Rightarrow \mathrm{OPT}(\psi) < (1 - \varepsilon_M)m$

$m = \#$ *Klauseln von* ψ

Beweis des Satzes

Transformiere Instanz ϕ von Sat auf Max k-Function Sat.

$\rightsquigarrow f_r$: n^c boolesche Funktion auf q Variablen.

Schreibe f_r als Sat-Formel ψ_r mit höchstens 2^q Klauseln.
Klauseln von ψ_r enthalten höchstens q Literale.

Setze $\psi := \bigwedge_r \psi_r$

- ϕ erfüllbar $\Rightarrow$ ψ erfüllbar.
- Sonst für jede Belegung $> \frac{1}{2} n^c$ Klauseln von ψ nicht erfüllt.

Beweis des Satzes

Transformiere Instanz ϕ von Sat auf Max k-Function Sat.

$\rightsquigarrow$ f_r: n^c boolesche Funktion auf q Variablen.

Schreibe f_r als Sat-Formel ψ_r mit höchstens 2^q Klauseln.
Klauseln von ψ_r enthalten höchstens q Literale.

Setze $\psi := \bigwedge_r \psi_r$

- ϕ erfüllbar $\Rightarrow$ ψ erfüllbar.
- Sonst für jede Belegung $> \frac{1}{2} n^c$ Klauseln von ψ nicht erfüllt.

Erzeuge zu ψ äquivalente 3Sat-Formel ψ'.

- ψ' hat höchstens $n^c 2^q (q-2)$ Klauseln.
- ϕ erfüllbar $\Rightarrow$ ψ' erfüllbar.
- Sonst für jede Belegung $> \frac{1}{2} n^c$ Klauseln nicht erfüllt.

Beweis des Satzes

Transformiere Instanz ϕ von Sat auf Max k-Function Sat.

$\rightsquigarrow$ f_r: n^c boolesche Funktion auf q Variablen.

Schreibe f_r als Sat-Formel ψ_r mit höchstens 2^q Klauseln.
Klauseln von ψ_r enthalten höchstens q Literale.

Setze $\psi := \bigwedge_r \psi_r$

- ϕ erfüllbar $\Rightarrow$ ψ erfüllbar.
- Sonst für jede Belegung $> \frac{1}{2}n^c$ Klauseln von ψ nicht erfüllt.

Erzeuge zu ψ äquivalente 3Sat-Formel ψ'.

- ψ' hat höchstens $n^c 2^q(q-2)$ Klauseln.
- ϕ erfüllbar $\Rightarrow$ ψ' erfüllbar.
- Sonst für jede Belegung $> \frac{1}{2}n^c$ Klauseln nicht erfüllt.

$\rightsquigarrow$ Satz für $\varepsilon_M = 1/(2^{q+1}(q-2))$

Max3Sat(k)

Problem (Max3Sat(k))

Max3Sat(k) *ist Einschränkung von* Max3Sat *auf Formeln, in denen jede Variable höchstens* k *mal vorkommt.*

Satz

Es gibt eine lückenerhaltende Reduktion von Max3Sat *auf* Max3Sat*(29) die* ϕ *transformiert zu* ψ *mit:*

- $\mathrm{OPT}(\phi) = m \Rightarrow \mathrm{OPT}(\psi) = m'$
- $\mathrm{OPT}(\phi) < (1 - \varepsilon_M)m \Rightarrow \mathrm{OPT}(\psi) < (1 - \varepsilon_b)m'$

m, m' *Anzahl Klauseln von* ϕ, ψ.
$\varepsilon_b = \varepsilon_m/43$

Expander-Graphen

Definition (Expander-Graph)

$G = (V, E)$ heißt Expander, wenn jeder Knoten denselben Grad hat, und für jede nichtleere Menge $S \subseteq V$:

$$|E(S, \overline{S})| > \min(|S|, |\overline{S}|)$$

Für Variable x erzeuge Expander G_x vom Grad 14 mit k Knoten.
Knoten von G_x entspricht Variable x_i.

Erzeuge Formel ψ_x:
Für $x_i x_j$ in G_x füge $(\overline{x_i} \vee x_j)$, $(x_i \vee \overline{x_j})$ zu ψ_x hinzu.

$V_x = \{x_1, \ldots, x_k\}$
Belegung von $x_1, \ldots, x_k$ *konsistent*, wenn alle x_i gleich belegt.
Inkonsistente Belegung definiert Schnitt $(S, \overline{S})$.

$$\Rightarrow \text{mindestens } \min(|S|, |\overline{S}|) + 1 \text{ verletzte Klauseln in } \psi_x.$$

Reduktion I

ϕ Max3Sat-Instanz, B Menge der Variablen in ϕ.

x Variable in B mit k Vorkommnissen in ϕ.

