

# Approximationskomplexität

# Approximationsalgorithmen

- Ein Optimierungsproblem **P** besteht aus einer Menge **I** von Instanzen, einer Zielfunktion **f**, die nur nicht-negative reelle Zahlen annimmt, und für jede Instanz  $x \in I$  aus einer Menge **L(x)** von Lösungen.

- ▶ Für ein Minimierungsproblem ist

$$\mathbf{opt}_P(x) = \min\{f(x, y) \mid y \in L(x)\}$$

der optimale Wert für Instanz  $x$ .

Für ein Maximierungsproblem definiere  $\mathbf{opt}_P(x)$  entsprechend.

- Ein Approximationsalgorithmus **A** für **P** bestimmt für jede Instanz  $x$  eine Lösung  $A(x) \in L(x)$ .

- ▶ Der Approximationsfaktor für Instanz  $x$  ist

$$r_A(x) = \min\left\{\frac{f(x, A(x))}{\mathbf{opt}(x)}, \frac{\mathbf{opt}(x)}{f(x, A(x))}\right\}.$$

- ▶ Der Approximationsfaktor für Instanzen der Länge  $n$  ist

$$r_A(n) = \min_{x, |x|=n} r_A(x).$$

# Der Approximationsfaktor $r_A(x) = \min\left\{\frac{f(x, A(x))}{\text{opt}(x)}, \frac{\text{opt}(x)}{f(x, A(x))}\right\}$

- Es ist stets  $r_A(n) \leq 1$ .
- $A$  ist genau dann optimal, wenn  $r_A(n) = 1$ .

**A sei ein Approximationsalgorithmus** mit zusätzlicher Eingabe  $\varepsilon > 0$ . Setze

$$r_A(\varepsilon, x) := \min\left\{\frac{f(x, A(\varepsilon, x))}{\text{opt}(x)}, \frac{\text{opt}(x)}{f(x, A(\varepsilon, x))}\right\}.$$

- **A** ist ein **volles polynomielles Approximationsschema**, wenn
  - ▶ die Laufzeit von  $A(\varepsilon, x)$  polynomiell in  $|x| + \frac{1}{\varepsilon}$  ist und
  - ▶  $r_A(\varepsilon, x) \geq 1 - \varepsilon$  gilt.
- **A** ist ein **polynomielles Approximationsschema**, wenn
  - ▶ die Laufzeit von  $A(\varepsilon, x)$  für jedes  $\varepsilon$  polynomiell in  $|x|$  ist und
  - ▶  $r_A(\varepsilon, x) \geq 1 - \varepsilon$  gilt.

**APX** besteht aus allen Optimierungsproblemen mit effizienten Approximationsalgorithmen  $A$ , so dass  $r_A(x) \geq \delta$  für eine Konstante  $\delta > 0$  gilt.

- Suchprobleme in NP (**gibt es eine Lösung?**) sind mehr oder minder gleich schwer.
  - ▶ Fast alle wichtigen Sprachen in NP sind effizient lösbar oder NP-vollständig.
  - ▶ Wichtige Ausnahmen sind die **Faktorisierung** und die **Graph-Isomorphie**.
- Ihre Optimierungsversionen (**bestimme eine möglichst gute Lösung**) zeigen hingegen große Individualität.

# Das Rucksack Problem

**Eingabe:** Ein Rucksack mit Kapazität  $K$  sowie  $n$  Objekte, wobei Objekt  $i$  das Gewicht  $g_i$  und den Wert  $w_i \in \mathbb{N}$  hat.

**Aufgabe:** Bepacke den Rucksack mit einer Auswahl aus den  $n$  Objekten, so dass

- das Gewicht der eingepackten Objekte die Kapazität  $K$  nicht übersteigt und
- der „eingepackte Wert“ größtmöglich ist.

- Bestimme „Gewicht $_i(w)$  = das minimale Gewicht einer Auswahl aus den ersten  $i$  Objekten mit Wertesumme  $w$ “ mit dynamischer Programmierung.
  - ▶ Die Rekursionsgleichung:  
$$\text{Gewicht}_{i+1}(w) = \min\{ \text{Gewicht}_i(w), \text{Gewicht}_i(w - w_{i+1}) + g_{i+1} \}.$$
  - ▶ Für  $W = \sum_{i=1}^n w_i$  gibt es  $n \cdot W$  Teilprobleme: Die Laufzeit beträgt  $O(n \cdot W)$ .
- Spare Laufzeit durch das Skalieren der Werte ( $w_i \mapsto \lfloor \frac{nw_i}{\varepsilon \max_i w_i} \rfloor$ )

Rucksack hat ein volles polynomielles Approximationsschema mit Laufzeit  $\frac{n^3}{\varepsilon}$ .

*Eingabe:*  $n$  Objekte mit den Gewichten  $0 \leq g_1, \dots, g_n \leq 1$ .

*Aufgabe:* Verteile die Objekte auf eine möglichst geringe Anzahl von Behältern, so dass jeder Behälter nur Objekte mit Gesamtgewicht höchstens 1 erhält.

BIN PACKING besitzt **kein** volles polynomielles Approximationsschema, wohl aber ein polynomielles Approximationsschema:

*Für jedes  $\varepsilon > 0$  kann eine Verteilung mit höchstens  $(1 + \varepsilon)$ opt Behältern in Zeit  $\text{poly}(n)$  gefunden werden.*

# VERTEX COVER

*Eingabe:* Ein ungerichteter Graph  $G = (V, E)$ .

*Aufgabe:* Bestimme eine Knotenmenge  $W \subseteq V$  kleinster Größe, so dass jede Kante in  $E$  mindestens einen Endpunkt in der Menge  $W$  besitzt. (Wir sagen, dass  $W$  die Kantenmenge  $E$  überdeckt.)

- VERTEX COVER besitzt einen Algorithmus  $A$  mit  $r_A = \frac{1}{2}$ .
  - 1 Beginne mit der leeren Menge  $W$ .
  - 2 Füge beide Endpunkte  $a, b$  einer beliebigen Kante  $\{a, b\}$  zu  $W$  hinzu.

