Approximierbarkeit von NP Problemen

- Approximations-Algorithmen: in polynomieller Zeit wird ein Resultat gefunden, das garantiert höchstens einen vorgegebenen Abstand zur optimalen Lösung aufweist.
- Obwohl sich alle **NP**-vollständigen Probleme ineinander mit polynomiellem Aufwand reduzieren lassen, verhalten sie sich in Bezug auf ihre Approximierbarkeit höchst unterschiedlich.
- In den letzten Jahren hat eine dramatische Entwicklung durch eine neue Charakterisierung von **NP** eingesetzt: $\rightarrow$ durch das faszinierende **PCP**-Theorem konnte die Approximierbarkeit bzw Nichtapproximierbarkeit von vielen kombinatorischen Optimierungsproblemen wesentlich genauer ermittelt werden.

Definition: NPO: Klasse aller Optimierungsprobleme, deren Entscheidungsvariante in **NP** liegt:

Gegeben: $A = (I, L, w, z) \in$ **NPO** wobei

- I ... Menge der Instanzen von A, erkennbar in polynomieller Zeit.
- $L(x)$ mit $x \in I$ ... Menge der Lösungen von Instanz x. Die einzelnen Lösungen sind dabei polynomiell längenbegrenzt, dh es gibt ein Polynom p, sodaß $\forall x \in I$ und $\forall s \in L(x)$ gilt: $|s| \leq p(|x|)$. Weiters gilt, daß $\forall y \,/\, |y| \leq p(|x|)$, die Frage ob $y \in L(x)$ in polynomieller Zeit entscheidbar ist.
- $w(x, s)$ mit $s \in L(x)$ ... „Wert" von Lösung s von x, ebenfalls in polynomieller Zeit berechenbar.
- $z \in \{\max, \min\}$ ... Ziel, entscheidet, ob es sich um ein Maximierungs- oder Minimierungsproblem handelt.

▷

Gefragt: Optimale Lösung y und deren „Wert“

$$\mathrm{opt}(x) \stackrel{\mathrm{def}}{=} w(x,y) = z\{w(x,s) \,/\, s \in L(x)\}$$

Bemerkung: Falls ein **NPO** Problem in polynomieller Zeit lösbar ist (Klasse **PO**), dann ist es seine Entscheidungsvariante klarerweise ebenfalls.

Definition: Kürzester Pfad (SHORTEST PATH):

Gegeben: $G = (V,E)$ und Knoten $a, e \in V$.

Gefragt: Kürzester Pfad von a nach e.

Theorem: Kürzester Pfad $\in$ **PO** .

Beweis: Breitensuche mit Markierung von bereits besuchten Knoten. $\surd$

Umgekehrt gilt, daß falls **P** $\neq$ **NP** und das Entscheidungsproblem **NP**-vollständig ist, dann kann das entsprechende **NPO** Problem nicht in polynomieller Zeit lösbar sein. Daher sucht man Lösungen mit Approximationsgarantien.

Definition: Sei $A = (I, L, w, z) \in$ **NPO** und $x \in I$, $y \in L(x)$. Dann ist das performance ratio von y in Bezug auf x

$$R(x,y) \stackrel{\text{def}}{=} \max\left\{\underbrace{\frac{w(x,y)}{\text{opt}(x)}}_{z=\min}, \underbrace{\frac{\text{opt}(x)}{w(x,y)}}_{z=\max}\right\} \geq 1\,.$$

⚠ Warnung ⚠ $\rightarrow$ Verwirrendere Alternativen:

ratio bound: $R'(x,y) \stackrel{\text{def}}{=} \frac{1}{R(x,y)} \in (0,1]$

relative error bound: $R''(x,y) \stackrel{\text{def}}{=} 1 - R'(x,y) \in [0,1)$

Definition: M ist ein $r(n)$-approximations Algorithmus für $A = (I, L, w, z) \in$ **NPO** und $r : \mathbb{N} \mapsto [1, \infty)$ wenn

$$\forall x \in I,\ M(x) \in L(x) \text{ und } R(x, M(x)) \leq r(|x|)\,.$$

Definition: Falls M außerdem polynomiell zeitbeschränkt ist, dann nennen wir A $r(n)$-approximierbar.