1. $V_x = \{x_1, \ldots, x_k\}$.
2. Konstruiere G_x, ψ_x wie vorher.
3. Ersetze Vorkommen von x in ϕ durch eindeutiges $x_i \in V_x$.
   $\Rightarrow$ Erhalten Formel ϕ'.
4. Definiere ψ:

$$\psi = \phi' \wedge (\bigwedge_{x \in B} \psi_x).$$

τ optimale Wahrheitbelegung $\Rightarrow$ τ konsistent auf $V_x \quad \forall x \in B$

Reduktion II

- ϕ hat m Klauseln, ψ hat m' Klauseln
- höchstens $3m$ Auftreten von Variablen in ϕ
- Gesamt: $m' \leq 43m$

- ϕ erfüllbar $\Rightarrow$ ψ erfüllbar
- $\mathrm{OPT}(\phi) < (1 - \varepsilon_M)m \Rightarrow \mathrm{OPT}(\psi) < (1 - \varepsilon_M/43)m'$

Reduktion II

- ϕ hat m Klauseln, ψ hat m' Klauseln
- höchstens $3m$ Auftreten von Variablen in ϕ
- Gesamt: $m' \leq 43m$

- ϕ erfüllbar $\Rightarrow$ ψ erfüllbar
- $\mathrm{OPT}(\phi) < (1 - \varepsilon_M)m \Rightarrow \mathrm{OPT}(\psi) < (1 - \varepsilon_M/43)m'$

Korollar

Es gibt eine lückenerhaltende Reduktion von Max3Sat*(29) auf* Max3Sat*(5).*

VertexCover

Definition

Für $d \geq 1$ ist $\mathrm{VC}(d)$ die Einschränkung von VertexCover auf Instanzen mit Maximalgrad d.

Satz

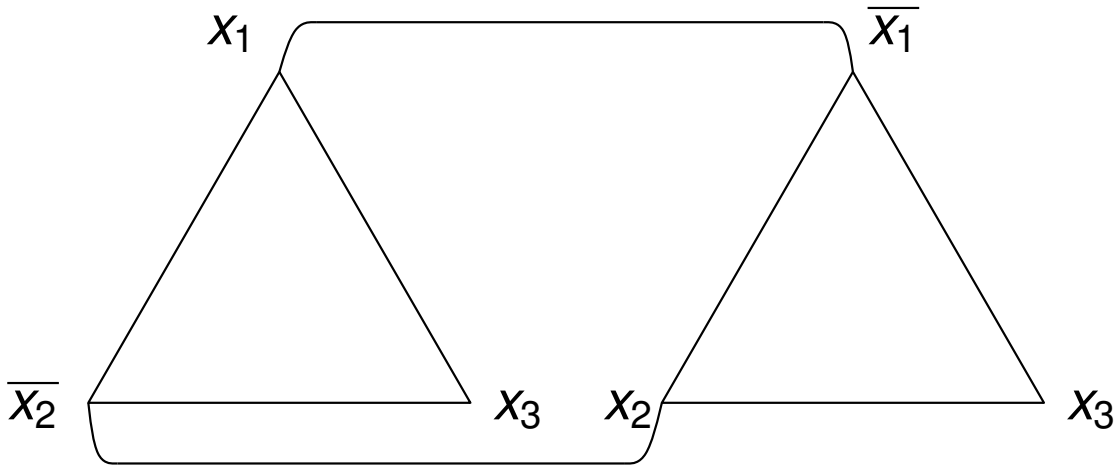
Es gibt lückenerhaltende Reduktion von Max3Sat*(29) auf* $\mathrm{VC}(30)$, *die Instanz* ϕ *einen Graphen* G *zuordnet, mit*

- $\mathrm{OPT}(\phi) = m \Rightarrow \mathrm{OPT}(G) \leq \frac{2}{3}|V|$
- $\mathrm{OPT}(\phi) < (1 - \varepsilon_b)m \Rightarrow \mathrm{OPT}(G) > (1 + \varepsilon_v)\frac{2}{3}|V|$

$m =$ #*Klauseln von* ϕ, $\varepsilon_v = \varepsilon_b/2$

VertexCover

ϕ Instanz von Max3Sat(29) mit m Klauseln



unabhängige Menge I entspricht Variablenbelegung

- I^c ist Vertex cover
- $\mathrm{OPT}(\phi) = m \Rightarrow \mathrm{OPT}(G) = 2m$
- $\mathrm{OPT}(\phi) < (1 - \varepsilon_b)m \Rightarrow \mathrm{OPT}(G) > (2 + \varepsilon_b)m$

