PCP-Theorem und Approximationsschwere

Andreas Weise

29.06.2010

Worum geht es beim PCP-Theorem?

- Reicht es, einen Beweis stichprobenartig zu lesen, um Fehler mit hoher Wahrscheinlichkeit zu finden?
- Kann man selbst dann effizient beliebig genaue Näherungslösungen für NP-schwere Optimierungsprobleme berechnen, wenn P $\neq$ NP?

Approximation NP-schwerer Optimierungsprobleme

Der Wert $val(\varphi)$ einer Formel

Definition

Der *Wert* einer Formel φ in 3KNF mit n Klauseln ist definiert als die größte (rationale) Zahl $q \in [0,1]$, für die gilt: es existiert eine Belegung der Variablen von φ, die m Klauseln erfüllt, und $q = \frac{m}{n}$.
Wir schreiben dafür $val(\varphi)$.
Offensichtlich gilt: φ ist erfüllbar $\Leftrightarrow$ $val(\varphi) = 1$.

Das Optimierungsproblem MAX-3SAT

Definition

Das Optimierungsproblem MAX-3SAT besteht darin, zu einer gegebenen logischen Formel φ eine Belegung zu finden, die die Anzahl erfüllter Klauseln maximiert, also gerade $val(\varphi) \cdot n$ Klauseln wahr macht, wobei n die Anzahl der Klauseln in φ ist. MAX-3SAT ist NP-schwer, da 3SAT NP-vollständig ist.

ρ-Approximation für MAX-3SAT

Definition

Sei $\rho \in [0,1]$. Ein Algorithmus A heißt *ρ-Approximations-Algorithmus* für MAX-3SAT, wenn A zu jeder beliebigen eingegebenen Formel φ in 3KNF eine Belegung ausgibt, die mindestens $\rho \cdot val(\varphi) \cdot n$ Klauseln in φ erfüllt.

Beispiel: $\frac{1}{2}$-Approximation für MAX-3SAT

- Eingabe: Formel φ in 3KNF (n Klauseln, m Variablen)
- für i $= 1$ bis m:
  1. belege Variable i mit dem Wert, der mindestens die Hälfte der verbleibenden Klauseln wahr macht, in denen die Variable vorkommt (bei Gleichstand zufällig wählen)
  2. streiche alle erfüllten Klauseln

Die erzeugte Belegung erfüllt mindestens die Hälfte aller Klauseln also auch mindestens halb so viele Klauseln wie maximal möglich.

Der Algorithmus am Beispiel

- $(a \vee \neg b \vee c) \wedge (b \vee c \vee \neg d) \wedge (a \vee b \vee c) \wedge (b \vee \neg c \vee d) \wedge (\neg a \vee \neg b \vee \neg c) \wedge (a \vee \neg b \vee \neg c) \wedge (\neg a \vee c \vee d)$

Der Algorithmus am Beispiel

- $(a \vee \neg b \vee c) \wedge (b \vee c \vee \neg d) \wedge (a \vee b \vee c) \wedge (b \vee \neg c \vee d) \wedge (\neg a \vee \neg b \vee \neg c) \wedge (a \vee \neg b \vee \neg c) \wedge (\neg a \vee c \vee d)$
- 3 Klauseln mit a, 2 mit $\neg a \rightarrow a$ wird wahr gesetzt

Der Algorithmus am Beispiel

- $(a \vee \neg b \vee c) \wedge (b \vee c \vee \neg d) \wedge (a \vee b \vee c) \wedge (b \vee \neg c \vee d) \wedge (\neg a \vee \neg b \vee \neg c) \wedge (a \vee \neg b \vee \neg c) \wedge (\neg a \vee c \vee d)$
- 3 Klauseln mit a, 2 mit $\neg a \rightarrow a$ wird wahr gesetzt
- ~~$(a \vee \neg b \vee c) \wedge$~~ $(b \vee c \vee \neg d) \wedge$ ~~$(a \vee b \vee c) \wedge$~~ $(b \vee \neg c \vee d) \wedge (\neg a \vee \neg b \vee \neg c) \wedge$ ~~$(a \vee \neg b \vee \neg c) \wedge$~~ $(\neg a \vee c \vee d)$

