

# Näherungsalgorithmen (Approximationsalgorithmen)

WiSe 2012/13 in Trier

Henning Fernau  
Universität Trier  
fernau@uni-trier.de

30. Januar 2013

# Näherungsalgorithmen

Gesamtübersicht

- Organisatorisches
- Einführung / Motivation
- Grundtechniken für Näherungsalgorithmen
- Approximationsklassen (Approximationstheorie)

## Approximationstheorie

- Absolute Approximation
- Relative Approximation: die Klasse APX
- Polynomzeit-Approximationsschemata PTAS
- Zwischen APX und NPO
- Zwischen PTAS und APX
- Approximationsklassen und Reduktionen

## Bekannte Begriffe

- Turing-Reduktion (sehr allgemein)
- Karp-Reduktion (abgeschwächter Begriff)

Ein Entscheidungsproblem  $\mathcal{P}_1$  heißt *Karp-reduzierbar* (oder *many-one-reduzierbar*) auf ein Entscheidungsproblem  $\mathcal{P}_2$ , wenn es einem (Polynomzeit-)Algorithmus  $R$  gibt, der eine Instanz  $x$  von  $\mathcal{P}_1$  in eine Instanz  $y$  von  $\mathcal{P}_2$  überführt in einer Weise, dass  $x$  eine JA-Instanz von  $\mathcal{P}_1$  ist gdw.  $y$  eine JA-Instanz von  $\mathcal{P}_2$  ist.

## Erinnerung: NP-Theorie

**Zentrales Anliegen:** Probleme zu kennen, die hart für NP sind in dem Sinne, dass ein deterministischer Polynomzeitalgorithmus für *ein* solches Problem die Existenz deterministischer Polynomzeitalgorithmen für *alle* NP-vollständigen Probleme nach sich ziehen würde.

Wir wollen etwas Entsprechendes auch im Falle der Optimierungsprobleme entwickeln, müssen uns aber erst einmal an die wichtigsten Dinge aus der NP-Vollständigkeitstheorie erinnern, da die Verhältnisse dort einfacher sind.

## Das „generische“ NP-vollständige Problem ist:

**Ggb:** nichtdeterministische Turing-Maschine, Eingabe  $x$  von TM, Polynom  $p$

**Frage:** Akzeptiert TM das Wort  $x$  in höchstens  $p(|x|)$  Schritten?

Dieses Problem liegt in NP, und würde es in P liegen, so wäre  $P=NP$ .

Das vielleicht wichtigste NP-vollständige Problem ist das **Erfüllbarkeitsproblem (SAT)**:

**Ggb:** KNF Formel  $\mathcal{F}$  auf einer Menge  $V$  von Booleschen Variablen.

**Frage:** Ist  $\mathcal{F}$  erfüllbar? D.h., gibt es eine Variablenbelegung  $f : V \rightarrow \{\text{true}, \text{false}\}$ , die  $\mathcal{F}$  „wahr macht“?

**Satz:** (Satz von Cook(-Levin))

Das Erfüllbarkeitsproblem ist NP-vollständig.

Zum Beweis verweisen wir auf andere Vorlesungen.

Die Beweisidee besteht in der formelmäßigen Darstellung des „Rechenteppichs“ einer Turing-Maschine.

Wir wollen den Karpischen Reduktionsbegriff an zwei Beispielen üben.

## Beispiel: $\{0, 1\}$ -Lineares Programmieren

**Ggb:** Menge von Variablen  $Z = \{z_1, \dots, z_n\}$ , die Werte aus  $\{0, 1\}$  annehmen können; Menge  $I$  von linearen Ungleichungen (mit Variablen aus  $Z$  und ganzzahligen Koeffizienten).