Eine optimale Kantenüberdeckung muss mindestens einen Endpunkt wählen.

Entferne alle Kanten mit Endpunkt  $a$  oder  $b$ .
  - 3 Wiederhole dieses Vorgehen solange bis alle Kanten entfernt wurden.
- Also gehört VERTEX COVER zu APX. Wir zeigen später:  
Wenn  $P \neq NP$ , dann hat VERTEX COVER kein polynomielles Approximationsschema.

*Eingabe:* Eine Menge  $K$  von Klauseln mit Gewichten  $w_k \geq 0$  für jede Klausel  $k \in K$ .

*Aufgabe:* Bestimme eine Belegung, die Klauseln mit maximalem Gesamtgewicht erfüllt

- MAX-SAT gehört wie VERTEX COVER zu APX:  
Eine zufällig ausgewürfelte Belegung wird im Mittel mindestens die Hälfte des gesamten Klauselgewichts erfüllen.
- Wir zeigen später, dass polynomielle Approximationsschemata nicht existieren, wenn  $P \neq NP$ .



*Eingabe:* Ein Universum  $U = \{1, \dots, m\}$  und Teilmengen  $T_1, \dots, T_n \subseteq U$  mit  $\bigcup_i T_i = U$ .

*Aufgabe:* Wähle möglichst wenige Teilmengen aus, die weiterhin  $U$  überdecken.

- Man kann für jedes  $\varepsilon > 0$  zeigen, dass Überdeckungen mit höchstens  $(1 - \varepsilon) \cdot \ln(m)$  **opt** Teilmengen **nicht** effizient bestimmt werden können.
- Der gierige Überdeckungsalgorithmus wählt stets eine Teilmenge, die die meisten noch nicht überdeckten Elemente überdeckt.  
Man kann zeigen, dass höchstens  $(1 + \ln(m)) \cdot \mathbf{opt}$  Mengen gewählt werden.

*Eingabe:* Ein ungerichteter Graph  $G = (V, E)$ .

*Aufgabe:* Bestimme eine Knotenmenge  $W \subseteq V$  maximaler Größe, so dass je zwei Knoten in  $W$  durch eine Kante miteinander verbunden sind.

Es kann gezeigt werden, dass CLIQUE für keine Konstante  $\varepsilon > 0$  effiziente Algorithmen besitzt, die eine Clique der Größe mindestens  $\frac{\text{opt}}{n^{1-\varepsilon}}$  berechnen.

# Die Fragestellungen

- Wie weit kann der Approximationsfaktor effizienter Algorithmen für ein Optimierungsproblem **P** verbessert werden?
- Wann besitzt das Optimierungsproblem **P** keine effizienten Approximationsalgorithmen mit scharfen Approximationen?
- Zum Beispiel:
  - ▶ Besitzt **VERTEX COVER** ein polynomielles Approximationsschema?
  - ▶ Was ist die Approximationskomplexität von **INDEPENDENT SET** „Bestimme eine möglichst große Knotenmenge, so dass keine zwei Knoten durch eine Kante verbunden sind, bzw von **CLIQUE**?
  - ▶ Bestimme die Approximationskomplexität von **MAX-SAT**.

*Wir benötigen eine völlig neue Sichtweise der Klasse NP, um diese Fragen beantworten zu können.*

$[f, g]$ -gap-**P**,  
eine Entscheidungsversion von **P**

# Das Promise Problem $[f, g]$ -gap-P

Sei  $\mathbf{P}$  ein Maximierungsproblem und  $\mathbf{opt}_{\mathbf{P}}(x)$  der optimale Wert für Instanz  $x$ .

- Für Funktionen  $f, g : \Sigma^* \rightarrow \mathbb{R}$  mit  $\mathbf{f}(x) < \mathbf{g}(x)$  für alle  $x \in \Sigma^*$  setze

$$\mathbf{YES}_{\mathbf{P}} = \{x \mid \mathbf{opt}_{\mathbf{P}}(x) \geq \mathbf{g}(x)\}$$

$$\mathbf{NO}_{\mathbf{P}} = \{x \mid \mathbf{opt}_{\mathbf{P}}(x) < \mathbf{f}(x)\}.$$

- Für ein Minimierungsproblem kehre die Antworten um.
- Algorithmus  $A$  löst das Promise Problem  $[f, g]$ -gap-P, wenn  $A$ 
  - ▶ für jedes  $x \in \mathbf{YES}_{\mathbf{P}}$  mit “ja” und
  - ▶ für jedes  $x \in \mathbf{NO}_{\mathbf{P}}$  mit “nein” antwortet.

Wir verlangen eine fehlerfreie Charakterisierung der Instanz  $x$ , solange  $x$  nicht „in die Lücke“  $\mathbf{f}(x) \leq \mathbf{opt}_{\mathbf{P}}(x) < \mathbf{g}(x)$  fällt. In anderen Worten, wir **versprechen**  $\mathbf{opt}_{\mathbf{P}}(x) < \mathbf{f}(x)$ , bzw.  $\mathbf{opt}_{\mathbf{P}}(x) \geq \mathbf{g}(x)$ .

# Wann ist $[f, g]$ -gap- $\mathbf{P}$ schwierig?

Sei  $\mathbf{P}$  ein Maximierungsproblem.

Wir sagen:

$[f, g]$ -gap- $\mathbf{P}$  ist ein NP-hartes Problem, wenn es für jede Sprache  $L \in \text{NP}$  eine effiziente Transformation  $T$  gibt, so dass

- für  $w \in L$  gilt  $T(w) \in \text{YES}_{\mathbf{P}}$ , also

$$\text{opt}_{\mathbf{P}}(T(w)) \geq g(T(w)),$$

- für  $w \notin L$  gilt  $T(w) \in \text{NO}_{\mathbf{P}}$ , also

$$\text{opt}_{\mathbf{P}}(T(w)) < f(T(w)).$$

Ist  $\mathbf{P}$  ein Minimierungsproblem, dann reduziere Eingaben  $w \in L$  auf Instanzen aus  $\text{NO}_{\mathbf{P}}$  und Eingaben  $w \notin L$  auf Instanzen aus  $\text{YES}_{\mathbf{P}}$ .