Der Algorithmus am Beispiel

- $\cancel{(a \vee \neg b \vee c) \wedge} (b \vee c \vee \neg d) \wedge \cancel{(a \vee b \vee c) \wedge} (b \vee \neg c \vee d) \wedge (\neg a \vee \neg b \vee \neg c) \wedge \cancel{(a \vee \neg b \vee \neg c) \wedge} (\neg a \vee c \vee d)$

Der Algorithmus am Beispiel

- ~~$(a \vee \neg b \vee c) \wedge$~~ $(b \vee c \vee \neg d) \wedge$ ~~$(a \vee b \vee c) \wedge$~~ $(b \vee \neg c \vee d) \wedge$ $(\neg a \vee \neg b \vee \neg c) \wedge$ ~~$(a \vee \neg b \vee \neg c) \wedge$~~ $(\neg a \vee c \vee d)$
- 2 Klauseln mit b, 1 mit $\neg b \to b$ wird wahr gesetzt

Der Algorithmus am Beispiel

- ~~$(a \vee \neg b \vee c) \wedge$~~ $(b \vee c \vee \neg d) \wedge$ ~~$(a \vee b \vee c) \wedge$~~ $(b \vee \neg c \vee d) \wedge$ $(\neg a \vee \neg b \vee \neg c) \wedge$ ~~$(a \vee \neg b \vee \neg c) \wedge$~~ $(\neg a \vee c \vee d)$
- 2 Klauseln mit b, 1 mit $\neg b \to b$ wird wahr gesetzt
- ~~$(a \vee \neg b \vee c) \wedge (b \vee c \vee \neg d) \wedge (a \vee b \vee c) \wedge (b \vee \neg c \vee d) \wedge$~~ $(\neg a \vee \neg b \vee \neg c) \wedge$ ~~$(a \vee \neg b \vee \neg c) \wedge$~~ $(\neg a \vee c \vee d)$

Der Algorithmus am Beispiel

- ~~$(a \vee \neg b \vee c) \wedge (b \vee c \vee \neg d) \wedge (a \vee b \vee c) \wedge (b \vee \neg c \vee d) \wedge$~~
  $(\neg a \vee \neg b \vee \neg c) \wedge$ ~~$(a \vee \neg b \vee \neg c) \wedge$~~ $(\neg a \vee c \vee d)$

Der Algorithmus am Beispiel

- ~~$(a \vee \neg b \vee c) \wedge (b \vee c \vee \neg d) \wedge (a \vee b \vee c) \wedge (b \vee \neg c \vee d) \wedge$~~
  $(\neg a \vee \neg b \vee \neg c) \wedge$ ~~$(a \vee \neg b \vee \neg c) \wedge$~~ $(\neg a \vee c \vee d)$
- je 1 Klausel mit c bzw. $\neg c \to c$ wahr oder falsch

Der Algorithmus am Beispiel

- ~~$(a \vee \neg b \vee c) \wedge (b \vee c \vee \neg d) \wedge (a \vee b \vee c) \wedge (b \vee \neg c \vee d) \wedge$~~
  $(\neg a \vee \neg b \vee \neg c) \wedge$ ~~$(a \vee \neg b \vee \neg c) \wedge$~~ $(\neg a \vee c \vee d)$

- wird c wahr gesetzt, liefert der Algorithmus eine Belegung, die nur 6 von 7 Klauseln erfüllt

Der Algorithmus am Beispiel

- $(a \vee \neg b \vee c) \wedge (b \vee c \vee \neg d) \wedge (a \vee b \vee c) \wedge (b \vee \neg c \vee d) \wedge (\neg a \vee \neg b \vee \neg c) \wedge (a \vee \neg b \vee \neg c) \wedge (\neg a \vee c \vee d)$

- wird c wahr gesetzt, liefert der Algorithmus eine Belegung, die nur 6 von 7 Klauseln erfüllt
- die Formel ist jedoch erfüllbar durch $a = 0$, $b = 0$, $c = 1$ und $d = 1$