Definition: Ein **NPO** Problem A ist in der Klasse **APX** wenn es ε-approximierbar für ein konstantes $\varepsilon > 1$ ist.

Beispiel: Minimum Vertex Cover

Gegeben: Graph $G = (V, E)$.

Gefragt: Kleinster Vertex Cover (Wh: Ein Vertex Cover ist eine Teilmenge $V' \subseteq V$, sodaß $\forall \{u, v\} \in E$ gilt: $u \in V'$ oder $v \in V'$, $\rightarrow$ jede Kante aus E wird von mindestens einem Knoten aus V' abgedeckt, dh berührt).

Theorem: Minimum Vertex Cover ist 2-approximierbar, dh

$$\text{Minimum Vertex Cover} \in \textbf{APX}.$$

Beweis: Das entsprechende Entscheidungsproblem Vertex Cover ist **NP**-vollständig. Wir führen einen polynomiellen 2-approximations Algorithmus für Minimum Vertex Cover vor.

▷

Algorithmus VertexCover-2-Approx(V,E)
while $E \neq \{\}$ **do**
  Nimm eine beliebige Kante $\{u,v\} \in E$.
  Füge sowohl u als auch v zum Vertex Cover hinzu.
  Entferne alle von u oder v abgedeckten Kanten aus E.

$\longrightarrow$ Das Resultat ist sicher ein Vertex Cover.

Frage: Ist es sicher höchstens doppelt so groß wie das Optimum?

Antwort: Ja: Betrachte nur die Kanten, die von dem Algorithmus ausgewählt wurden. Keine zwei davon können einen gemeinsamen Knoten haben. Daher muß bereits ein Vertex Cover von nur diesen Kanten jeweils mindestens einen der beiden Knoten enthalten, dh mindestens halb so groß sein wie der gefundene Vertex Cover. $\surd$

Beispiel: Billigste Rundreise für Handelsreisenden (TSP).

Theorem: Wäre TSP für ein bestimmtes $\varepsilon > 1$ ε-approximierbar, dann würde dies **P** $=$ **NP** bedeuten und wir könnten TSP auch exakt in **P** lösen:

$$\text{Daher:}\quad \mathbf{P} \neq \mathbf{NP} \;\Leftrightarrow\; \text{TSP} \notin \mathbf{APX}.$$

Beweis: Wir reduzieren mit polynomiellem Aufwand ein bekanntes **NP**-vollständiges Problem (beliebige Hamiltonian Cycle Instanz $G = (V, E)$) auf eine spezielle TSP Instanz mit $|V|$ Städten, für die wir die Existenz eines polynomiellen ε-approximations Algorithmus annehmen, sodaß sich daraus ein polynomieller Algorithmus für das **NP**-vollständige Problem ergäbe. Wir schließen daraus, daß, falls **P** $\neq$ **NP**, es kein solches $\varepsilon > 1$ geben kann. Die Reduktion sieht wie folgt aus:

$$M_{i,j} = \begin{cases} 1 & \text{falls } \{i,j\} \in E \\ \varepsilon|V| & \text{falls } \{i,j\} \notin E \end{cases}$$

▷

Wir lassen unseren hypothetischen polynomiellen ε-approximations Algorithmus auf die so konstruierte TSP Instanz los. Es kann nur zwei Fälle geben (gap-Technik):

1. $w(x,y) = |V|$: Da $\varepsilon > 1$ konnten nur Kanten mit $M_{i,j} = 1$, dh Kanten aus G durchlaufen werden, und die Rundreise ist daher auch ein Hamiltonian Cycle.

2. Falls umgekehrt auch nur eine Kante in der Rundreise durchlaufen wurde, die nicht in G ist, dann muß $w(x,y) = \varepsilon|V| + \gamma$ mit $\gamma > 0$ sein. Da es sich um einen ε-approximations Algorithmus handelt, gilt:
$$R(x,y) = \frac{\varepsilon|V|+\gamma}{\mathrm{opt}(x)} \leq \varepsilon \quad \Rightarrow \quad |V| < \mathrm{opt}(x)$$
und daher kann G keinen Hamiltonian Cycle haben.

