PCP & Hardness of Approximation

Vasilis Margonis

Advanced topics in Algorithms

April 25, 2018
Outline

1. Introduction

2. The **PCP** Theorem, a new characterization of **NP**

3. The Hardness of Approximation View

4. An optimal inapproximability result for **MAX-3SAT**

5. Inapproximability results for other known problems

6. Appendix: Derandomization via Conditional Expectations
Outline

1. Introduction

2. The PCP Theorem, a new characterization of $\text{NP}$

3. The Hardness of Approximation View

4. An optimal inapproximability result for $\text{Max-3SAT}$

5. Inapproximability results for other known problems

6. Appendix: Derandomization via Conditional Expectations
Suppose a mathematician circulates a proof of an important result, say Riemann Hypothesis, fitting 10 thousand pages.

To verify it would take us several years, going through all of those pages.

Weird question: Can we do better than that? (e.g. ignore most part of the proof)

Even weirder answer: Yes, according to the PCP theorem.
The idea behind **PCP**

So, the mathematician can rewrite his proof in a certain format: the **PCP** format, so we can verify it by probabilistically selecting a constant number of bits to examine it. Furthermore, this verification has the following properties:

1. A correct proof will always convince us.
2. A false proof will convince us with only negligible probability ($2^{-100}$ if we examine 300 bits).
The idea behind **PCP**

- In general, a mathematical proof is invalid if it has even a single error somewhere, which can be very difficult to detect.
- What the **PCP** theorem tells us is that there is a mechanical way to rewrite the proof so that the error is almost everywhere!

A nice analogue is the following:

![Initial Proof](image1)

**PCP transformation**

![PCP format](image2)
1. Introduction

2. The **PCP** Theorem, a new characterization of **NP**

3. The Hardness of Approximation View

4. An optimal inapproximability result for **MAX-3SAT**

5. Inapproximability results for other known problems

6. Appendix: Derandomization via Conditional Expectations
Standard definitions of \( \textbf{NP} \)

From now on, we shall refer to languages \( L \subseteq \{0, 1\}^* \).

**Definition (Classic definition)**

\[
\textbf{NP} = \bigcup_{c \in \mathbb{N}} \text{NTIME}(n^c)
\]

**Definition (YES-certificate definition)**

A language \( L \) is in \( \textbf{NP} \) if there exists a polynomial \( p \) and a polynomial-time TM \( V \) (called verifier) such that, given an input \( x \), verifies certificates (proofs), denoted \( \pi \):

\[
\begin{align*}
x \in L & \Rightarrow \exists \pi \in \{0, 1\}^{p(|x|)} : V(x, \pi) = 1 \\
x \notin L & \Rightarrow \forall \pi \in \{0, 1\}^{p(|x|)} : V(x, \pi) = 0
\end{align*}
\]

If \( V(x, \pi) = 1 \), then we call \( \pi \) a correct proof for \( x \).
Towards a new characterization of \textbf{NP}

\textbf{Definition (PCP verifier)}

Let $L$ be a language and $r, q : \mathbb{N} \rightarrow \mathbb{N}$. We say that $L$ has an $[r(n), q(n)]$-\textbf{PCP} verifier if there is a polynomial-time TM $V$ such that:

On input $x \in \{0, 1\}^n$ and a string $\pi \in \{0, 1\}^*$, $V$ uses at most $r(n)$ random coins and makes at most $q(n)$ non-adaptive queries to locations of $\pi$, satisfying

- **Completeness:** $x \in L \Rightarrow \exists \pi \in \{0, 1\}^* : \Pr[V(x, \pi) = 1] = 1$.
- **Soundness:** $x \notin L \Rightarrow \forall \pi \in \{0, 1\}^* : \Pr[V(x, \pi) = 1] \leq \frac{1}{2}$.

We say that $L \in \textbf{PCP}[r(n), q(n)]$, if $L$ has a $[r(n), q(n)]$-\textbf{PCP} verifier.
Notes:

1. Proofs checkable by an \([r, q]-\text{PCP}\) verifier are of length at most \(q2^r\). The verifier looks at only \(q\) bits of the proof for any particular choice of its random coins, and there are only \(2^r\) such choices.

2. The constant \(1/2\) in the soundness condition is arbitrary, in the sense that we can execute the verifier multiple times to make the constant as small as we want.
By the definitions of $P$ and $NP$:

- $P = PCP[0, 0]$
- $NP = PCP[0, poly(n)]$

Surprisingly...

**Theorem (Arora, Safra, Lund, Motwani, Sudan, Szegedy)**

$$NP = PCP[O(\log n), O(1)]$$
Lemma

\[ \text{PCP}[O(\log n), O(1)] \subseteq \text{NP} \]

Proof.