PCP-Systeme

PCP-Systeme

Definition

Seien L eine Sprache und $q, r : \mathbb{N} \to \mathbb{N}$ Funktionen. L hat ein $(r(n), q(n))$-*PCP-System*, wenn es einen probabilistischen polynomialzeitbeschränkten Algorithmus V gibt, der die folgenden drei Bedingungen erfüllt:

PCP-Systeme

Definition

Seien L eine Sprache und $q, r : \mathbb{N} \to \mathbb{N}$ Funktionen. L hat ein $(r(n), q(n))$-*PCP-System*, wenn es einen probabilistischen polynomialzeitbeschränkten Algorithmus V gibt, der die folgenden drei Bedingungen erfüllt:

1. Effizienz: Bei Eingabe $x \in \{0,1\}^n$ und wahlfreiem Zugriff auf $\pi \in \{0,1\}^*$ mit $|\pi| \leq q(n) \cdot 2^{r(n)}$ nutzt V höchstens $r(n)$ Münzwürfe und stellt höchstens $q(n)$ nicht-adaptive Anfragen an Zeichen in π. Dann gibt er 1 oder 0 aus.

PCP-Systeme

Definition

Seien L eine Sprache und $q, r : \mathbb{N} \to \mathbb{N}$ Funktionen. L hat ein $(r(n), q(n))$-*PCP-System*, wenn es einen probabilistischen polynomialzeitbeschränkten Algorithmus V gibt, der die folgenden drei Bedingungen erfüllt:

1. Effizienz: Bei Eingabe $x \in \{0,1\}^n$ und wahlfreiem Zugriff auf $\pi \in \{0,1\}^*$ mit $|\pi| \leq q(n) \cdot 2^{r(n)}$ nutzt V höchstens $r(n)$ Münzwürfe und stellt höchstens $q(n)$ nicht-adaptive Anfragen an Zeichen in π. Dann gibt er 1 oder 0 aus.
2. Vollständigkeit:
   $x \in L \to \exists \pi \in \{0,1\}^* : \Pr[V^{\pi}(x) = 1] = 1 \wedge |\pi| \leq q(n) \cdot 2^{r(n)}$.

PCP-Systeme

Definition

Seien L eine Sprache und $q, r : \mathbb{N} \to \mathbb{N}$ Funktionen. L hat ein $(r(n), q(n))$-*PCP-System*, wenn es einen probabilistischen polynomialzeitbeschränkten Algorithmus V gibt, der die folgenden drei Bedingungen erfüllt:

1. Effizienz: Bei Eingabe $x \in \{0,1\}^n$ und wahlfreiem Zugriff auf $\pi \in \{0,1\}^*$ mit $|\pi| \leq q(n) \cdot 2^{r(n)}$ nutzt V höchstens $r(n)$ Münzwürfe und stellt höchstens $q(n)$ nicht-adaptive Anfragen an Zeichen in π. Dann gibt er 1 oder 0 aus.
2. Vollständigkeit:
   $x \in L \to \exists \pi \in \{0,1\}^* : \Pr[V^\pi(x) = 1] = 1 \wedge |\pi| \leq q(n) \cdot 2^{r(n)}$.
3. Korrektheit:
   $x \notin L \to \forall \pi \in \{0,1\}^* : \Pr[V^\pi(x) = 1] \leq \frac{1}{2} \wedge |\pi| \leq q(n) \cdot 2^{r(n)}$.

Anmerkungen

- PCP steht für *probabilistically checkable proofs*

Anmerkungen

- PCP steht für *probabilistically checkable proofs*
- $V^{\pi}(x)$ bezeichnet die Zufallsvariable , die die Ausgabe von V bei Eingabe x und wahlfreiem Zugriff auf π repräsentiert.

Anmerkungen

- PCP steht für *probabilistically checkable proofs*
- $V^{\pi}(x)$ bezeichnet die Zufallsvariable , die die Ausgabe von V bei Eingabe x und wahlfreiem Zugriff auf π repräsentiert.
- π wird im Folgenden *Beweis* genannt, ein π mit $\Pr[V^{\pi}(x) = 1] = 1$ heißt *korrekter Beweis* von x.