# Und wenn $[f, g]$ -gap-P ein NP hartes Problem ist?

Sei  $P$  ein Maximierungsproblem.  $f$  sei effizient berechenbar.

- Angenommen,  $A$  ist ein Approximationsalgorithmus mit

$$A(x) \geq \frac{f(x)}{g(x)} \cdot \text{opt}_P.$$

- Wir lösen das NP-harte Problem  $[f, g]$ -gap-P:

- ▶ Wenn  $A(x) \geq f(x)$ , dann antworte mit „ja“ (denn  $\text{opt}_P \geq f(x)$ ).
- ▶ Wenn  $A(x) < f(x)$ , dann antworte mit „nein“  
(denn der Approximationsfaktor höchstens  $\frac{g(x)}{f(x)}$  garantiert, dass  $\text{opt}_P < g(x)$ ).

$f$  sei effizient berechenbar.

Wenn  $[f, g]$ -gap-P ein NP-hartes Problem ist, dann besitzt  $P$  keine effizienten  $\frac{g(x)}{f(x)}$ -approximativen Algorithmen, solange  $P \neq NP$  gilt.

# Lückenerhaltende Reduktionen

$P_1$  und  $P_2$  seien zwei Optimierungsprobleme.

$[f_1, g_1]$ -gap- $P_1$  ist **lückenerhaltend** auf  $[f_2, g_2]$ -gap- $P_2$  reduzierbar, wenn es eine effiziente Transformation  $T$  gibt, so dass für alle Instanzen  $x$  von  $P_1$  gilt:

- Wenn  $\text{opt}_{P_1}(x) \geq g_1(x)$ , dann  $\text{opt}_{P_2}(T(x)) \geq g_2(T(x))$ .
- Wenn  $\text{opt}_{P_1}(x) < f_1(x)$ , dann  $\text{opt}_{P_2}(T(x)) < f_2(T(x))$ .

- Wenn  $[f_1, g_1]$ -gap- $P_1$  lückenerhaltend auf  $[f_2, g_2]$ -gap- $P_2$  reduzierbar ist und wenn  $[f_1, g_1]$ -gap- $P_1$  ein NP-hartes Problem ist, dann ist auch  $[f_2, g_2]$ -gap- $P_2$  ein NP-hartes Problem.
- Mit einem ersten NP-harten Gap-Problem können wir eine Lawine weiterer Nicht-Approximierbarkeitsergebnisse losstreuen.



# Ist der Nachweis der NP-Härte schwierig?

**[f, g]-gap-P** ist ein **Promise Problem**: Disqualifiziere eine Instanz  $x$  mit

$$f(x) \leq \mathbf{opt_P}(x) < g(x).$$

Die Herausnahme der Lücke kann das Entscheidungsproblem zu stark vereinfachen!

- Um die Bestimmung der Approximationskomplexität auf ein Entscheidungsproblem zurückzuführen, arbeiten wir mit **[f, g]-gap-P**.
- Der Nachweis der NP-Härte wird aber neue Methoden verlangen.

# Probabilistically Checkable Proofs

# NP: Die alte Sichtweise

Wenn eine Sprache  $L$  zu NP gehört, dann gibt es einen **Prover** und einen **Verifier** mit den folgenden Eigenschaften:

- Der Prover (entspricht einem nichtdeterministischen Algorithmus  $P$  für  $L$ ) präsentiert für Eingabe  $w$  einen **möglicherweise falschen** Beweis  $b$  für Eingabe  $w$  ( $P$  hat möglicherweise falsch geraten).
- Der Verifier muss in polynomieller Zeit deterministisch überprüfen, ob  $b$  ein korrekter Beweis ist.
- Die Aufgaben des Verifiers.
  - ▶ **Vollständigkeit**: Wenn Eingabe  $w$  zu  $L$  gehört, dann muss der Verifier einen Beweis polynomieller Länge in  $|w|$  akzeptieren.
  - ▶ **Soundness**: Wenn Eingabe  $w$  **nicht** zu  $L$  gehört, dann darf der Verifier keinen Beweis akzeptieren.

$r, q : \mathbb{N} \rightarrow \mathbb{N}$  sind vorgegebene Funktionen.

- Ein  $(r, q)$ -Verifier  $V$  ist ein deterministischer Algorithmus, der neben der Eingabe  $w$  auf  $r(|w|)$  Zufallsbits zugreifen kann.
- Wie arbeitet  $V$ ? Für eine Eingabe  $w$  und eine Zufallsfolge  $\sigma$ 
  - ▶ berechnet  $V$  eine Folge von höchstens  $q(|w|)$  Bitpositionen und **fordert die Beweisbits an diesen Positionen vom Prover an.**
  - ▶ Die nachfolgende Rechnung hängt nur von  $w, \sigma$  und den nachgefragten Beweisbits ab.
- Insgesamt muss  $V$  in polynomieller Zeit in  $n$  rechnen.

Eine Sprache  $L$  gehört genau dann zur Komplexitätsklasse  $\text{PCP}(r, q)$ , wenn es einen  $(r, q)$ -Verifizier  $V$  und eine Konstante  $\alpha$  ( $0 < \alpha < 1$ ) gibt mit:

(a) **Vollständigkeit:**

Wenn die Eingabe  $w$  zur Sprache  $L$  gehört, dann akzeptiert  $V$  einen Beweis  $b$  mit Wahrscheinlichkeit 1. (Wir sagen auch, dass  $V$  die Vollständigkeit 1 besitzt.)

(b) **Soundness:**

Wenn die Eingabe  $w$  nicht zur Sprache  $L$  gehört, dann wird  $V$  für jeden Beweis  $b$  mit Wahrscheinlichkeit höchstens  $\alpha$  akzeptieren.

