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 $\mathbf{x} \in \mathbf{I}$ aus einer Menge $\mathbf{L(x)}$ von Lösungen.
  - Für ein Minimierungsproblem ist

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

    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 $\mathbf{A(x)} \in \mathbf{L(x)}$.
  - Der Approximationsfaktor für Instanz *x* ist

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

  - Der Approximationsfaktor für Instanzen der Länge *n* ist

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

Der Approximationsfaktor $r_A(x) = \min\{\frac{\mathbf{f(x,A(x))}}{\mathbf{opt(x)}}, \frac{\mathbf{opt(x)}}{\mathbf{f(x,A(x))}}\}$

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

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

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

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

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

Individualität von Optimierungsproblemen

- 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 „$\text{Gewicht}_i(w)$ = 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 ($w_i \mapsto \lfloor \frac{n w_i}{\varepsilon \max_i w_i} \rfloor$)

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

Bin Packing

Eingabe: n Objekte mit den Gewichten $0 \leqslant g_1, \ldots, g_n \leqslant 1$.

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 $(\mathbf{1} + \varepsilon)\boldsymbol{opt}$ Behältern in Zeit 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.

MAX-SAT

Eingabe: Eine Menge K von Klauseln mit Gewichten $w_k \geqslant 0$ 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.

SET COVER

Eingabe: Ein Universum $U = \{1, \ldots, m\}$ und Teilmengen $T_1, \ldots, T_n \subseteq U$ 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 $(\mathbf{1} - \varepsilon) \cdot \ln(\mathbf{m})$ **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 $(\mathbf{1} + \ln(\mathbf{m}))\cdot$ **opt** Mengen gewählt werden.

CLIQUE

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 **P** ein Maximierungsproblem und $\mathbf{opt_P}(x)$ der optimale Wert für Instanz x.

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

$$\begin{aligned} \mathbf{YES_P} &= \{\mathbf{x} \mid \mathbf{opt_P(x)} \geqslant \mathbf{g(x)}\} \\ \mathbf{NO_P} &= \{\mathbf{x} \mid \mathbf{opt_P(x)} < \mathbf{f(x)}\}. \end{aligned}$$

- Für ein Minimierungsproblem kehre die Antworten um.
- Algorithmus A löst das Promise Problem $[\mathbf{f}, \mathbf{g}]$**-gap-P**, wenn A
  - für jedes $x \in \mathrm{YES_P}$ mit "ja" und
  - für jedes $x \in \mathrm{NO_P}$ mit "nein" antwortet.

Wir verlangen eine fehlerfreie Charakterisierung der Instanz x, solange x nicht „in die Lücke" $f(x) \leqslant \mathbf{opt_P}(x) < g(x)$ fällt. In anderen Worten, wir **versprechen** $\mathbf{opt_P}(x) < f(x)$, bzw. $\mathbf{opt_P}(x) \geqslant g(x)$).

Wann ist $[f, g]$-gap-**P** schwierig?

Sei **P** ein Maximierungsproblem.

Wir sagen:

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

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

$$\mathbf{opt_P(T(w)) \geqslant g(T(w)),}$$

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

$$\mathbf{opt_P(T(w)) < f(T(w)).}$$

Ist **P** ein Minimierungsproblem, dann reduziere Eingaben $w \in L$ auf Instanzen aus $\mathbf{NO_P}$ und Eingaben $w \notin L$ auf Instanzen aus $\mathbf{YES_P}$.

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

$$\mathbf{A(x)} \geqslant \frac{\mathbf{f(x)}}{\mathbf{g(x)}} \cdot \mathbf{opt_P}.$$

- Wir lösen das NP-harte Problem **[f, g]-gap-P**:
  - Wenn $A(x) \geqslant f(x)$, dann antworte mit „ja“ (denn $\mathbf{opt_P} \geqslant \mathbf{f(x)}$).
  - Wenn $A(x) < f(x)$, dann antworte mit „nein“
    (denn der Approximationsfaktor höchstens $\frac{g(x)}{f(x)}$ garantiert, dass $\mathbf{opt_P} < \mathbf{g(x)}$).

f sei effizient berechenbar.

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

Lückenerhaltende Reduktionen

$\mathbf{P_1}$ und $\mathbf{P_2}$ seien zwei Optimierungsprobleme.

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

- Wenn $\mathbf{opt_{P_1}(x)} \geqslant \mathbf{g_1(x)}$, dann $\mathbf{opt_{P_2}(T(x))} \geqslant \mathbf{g_2(T(x))}$.
- Wenn $\mathbf{opt_{P_1}(x)} < \mathbf{f_1(x)}$, dann $\mathbf{opt_{P_2}(T(x))} < \mathbf{f_2(T(x))}$.