Damit haben wir aber in polynomieller Zeit das **NP-vollständige** Problem entschieden $\longrightarrow$ Widerspruch. $\surd$

Definition: Sei $A = (I, L, w, z) \in$ **NPO**. Ein Algorithmus M ist ein approximation scheme für A falls $\forall x \in I$ und $\forall\, \varepsilon > 1$,

$$R(x, M(x, \varepsilon)) \leq \varepsilon\,.$$

Definition: Ein **NPO** Problem ist in der Klasse **PTAS** wenn es ein polynomial-time approximation scheme hat.

⚠ Warnung ⚠ Die Konstanten im Polynom könnten exponentiell wachsen wenn ε sich an 1 annähert (zB Polynome der Form $2^{1/(\varepsilon-1)} p(|x|)$ oder $|x|^{1/(\varepsilon-1)}$). Um das Wachstum auf polynomielles einzuschränken definieren wir:

Definition: Ein **NPO** Problem ist in der Klasse **FPTAS**, wenn es ein fully polynomial-time approximation scheme dafür gibt, dessen Ausführungszeit also durch $q(|x|, 1/(\varepsilon - 1))$ begrenzt ist, wobei q ein Polynom ist.

Beispiel: **Theorem:** Rucksack $\in$ **FPTAS**.

Beweis: Sei x eine Rucksack-Instanz wie folgt gegeben: n Objekte, jedes mit Wert $v_i \in \mathbb{N}$ und Gewicht $w_i \in \mathbb{N}$, maximale Kapazität $W \in \mathbb{N}$, gesucht ist $S \subseteq \{1, 2, \ldots, n\}$ mit $\sum_{i \in S} w_i \leq W$ mit $\sum_{i \in S} v_i$ maximal.

Der früher besprochene pseudopolynomielle Algorithmus für Rucksack verhielt sich proportional zu W. Dual geht das auch mit den Werten statt mit den Gewichten. Sei $V \stackrel{\text{def}}{=} \max\{v_1, \ldots, v_n\}$. Wir definieren für $i \in \{0, 1, \ldots, n\}$ und $v \in \{0, \ldots, nV\}$ den Wert $W(i, v)$ als das kleinste Gewicht, wenn man genau Wert v aus den ersten i Objekten auswählt: $\forall i \;\; W(i, 0) \stackrel{\text{def}}{=} 0, \;\; \forall v \neq 0 \;\; W(0, v) \stackrel{\text{def}}{=} \infty$:

$$W(i+1, v) \stackrel{\text{def}}{=} \min\{\underbrace{W(i, v)}_{i+1 \notin}, \underbrace{W(i, v - v_{i+1}) + w_{i+1}}_{i+1 \in}\}$$

$\triangleright$

Am Ende wird der größte Wert $v \,/\, W(n,v) \leq W$ ausgewählt. Offensichtlich löst dieser dynamic programming Algorithmus das Problem in $O(n^2 V)$. Da wieder ein exponentieller Sprung bezogen auf Länge der Eingabe ($\rightarrow \log(V) + \ldots$) vorliegt, ist es natürlich wieder nur ein pseudopolynomieller Algorithmus.

Allerdings können wir uns nun höhere Geschwindigkeit durch geringere Genauigkeit erkaufen: Wir konstruieren uns eine neue Instanz x', in der $v'_i \stackrel{\text{def}}{=} 2^b \left\lfloor \frac{v_i}{2^b} \right\rfloor$, dh die jeweils b least significant bits in den Werten werden auf Null gesetzt. Eigentlich können wir also auch ohne diese Nullen rechnen und durch Multiplikation des Endresultats mit 2^b erhalten wir eine Approximation für die ursprüngliche Instanz. Dadurch verringert sich der Zeitbedarf auf $O(n^2 \frac{V}{2^b})$. ▷

Ist es damit aber auch in FPTAS? Ja:

$$\sum_{i\in S'} v_i \underset{v_i \geq v_i'}{\overset{\updownarrow}{\geq}} \sum_{i\in S'} v_i' \underset{\mathrm{opt}(x')}{\overset{\updownarrow}{\geq}} \sum_{i\in S} v_i' \underset{v_i' \geq v_i - 2^b}{\overset{\updownarrow}{\geq}} \sum_{i\in S} (v_i - 2^b) \underset{|S|\leq n}{\overset{\updownarrow}{\geq}} \sum_{i\in S} v_i - n2^b$$