(Wir sagen auch, dass  $V$  die Soundness  $\alpha$  besitzt.)

- $\text{PCP}(0, 0) = \text{P}$ .
- $\text{PCP}(0, \text{poly}(n)) = \text{NP}$ .

# Was kann man mit Zufallsbits anfangen?

Wir möchten überprüfen, ob zwei Graphen  $G_1, G_2$  nicht-isomorph sind und fordern dazu  $O(n \log_2 n)$  Zufallsbits und ein Beweisbit an.

- $V$  wählt zufällig einen Graphen  $G_i$ , und permutiert  $G_i$  mit einer zufälligen Permutation  $\pi$ , um den Graphen  $H$  zu erhalten.
- $V$  erwartet, dass der permutierte Graph genannt wird.
- Erhält  $V$  die Antwort  $b = i$ , dann wird der Beweis akzeptiert und ansonsten verworfen.
  - ▶ Ist  $G_1 \not\equiv G_2$ , dann gibt es einen Beweis, den  $V$  mit Wahrscheinlichkeit 1 akzeptiert.
  - ▶ Ist  $G_1 \equiv G_2$ , dann akzeptiert  $V$  nur mit Wahrscheinlichkeit  $1/2$ .

GRAPH-NICHTISOMORPHIE gehört zu  $PCP(n \cdot \log_2 n, 1)$ . Ein Beweisbit reicht.

# Das PCP-Theorem

Es gilt  $\text{PCP}(O(\log_2 n), O(1)) = \text{NP}$ .

- Die New York Times: New Short Cut Found For Long Math Proofs.
- Die Beziehung  $\text{PCP}(O(\log_2 n), O(1)) \subseteq \text{NP}$  ist offensichtlich. Warum?
- Die ungemein überraschende Aussage ist die Inklusion  
$$\text{NP} \subseteq \text{PCP}(O(\log_2 n), O(1)).$$

*Für eine erfüllbare 3KNF Formel  $\phi$  gibt es also einen polynomiell langen Beweis, so dass die Inspektion von **konstant vielen** Beweisbits bereits überzeugend ist.*



# Die Konsequenzen für die Approximierbarkeit

$g(x)$  sei die Anzahl der Klauseln der KNF-Formel  $x$ .

Die beiden folgenden Aussagen sind äquivalent:

- (a) Es gibt eine Konstante  $0 < \alpha < 1$ , so dass  $[\alpha \cdot g, g]$ -**gap-MAX-3SAT** ein NP-hartes Problem ist.
  - (b)  $\text{PCP}(O(\log_2 n), O(1)) = \text{NP}$ .
- 
- $(b) \Rightarrow (a)$ : Das PCP-Theorem liefert eine lückenschaffende Reduktion für MAX-3SAT: MAX-3SAT besitzt kein Approximationschema!
  - $(a) \Rightarrow (b)$ : Jede lückenschaffende Reduktion für MAX-3SAT erzwingt das PCP-Theorem.

**Die neue Sichtweise von NP ist hinreichend, aber auch notwendig für den Nachweis der Nicht-Approximierbarkeit von Max-3SAT.**

# Nicht-Approximierbarkeit $\Rightarrow$ PCP-Theorem

**Annahme:**  $[\alpha \cdot g, g]$ -gap-MAX-3SAT ist ein NP-hartes Problem.

Für jedes  $L \in \text{NP}$  gibt es eine effiziente Transformation  $T$

- für Eingaben  $w \in L$  sind alle  $g(T(w))$  Klauseln von  $T(w)$  erfüllbar,
- für Eingaben  $w \notin L$  sind weniger als  $\alpha \cdot g(T(w))$  Klauseln erfüllbar.

- Wir konstruieren einen Verifier  $V$ , der annimmt, dass der Beweis eine erfüllende Belegung für  $T(w)$  ist.
  - ▶  $V$  wählt eine Klausel  $k$  von  $T(w)$  zufällig aus,
  - ▶ fragt die Wahrheitswerte der drei Variablen von  $k$  ab
  - ▶ und akzeptiert genau dann, wenn  $k$  erfüllt wird.
- $V$  ist ein  $(O(\log_2 n), O(1))$ -Verifier.
  - ▶ Wenn  $w \in L$ , dann akzeptiert  $V$  mit Wahrscheinlichkeit 1.
  - ▶ Ist  $w \notin L$ , dann sind weniger als  $\alpha \cdot g(w)$  Klauseln erfüllbar und  $V$  akzeptiert fälschlicherweise mit Wahrscheinlichkeit höchstens  $\alpha$ .

# PCP-Theorem $\Rightarrow$ Nicht-Approximierbarkeit

Zeige, dass es  $\alpha < 1$  gibt, so dass jede Sprache  $L \in \text{NP}$  auf  $[\alpha \cdot \mathbf{g}, \mathbf{g}]$ -gap-MAX-3SAT reduzierbar ist.

- Da  $L \in \text{NP}$  ist  $L \in \text{PCP}(O(\log n), O(1))$ .
  - ▶ Für Eingabe  $w$  sei  $n = |w|$ .
  - ▶ Ein Verifier  $V$  akzeptiert  $L$  mit  $O(\log_2 n)$  Zufallsbits  $\sigma$  und  $k = O(1)$  Beweisbits eines unbekanntes Beweises  $b = b_1 b_2 \cdots b_{n^d}$ .
- Beschreibe mit einer DNF-Formel  $D_{w,\sigma}(x_1, \dots, x_{n^d})$  für welche Werte der Beweisbits der Verifier verwirft, wenn  $\sigma$  die Zufallsfolge ist.
  - ▶ Höchstens  $k$  Literale pro Monom reichen aus.
- Die  $k$ -KNF Formel  $K_w = \bigwedge_{\sigma} \neg D_{w,\sigma}(x)$  ist höchstens polynomiell lang: Die Länge von  $K_w$  ist polynomiell in  $2^k$  und  $2^{O(\log_2 |w|)}$ .
  - ▶ Wenn  $w \in L$ , dann ist  $K_w$  erfüllbar.
  - ▶ Wenn  $w \notin L$ , dann akzeptiert  $V$  mit Wahrscheinlichkeit höchstens  $\alpha$ . Also ist ein konstanter Bruchteil  $\alpha'$  aller Klauseln von  $K_w$  nicht erfüllbar.