Anmerkungen

- PCP steht für *probabilistically checkable proofs*
- $V^{\pi}(x)$ bezeichnet die Zufallsvariable , die die Ausgabe von V bei Eingabe x und wahlfreiem Zugriff auf π repräsentiert.
- π wird im Folgenden *Beweis* genannt, ein π mit $\Pr[V^{\pi}(x) = 1] = 1$ heißt *korrekter Beweis* von x.
- Eine nicht-adaptive Anfrage hängt nur von der Eingabe und den Münzwürfen ab, jedoch nicht von Ergebnissen vorheriger Anfragen.

Anmerkungen

- PCP steht für *probabilistically checkable proofs*
- $V^{\pi}(x)$ bezeichnet die Zufallsvariable , die die Ausgabe von V bei Eingabe x und wahlfreiem Zugriff auf π repräsentiert.
- π wird im Folgenden *Beweis* genannt, ein π mit $\Pr[V^{\pi}(x) = 1] = 1$ heißt *korrekter Beweis* von x.
- Eine nicht-adaptive Anfrage hängt nur von der Eingabe und den Münzwürfen ab, jedoch nicht von Ergebnissen vorheriger Anfragen.
- Die Forderung, dass $|\pi| \leq q(n) \cdot 2^{r(n)}$ ist eigentlich unnötig, da höchstens diese Anzahl verschiedener nicht-adaptiver Anfragen mit Wahrscheinlichkeit größer 0 möglich ist.

Die PCP-Sprachklassen

Definition

Eine Sprache L ist in $\mathrm{PCP}(r(n), q(n))$, wenn Konstanten $c, d > 0$ existieren, sodass L ein $(c \cdot r(n), d \cdot q(n))$-PCP-System hat.

Die PCP-Sprachklassen

Definition

Eine Sprache L ist in $\mathrm{PCP}(r(n), q(n))$, wenn Konstanten $c, d > 0$ existieren, sodass L ein $(c \cdot r(n), d \cdot q(n))$-PCP-System hat.

Statt $\frac{1}{2}$ hätte auch jede andere positive Konstante kleiner als 1 in der Definition der Korrektheit verwendet werden können ohne die definierten Sprachklassen zu verändern.

Beispiel für ein PCP-System

Die Sprache GNI aller Paare nicht-isomorpher Graphen ist in $\mathrm{PCP}(poly(n), 1)$.

Beispiel für ein PCP-System

Die Sprache GNI aller Paare nicht-isomorpher Graphen ist in $\mathrm{PCP}(poly(n), 1)$.

Beweis durch Angabe des Systems:

Beispiel für ein PCP-System

Die Sprache GNI aller Paare nicht-isomorpher Graphen ist in $\mathrm{PCP}(poly(n), 1)$.

Beweis durch Angabe des Systems:

- Eingabe x: ein Paar $< G_0, G_1 >$ von Graphen mit n Knoten

Beispiel für ein PCP-System

Die Sprache GNI aller Paare nicht-isomorpher Graphen ist in $\mathrm{PCP}(poly(n), 1)$.

Beweis durch Angabe des Systems:

- Eingabe x: ein Paar $< G_0, G_1 >$ von Graphen mit n Knoten
- Beweis π: ein Bit für jeden Graphen H mit n Knoten.

$$\pi[H] = \begin{cases} 0, \text{ falls } H \text{ nur isomorph zu } G_0 \\ 1, \text{ falls } H \text{ nur isomorph zu } G_1 \\ \text{beliebig, falls } H \text{ isomorph zu beiden oder keinem} \end{cases}$$

Beispiel für ein PCP-System

Die Sprache GNI aller Paare nicht-isomorpher Graphen ist in $\mathrm{PCP}(poly(n), 1)$.