Wir nehmen o.V.d.A. $\forall i : w_i \leq W$ an. Dann gilt $\sum_{i\in S} v_i = \mathrm{opt}(x) \geq V > 0$. Daraus ergibt sich

$$\frac{n2^b}{V} \geq \frac{\sum_{i\in S} v_i - \sum_{i\in S'} v_i}{V} \geq \frac{\mathrm{opt}(x) - w(x,S')}{\mathrm{opt}(x)}$$

$$\Rightarrow \quad R(x,S') \leq \frac{V}{V - n2^b} = \varepsilon$$

Durch Wahl von $b \stackrel{\text{def}}{=} \left\lceil \log\left(\frac{V(\varepsilon-1)}{n\varepsilon}\right) \right\rceil$ und Einsetzen erhalten wir die passende Zeitschranke $O\left(\frac{n^3}{(\varepsilon-1)}\right)$ mit $q(a,b) \stackrel{\text{def}}{=} ca^3 b$ als dem Polynom in a und b für beliebig kleines $\varepsilon > 1$. $\surd$

Probabilistically Checkable "holographic" Proofs

Theorem: (Arora et al. 1992) $\boxed{\textsf{NP} = \textsf{PCP}(\log n, 1)}$

- Exponentieller Beweiser („Prover, Autor").
- Polynomieller Überprüfer („Checker, Editor").
- $O(1)$, dh konstant viele 1-Bit checks (Aufwand $\searrow$).
- $O(\log n)$ zufällige Bits $\rightarrow$ Probabilistisch:
  Beweis korrekt $\Rightarrow$ 100% akzeptiert („Vollständigkeit").
  Beweis inkorrekt $\Rightarrow$ mit Wahrscheinlichkeit $> 1 - \delta$ mit konstantem $\delta > 0$ verworfen („Korrektheit" $\searrow$).
- Wiederholung erlaubt es, δ beliebig klein zu machen und zB $\delta = \frac{1}{2}$ sicherzustellen, dabei genügen etwa 36 Bit checks. Nach 500 Durchläufe damit $\rightarrow$ falscher Beweis wird mit Wahrscheinlichkeit $1 - 2^{-500}$ verworfen.

Bemerkung: NP $=$ PCP$(0, \text{Poly})$ (Definition von **NP**).

Der Beweis des **PCP**-Theorems ist lang (> 50 Seiten) $\rightarrow$ Bücher, Spezialvorlesungen. Wir zeigen hier nur den kürzeren Teil, nämlich daß **PCP**$(\log n, 1) \subseteq$ **NP** .

Lemma: $\forall L \in$ **PCP**$(\log n, 1)\, \exists c, d, k \in \{1, 2, \ldots\}\, \forall x \in \Sigma^n$: es gibt n^c boolsche Funktionen $f_1, f_2, \ldots$ in insgesamt n^d boolschen Variablen, so daß:

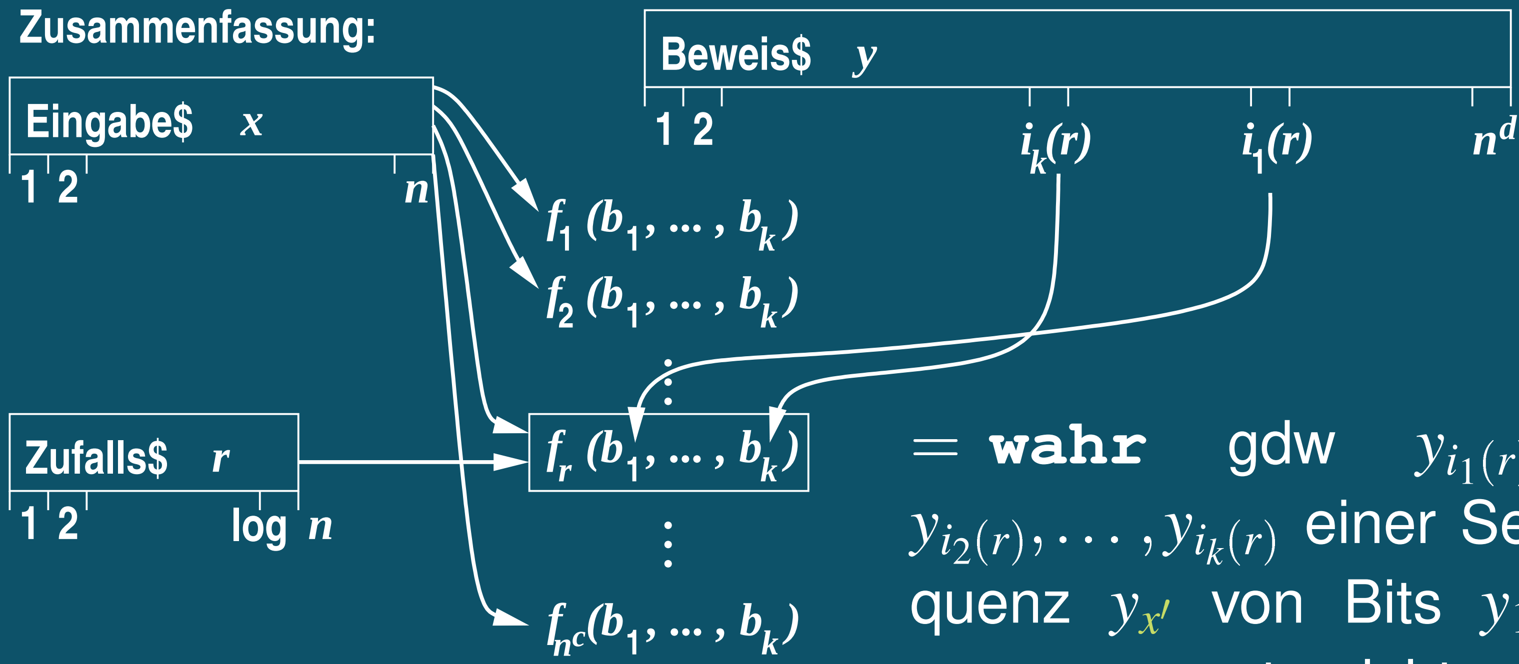
1. Jedes f_i ist eine Funktion von k Variablen, und seine Wahrheitswertetabelle kann in polynomieller Zeit aus x und i berechnet werden.
2. $x \in L \;\Rightarrow$ Es gibt eine Belegung, die alle f_i's **`wahr`** macht.
3. $x \notin L \Rightarrow$ Keine Belegung macht mehr als die Hälfte der n^c Funktionen f_i **`wahr`**.

$\triangleright$

Beweis: Der Überprüfer verwende $c\log n$ Zufallsbits. Es gibt also $2^{c\log n} = n^c$ mögliche Strings $r \in \{0,1\}^{c\log n}$, jeder davon fixiert die Sequenz von $k = O(1)$ Bit-Adressen, die vom Überprüfer gecheckt werden. O.V.d.A. können wir annehmen, daß jeder Beweis höchstens in $O(n^c) = n^d$ boolschen Variablen $y_1, y_2, \ldots, y_{n^d}$ kodiert ist. Sei die Sequenz von gecheckten Bit-Adressen $i_1(r), i_2(r), \ldots, i_k(r)$. Die Entscheidung des Überprüfers hängt also nur von der Belegung von $y_{i_1(r)}, y_{i_2(r)}, \ldots, y_{i_k(r)}$ ab. Sei $f_r(b_1, \ldots, b_k) = \mathtt{\textbf{wahr}} \Leftrightarrow$ „Der Überprüfer akzeptiert, wenn $y_{i_1(r)}, \ldots, y_{i_k(r)}$ die Werte $b_1, \ldots, b_k$ haben.“ Die Wahrheitswertetabelle von f_r kann durch $2^k = O(1)$-maligen Aufruf des polynomiellen Überprüfers, also insgesamt in polynomieller Zeit berechnet werden (1.). Per Definition von **PCP**$(\log n, 1)$ und f_r, wenn $x \in L$, dann gibt es eine y-Belegung, die alle f_r **`wahr`** macht (2.), und wenn $x \notin L$, dann macht keine mehr als $n^c/2$ **`wahr`** (3.). $\surd$