$w \mapsto K_w$  ist die verlangte Reduktion. Achtung: Das ist eine Reduktion auf  $[\alpha \cdot \mathbf{g}, \mathbf{g}]$ -gap-MAX-3SAT.

Wir nehmen an, dass das PCP-Theorem gilt und wissen, dass MAX-3SAT **kein** polynomielles Approximationsschema besitzt.

- Wir bestimmen die Approximationskomplexität von VERTEX COVER, INDEPENDENT SET (und CLIQUE).
- Dann betrachten wir MAX-3SAT und bestimmen den besten, effizient erreichbaren Approximationsfaktor.

# Independent Set und Vertex Cover

Bestimme die Approximationskomplexität von INDEPENDENT SET für Graphen von beschränktem Grad.

- MAX-3SAT<sub>B</sub>: Erfülle möglichst viele Klauseln einer 3KNF Formel  $\phi$ , wenn jede Variable von  $\phi$  in höchstens  $B$  Klauseln vorkommt.
  - Bestimme die Approximationskomplexität von MAX-3SAT<sub>B</sub>.
- 
- MAX-3SAT besitzt kein polynomielles Approximationsschema.
  - Konstruiere eine lückenerhaltende Reduktion von MAX-3SAT auf MAX-3SAT<sub>B</sub> für eine geeignete Konstante  $B$ .
    - ▶ Die 3KNF-Formel  $\phi$  besitze  $n$  Variablen  $x_1, \dots, x_n$  und  $m$  Klauseln.
      - ★ Die  $i$ te Variable komme  $m_i$ -mal in den Klauseln von  $\phi$  vor.
      - ★ Setze  $N = \sum_i m_i$  und  $N \leq 3 \cdot m$  folgt.
    - ▶ Weise  $\phi$  eine 3KNF Formel  $\psi$  zu, wobei jede Variable von  $\psi$  in höchstens  $B$  Klauseln vorkomme.
    - ▶ Ersetze das  $j$ te Vorkommen von  $x_i$  durch die neue Variable  $x_{i,j}$ .

Erzwinge  $x_{i,1} = \dots = x_{i,m_i}$  für optimale Belegungen.

## Expander-Graphen

Es gibt  $B \in \mathbb{N}$ , so dass ungerichtete Graphen  $G_m = (V_m, E_m)$  in Zeit  $\text{poly}(m)$  konstruiert werden können, wobei

- $|V_m| = m$  und jeder Knoten besitzt genau  $B$  Nachbarn.
- $G_m$  **expandiert**: Für jede Knotenmenge  $W \subseteq V_m$  gibt es mindestens  $1 + \min\{|W|, |V_m - W|\}$  Kanten, die einen Knoten in  $W$  mit einem Knoten aus  $V_m - W$  verbinden.

Fasse die Variablen  $x_{i,1}, \dots, x_{i,m_i}$  als Knoten von  $G_{m_i}$  auf.

- Für jede Kante  $\{r, s\}$  in  $G_{m_i}$  füge die neuen Klauseln  $x_{i,r} \vee \neg x_{i,s}$  und  $\neg x_{i,r} \vee x_{i,s}$  zu  $\psi$  hinzu.
- Die Formel  $\psi$  besteht insgesamt aus  $B \cdot \sum_i m_i = B \cdot N \leq 3B \cdot m$  neuen Klauseln und  $m$  modifizierten alten Klauseln.

Jede optimale Belegung von  $\psi$  erfüllt **alle** neuen Klauseln.

- Angenommen, eine Belegung weist den Variablen  $x_{i,1}, \dots, x_{i,m_i}$  **unterschiedliche** Wahrheitswerte zu.
- O.B.d.A. habe

$$W = \{j \mid x_{i,j} = 0\}$$

höchstens  $\frac{m_i}{2}$  Elemente.

- ▶ Expansion: Mindestens  $1 + |W|$  Kanten von  $G_{m_i}$  verbinden einen Endpunkt in  $W$  mit einem Endpunkt in  $V_{m_i} \setminus W$ .
- Flippe den Wahrheitswert der Variablen in  $W$ :
  - **Möglicherweise erfüllen wir  $|W|$  alte Klauseln nicht mehr.**
  - **Im Gegenzug erfüllen wir aber mindestens  $1 + |W|$  neue Klauseln.**



$g(\phi) = m$  sei die Anzahl der Klauseln von  $\phi$ .

Für jedes  $\frac{1}{2} < \alpha < 1$  gibt es  $\frac{1}{2} < \alpha' < 1$  und eine lückenerhaltende Reduktion von  $[\alpha g(\phi), g(\phi)]$ -gap-MAX-3SAT auf  $[\alpha' g(\phi), g(\phi)]$ -gap-MAX-3SAT<sub>B</sub>.

- Die Reduktion wird durch  $\phi \mapsto \psi$  definiert.
- Jede optimale Belegung von  $\psi$  erfüllt **alle** neuen Klauseln
  - ▶ und für alle  $i$  gilt  $x_{i,1} = \dots = x_{i,m_i}$ .
  - ▶ Eine optimale Belegung für  $\phi$  erfüllt  $m'$  Klauseln  $\Leftrightarrow$   
Eine optimale Belegung für  $\psi$  erfüllt  $m' + BN$  Klauseln.
- Setze  $g_B = g$  und  $f_B = \alpha' g_B$  für eine zu bestimmende Konstante  $\alpha'$ .
  - ▶ Wenn  $m' \geq g(\phi) = m$ , dann ist  $m' + BN \geq g_B(\psi) = g(\psi) = m + BN$ .
  - ▶ Wenn  $m' < f(\phi) = \alpha g(\phi)$ , dann  $m' + BN < f_B(\psi) = \alpha' g_B(\psi) = \alpha'(m + BN)$ .
- Die durch das PCP-Theorem geschaffene Lücke für MAX-3SAT wird für MAX-3SAT<sub>B</sub> nicht zerstört, sondern höchstens verkleinert!