**Frage:** Hat  $I$  eine Lösung, d.h. irgendeine Variablenbelegung, die alle Ungleichungen erfüllt?

**Lemma:**  $\{0, 1\}$ -lineares Programmieren ist NP-hart.

**Beweis:** Betrachte eine Instanz  $\chi = (V, \mathcal{F})$  von SAT mit  $V = \{x_1, \dots, x_n\}$ . Es sei  $l_{j_1} \vee \dots \vee l_{j_{n_j}}$  die  $j$ -te Klausel in  $\mathcal{F}$ . Als entsprechende Ungleichung sehen wir  $\rho_{j_1} + \dots + \rho_{j_{n_j}} \geq 1$  an mit  $\rho_{j_x} = z_i$ , falls  $l_{j_k} = x_i$ , und  $\rho_{j_k} = (1 - z_i)$ , falls  $l_{j_k} = \bar{x}_i$ . Dadurch ergibt sich eine  $\{0, 1\}$ -LP-Instanz  $y = (Z, I)$ . Ist  $f : V \rightarrow \{\text{true}, \text{false}\}$  eine Wahrheitsbelegung, so ist  $f(\mathcal{F}) = \text{true}$  gdw.  $f'$  erfüllt alle Ungleichungen in  $I$ , wobei  $f'(z_i) = 1$  gdw.  $f(x_i) = \text{true}$ . □



**Beispiel:** 3SAT Def. wie SAT, nur dass jede Klausel (höchstens) drei Literale enthält.

**Lemma:** 3SAT ist NP-hart.

Beweis: Wir zeigen, wie allgemeine SAT-Formeln in (hinsichtlich der Erfüllbarkeit) äquivalente 3SAT-Formeln überführt werden können. Ist  $l_{j,1} \vee \dots \vee l_{j,n_j}$  eine Klausel mit  $n_j > 3$ , so kann durch Einführen von  $n_j - 3$  Variablen  $y_{j,1}, \dots, y_{j,n-3}$  und insgesamt  $n_j - 2$  Klauseln die 3SAT-Restriktion erfüllt werden. Die Klauseln sehen dafür wie folgt aus:

$$(l_{j,1} \vee l_{j,2} \vee y_{j,1}), (\overline{y_{j,1}} \vee l_{j,3} \vee y_{j,2}), \dots, (\overline{y_{j,n_j-4}} \vee l_{j,n_j-2} \vee y_{j,n_j-3}), (\overline{y_{j,n_j-3}} \vee l_{j,n_j-1} \vee l_{j,n_j})$$

□

## Die Welt von NPO-Problemen

Betrachten wir zunächst die folgende, den Begriff eines  $r$ -approximativen Algorithmus nur verallgemeinernde Definition:

Ist  $\mathcal{P}$  ein NPO-Problem,  $\mathcal{A}$  ein Approximationsalgorithmus für  $\mathcal{P}$  und  $r : \mathbb{N} \rightarrow (1, \infty)$  eine Abbildung, so heißt  $\mathcal{A}$   $r(n)$ -Approximation, falls für jede Instanz  $x \in I_{\mathcal{P}}$  mit  $S_{\mathcal{P}}(x) \neq \emptyset$  die Leistungsgüte der zulässigen Lösung  $\mathcal{A}(x)$  der Ungleichung  $R(x, \mathcal{A}) \leq r(|x|)$  genügt.

Das Verhalten des Algorithmus  $\mathcal{A}$  ist bei Eingaben, die keine zulässige Lösungen haben, unbestimmt. Natürlich wird keine Lösung zurückgeliefert.

Ist  $\mathcal{F}$  eine Klasse von Funktionen  $f : \mathbb{N} \rightarrow (0, \infty)$  so bezeichnet  $\mathcal{F}$ -APX die Klasse der Probleme, für die ein  $r(n)$ -approximativer Polynomzeitalgorithmus (für ein  $r \in \mathcal{F}$ ) existiert. Spezielle Funktionsklassen sind:

- LOG:=  $O(\log(n))$
- POLY:=  $\bigcup_{k>0} O(n^k)$
- EXP:=  $\bigcup_{k>0} O(2^{n^k})$

Satz:

$$\text{PTAS} \subseteq \text{APX} \subseteq \text{LOG} - \text{APX} \subseteq \text{POLY} - \text{APX} \subseteq \text{EXP} - \text{APX} \subseteq \text{NPO}.$$

Gilt vielleicht  $EXP - APX = NPO$  ?