Beweis durch Angabe des Systems:

- Eingabe x: ein Paar $< G_0, G_1 >$ von Graphen mit n Knoten
- Beweis π: ein Bit für jeden Graphen H mit n Knoten.

$$\pi[H] = \begin{cases} 0, \text{ falls } H \text{ nur isomorph zu } G_0 \\ 1, \text{ falls } H \text{ nur isomorph zu } G_1 \\ \text{beliebig, falls } H \text{ isomorph zu beiden oder keinem} \end{cases}$$

- Algorithmus:

Beispiel für ein PCP-System

Die Sprache GNI aller Paare nicht-isomorpher Graphen ist in $\mathrm{PCP}(poly(n), 1)$.

Beweis durch Angabe des Systems:

- Eingabe x: ein Paar $< G_0, G_1 >$ von Graphen mit n Knoten
- Beweis π: ein Bit für jeden Graphen H mit n Knoten.

$$\pi[H] = \begin{cases} 0, \text{ falls } H \text{ nur isomorph zu } G_0 \\ 1, \text{ falls } H \text{ nur isomorph zu } G_1 \\ \text{beliebig, falls } H \text{ isomorph zu beiden oder keinem} \end{cases}$$

- Algorithmus:
  1. erzeuge eine Zufallszahl $b \in \{0, 1\}$ und eine zufällige Permutation der Zahlen von 1 bis n

Beispiel für ein PCP-System

Die Sprache GNI aller Paare nicht-isomorpher Graphen ist in $\mathrm{PCP}(poly(n), 1)$.

Beweis durch Angabe des Systems:

- Eingabe x: ein Paar $< G_0, G_1 >$ von Graphen mit n Knoten
- Beweis π: ein Bit für jeden Graphen H mit n Knoten.

$$\pi[H] = \begin{cases} 0, \text{ falls } H \text{ nur isomorph zu } G_0 \\ 1, \text{ falls } H \text{ nur isomorph zu } G_1 \\ \text{beliebig, falls } H \text{ isomorph zu beiden oder keinem} \end{cases}$$

- Algorithmus:
  1. erzeuge eine Zufallszahl $b \in \{0, 1\}$ und eine zufällige Permutation der Zahlen von 1 bis n
  2. erzeuge $H \simeq G_b$ durch Anwendung dieser Permutation auf G_b

Beispiel für ein PCP-System

Die Sprache GNI aller Paare nicht-isomorpher Graphen ist in $\mathrm{PCP}(poly(n), 1)$.

Beweis durch Angabe des Systems:

- Eingabe x: ein Paar $< G_0, G_1 >$ von Graphen mit n Knoten
- Beweis π: ein Bit für jeden Graphen H mit n Knoten.

$$\pi[H] = \begin{cases} 0, \text{ falls } H \text{ nur isomorph zu } G_0 \\ 1, \text{ falls } H \text{ nur isomorph zu } G_1 \\ \text{beliebig, falls } H \text{ isomorph zu beiden oder keinem} \end{cases}$$

- Algorithmus:
  1. erzeuge eine Zufallszahl $b \in \{0, 1\}$ und eine zufällige Permutation der Zahlen von 1 bis n
  2. erzeuge $H \simeq G_b$ durch Anwendung dieser Permutation auf G_b
  3. gib 1 aus, falls $\pi[H] = b$ und 0 sonst

Effizienz, Vollständigkeit und Korrektheit des Systems

1. Für b wird nur ein Münzwurf benötigt und die Permutation lässt sich mit einer polynomiellen Anzahl an Münzwürfen erzeugen (zum Beispiel mit $O(n \log n)$). Es wird genau eine Anfrage an den Beweis gestellt.

Effizienz, Vollständigkeit und Korrektheit des Systems

1. Für b wird nur ein Münzwurf benötigt und die Permutation lässt sich mit einer polynomiellen Anzahl an Münzwürfen erzeugen (zum Beispiel mit $O(n \log n)$). Es wird genau eine Anfrage an den Beweis gestellt.
2. Wenn G_0 und G_1 nicht isomorph sind, dann wird das System bei korrekt konstruiertem π in $\pi[H]$ stets das erwartete b vorfinden und akzeptieren.