- Wenn $[\mathbf{f_1}, \mathbf{g_1}]$-**gap-**$\mathbf{P_1}$ lückenerhaltend auf $[\mathbf{f_2}, \mathbf{g_2}]$-**gap-**$\mathbf{P_2}$ reduzierbar ist und wenn $[\mathbf{f_1}, \mathbf{g_1}]$-**gap-**$\mathbf{P_1}$ ein NP-hartes Problem ist, dann ist auch $[\mathbf{f_2}, \mathbf{g_2}]$-**gap-**$\mathbf{P_2}$ ein NP-hartes Problem.

- Mit einem ersten NP-harten Gap-Problem können wir eine Lawine weiterer Nicht-Approximierbarkeitsergebnisse lostreten.

Ist der Nachweis der NP-Härte schwierig?

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

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

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) päsentiert 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.

NP: Die neue Sichtweise (1/2)

$r, q : \mathbb{N} \to \mathbb{N}$ 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 σ
  - 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, σ und den nachgefragten Beweisbits ab.
- Insgesamt muss V in polynomieller Zeit in n rechnen.

NP: Die neue Sichtweise (2/2)

Eine Sprache L gehört genau dann zur Komplexitätsklasse PCP(r, q), wenn es einen (r, q)-Verifier V und eine Konstante α $(0 < \alpha < 1)$ 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 α akzeptieren.
(Wir sagen auch, dass V die Soundness α besitzt.)

- PCP$(0, 0) =$ P.
- PCP$(0, \text{poly}(n)) =$ 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 π, 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 $\mathrm{PCP}(n \cdot \log_2 n, 1)$. Ein Beweisbit reicht.

Das PCP-Theorem

Das PCP-Theorem

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

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

$$\mathrm{NP} \subseteq \mathrm{PCP}(O(\log_2 n), O(1)).$$

Für eine erfüllbare 3KNF Formel ϕ 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 $[\alpha \cdot \mathbf{g}, \mathbf{g}]$-**gap-MAX-3SAT** ein NP-hartes Problem ist.

(b) $\mathrm{PCP}(O(\log_2 n), O(1)) = \mathrm{NP}$.

- $(b) \Rightarrow (a)$: Das PCP-Theorem liefert eine lückenschaffende Reduktion für MAX-3SAT: MAX-3SAT besitzt kein Approximationsschema!
- $(a) \Rightarrow (b)$: 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: $[\alpha \cdot \mathbf{g}, \mathbf{g}]$**-gap-MAX-3SAT** ist ein NP-hartes Problem.

Für jedes $L \in$ 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 α.

PCP-Theorem $\Rightarrow$ Nicht-Approximierbarkeit

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

- Da $L \in \mathrm{NP}$ ist $L \in \mathrm{PCP}(O(\log n), O(1))$.
  - Für Eingabe w sei $n = |w|$.
  - Ein Verifier V akzeptiert L mit $O(\log_2 n)$ Zufallsbits σ und $k = O(1)$ Beweisbits eines unbekannten Beweises $b = b_1 b_2 \cdots b_{n^d}$.
- Beschreibe mit einer DNF-Formel $D_{w,\sigma}(x_1, \ldots, x_{n^d})$ für welche Werte der Beweisbits der Verifier verwirft, wenn σ 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 α. Also ist ein konstanter Bruchteil α' aller Klauseln von K_w nicht erfüllbar.

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

Wie geht es weiter?

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

MAX-3SAT$_B$ (1/4)

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

- MAX-3SAT$_B$: Erfülle möglichst viele Klauseln einer 3KNF Formel ϕ, wenn jede Variable von ϕ in höchstens B Klauseln vorkommt.
- Bestimme die Approximationskomplexität von MAX-3SAT$_B$.

- MAX-3SAT besitzt kein polynomielles Approximationsschema.
- Konstruiere eine lückenerhaltende Reduktion von MAX-3SAT auf MAX-3SAT$_B$ für eine geeignete Konstante B.
  - Die 3KNF-Formel ϕ besitze n Variablen $x_1, \ldots, x_n$ und m Klauseln.
    - Die $i$te Variable komme m_i-mal in den Klauseln von ϕ vor.
    - Setze $N = \sum_i m_i$ und $N \leqslant 3 \cdot m$ folgt.
  - Weise ϕ eine 3KNF Formel ψ zu, wobei jede Variable von ψ in höchstens B Klauseln vorkomme.
  - Ersetze das $j$te Vorkommen von x_i durch die neue Variable $x_{i,j}$.

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

Expander-Graphen

Es gibt $B \in \mathbb{N}$, so dass ungerichtete Graphen $G_m = (V_m, E_m)$ in Zeit 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 $x_{i,1}, \ldots, x_{i,m_i}$ 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 ψ hinzu.
- Die Formel ψ besteht insgesamt aus $B \cdot \sum_i m_i = B \cdot N \leqslant 3B \cdot m$ neuen Klauseln und m modifizierten alten Klauseln.