▷

Zusammenfassung:



$= \texttt{wahr}$ gdw $y_{i_1(r)}$, $y_{i_2(r)}, \dots, y_{i_k(r)}$ einer Sequenz $y_{x'}$ von Bits y_1, $y_2, \dots, y_{n^d}$ entspricht, so daß $y_{x'}$ beweist, daß $x' \in L$, wobei die anderen $n^d - k$ Bits nicht mit $y_{x'}$ übereinstimmen müssen, was die seltene fälschliche Akzeptanz ermöglicht.

$$\underbrace{\left.\begin{array}{l} f_r(0,0,\dots,0,0) \\ f_r(0,0,\dots,0,1) \\ f_r(0,0,\dots,1,0) \\ \vdots \\ f_r(1,1,\dots,1,0) \\ f_r(1,1,\dots,1,1) \end{array}\right\} 2^k = O(1)}_{\text{in polynomieller Zeit berechenbar}}$$

Theorem: **PCP**$(\log n, 1) \subseteq$ **NP** .

Beweis: $\forall L \in$ **PCP**$(\log n, 1)\, \forall x \in \Sigma^n$: Konstruiere die f_i's des Lemmas (polynomiell möglich, da poly$(n) \times n^c$ ebenfalls ein Polynom ist). Rate nichtdeterministisch den Beweis y ($x \in L \Leftrightarrow$ Es gibt eine Belegung von $y = y_1, \ldots, y_{n^d}$ sodaß alle f_i's **`wahr`** sind). Wieder kann polynomiell überprüft werden, daß der geratene Beweis tatsächlich alle f_i's **`wahr`** macht, indem man die entsprechenden Bits des Beweises in die n^c vielen Funktionen f_i's einsetzt und (wieder in polynomieller Zeit) den jeweiligen Wert berechnet und auf **`wahr`** überprüft.

Es kann also jede Sprache, die durch die **PCP**$(\log n, 1)$-Maschinerie akzeptiert wird, auch durch die **NP**-Maschinerie akzeptiert werden. $\surd$

Anwendungsbeispiel: Wir zeigen mit dem **PCP**-Theorem:

Theorem: **P** $\neq$ **NP** $\Leftrightarrow$ MAX3CNF $\notin$ **PTAS** .

Beweis: Seien $L \in \mathbf{NP} = \mathbf{PCP}(\log n, 1)$, $x \in \Sigma^n$ beliebig. Dann kann jedes f_i vom Lemma (Seite 14) als k-CNF Formel $f_i = \bigwedge_{j=1}^{2^k} C_{i,j}$ in $y_1, \ldots, y_{n^d}$ ausgedrückt werden, zB:

$$
\begin{array}{c|c|c|c|l}
f_i(a,b,c) & a & b & c & f_i(a,b,c) \equiv \\
\hline
0 & 0 & 0 & 0 & (a \vee b \vee c) \wedge \\
0 & 0 & 0 & 1 & (a \vee b \vee \neg c) \wedge \\
0 & 0 & 1 & 0 & (a \vee \neg b \vee c) \wedge \\
1 & 0 & 1 & 1 & \quad — \\
0 & 1 & 0 & 0 & (\neg a \vee b \vee c) \wedge \\
1 & 1 & 0 & 1 & \quad — \\
0 & 1 & 1 & 0 & (\neg a \vee \neg b \vee c) \wedge \\
0 & 1 & 1 & 1 & (\neg a \vee \neg b \vee \neg c)
\end{array}
\left.\vphantom{\begin{array}{c}1\\1\\1\\1\\1\\1\\1\\1\end{array}}\right\} 2^k
$$