Für jedes  $B \in \mathbb{N}$  für jede 3-KNF Formel  $\phi$  mit höchstens  $B$  Vorkommen einer Variable gibt es eine Transformation

$$\phi \mapsto G_\phi$$

so dass eine optimale Belegung für  $\phi$  genau dann  $m'$  Klauseln erfüllt, wenn eine optimale unabhängige Menge für  $G_\phi$  die Größe  $m'$  hat.

- Konstruktion von  $G_\phi$ :
  - ▶  $G_\phi$  besitzt für jede Klausel eine Gruppe von 3 Knoten (mit einem Knoten pro Literal).
  - ▶ Je zwei Knoten einer Gruppe werden verbunden.
  - ▶ Knoten verschiedener Gruppen werden genau dann verbunden, wenn sich ihre Literale widersprechen.
- Jeder Knoten  $v$  von  $G_\phi$  ist mit höchstens  $B - 1$  „negierten Knoten“ aus anderen Klauseln und mit maximal zwei Knoten aus der eigenen Klausel verbunden:  $v$  hat höchstens  $B + 1$  Nachbarn.

- Sei  $U$  eine unabhängige Knotenmenge in  $G_\phi$ .
  - ▶  $U$  besitzt offensichtlich höchstens einen Knoten pro Gruppe und die Literale der Knoten in  $U$  widersprechen sich nicht.
  - ▶ Erfülle alle Literale zu Knoten in  $U$ :  
Mindestens  $|U|$  Klauseln werden simultan erfüllt.
- Sei  $K$  eine Menge simultan erfüllbarer Klauseln von  $\phi$ .
  - ▶ Mindestens ein Literal pro Klausel in  $K$  wird erfüllt.
  - ▶ Es gibt eine unabhängige Menge in  $G_\phi$  der Größe  $|K|$ .
- Wir haben eine lückenbewahrende Reduktion von  $\text{MAX-3SAT}_B$  auf  $\text{INDEPENDENT SET}_{B+1}$  gefunden.

$\text{INDEPENDENT SET}_{B+1}$  hat kein polynomielles Approximationsschema.

# Die Konsequenzen

- Es gibt eine Konstante  $B$ , so dass

weder  $\text{MAX-3SAT}_B$  noch  $\text{INDEPENDENT-SET}_{B+1}$

ein polynomielles Approximationsschema besitzt.

- $\text{VERTEX COVER}$  hat kein polynomielles Approximationsschema.

Warum?

- $W \subseteq V$  ist genau dann ein Vertex Cover in  $G = (V, E)$ , wenn  $V \setminus W$  eine unabhängige Menge ist.
- Was bedeutet eine sehr scharfe Approximation eines kleinsten Vertex Covers für Graphen  $G$  mit Grad höchstens  $B + 1$ ?
  - ▶  $G$  hat einen Vertex Cover der Größe höchstens  $|V| - \frac{|V|}{B+1} = |V| \cdot (1 - \frac{1}{B+1})$ .
  - ▶ Eine größte unabhängige Menge ist scharf approximierbar: **Unmöglich!**

CLIQUE hat keinen effizienten Algorithmus, der eine Clique der Größe mindestens  $\frac{\text{opt}}{n^\epsilon}$ , für eine Konstante  $\epsilon > 0$ , bestimmt.

- CLIQUE und INDEPENDENT SET sind äquivalente Probleme:
  - ▶ Die maximale Größe einer Clique in  $G$  stimmt überein mit der maximalen Größe einer unabhängigen Menge im Komplementgraphen  $\overline{G}$ .
  - ▶ Auch INDEPENDENT SET kann nur sehr schlecht approximiert werden.
- Das Graph-Produkt ist ein zentrales technisches Hilfsmittel.

## Das Graph-Produkt

$$G = G_1 \times G_2 = (V, E)$$

für ungerichtete Graphen  $G_1 = (V_1, E_1)$  und  $G_2 = (V_2, E_2)$ :

- Es ist  $V = V_1 \times V_2$  und
- $E = \{ \{(u_1, v_1), (u_2, v_2)\} \mid \{u_1, u_2\} \in E_1, \{v_1, v_2\} \in E_2, \}$ .

$$\text{clique}(G_1 \times G_2) = \text{clique}(G_1) \cdot \text{clique}(G_2).$$

wenn  $\text{clique}(H)$  die maximale Größe einer Clique im Graphen  $H$  ist.

Wenn CLIQUE nicht mit dem Faktor  $\alpha$  effizient approximierbar ist, dann ist CLIQUE auch nicht mit dem Faktor  $\alpha^2$  effizient approximierbar.

- Wir wissen bereits, dass INDEPENDENT SET, und damit auch CLIQUE, kein Approximationsschema besitzt.
  - ▶ Das Graph-Produkt: Es gibt auch keine effizienten Algorithmen, die CLIQUE mit **irgendeinem konstanten Faktor** approximieren.
- Es scheint sogar, dass sich ein Approximationsfaktor  $\alpha$  für  $H$  in einen Approximationsfaktor  $\alpha^r$  für  $H^r$  übersetzt.
  - ▶ Dem ist nicht so, denn  $H^r$  ist wesentlich größer als  $H$ .
  - ▶ Approximationsalgorithmen dürfen auf der sehr viel größeren Eingabe  $H^r$  dementsprechend mehr Laufzeit investieren!