Effizienz, Vollständigkeit und Korrektheit des Systems

1. Für b wird nur ein Münzwurf benötigt und die Permutation lässt sich mit einer polynomiellen Anzahl an Münzwürfen erzeugen (zum Beispiel mit $O(n \log n)$). Es wird genau eine Anfrage an den Beweis gestellt.
2. Wenn G_0 und G_1 nicht isomorph sind, dann wird das System bei korrekt konstruiertem π in $\pi[H]$ stets das erwartete b vorfinden und akzeptieren.
3. Sind G_0 und G_1 isomporph, kann die Wahrscheinlichkeit, dass in $\pi[H]$ das erwartete b steht, für kein π größer als $\frac{1}{2}$ sein.

PCP-Theorem

Die zwei Sichtweisen des Theorems

- Theorem 1, erste Sichtweise: Charakterisierung von NP
  $\mathrm{NP} = \mathrm{PCP}(\log n, 1)$

Die zwei Sichtweisen des Theorems

- Theorem 1, erste Sichtweise: Charakterisierung von NP
  $\mathrm{NP} = \mathrm{PCP}(\log n, 1)$
- Theorem 2, zweite Sichtweise: Approximationsschwere
  Es gibt ein $\rho < 1$, sodass für alle $L \in \mathrm{NP}$ eine in polynomieller Zeit berechenbare Funktion f existiert, die Wörter auf (Repräsentationen von) Formeln in 3KNF abbildet, sodass gilt:
  1. $x \in L \Rightarrow val(f(x)) = 1$
  2. $x \notin L \Rightarrow val(f(x)) < \rho$

Anmerkungen

- Allgemein gilt $\mathrm{PCP}(r(n), q(n)) \subseteq \mathrm{NTIME}(2^{O(r(n))} q(n))$.
  Es folgt $\mathrm{PCP}(\log n, 1) \subseteq \mathrm{NTIME}(2^{O(\log n)}) = \mathrm{NP}$.

Anmerkungen

- Allgemein gilt $\mathrm{PCP}(r(n), q(n)) \subseteq \mathrm{NTIME}(2^{O(r(n))} q(n))$. Es folgt $\mathrm{PCP}(\log n, 1) \subseteq \mathrm{NTIME}(2^{O(\log n)}) = \mathrm{NP}$.
- Die PCP-Systeme zu verschiedenen Sprachen in NP können eine verschiedene (konstante) Anzahl an Münzwürfen verwenden. Die Konstante für das PCP-System von 3SAT stellt eine universelle obere Schranke dar.

Anmerkungen

- Allgemein gilt $\mathrm{PCP}(r(n), q(n)) \subseteq \mathrm{NTIME}(2^{O(r(n))} q(n))$. Es folgt $\mathrm{PCP}(\log n, 1) \subseteq \mathrm{NTIME}(2^{O(\log n)}) = \mathrm{NP}$.
- Die PCP-Systeme zu verschiedenen Sprachen in NP können eine verschiedene (konstante) Anzahl an Münzwürfen verwenden. Die Konstante für das PCP-System von 3SAT stellt eine universelle obere Schranke dar.
- Als Korollar folgt aus der zweiten Sichtweise: Es existiert ein $\rho < 1$, sodass aus der Existenz eines polynomialzeitbeschränkten ρ-Approximations-Algorithmus für MAX-3SAT folgt, dass $\mathrm{P} = \mathrm{NP}$.

Äquivalenz der beiden Sichtweisen

Constraint Satisfaction Problems

Definition

Sei $q \in \mathbb{N}$. Eine qCSP-Instanz φ ist eine Menge von Funktionen $\varphi_1, \ldots, \varphi_m$ von $\{0,1\}^n$ nach $\{0,1\}$, *constraints* genannt, wobei für jedes $i \in \{1, \ldots, m\}$ Indizes $j_1, \ldots, j_q \in \{1, \ldots, n\}$ und eine Funktion f von $\{0,1\}^q$ nach $\{0,1\}$ existieren, sodass für alle $u \in \{0,1\}^n$ gilt: $\varphi_i(u) = f(u_{j_1}, \ldots, u_{j_q})$.