An \([O(\log n), O(1)]\)-\text{PCP} verifier can check proofs of length at most

\[ 2^{O(\log n)} O(1) = O(n^c). \]

Hence, a nondeterministic machine could “guess” the proof in \(O(n^c)\) time, and verify it deterministically by running the verifier for all \(2^{O(\log n)} = n^c\) possible outcomes of its random coin tosses. If the verifier accepts for all these possible coin tosses then the nondeterministic machine accepts.

Let \(p(n)\) be the running time of the verifier. Then,

\[ \text{PCP}[O(\log n), O(1)] \subseteq \text{NTIME}[O(n^c) + n^c \cdot p(n)] \subseteq \text{NP}. \]
Proof of the **PCP** theorem - hard direction

**Lemma**

\[ \text{NP} \subseteq \text{PCP}[O(\log n), O(1)] \]

The original proof is very extensive and outside the scope of this presentation. However, Irit Dinur gave a significantly simpler (but still hard) proof in 2007.
1. Introduction

2. The 
   PCP Theorem, a new characterization of NP

3. The Hardness of Approximation View

4. An optimal inapproximability result for MAX-3SAT

5. Inapproximability results for other known problems

6. Appendix: Derandomization via Conditional Expectations
Motivation: Approximate solutions to NP-hard problems

• Since the discovery of NP-completeness in 1972, researchers tried to efficiently compute good approximate solutions to NP-hard optimization problems.

• After failing to design good approximation algorithms for some problems, they tried to give inapproximability results, but this effort also stalled.

• Researchers slowly began to realize that classic Cook/Karp style reductions do not suffice for proving limits on approximation algorithms. (apart from few isolated successes)

• The PCP Theorem, not only gave a new characterization of NP, but also provided a new type of reductions suitable for proving hardness of approximation, the gap-producing reductions.
Case Study: \textbf{Max-3Sat}

\textbf{Input:} A 3CNF formula $\phi$, with $n$ variables and $m$ clauses.

\text{e.g. } \phi = (x_1 \lor \bar{x}_2 \lor x_4) \land \ldots \land (x_2 \lor \bar{x}_3 \lor \bar{x}_n)

\textbf{Goal:} Find an assignment that satisfies as many clauses as possible.

\textbf{Definition}

\begin{itemize}
  \item $\text{val}(\phi)$ denotes the maximum \textit{fraction} of clauses that can be satisfied by any assignment. For example, $\phi$ is satisfiable iff $\text{val}(\phi) = 1$.
  \item Let $\rho < 1$. An algorithm $A$ is an $\rho$-approximation algorithm for \textbf{Max-3Sat} if for every 3CNF formula $\phi$ with $m$ clauses,

  \[ \text{SOL}(A) \geq \rho \cdot \text{val}(\phi) \cdot m \cdot \frac{1}{\text{OPT}} \]
\end{itemize}
A simple randomized algorithm for \textsc{Max-3Sat}

Algorithm

For every variable $x_i$, set $x_i = 1$ with probability $\frac{1}{2}$, independently.

Claim

This is a $\frac{7}{8}$-approximation algorithm (in expectation).

Proof: We define the following random variable for every clause $C_j$

$$Y_j = \begin{cases} 
1, & \text{clause } j \text{ is satisfied} \\
0, & \text{otherwise} 
\end{cases}$$

Then, the number of clauses satisfied by the algorithm is

$$\text{SOL} = \sum_{j=1}^{m} Y_j$$
A simple randomized algorithm for $\text{MAX-3SAT}$

- For every clause $C_j$
  
  \[ \Pr[Y_j = 1] = 1 - \left(\frac{1}{2}\right)^3 = \frac{7}{8}. \]

- Hence,
  
  \[ \mathbb{E}[\text{SOL}] = \mathbb{E} \left[ \sum_{j=1}^{m} Y_j \right] = \sum_{j=1}^{m} \Pr[Y_j = 1] = \sum_{j=1}^{m} \frac{7}{8} = \frac{7}{8} m \geq \frac{7}{8} \text{OPT}. \]

**Remark**

*This algorithm can be derandomized via the method of conditional expectations. (See appendix section)*
Any hope for a PTAS or an FPTAS?

The PCP Theorem implies that the answer is NO (unless $P = NP$). The reason is that it is equivalent to the following theorem.