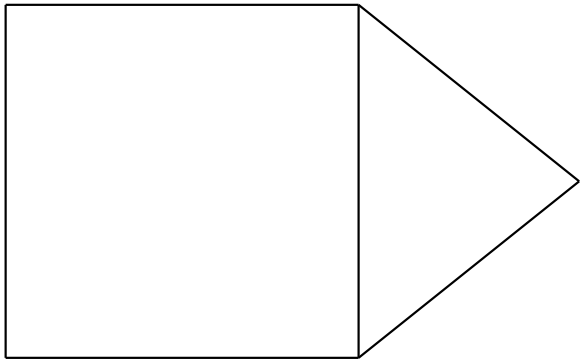
STEINER-BAUM

Satz

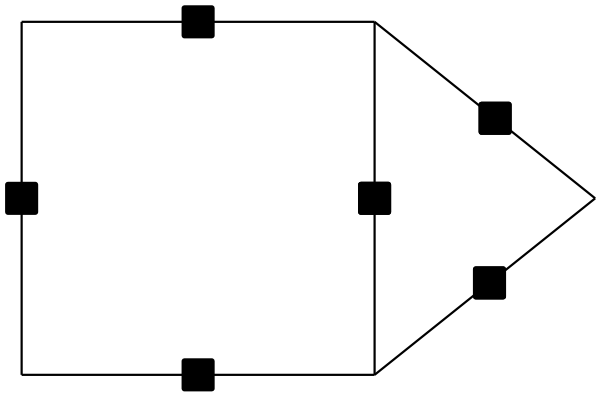
Es gibt eine lückenerhaltende Reduktion von VC(30)-Instanz G auf STEINER-BAUM *auf Steiner-Baum-Instanz $H = (R, S, \text{cost})$:*

- $\text{OPT}(G) \leq \frac{2}{3}|V| \Rightarrow \text{OPT}(H) \leq |R| + \frac{2}{3}|S| - 1$
- $\text{OPT}(G) > (1+\varepsilon_v)\frac{2}{3}|V| \Rightarrow \text{OPT}(H) > (1+\epsilon_s)(|R|+\frac{2}{3}|S|-1)$

Beweis des Satzes



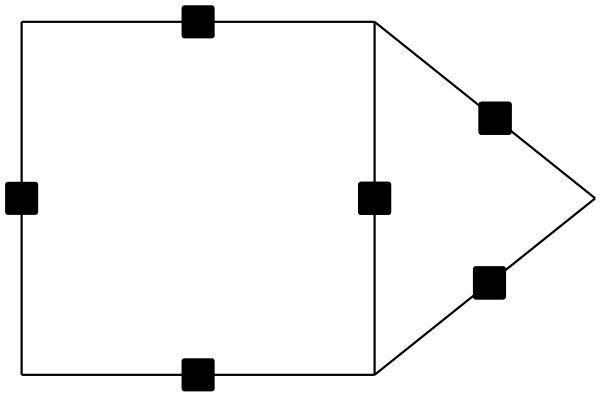
Beweis des Satzes



■: Terminalknoten

Rest: Steinerknoten

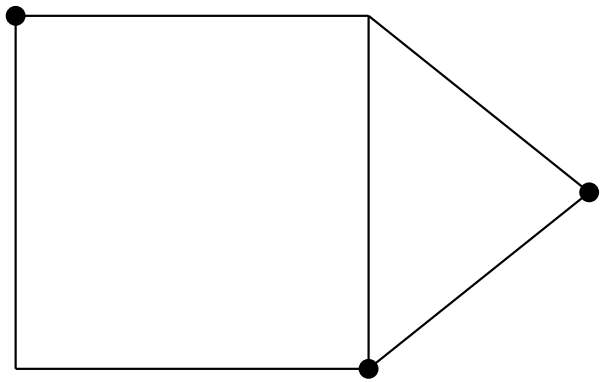
Beweis des Satzes



■: Terminalknoten
Rest: Steinerknoten

- Kanten zw. adjazentem Steinerknoten und Terminal: Kosten 1
- Kanten zwischen Steiner-Knoten: Kosten 1
- Alle anderen Kanten: Kosten 2

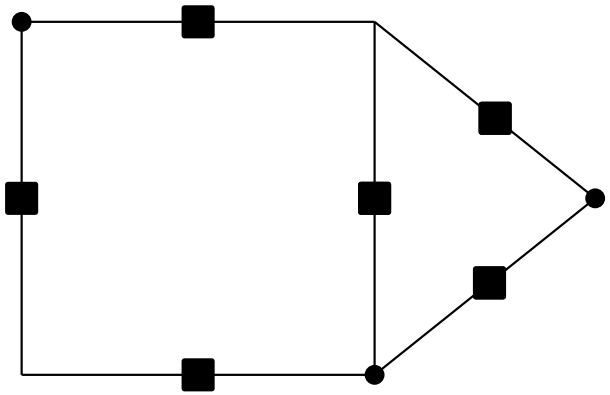
Beweis des Satzes



■: Terminalknoten
Rest: Steinerknoten

- Kanten zw. adjazentem Steinerknoten und Terminal: Kosten 1
- Kanten zwischen Steiner-Knoten: Kosten 1
- Alle anderen Kanten: Kosten 2