Constraint Satisfaction Problems

Definition

Sei $q \in \mathbb{N}$. Eine qCSP-Instanz φ ist eine Menge von Funktionen $\varphi_1, \ldots, \varphi_m$ von $\{0,1\}^n$ nach $\{0,1\}$, *constraints* genannt, wobei für jedes $i \in \{1, \ldots, m\}$ Indizes $j_1, \ldots, j_q \in \{1, \ldots, n\}$ und eine Funktion f von $\{0,1\}^q$ nach $\{0,1\}$ existieren, sodass für alle $u \in \{0,1\}^n$ gilt: $\varphi_i(u) = f(u_{j_1}, \ldots, u_{j_q})$.
Eine *Belegung* $u \in \{0,1\}^n$ *erfüllt* constraint φ_i, wenn $\varphi_i(u) = 1$.
Der Anteil der durch die Belegung u erfüllten constraints ist $(\sum_{i=1}^{m} \varphi_i(u))/m$ und $val(\varphi)$ bezeichne das Maximum dieses Wertes für alle $u \in \{0,1\}^n$. Wenn $val(\varphi) = 1$, heißt φ *erfüllbar*.

GAP CSP

Definition

Seien $q \in \mathbb{N}$ und $\rho \in (0,1)$. Das Problem ρ-GAPqCSP besteht darin, zu einer gegebenen qCSP-Instanz φ zu bestimmen, ob (1) $val(\varphi) = 1$ oder (2) $val(\varphi) < \rho$. Fall (1) werde als JA-Instanz von ρ-GAPqCSP bezeichnet, Fall (2) als NEIN-Instanz.

GAP CSP

Definition

Seien $q \in \mathbb{N}$ und $\rho \in (0,1)$. Das Problem ρ-GAPqCSP besteht darin, zu einer gegebenen qCSP-Instanz φ zu bestimmen, ob (1) $val(\varphi) = 1$ oder (2) $val(\varphi) < \rho$. Fall (1) werde als JA-Instanz von ρ-GAPqCSP bezeichnet, Fall (2) als NEIN-Instanz.
Das Problem ρ-GAPqCSP wird NP-schwer genannt, wenn für jede Sprache $L \in$ NP eine in polynomieller Zeit berechenbare Funktion f existiert, die Wörter auf (Repräsentationen von) qCSP-Instanzen abbildet, sodass gilt:

1. Vollständigkeit: $x \in L \Rightarrow val(f(x)) = 1$
2. Korrektheit: $x \notin L \Rightarrow val(f(x)) < \rho$

Ein Hilfs-Theorem

Theorem 3:
Es existieren $q \in \mathbb{N}$ und $\rho \in (0,1)$, sodass ρ-GAPqCSP NP-schwer ist.

Die drei Theoreme im Überblick

- Theorem 1:
  NP $=$ PCP$(\log n, 1)$
- Theorem 3:
  Es existieren $q \in \mathbb{N}$ und $\rho \in (0,1)$, sodass ρ-GAPqCSP NP-schwer ist.
- Theorem 2:
  Es gibt ein $\rho < 1$, sodass für alle $L \in$ NP eine in polynomieller Zeit berechenbare Funktion f existiert, die Wörter auf (Repräsentationen von) Formeln in 3KNF abbildet, sodass gilt:
  1. $x \in L \Rightarrow val(f(x)) = 1$
  2. $x \notin L \Rightarrow val(f(x)) < \rho$

Äquivalenzbeweis siehe Tafel

Zusammenfassung

- Reicht es, einen Beweis stichprobenartig zu lesen, um Fehler mit hoher Wahrscheinlichkeit zu finden?
  überraschenderweise JA
- Kann man selbst dann effizient beliebig genaue Näherungslösungen für NP-schwere Optimierungsprobleme berechnen, wenn P $\neq$ NP?
  leider NEIN

Vielen Dank für die Aufmerksamkeit.

Gibt es noch Fragen?