Für  $n, k \in \mathbb{N}$  und  $\alpha \in \mathbb{R}$  und  $[n] = \{1, \dots, n\}$  ist

- ein  $(n, k, \delta)$ -Booster  $B$  eine Menge  $k$ -elementiger Teilmengen von  $[n]$ .
- Für jede Teilmenge  $A \subseteq [n]$  gilt

$$\left(\frac{|A|}{n} - \delta\right)^k \cdot |B| \leq |\{T \in B \mid T \subseteq A\}| \leq \left(\frac{|A|}{n} + \delta\right)^k \cdot |B|.$$

- Wenn  $B$  aus allen  $k$ -elementigen Teilmengen von  $[n]$  besteht:
  - ▶  $\left(\frac{|A|}{n}\right)^k$  ist die „ungefähre“ Wahrscheinlichkeit, dass eine zufällig gezogene Menge  $T \in B$  in der Menge  $A$  enthalten ist.
- $(n, k, \delta)$ -Booster liefern gute Schätzungen der Größe von  $A$ .
  - ▶ Der Booster sollte aus nicht zu vielen, effizient konstruierbaren Teilmengen bestehen.
  - ▶ Interessant sind Booster für  $k = O(\log_2 n)$  und kleines  $\delta > 0$ .



Ohne Beweis: Für jedes  $k = O(\log_2 n)$  und jedes  $\delta > 0$  gibt es  $(n, k, \delta)$ -Booster, die in polynomieller Zeit in  $n$  konstruiert werden können.

- Sei  $G$  ein ungerichteter Graph mit Knotenmenge  $\{1, \dots, n\}$ .
- Statt  $G$  betrachten wir das Booster-Produkt  $B(G)$ :
  - ▶ Die Knoten entsprechen den Teilmengen des Boosters  $B$ .
  - ▶ Wir verbinden zwei Boostermengen  $S_1, S_2 \in B$  in  $B(G)$  durch eine Kante, falls  $S_1 \cup S_2$  eine Clique in  $G$  ist.
- Für jeden Graphen  $G$  und für jeden  $(n, k, \delta)$ -Booster  $B$  gilt:

$$\left(\frac{\text{clique}(G)}{n} - \delta\right)^k \cdot |B| \leq \text{clique}(B(G)) \leq \left(\frac{\text{clique}(G)}{n} + \delta\right)^k \cdot |B|.$$

Wenn  $\text{clique}(G)$  nur **mittelmäßig** approximierbar ist, dann „sollte“  $\text{clique}(B(G))$  **sehr schlecht** approximierbar sein.

- $A \subseteq \{1, \dots, n\}$  sei eine größte Clique in  $G$ .
  - ▶ Dann ist  $|A| = \text{clique}(G)$ .
  - ▶ Der Booster besitzt  $\geq \left(\frac{\text{clique}(G)}{n} - \delta\right)^k \cdot |B|$  Teilmengen von  $A$ .
  - ▶ Die in  $A$  enthaltenen Boostermengen bilden eine Clique in  $B(G)$ :

$$\left(\frac{\text{clique}(G)}{n} - \delta\right)^k \cdot |B| \leq \text{clique}(B(G)).$$

- Wenn  $A'$  die größte Clique in  $B(G)$  ist:
  - ▶  $A$  sei die Vereinigung aller Mengen des Boosters, die den Elementen von  $A'$  entsprechen:  $A$  ist eine Clique in  $G$  und  $|A| \leq \text{clique}(G)$ .
  - ▶ Der Booster besitzt höchstens  $\left(\frac{|A|}{n} + \delta\right)^k \cdot |B| \leq \left(\frac{\text{clique}(G)}{n} + \delta\right)^k \cdot |B|$  viele Teilmengen von  $A$ .
  - ▶ Und, nach Definition eines  $(n, k, \delta)$ -Boosters

$$\text{clique}(B(G)) = |A'| = |\{T \in B \mid T \subseteq A\}| \leq \left(\frac{\text{clique}(G)}{n} + \delta\right)^k \cdot |B|.$$

# Zusammenfassung

$[\alpha n, \beta n]$ -gap-INDEPENDENT SET $_{B+1}$  ist NP-hart (für geeignete  $\alpha, \beta$  und  $B$ ).

- $[\alpha n, \beta n]$ -gap-CLIQUE $_{n-B-1}$  ist ein NP-hartes Problem.

Hier sind aber nur Graphen  $G$  interessant mit

$$\text{clique}(G) < \alpha \cdot n \text{ oder } \beta \cdot n \leq \text{clique}(G).$$

- Für einen  $(n, \log_2 n, \delta)$ -Booster  $B$  gilt

$$\text{clique}(B(G)) < (\alpha + \delta)^{\log_2 n} \cdot |B| \text{ oder } (\beta - \delta)^{\log_2 n} \cdot |B| \leq \text{clique}(B(G)).$$

Für ein genügend kleines  $\delta$  haben wir die Lücke

$$\text{von vorher } \frac{\beta}{\alpha} \text{ auf jetzt } \left(\frac{\beta - \delta}{\alpha + \delta}\right)^{\log_2 n}$$

polynomiell amplifiziert.

- **CLIQUE** besitzt für kein  $\varepsilon < 0$  effiziente  $n^{1-\varepsilon}$ -approximative Algorithmen.
- **COLORING** besitzt keine effizienten  $n^{1-\varepsilon}$ -approximativen Algorithmen.
  - ▶ In COLORING ist ein ungerichteter Graph  $G$  gegeben.
  - ▶ Färbe die Knoten mit möglichst wenigen Farben, so dass keine zwei benachbarten Knoten in  $G$  gleichgefärbt sind.

Unser Wissen ist trotzdem noch sehr eingeschränkt:

- Selbst wenn bekannt ist, daß ein gegebener Graph **3**-färbbar ist, sind die besten effizienten Algorithmen bisher nur im Stande, eine Färbung mit  $O(n^{0,25} \cdot \log n)$  Farben zu finden.
- Der kleinste Approximationsfaktor für CLIQUE ist  $O\left(\frac{n}{\log^2 n}\right)$ .

# 3-Bit PCP

# Wieviele Beweisbits sind notwendig?

- Zwei Bits reichen nicht aus.
- Modelliere einen Verifier, der höchstens zwei Beweisbits nachfragt,  
durch eine 2-KNF Formel.
- Es ist  $PCP(O(\log n), 2) = P$ .