Ein verführerisches Argument ist das Folgende:

Wegen der polynomiellen Schranke auf der Rechenzeit für die Maßfunktion  $m_{\mathcal{P}}$  ist doch jedes NPO-Problem  $\mathcal{P}$   $h \cdot 2^{n^k}$ -approximierbar für geeignete  $h$  und  $k$ .

**ABER:** Es gibt eben Probleme, für die bereits die Frage, ob eine zulässige Lösung existiert, NP-hart ist. Dazu gibt es im Folgenden Beispiele.

**Satz:** Wenn  $P \neq NP$ , so  $EXP - APX \neq NPO$ .

Beweis: Betrachten wir das folgende NPO-Problem:

**Minimum  $\{0, 1\}$ -LP**

$I = A \in \mathbb{Z}^{m \times n}, b \in \mathbb{Z}^m, w \in \mathbb{N}^n$

$S : x \in \{0, 1\}^n$  mit  $Ax \geq b$

$m : \sum w_i x_i$  (Skalarprodukt von  $w$  und  $x$ )

opt : min.

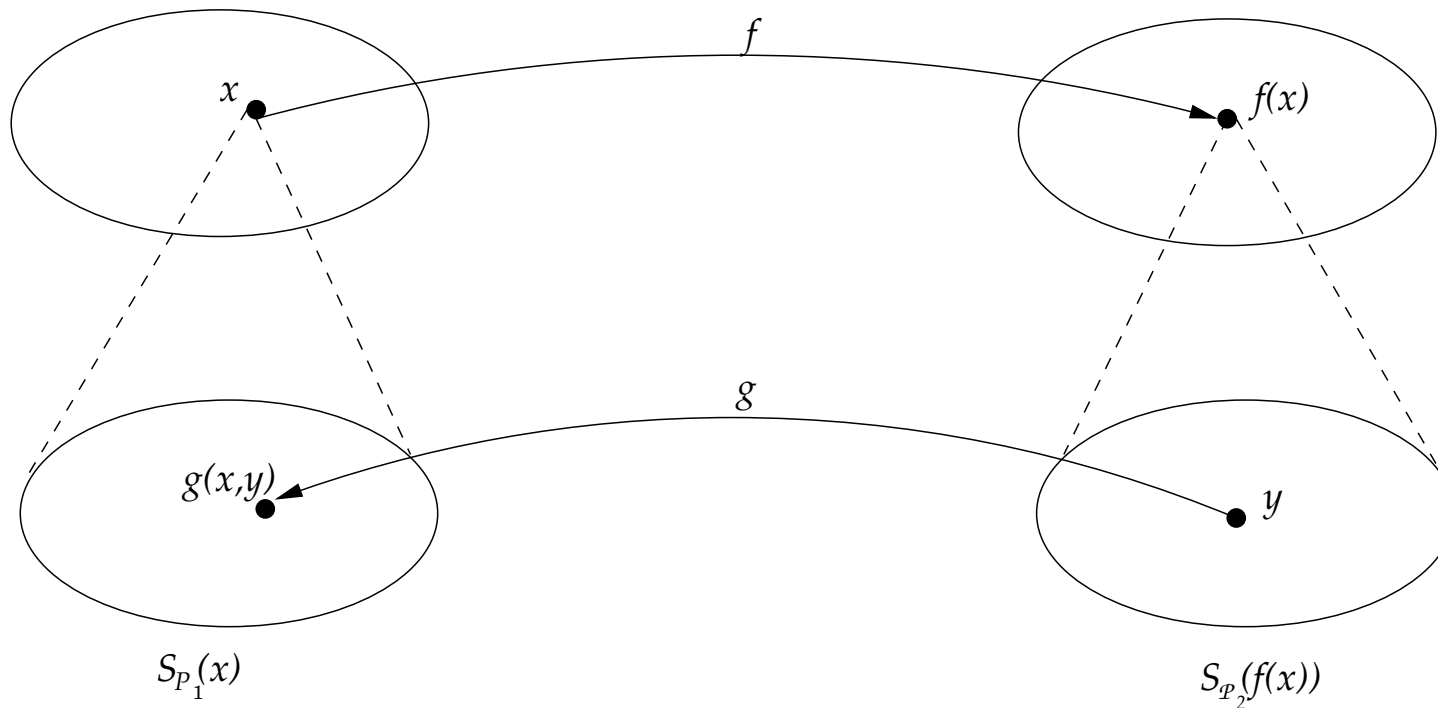
Wäre Minimum  $\{0, 1\}$  - LP  $\in EXP - APX$ , so könnten wir an dem Verhalten einer Polynomzeit-Approximation für eine Instanz  $x$  ablesen, ob dieselbe Instanz  $x$ , aufgefasst als  $\{0, 1\}$ -LP-Instanz, eine JA-Instanz ist oder nicht. Das Problem, überhaupt eine zulässige Lösung zu finden, haben wir im ersten Lemma betrachtet. □