**Theorem (Gap-producing reduction)**

There exists $\rho < 1$ such that $\forall L \in \text{NP}$ there is a polynomial-time function $f$ mapping strings to 3CNF formulas such that:

\begin{align*}
    x \in L &\Rightarrow \text{val}(f(x)) = 1 \\
    x \notin L &\Rightarrow \text{val}(f(x)) < \rho
\end{align*}
**Max-3SAT is APX-hard**

---

**Corollary**

*There exists some constant $\rho < 1$ such that there is no polynomial-time $\rho$-approximation algorithm for Max-3SAT, unless $P = \text{NP}$.*

- Indeed, we can convert a $\rho$-approximation algorithm $A$ for Max-3SAT into an algorithm deciding $L$.
- We apply the reduction $f$ on $x$ and then run the approximation algorithm to the resultant 3CNF formula $f(x)$.
- (1) and (2) together imply that $x \in L$ iff $A(f(x))$ returns an assignment that satisfies at least a $\rho$ fraction of $f(x)$’s clauses.
Outline

1. Introduction

2. The **PCP** Theorem, a new characterization of **NP**

3. The Hardness of Approximation View

4. An optimal inapproximability result for **Max-3SAT**

5. Inapproximability results for other known problems

6. Appendix: Derandomization via Conditional Expectations
Recap

- There is a deterministic $\frac{7}{8}$-approximation algorithm.
- There exists a constant $\rho < 1$ such that there is no $\rho$-approximation unless $P = NP$.
- No hope for a PTAS.

**Question:** Can we do better than $7/8$?

**More important question:** what is the actual value of $\rho$?

Håstad answered both...
Theorem (Håstad, 1997)

\[ \text{NP} = \text{PCP}_{1-\varepsilon, \frac{1}{2}+\varepsilon}[O(\log n), 3], \ \forall \varepsilon > 0 \]

Moreover, the tests used by \( V \) are linear: Given a proof \( \pi \in \{0,1\}^m \), \( V \) chooses a random triple \((i,j,k)\) and a bit \( b_{ijk} \in \{0,1\} \) according to some distribution and accepts iff \( \pi_i \oplus \pi_j \oplus \pi_k = b_{ijk} \).

- **Completeness:**

  \[ x \in L \Rightarrow \exists \pi \in \{0,1\}^m : \Pr_{(i,j,k) \in [m]^3} \left[ \pi_i \oplus \pi_j \oplus \pi_k = b_{ijk} \right] \geq 1 - \varepsilon \]

- **Soundness:**

  \[ x \notin L \Rightarrow \forall \pi \in \{0,1\}^m : \Pr_{(i,j,k) \in [m]^3} \left[ \pi_i \oplus \pi_j \oplus \pi_k = b_{ijk} \right] \leq \frac{1}{2} + \varepsilon \]
3-bit **PCP** and **Max-E3LIN**

- We can convert the computation of Håstad’s 3-bit **PCP** into an instance of a problem called **Max-E3LIN**, as follows:

  \[
  \pi_1 \oplus \pi_1 \oplus \pi_1 = b^{111} \\
  \pi_1 \oplus \pi_1 \oplus \pi_2 = b^{112} \\
  \vdots \\
  \pi_m \oplus \pi_m \oplus \pi_m = b^{mmm}
  \]

  - If \( x \in L \), \( \exists \pi = (\pi_1, \ldots, \pi_m) \) that satisfies all constraints.
  - If \( x \notin L \), \( \forall \pi = (\pi_1, \ldots, \pi_m) \) at most a \((1/2 + \varepsilon)\) fraction of constraints can be satisfied.

**Corollary**

*Håstad’s Theorem implies that there is no \((1/2 + \varepsilon)\)-approximation for **Max-E3LIN**, for every \( \varepsilon > 0 \).*

- This is a tight result! The problem has a simple \(1/2\)-approximation algorithm.
Hardness of approximating \textsc{Max-3SAT}

**Corollary**

For every $\varepsilon > 0$, $(7/8 + \varepsilon)$-approximation to \textsc{Max-3SAT} is \textbf{NP-hard}.

**Proof:** Take an instance of \textsc{Max-E3LIN} with $n$ variables and $m$ constraints, where we want to determine whether at least a $(1 - \nu)$ or at most a $(1/2 + \nu)$ fraction of constraints can be satisfied. We construct an instance of \textsc{Max-3SAT} with $4m$ clauses and $n$ variables:

- $E_1 : x_1 \oplus x_2 \oplus x_3 = 0$
- $E_2 : x_4 \oplus x_1 \oplus x_7 = 1$
- $\vdots$
- $E_m : x_5 \oplus x_n \oplus x_9 = 1$

$$E_1 : \begin{cases} (\overline{x}_1 \lor x_2 \lor x_3), (x_1 \lor \overline{x}_2 \lor x_3), \\ (x_1 \lor x_2 \lor \overline{x}_3), (\overline{x}_1 \lor \overline{x}_2 \lor \overline{x}_3). \end{cases}$$

$$E_m : \begin{cases} (\overline{x}_5 \lor \overline{x}_n \lor x_9), (x_5 \lor \overline{x}_n \lor \overline{x}_9), \\ (\overline{x}_5 \lor x_n \lor \overline{x}_9), (x_5 \lor x_n \lor x_9). \end{cases}$$
Hardness of approximating Max-3SAT

• If $x_i, x_j, x_k$ satisfy a linear constraint, then they satisfy all four corresponding clauses. Otherwise, they satisfy exactly three clauses.

• **Completeness:** If at least a $(1 - \nu)$ fraction of constraints can be satisfied, then the fraction of clauses that can be satisfied is at least

$$
(1 - \nu) \cdot \frac{4}{4} + \nu \cdot \frac{3}{4} = \left(1 - \frac{\nu}{4}\right)
$$

• **Soundness:** If at most a $(1/2 + \nu)$ fraction of constraints can be satisfied, then the fraction of clauses that can be satisfied is at most

$$
\left(\frac{1}{2} + \nu\right) \cdot \frac{4}{4} + \left(\frac{1}{2} - \nu\right) \cdot \frac{3}{4} = \left(\frac{7}{8} + \frac{\nu}{4}\right) = \rho
$$

• Therefore, it is **NP**-hard to approximate Max-3SAT within a factor better than $\rho = (7/8 + \nu/4) = (7/8 + \varepsilon)$.
Outline

1. Introduction

2. The PCP Theorem, a new characterization of \textbf{NP}

3. The Hardness of Approximation View

4. An optimal inapproximability result for \textsc{Max-3SAT}

5. Inapproximability results for other known problems

6. Appendix: Derandomization via Conditional Expectations
**Vertex Cover & Independent Set**

**Vertex Cover:**
- Current best: \(2 - \Theta \left(1/\sqrt{\log |V|}\right)\)-apx. [Karakostas, 2009]
- **NP**-hard to approximate within a factor of 1.3606. [Dinur & Safra, 2005]
- If UGC is true, **Vertex Cover** cannot be approximated within any constant factor better than 2. [Khot & Regev, 2008]

**Independent Set:**
- Trivial \((1/n)\)-approximation: return any vertex of the graph.
- For every \(\varepsilon > 0\) there is no \((1/n^{1-\varepsilon})\)-approximation algorithm. [Zuckerman, 2007]
- No \(2^{O(\sqrt{\log d})/d}\)-approximation algorithm exists, where \(d\) is the graph’s maximum degree. [Trevisan, 2001]
Max-Cut:

- **NP-hard** to approximate with a ratio better than $16/17 \approx 0.941$. [Håstad, 2001]
- Using semidefinite programming, there is an approximation algorithm with a ratio of $\alpha \approx 0.878$. [Goemans & Williamson, 1995]
- If UGC is true, this is the best possible approximation ratio for **Max Cut**. [Khot et al., 2007]

Metric TSP:

- The best known approximation ratio is $3/2$. [Christofides, 1976]
- $5500$-apx for asymmetric distances. [Svensson, Tarnawski & Végn, 2017]
- There is no polynomial time algorithm for Metric TSP with performance ratio better that $123/122$ (and $75/74$ for asymmetric distances). [Karpinski, Lampis & Schmied, 2013]
Colorability

- For every $\varepsilon > 0$, there is no $n^{1-\varepsilon}$-approximation algorithm. [Zuckerman, 2007]

An interesting special case of the problem is to devise algorithms that color a 3-colorable graph with a minimum number of colors.

- There is a polynomial time algorithm that colors every 3-colorable graph with at most $\tilde{O}(n^{3/14} \approx 0.214)$ colors. [Karger & Blum, 1997]

- There is no polynomial time algorithm that colors every 3-colorable graph using at most 4 colors. [Khanna, Linial & Safra, 1993]

This is one of the largest gaps between known approximation algorithms and known inapproximability results.
1. Introduction

2. The **PCP** Theorem, a new characterization of **NP**

3. The Hardness of Approximation View

4. An optimal inapproximability result for **MAX-3SAT**

5. Inapproximability results for other known problems

6. Appendix: Derandomization via Conditional Expectations
Deterministic choices

We saw the following randomized algorithm for Max-3SAT:

Algorithm

For every variable $x_i$, set $x_i = 1$ with probability $\frac{1}{2}$, independently.

and we proved that

$$
E[SOL] = E\left[\sum_{j=1}^{m} Y_j\right] = \sum_{j=1}^{m} Pr[Y_j = 1] = \sum_{j=1}^{m} \frac{7}{8} = \frac{7}{8}m \geq \frac{7}{8}\text{OPT}.
$$

Idea: What if we set $x_1 = b_1 \in \{0, 1\}$ deterministically and all others 1 with probability $\frac{1}{2}$?