Jede optimale Belegung von ψ erfüllt **alle** neuen Klauseln.

- Angenommen, eine Belegung weist den Variablen $x_{i,1}$,.., 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.

MAX-3SAT$_B$ (4/4)

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

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$_B$.

- Die Reduktion wird durch $\phi \mapsto \psi$ definiert.
- Jede optimale Belegung von ψ erfüllt **alle** neuen Klauseln
  - und für alle i gilt $x_{i,1} = \cdots = x_{i,m_i}$.
  - Eine optimale Belegung für ϕ erfüllt m' Klauseln $\Leftrightarrow$ Eine optimale Belegung für ψ erfüllt $m' + BN$ Klauseln.
- Setze $g_B = g$ und $f_B = \alpha' g_B$ für eine zu bestimmende Konstante α'.
  - Wenn $m' \geqslant g(\phi) = m$, dann ist $m' + BN \geqslant g_B(\psi) = g(\psi) = m + BN$.
  - 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$_B$ nicht zerstört, sondern höchstens verkleinert!

INDEPENDENT SET$_{B+1}$ (1/2)

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

$$\phi \mapsto G_\phi$$

so dass eine optimale Belegung für ϕ genau dann m' Klauseln erfüllt, wenn eine optimale unabhängige Menge für G_ϕ die Größe m' hat.

- Konstruktion von G_ϕ:
  - G_ϕ 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_ϕ 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.

INDEPENDENT SET$_{B+1}$ (2/2)

- Sei U eine unabhängige Knotenmenge in G_ϕ.
  - 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 ϕ.
  - Mindestens ein Literal pro Klausel in K wird erfüllt.
  - Es gibt eine unabhängige Menge in G_ϕ der Größe $|K|$.
- Wir haben eine lückenbewahrende Reduktion von MAX-3SAT$_B$ auf INDEPENDENT SET$_{B+1}$ gefunden.

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

Die Konsequenzen

- Es gibt eine Konstante B, so dass

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

  ein polynomielles Approximationsschema besitzt.
- 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 ist Hammer-hart.

CLIQUE hat keinen effizienten Algorithmus, der eine Clique der Größe mindestens $\frac{\text{opt}}{n^\varepsilon}$, für eine Konstante $\varepsilon > 0$, 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 (1/2)

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 clique(H) die maximale Größe einer Clique im Graphen H ist.

Wenn CLIQUE nicht mit dem Faktor α effizient approximierbar ist, dann ist CLIQUE auch nicht mit dem Faktor α^2 effizient approximierbar.

Das Graph-Produkt (2/2)

- 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 α für H in einen Approximationsfaktor α^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!

Das Booster-Produkt (1/3)

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

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

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

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

Das Booster-Produkt (2/3)

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

- Sei G ein ungerichteter Graph mit Knotenmenge $\{1, \ldots, n\}$.
- Statt G^r 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, δ)-Booster B gilt:

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

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

Das Booster-Produkt (3/3)

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

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

- 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 $|A| \leqslant \text{clique}(G)$.
  - Der Booster besitzt höchstens
    $(\frac{|A|}{n} + \delta)^k \cdot |B| \leqslant (\frac{\text{clique}\ (G)}{n} + \delta)^k \cdot |B|$ viele Teilmengen von A.
  - Und, nach Definition eines (n, k, δ)-Boosters

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

Zusammenfassung

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

- $[\alpha n, \beta n]$-gap-CLIQUE$_{n-B-1}$ ist ein NP-hartes Problem.
  Hier sind aber nur Graphen G interessant mit

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

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

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

Für ein genügend kleines δ haben wir die Lücke

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

polynomiell amplifiziert.

Der Status Quo

- 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 $O(\frac{n}{\log^2 n})$.

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 $\mathrm{PCP}(O(\log n), 2) = \mathrm{P}$.

3-Bit PCP

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 $\alpha \cdot x + \beta \cdot y + \gamma \cdot z \bmod 2$.

(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

MAX-3LIN

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

$$A \cdot x = b \bmod 2$$

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 ϕ gibt es einen Verifier, der für jede Folge σ 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:
  - ϕ 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 ϕ mit $\mathcal{A}$ entscheiden und das geht nun mal nicht.

MAX-3SAT

Reduziere MAX-3LIN auf MAX-3SAT.

- Schreibe jede Gleichung des Systems $A \cdot x = b \bmod 2$ als Konjunktionen von vier Klauseln.
- Anstelle des linearen Systems von m Gleichungen erhalten wir eine 3KNF Formel ϕ mit $4m$ Klauseln.
  - Wenn $A \cdot x = b \bmod 2$ „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

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^ε-approximative Algorithmen für jedes $\varepsilon > 0$.
- Wir haben das 3-Bit PCP benutzt, um zu zeigen, dass MAX-3SAT keine effizienten $(8/7 - \varepsilon)$-approximativen Algorithmen besitzt.