## AP-Reduzierbarkeit

Bei Entscheidungsproblemen genügt es, einen Reduktionsbegriff von  $\mathcal{P}_1$  auf  $\mathcal{P}_2$  so zu definieren, dass man  $\mathcal{P}_1$  „mit Hilfe von“  $\mathcal{P}_2$  lösen kann, was beim Karp-schen Reduktionsbegriff bedeutet, dass Instanzen von  $\mathcal{P}_1$  in Instanzen von  $\mathcal{P}_2$  (in Polynomzeit) umgerechnet werden können. Dies genügt für einen Approximationsreduktionsbegriff nicht; vielmehr benötigen wir einen weiteren Algorithmus, der Lösungen von  $\mathcal{P}_2$  in solche für  $\mathcal{P}_1$  zurück rechnet, und letztere Rechnung sollte natürlich (in einem noch zu detaillierenden Sinne) die Näherungsgüte bewahren.

Schematisch können wir uns eine solche Approximationsreduktion wie folgt vorstellen.

## Schema einer AP-Reduktion



## Approximationsgütererhaltung am Bsp.: Knotenüberdeckung $\rightsquigarrow$ MaxClique

Ist  $G = (V, E)$  ein Graph, so ist der *Komplementgraph*  $G^c = (V, E^c)$  definiert durch  $E^c = \{\{v_1, v_2\} \subseteq V \mid v_1 \neq v_2, \{v_1, v_2\} \notin E\}$ .

**Lemma:**  $V' \subseteq V$  ist Knotenüberdeckung in  $G$  gdw,  $V \setminus V'$  ist Clique in  $G^c$ .

Beweis: Angenommen  $V' \subseteq V$  ist Knotenüberdeckung. Gäbe es eine „Kante“  $\{u, v\} \notin E^c, u, v \in V \setminus V'$ , so wäre  $\{u, v\} \in E$  und  $\{u, v\} \cap V' = \emptyset$ , also  $V'$  keine Überdeckung.

Ist  $V \setminus V'$  Clique, so betrachte hypothetisch eine von  $V'$  unüberdeckte Kante  $\{u, v\}$ , also  $\{u, v\} \cap V' = \emptyset$ .  $\rightsquigarrow \{u, v\} \subseteq V \setminus V'$ , d. h.  $\{u, v\} \in E^c$ . Kanten aus  $E$  sind also durch  $V'$  abgedeckt.  $\square$



Das Lemma zeigt, dass das Knotenüberdeckungsproblem (Frage nach der Existenz einer Knotenüberdeckung mit höchstens  $k$  Knoten) auf das Cliquesproblem (Frage nach der Existenz einer Clique der Größe mindestens  $|V| - k$ ) reduzieren lässt und umgekehrt (im Karpischen Sinne). In obiger Notation haben wir (für beide Reduktionsrichtungen!):

$$f(G) = G^c \text{ und, für } V' \subseteq V, g(G, V') = V \setminus V'$$

Diese Approximationsreduktion erhält aber *nicht* die Approximationsgüte:

Betrachte die Graphenschar  $(G_n)_{n \geq 1}$ , wobei  $G_n$  aus zwei Cliques mit jeweils  $n$  Knoten besteht, wobei der  $i$ -te Knoten der ersten Clique mit allen Knoten der zweiten Clique —mit Ausnahme des  $i$ -ten Knoten der zweiten Clique— verbunden ist. Jede maximale Clique von  $G_n$  enthält  $n$  Knoten.