Then, the number of clauses satisfied would be $E[SOL|x_1 = b_1]$. 
Maximization of conditional expectations

- We have

\[ E[\text{SOL}] = E[\text{SOL}|x_1 = 0] \cdot \Pr[x_1 = 0] + E[\text{SOL}|x_1 = 1] \cdot \Pr[x_1 = 1] \]

\[ = \frac{1}{2} \left( E[\text{SOL}|x_1 = 0] + E[\text{SOL}|x_1 = 1] \right). \]

- So \( E[\text{SOL}] \) is a convex combination of \( E_0 \) and \( E_1 \). Hence,

\[ \max_{b_1 \in \{0,1\}} E[\text{SOL}|x_1 = b_1] \geq E[\text{SOL}] = \frac{7}{8} m. \]

- Suppose that we set \( x_1 = b_1 \in \{0,1\} \) so as to maximize the conditional expectation \( E[\text{SOL}|x_1 = b_1] \), and for the rest variables we continue with probability \( 1/2 \). Then, in expectation, we satisfy at least as many clauses as the full-randomized algorithm!
Do the same for all variables, sticking with every choice along the way.

Algorithm 1 \textbf{De-randomized}(\phi, n, m)

1: \textbf{for} i=1:n \textbf{do}
2: \hspace{1em} E_0 \leftarrow \mathbb{E}[\text{SOL} \mid x_1 = b_1, \ldots, x_{i-1} = b_{i-1}, x_i = 0];
3: \hspace{1em} E_1 \leftarrow \mathbb{E}[\text{SOL} \mid x_1 = b_1, \ldots, x_{i-1} = b_{i-1}, x_i = 1];
4: \hspace{1em} \textbf{if} \ E_0 \geq E_1 \textbf{ then}
5: \hspace{2em} b_i \leftarrow 0;
6: \hspace{1em} \textbf{else}
7: \hspace{2em} b_i \leftarrow 1;
8: \hspace{1em} \textbf{end if}
9: \textbf{end for}
10: \textbf{Output}: Assign $b_i$ to variable $x_i$. 
• In every iteration \( i \in [n] \), we choose \( b_i \) so as to maximize the conditional expectation

\[
E[SOL|x_1 = b_1, \ldots, x_i = b_i].
\]

• We already proved that

\[
E[SOL|x_1 = b_1] \geq E[SOL] = \frac{7}{8} m.
\]

• Using the same argument, for every \( i \in \{2, \ldots, n\} \)

\[
E[SOL|x_1 = b_1, \ldots, x_i = b_i] \geq E[SOL|x_1 = b_1, \ldots, x_{i-1} = b_{i-1}].
\]

• Then, by induction:

\[
E[SOL|x_1 = b_1, \ldots, x_n = b_n] \geq E[SOL|x_1 = b_1, \ldots, x_{n-1} = b_{n-1}]
\]

\[
\vdots
\]

\[
\geq E[SOL|x_1 = b_1] \geq E[SOL] = \frac{7}{8} m.
\]
Analysis

- \( \mathbb{E}[\text{SOL}|x_1 = b_1, \ldots, x_n = b_n] \) is the number of clauses satisfied by the algorithm. Thus, the derandomized algorithm is a deterministic 7/8-approximation.

- We must show that in each iteration \( i \) the conditional expectation can be computed in polynomial time.

- We have that

\[
\mathbb{E}[\text{SOL}|x_1 = b_1, \ldots, x_i = b_i] = \sum_{j=1}^{m} \Pr[Y_j = 1|x_1 = b_1, \ldots, x_i = b_i].
\]

- Let \( C_j = (l_1 \lor l_2 \lor l_3) \), then

\[
P_j = \begin{cases} 
1, & \text{one literal already true} \\
0, & \text{all literals already false} \\
3/4, & \text{one literal already false} \\
1/2, & \text{two literals already false}
\end{cases}
\]

- We compute \( P_j \) in \( O(1) \) time. So, naively, the running time of the algorithm is \( O(n \cdot m) \).
Chapter 11 and Section 22.4 of

Sanjeev Arora, Boaz Barak

Sections 5.1-5.2 of

D. P. Williamson, D. B. Shmoys
The Design of Approximation Algorithms.