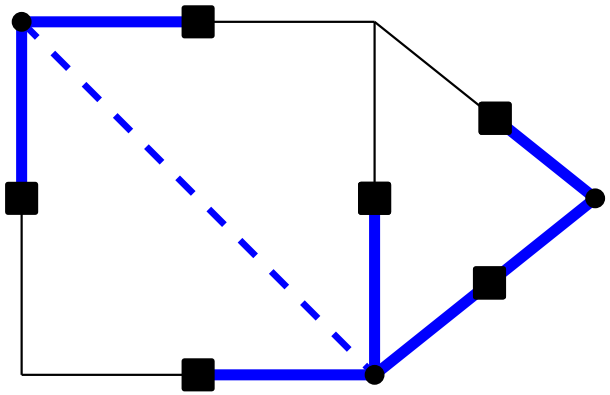
Beweis des Satzes



■: Terminalknoten
Rest: Steinerknoten

- Kanten zw. adjazentem Steinerknoten und Terminal: Kosten 1
- Kanten zwischen Steiner-Knoten: Kosten 1
- Alle anderen Kanten: Kosten 2

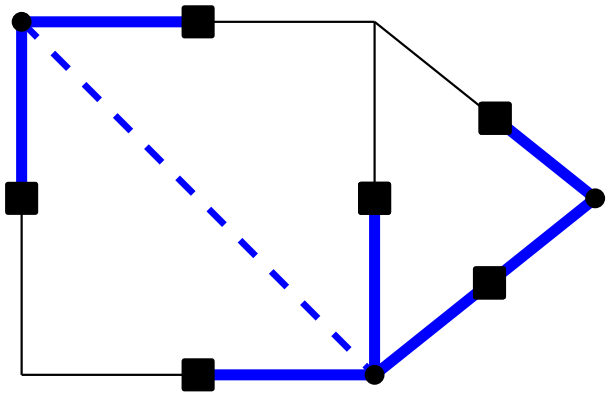
Beweis des Satzes



■: Terminalknoten
Rest: Steinerknoten

- Kanten zw. adjazentem Steinerknoten und Terminal: Kosten 1
- Kanten zwischen Steiner-Knoten: Kosten 1
- Alle anderen Kanten: Kosten 2

Beweis des Satzes

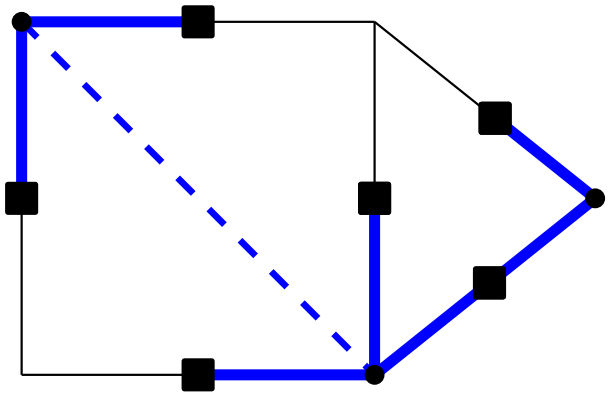


■: Terminalknoten
Rest: Steinerknoten

- Kanten zw. adjazentem Steinerknoten und Terminal: Kosten 1
- Kanten zwischen Steiner-Knoten: Kosten 1
- Alle anderen Kanten: Kosten 2

Vertex cover mit c Knoten $\equiv$ Baum mit Kosten $|R| + c - 1$

Beweis des Satzes



■: Terminalknoten
Rest: Steinerknoten

- Kanten zw. adjazentem Steinerknoten und Terminal: Kosten 1
- Kanten zwischen Steiner-Knoten: Kosten 1
- Alle anderen Kanten: Kosten 2

Vertex cover mit c Knoten $\equiv$ Baum mit Kosten $|R| + c - 1$

- $\mathrm{OPT}(G) \leq \frac{2}{3}|V| \Rightarrow \mathrm{OPT}(H) \leq |R| + \frac{2}{3}|S| - 1$
- $\mathrm{OPT}(G) > (1+\varepsilon_v)\frac{2}{3}|V|$
  $\Rightarrow \mathrm{OPT}(H) > |R| + (1+\epsilon_v)\frac{2}{3}|S| - 1 > (1+\epsilon_s)(|R| + \frac{2}{3}|S| - 1)$

Zusammenfassung