Der Komplementgraph  $G_n^c$  besteht aus  $n$  disjunkten Paaren miteinander verbundener Knoten. Daher hat die triviale Lösung des MVC-Problems (man nehme alle Knoten als Knotenüberdeckung) eine Leistungsgüte von 2. Geht man zurück zum Ursprungsproblem, dem Cliquesproblem, so wäre die der MVC-Leistung „entsprechende“ Cliqueslösung die leere Menge. Damit ist klar, dass die Näherungsgüte nicht erhalten bleibt bei dieser Reduktion.

## Approximationserhaltene Reduktionen

Betrachte  $\mathcal{P}_1, \mathcal{P}_2 \in \text{NPO}$ ,  $\mathcal{P}_1$  heißt *näherungserhaltend* auf  $\mathcal{P}_2$  *reduzierbar*, kurz  $\mathcal{P}_1$  ist *AP-reduzierbar* (AP bedeutet ausgeschrieben „approximation preserving“) auf  $\mathcal{P}_2$ , in Zeichen  $\mathcal{P}_1 \leq_{\text{AP}} \mathcal{P}_2$ , wenn es zwei Abbildungen  $f, g$  gibt und eine Konstante  $\alpha \geq 1$  derart, dass folgende Bedingungen erfüllt sind:

1.  $\forall x \in I_{\mathcal{P}_1} \forall r \in \mathbb{Q} \cap (1, \infty) : f(x, r) \in I_{\mathcal{P}_2}$ .
2.  $\forall x \in I_{\mathcal{P}_1} \forall r \in \mathbb{Q} \cap (1, \infty) : S_{\mathcal{P}_1}(x) \neq \emptyset \rightarrow S_{\mathcal{P}_2}(f(x, r)) \neq \emptyset$ .
3.  $\forall x \in I_{\mathcal{P}_1} \forall r \in \mathbb{Q} \cap (1, \infty) \forall y \in S_{\mathcal{P}_2}(f(x, r)) : g(x, y, r) \in S_{\mathcal{P}_1}(x)$ .
4.  $f, g$  sind durch Algorithmen  $\mathcal{A}_f, \mathcal{A}_g$  berechenbar, deren Laufzeit polynomiell ist für jedes feste  $r \in \mathbb{Q} \cap (1, \infty)$ .
5.  $\forall x \in I_{\mathcal{P}_1} \forall r \in \mathbb{Q} \cap (1, \infty) \forall y \in S_{\mathcal{P}_2}(f(x, r)) :$

$$R_{\mathcal{P}_2}(f(x, r), y) \leq r \rightarrow R_{\mathcal{P}_1}(x, g(x, y, r)) \leq 1 + \alpha(r - 1)$$

Ein einfaches **Beispiel** für eine AP-Reduktion liefern MAXCLIQUE und MAX-IS durch Übergang auf den Komplementgraphen; die Clique wird so zur unabhängigen Menge.

**Satz:** Betrachte  $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3 \in \text{NPO}$ .

1. Gilt  $\mathcal{P}_1 \leq_{\text{AP}} \mathcal{P}_2$  und  $\mathcal{P}_2 \leq_{\text{AP}} \mathcal{P}_3$ , so auch  $\mathcal{P}_1 \leq_{\text{AP}} \mathcal{P}_3$  (Transitivität)
2. Gilt  $\mathcal{P}_1 \leq_{\text{AP}} \mathcal{P}_2$  und  $\mathcal{P}_2 \in \text{APX}$ , so folgt  $\mathcal{P}_1 \in \text{APX}$ .
3. Gilt  $\mathcal{P}_1 \leq_{\text{AP}} \mathcal{P}_2$  und  $\mathcal{P}_2 \in \text{PTAS}$ , so folgt  $\mathcal{P}_1 \in \text{PTAS}$ .