$\underbrace{a \quad b \quad c}_{k \text{ Variable aus } y_1, \ldots, y_{n^d}}$

▷

Es gilt: $x \in L \Leftrightarrow \bigwedge_{i=1}^{n^c} f_i \equiv \bigwedge_{i=1}^{n^c} \bigwedge_{j=1}^{2^k} C_{i,j} \stackrel{\text{def}}{=} F_0$ ist erfüllbar, und wenn $x \notin L$, dann kann kein Beweis y mehr als die Hälfte der f_i's erfüllen **→ PCP-Eigenschaft ←**. Für jedes unerfüllte f_i gibt es mindestens eine unerfüllte Klausel, der Anteil unerfüllter Klauseln in F_0 ist also $\geq \frac{1}{2^{k+1}}$. Wir ersetzen jede k-Klausel $l_1 \vee l_2 \vee \ldots \vee l_k$ durch folgende erfüllungsäquivalente 3CNF Formel

$$(l_1 \vee l_2 \vee z_1) \wedge \bigwedge_{t=1}^{k-4} (\neg z_t \vee l_{t+2} \vee z_{t+1}) \wedge (l_{k-1} \vee l_k \vee \neg z_{k-3})$$

wobei $z_1, \ldots, z_{k-3}$ neue Variable sind ($\neq$ pro k-Klausel).

$$\left.\begin{array}{l}\text{Wenn } x \notin L\text{, dann gibt es für jede unerfüllte } k\text{-Klausel}\\ \text{mindestens eine unerfüllte Klausel in der resultierenden}\\ \text{Gesamt-3CNF Formel } F\ (m \stackrel{\text{def}}{=} n^c 2^k (k-2))\text{, und der An-}\\ \text{teil unerfüllbarer Klauseln in } F \text{ ist} \geq \frac{1}{2^{k+1}(k-2)} \stackrel{\text{def}}{=} 1-\delta.\end{array}\right\} (*)$$

$\triangleright$

Nehmen wir nun an, es gäbe einen $\varepsilon \stackrel{\text{def}}{=} \frac{(k-2)2^{k+1}}{(k-2)2^{k+1}-1/2}$-approximations Algorithmus für MAX3CNF. Ähnlich wie im Beweis der Nichtapproximierbarkeit von TSP könnten wir nun den Gap in der Akzeptanz-Wahrscheinlichkeit nützen, um das beliebige **NP** Problem $x \in L$ polynomiell zu entscheiden. Es kann wieder nur zwei Fälle geben:

1. Die Anzahl der durch den ε-approximations Algorithmus erfüllten Klauseln von F, $w(F,y) \geq \frac{m}{\varepsilon}$: Da $\frac{m}{\varepsilon} > m\delta$ muß dann aber wegen $\left(* : x \notin L \Rightarrow m - w(F,y) \geq m(1-\delta)\right)$ gelten, daß F erfüllbar ist und daher sicher $x \in L$ gilt.
2. $w(F,y) < \frac{m}{\varepsilon}$: Da wegen dem ε-approximations Algorithmus immer $\frac{\mathrm{opt}(F)}{\varepsilon} \leq w(F,y)$ gilt, muß $\mathrm{opt}(F) < m$ gelten, dh F ist nicht erfüllbar, dh es muß sicher $x \notin L$ gelten.

Damit haben wir aber in polynomieller Zeit eine beliebige Sprache aus **NP** entschieden $\rightarrow$ Widerspruch. $\surd$

- Vergleich mit Cooks Beweis $\rightarrow$ „robusteres" Format von y: wenn es keinen Beweis gibt, konnte bei Cook y bis auf ein Bit korrekt aussehen, während bei **PCP** mindestens die Hälfte der Bits falsch sind.

- **Bemerkung:** MAX3CNF ist 2-approximierbar, genauer: $1.\overline{142857}$-approximierbar aber nicht besser. Es gibt approximationserhaltende Reduktionen und MAX3CNF ist damit **APX**-vollständig (andere: MAX-2SAT, 4-degree Independent Set, Vertex Cover).

- **Bemerkung:** **PO** $\subseteq$ **FPTAS** $\subseteq$ **PTAS** $\subseteq$ **APX** $\subseteq$ **NPO**
  wobei alle Inklusionen echt sind gdw **P** $\neq$ **NP**.

- Vertiefende Vorlesungen „Approximative Algorithmen" und „Kombinatorische Algorithmen".

- Online Übersicht (incl Literatur):

  http://www.nada.kth.se/˜viggo/wwwcompendium/