Für **alle** Konstanten  $\varepsilon, \eta > 0$  und für alle Sprachen  $L \in \text{NP}$  gibt es einen Verifier  $V$  mit den folgenden Eigenschaften

- (a)  $V$  fordert nur drei Beweisbits  $x, y, z$  und  $O(\log_2 n)$  Zufallsbits an.
- (b)  $V$  entscheidet Akzeptanz über den Wert einer Summe  $\alpha \cdot x + \beta \cdot y + \gamma \cdot z \pmod 2$ .
- (c) Es gilt
  - ▶ „Vollständigkeit“: Für jede Eingabe  $w \in L$  gibt es einen Beweis, der mit Wahrscheinlichkeit  $1 - \varepsilon$  akzeptiert wird.
  - ▶ Soundness: Ist hingegen  $w \notin L$ , dann wird jeder Beweis mit Wahrscheinlichkeit höchstens  $1/2 + \eta$  akzeptiert.

# MAX-3LIN



Bestimme einen Vektor  $x$ , so dass möglichst viele Gleichungen des linearen Gleichungssystem

$$A \cdot x = b \pmod{2}$$

erfüllt sind. In jeder Gleichung kommen höchstens drei Variablen vor.

- Was passiert, wenn wir einen Vektor  $x$  zufällig auswürfeln?
  - ▶ Jede einzelne Gleichung wird mit Wahrscheinlichkeit  $1/2$  erfüllt.
  - ▶ Im Erwartungsfall erfüllen wir die Hälfte aller Gleichungen.
- Und wenn wir eine Belegung für eine MAX-3SAT Instanz zufällig auswürfeln, für die alle Klauseln genau drei Literale besitzen?
  - ▶ Eine Klausel wird mit Wahrscheinlichkeit  $7/8$  erfüllt.
  - ▶ Im Erwartungsfall erfüllen wir genau  $7/8$  aller Klauseln.

Und mehr ist weder für MAX-3SAT noch für MAX-3LIN nicht drin!

# MAX-3LIN und MAX-3SAT: Nichts geht mehr

Es gelte  $P \neq NP$ . Für jedes  $\varepsilon > 0$  besitzt

- (a) weder MAX-3LIN effiziente  $(2 - \varepsilon)$ -approximative Algorithmen
- (b) noch MAX-3SAT effiziente  $(8/7 - \varepsilon)$ -approximative Algorithmen.

Wir lösen das Erfüllbarkeitsproblem mit einem effizienten  $(2 - \varepsilon)$ -approximativen Algorithmus  $\mathcal{A}$  für MAX-3LIN.

- Für eine KNF-Formel  $\phi$  gibt es einen Verifier, der für jede Folge  $\sigma$  von logarithmisch vielen Zufallsbits akzeptiert, wenn die Gleichung

$$\alpha_\sigma \cdot x_\sigma + \beta_\sigma \cdot y_\sigma + \gamma_\sigma \cdot z_\sigma = b_\sigma$$

für die nachgefragten Beweisbits  $x_\sigma, y_\sigma, z_\sigma$  erfüllt ist.

- Wir erhalten ein Gleichungssystem:
  - ▶  $\phi$  ist zu akzeptieren, wenn fast alle Gleichungen erfüllt werden und
  - ▶ zu verwerfen, wenn nur wenig mehr als die Hälfte der Gleichungen erfüllt ist.
- Jetzt können wir  $\phi$  mit  $\mathcal{A}$  entscheiden und das geht nun mal nicht.

## Reduziere MAX-3LIN auf MAX-3SAT.

- Schreibe jede Gleichung des Systems  $A \cdot x = b \pmod 2$  als Konjunktionen von vier Klauseln.
- Anstelle des linearen Systems von  $m$  Gleichungen erhalten wir eine 3KNF Formel  $\phi$  mit  $4m$  Klauseln.
  - ▶ Wenn  $A \cdot x = b \pmod 2$  „fast“ erfüllbar ist, dann sind auch fast alle Klauseln erfüllbar.
  - ▶ Sind hingegen nur wenig mehr als die Hälfte aller Gleichungen erfüllbar, dann sind bei  $m$  Gleichungen ungefähr

$$4 \cdot \frac{m}{2} + 3 \cdot \frac{m}{2} = \frac{7}{2} \cdot m = \frac{7}{8} \cdot 4m$$

der  $4m$  Klauseln erfüllbar.

MAX-3SAT hat keine effizienten  $(8/7 - \varepsilon)$ -approximativen Algorithmen.

# Zusammenfassung

- In der Klasse  $PCP(r, q)$  der „Probabilistically Checkable Proofs“ sind alle Sprachen aufgeführt, die für jede Eingabe  $w$  mit höchstens  $r(w)$  Zufallsbits und  $q(w)$  Beweisbits effizient entscheidbar sind
  - ▶ Das PCP-Theorem besagt, dass  $NP = PCP(O(\log_2 n), O(1))$  gilt.
  - ▶ Das PCP-Theorem ist äquivalent zu der Aussage, dass das Gap-Problem für MAX-3SAT NP-hart ist.
- VERTEX COVER besitzt kein polynomielles Approximationsschema. Diese Aussage folgt, da wir nacheinander MAX-3SAT auf  $MAX-3SAT_B$  und dann auf  $INDEPENDENT SET_B$  lückenbewahrend reduziert haben.
- Wir haben die Lücke von  $INDEPENDENT SET_B$  mit Hilfe des Booster-Produkts amplifiziert: Weder CLIQUE noch  $INDEPENDENT SET$  besitzen effiziente  $n^\epsilon$ -approximative Algorithmen für jedes  $\epsilon > 0$ .
- Wir haben das 3-Bit PCP benutzt, um zu zeigen, dass MAX-3SAT keine effizienten  $(8/7 - \epsilon)$ -approximativen Algorithmen besitzt.