Beweis:

1. Ist intuitiv klar, wenn auch formal mühsam hinzuschreiben.
2. Sei  $(f, g, \alpha)$  eine AP-Reduktion von  $\mathcal{P}_1$  auf  $\mathcal{P}_2$ . Liegt  $\mathcal{P}_2$  in APX und ist  $\mathcal{A}_{\mathcal{P}_2}$  ein Algorithmus für  $\mathcal{P}_2$  mit Leistungsgüte höchstens  $r$ , so ist

$$\mathcal{A}_{\mathcal{P}_1}(x) := g(x, \mathcal{A}_{\mathcal{P}_2}(f(x, r)), r)$$

ein Polynomzeitalgorithmus der Leistungsgüte höchstens  $1 + \alpha(r - 1)$ .

3. Entsprechend überlegt man für Approximationsschemata, dass

$$\mathcal{A}_{\mathcal{P}_1}(x, r) = g(x, \mathcal{A}_{\mathcal{P}_2}(f(x, r'), r'), r')$$

mit  $r' = 1 + (r - 1)/\alpha$  ein Approximationsschema für  $\mathcal{P}_1$  ist, sobald  $\mathcal{A}_{\mathcal{P}_2}$  eines für  $\mathcal{P}_2$  ist.  $\square$

Wegen dem Satz ist die folgende Definition sinnvoll: Es sei  $C \subseteq \text{NPO}$ .  
Ein Problem  $\mathcal{P} [\in \text{NPO}]$  heißt *C-hart*, wenn für jedes  $\mathcal{P}' \in C$  gilt:

$$\mathcal{P}' \leq_{\text{AP}} \mathcal{P}.$$

Ein C-hartes Problem heißt *C-vollständig*, wenn es in C liegt.

In der Literatur werden verschiedene Reduktionsbegriffe für Approximationsprobleme betrachtet. Entsprechend gibt es auch verschiedene Härte- und Vollständigkeitsbegriffe. Näheres dazu im Buch von Ausiello et al., Kapitel 8. Im Folgenden werden wir noch einige konkrete AP-Vollständigkeitsbegriffe diskutieren. Dadurch wird auch der Umgang mit AP-Reduktionen geübt.

## NPO-Vollständigkeit

Als (nahezu generische) NPO-vollständige Probleme betrachten wir:

(a) MAXWSAT für Maximierungsprobleme aus NPO,

(b) MINWSAT für Minimierungsprobleme aus NPO.

Konkreter: MAXWSAT (Maximum Weighted Satisfiability)

I : Boolesche Formeln  $\varphi$  mit Variablen  $x_1, \dots, x_n$  und nichtnegativen Gewichten  $w_1, \dots, w_n$

S : Belegung I der Variablen, sodass  $\varphi$  erfüllt wird.

m :  $\max \{1, \sum_{i=1}^n w_i \tau(x_i)\}$ ; hierbei werden durch  $\tau$  die Booleschen Werte true und false mit 1 und 0 identifiziert.

opt : max

MINWSAT ist das entsprechende Minimierungsproblem (opt = min).

## Mitteilung:

- a) MAXWSAT ist vollständig für die Klasse der Maximierungsprobleme in NPO.
- b) MINWSAT ist vollständig für die Klasse der Minimierungsprobleme in NPO.

Der Beweis der Mitteilung ist analog zum Beweis des Satzes von Cook-Levin: Der Rechent Teppich einer geeigneten Turingmaschine wird „logisch ausgedrückt“.

Aus der Mitteilung alleine folgt *nicht*, dass MAXWSAT oder MINWSAT NPO-vollständig sind. Dies ergibt sich aber unmittelbar aus dem folgenden Satz.

**Satz:** MAXWSAT und MINWSAT sind aufeinander AP-reduzierbar.

**Satz:** MAXWSAT und MINWSAT sind aufeinander AP-reduzierbar.

Beweis: (Skizze)

Wir beschreiben genauer eine Reduktion von MAXWSAT auf MINWSAT, die hinsichtlich Bedingung 5 *keine* AP-Reduktion ist, da das sich ergebende „ $\alpha$ “ von  $r$  abhängt, also nicht konstant ist. Danach deuten wir an, wie sich die Konstruktion als Spezialfall einer Schar von Reduktionen deuten lässt; mindestens eine Reduktion aus dieser Schar ist auch eine AP-Reduktion. In ähnlicher Weise kann man eine AP-Reduktion von MINWSAT auf MAXWSAT angeben.



### Konstruktion einer „falschen“ AP-Reduktion von MAXWSAT auf MINWSAT:

Aus dem (nur angedeuteten) Beweis der vorigen Mitteilung ergibt sich, dass wir o.E. nur MAXWSAT-Instanzen mit Boolescher Formel betrachten müssen, die das Folgende erfüllen:

1.  $\varphi$  ist definiert über Variablen  $v_1, \dots, v_s$  mit Gewichten  $w(v_i) = 2^{s-i}$ ,  $i = 1, \dots, s$  sowie über einigen anderen Variablen vom Gewicht Null.
2. Jede Belegung, die  $\varphi$  erfüllt, weist wenigstens einer der  $v_i$  den Wert true zu.

Es sei  $x$  eine solchermaßen eingeschränkte Instanz von MAXWSAT mit Boolescher Formel  $\varphi$ .  
Definiere:

$$f(x) := \varphi \wedge \alpha_1 \wedge \dots \wedge \alpha_s \text{ mit } \alpha_i := (z_i \equiv (\overline{v_1} \wedge \dots \wedge \overline{v_{i-1}}) \wedge v_i));$$

$z_i$  sind dabei neue Variablen mit  $w(z_i) = 2^i$ ,  $1 \leq i \leq s$ . Alle anderen Variablen haben Gewicht Null in der  $f(x)$ -Instanz.

Ist  $y$  eine erfüllende Belegung für  $f(x)$ , so sei  $g(x, y)$  die Einschränkung von  $y$  auf die in  $\varphi$  vorkommenden Variablen.

Beachte: Genau eine der  $z_i$ -Variablen ist true in jeder erfüllenden Belegung von  $f(x)$ . Wäre keine der  $z_i$ -Variablen true, dann wären auch alle  $v_i$ -Variablen falsch, was 2. widerspricht. Nach Konstruktion der  $\alpha_i$  sind aber keine zwei  $z_i$ -Variablen wahr.

Also gilt für jede zulässige Lösung  $y$  von  $f(x)$ , dass  $m(f(x), y) = 2^i$  für ein  $1 \leq i \leq s$ .

$$\begin{aligned} m(f(x), y) = 2^i &\Leftrightarrow z_i = 1 \Leftrightarrow v_1 = v_2 = \dots v_{i-1} = 0 \wedge v_i = 1 \\ &\Leftrightarrow 2^{s-i} \leq m(x, g(x, y)) < 2 \cdot 2^{s-i} \\ &\rightsquigarrow \frac{2^s}{m(f(x), y)} \leq m(x, g(x, y)) < 2 \cdot \frac{2^s}{m(f(x), y)} \end{aligned}$$

für jede zulässige Lösung  $y$  von  $f(x)$ . [\*]

Dies gilt natürlich auch für eine optimale Lösung  $y_f^*$  von  $f(x)$ .

Ist  $\tilde{y}$  eine zulässige Lösung für  $x$ , also eine erfüllende Belegung von  $\varphi$ , so gibt es wegen 2) ein kleinstes  $i$ , für das  $v_i$  true ist. Durch  $z_i = \text{true}$  und  $z_j = \text{false}$  für  $j \neq i$  lässt sich diese Belegung zu einer erfüllenden Belegung  $\bar{y}$  von  $f(x)$  erweitern. Einer optimalen Lösung  $\tilde{y}^*$  von  $x$  entspricht so eine zulässige Lösung  $\bar{y}^*$  von  $f(x)$  mit der Eigenschaft  $g(x, \bar{y}^*) = \tilde{y}^*$ .

Für die Leistungsgüte von  $g(x, y)$  ergibt sich:

$$\begin{aligned} R(x, (x, y)) &= \frac{m^*(x)}{m(x, g(x, y))} = \frac{m(x, \tilde{y}^*)}{m(x, g(x, y))} \stackrel{[*]}{<} \frac{2 \cdot \frac{2^s}{m(f(x), \tilde{y}^*)}}{\frac{2^s}{m(f(x), y)}} \\ &\leq \frac{2 \cdot m(f(x), y)}{m^*(f(x))} = 2 \cdot R(f(x), y). \end{aligned}$$

Setzen wir diese Abschätzung in der letzten Bedingung der AP-Reduktions-Definition ein, so sehen wir, dass  $\alpha = (2r - 1)/(r - 1)$  keine Konstante ist. Betrachte nun folgende Schar von Reduktionen:

$$f_k(x) := \varphi \wedge \bigwedge_{\substack{i=1, \dots, s \\ b_1=0,1, \dots, b_k=0,1}} \alpha_{i, b_1, \dots, b_k}$$

mit

$$\alpha_{i, b_1, \dots, b_k} = (z_{i, b_1, \dots, b_k} \equiv (\overline{v_1} \wedge \dots \wedge \overline{v_{i-1}} \wedge v_i \wedge (v_{i+1} \equiv b_1) \wedge \dots \wedge (v_{i+k} \equiv b_k)))$$

(Falls  $i + j > s$ , entfallen die entsprechenden Bedingungen  $v_{i+j} \equiv b_j$ .)

Dafür sind  $z_{i, b_1, \dots, b_k}$   $2^k \cdot s$  viele neue Variablen.

Wie oben sind nur die  $z$ -Variablen solche mit nicht-verschwindendem Gewicht. Wir setzen hierbei

$$w(z_{i,b_1,\dots,b_k}) = \left[ \frac{c \cdot 2^s}{w(v_i) + \sum_{j=1}^k b_j w(v_{i+j})} \right]$$

für eine genügend große Konstante  $c$ .

Nach einiger (hier fortgelassener) Rechnung findet man

$$\frac{c \cdot 2^s}{m(f_k(x), y)} \leq m(x, g(x, y)) < \frac{c \cdot 2^s}{m(f_k(x), y)} \cdot (1 + 2^{-k})$$

Dabei ist  $g(x, y)$  wieder durch „Vergessen“ der  $z$ -Belegung definiert. Wie zuvor erhält man:

$$R(x, g(x, y)) < (1 + 2^{-k})R(f_k(x), y).$$

Unsere zuvor durchgeführte Rechnung entspricht dem Spezialfall  $k = 0$ .

Ist nun  $r > 1$  vorgegeben, so wählen wir  $k = k(r)$  so, dass  $2^{-k(r)} \leq (r - 1)/r$ . Dann folgt aus  $R(f_{k(r)}(x), y) \leq r$  nämlich

$$R(f_{k(r)}(x), y) < (1 + 2^{-k(r)})R(f_{k(r)}(x), y) \leq r + r2^{-k(r)} \leq r + r - 1 = 1 + 2(r - 1).$$

Mit  $f(x, r) := f_{k(r)}(x)$  ist  $(f, g, 2)$  eine AP-Reduktion von MAXWSAT auf MINWSAT. □

## Folgerungen

**Folgerung:** Maximum Weighted 3-SAT ist NPO-vollständig.

Beweis: Die Überführung in KNF ist in Polynomzeit möglich, ansonsten betrachte den klassischen Beweis, s.o. □

Analog sieht man:

**Folgerung:** Minimum Weighted 3-SAT ist NPO-vollständig. □

**Folgerung:** Minimum  $\{0, 1\}$ -LP ist NPO-vollständig.

Beweis: Kombiniere die vorige Folgerung und (den Beweis vom) 1. Lemma. □