PCP and Hardness of Approximation

Aman Bansal Adwait Godbole

October 2019

Trajectory

1. Relaxations of Hard Problems
   - Approximate Problems
   - Gap Problem
   - Connection between Approximation and Gap
2. A New Proof System
   - Probabilistically Checkable Proof Systems
   - Some Preliminaries
   - Proof of the PCP Theorem
3. Hardness of Approximation

Trajectory

Approximate Problems

Given an instance $\mathbf{x} \in \mathcal{X}$ of an NP-hard optimization problem with objective function $Obj : \mathcal{Y} \to \mathbb{R}$ (which is to be maximized[1]), a solution $\mathbf{y}$ is said to be an α-*approximate* (for $\alpha \leq 1$) solution to the instance if

$$\alpha \cdot Obj(\mathbf{y}^*) \leq Obj(\mathbf{y}) \leq Obj(\mathbf{y}^*)$$

where $\mathbf{y}^*$ is the true (not approximated) solution to the problem instance.

[1] If the problem is a minimization problem then we have a slightly different definition

Approximate Problems

α-approximate algorithms

An algorithm **A** is an α-approximate algorithm if for all **x** in the instance space $\mathcal{X}$, it returns an α-approximate solution **y**.

α-approximate problems

Problems that render (*poly-time*) α-approximate algorithms are called α-approximate problems.

Trajectory

1. Relaxations of Hard Problems
   - Approximate Problems
   - Gap Problem
   - Connection between Approximation and Gap
2. A New Proof System
   - Probabilistically Checkable Proof Systems
   - Some Preliminaries
   - Proof of the PCP Theorem
3. Hardness of Approximation

Promise Problem

A *promise* problem Π is specified by a pair of sets (YES, NO) such that YES, NO $\subseteq \mathcal{X}$ and YES $\cap$ NO $= \Phi$.

Any algorithm **A** solving Π, on input $\mathbf{x}$, should output 'yes' if $\mathbf{x} \in$ YES, 'no' if $\mathbf{x} \in$ NO and any output if $\mathbf{x}$ is a don't care instance

Recollect the PromiseΠ problem from the midsem.

Gap Problems

A gap problem is a promise problem parametrized by α (< 1). Let P be an NP-hard optimization problem with objective function $Obj : \mathcal{Y} \to \mathbb{R}$ (which is to be maximized), the corresponding gap problem gap_α-P is a promise problem with (YES, NO) sets as given below:

$$YES = \{\langle \mathbf{x}, k\rangle \mid \exists \mathbf{y} \in \mathcal{Y} \textit{ such that } Obj(\mathbf{y}) \geq k\}$$
$$NO = \{\langle \mathbf{x}, k\rangle \mid \forall \mathbf{y} \in \mathcal{Y}, Obj(\mathbf{y}) < \alpha k\}$$

> Intuitively, the 'gap' refers to the interval $(\alpha k, k)$.

Trajectory

1. Relaxations of Hard Problems
   - Approximate Problems
   - Gap Problem
   - Connection between Approximation and Gap
2. A New Proof System
   - Probabilistically Checkable Proof Systems
   - Some Preliminaries
   - Proof of the PCP Theorem
3. Hardness of Approximation

Intuition

Intuitively, it seems that approximate problems and gap problems are similar.

α-approximation is a relaxed variant of a search problem

gap$_\alpha$ is a relaxed variant of a decision problem

Indeed, this notion can be formalized.

α-approximate $\rightarrow$ *gap*$_\alpha$

Connnection between α-approximation and *gap*$_\alpha$

For any problem P and $0 < \alpha < 1$, α-approximating P is at least as hard as solving *gap*$_\alpha$-P.

Proof:
Let A be an α-approximate algorithm for P. The following algorithm solves *gap*$_\alpha$-P:

On input (ϕ, k):

1. Let $k' = A(\phi)$
2. Accept iff $k' \geq \alpha k$

Trajectory

Trajectory

Probabilistically Checkable Proof System

(r, q, m, t)-restricted Verifier

Let $r, q, m, t : \mathbb{N} \to \mathbb{N}$. A language L $\in PCP_{c,s}[r, q, m, t]$ if L has an (r, q, m, t) restricted verifier V such that

$$\forall x \in L, \exists \pi \ of \ size \ at \ most \ m(|x|), Pr_R[V^\pi[x; R] = acc] \geq c(|x|)$$
$$\forall x \notin L, \forall \pi \ of \ size \ at \ most \ m(|x|), Pr_R[V^\pi[x; R] = acc] < s(|x|)$$

Resource bounds

1. $r(|x|)$ is a bound on randomness used by V
2. $q(|x|)$ is a bound on the number of locations queried by V
3. $m(|x|)$ is a bound the length of the proof to which V has oracle access
4. $t(|x|)$ is a bound on the runtime of V

Completeness and Soundness constraints

$c(|x|)$ and $s(|x|)$ are the completeness and soundness specifications

Remarks

- The verifier could be adaptive or non-adaptive
- If the verifier is non-adaptive then $m(n) \leq q(n) \cdot 2^{r(n)}$
- $q(n) \leq t(n)$
- $PCP_{c,s}[r, q] \subseteq NTIME(q(n) \cdot 2^{r(n)})$
- $NP = PCP_{1,0}[0, poly(n)]$
- $BPP = PCP_{\frac{2}{3},\frac{1}{3}}[poly(n), 0]$

PCP Theorem

PCP Theorem

$NP = PCP_{1,\frac{1}{2}}[log(n), 1]$

PCP Theorem

PCP Theorem

$NP = PCP_{1,\frac{1}{2}}[log(n), 1]$

We will today present a weaker version of the PCP theorem.

PCP Theorem [Weaker]

$NP = PCP_{1,\frac{1}{2}}[poly(n), 1]$

Trajectory

1. Relaxations of Hard Problems
   - Approximate Problems
   - Gap Problem
   - Connection between Approximation and Gap
2. A New Proof System
   - Probabilistically Checkable Proof Systems
   - Some Preliminaries
   - Proof of the PCP Theorem
3. Hardness of Approximation

Subset XOR

Consider the function $f_{\mathbf{u}}(\mathbf{x}) = \mathbf{x} \odot \mathbf{u}$.
Here for $\mathbf{x}, \mathbf{y} \in \{0,1\}^n, \mathbf{x} \odot \mathbf{y} = \sum_i \mathbf{x}_i \mathbf{y}_i \ (mod\ 2)$.

Subset XOR

Consider the function $f_{\mathbf{u}}(\mathbf{x}) = \mathbf{x} \odot \mathbf{u}$.
Here for $\mathbf{x}, \mathbf{y} \in \{0,1\}^n, \mathbf{x} \odot \mathbf{y} = \sum_i \mathbf{x}_i \mathbf{y}_i \ (mod\ 2)$.

Note that $f_{\mathbf{u}}$ is equivalent to choosing a subset (given by $\mathbf{u}$) of elements from $[n]$ and evaluating parity over this subset.

Subset XOR

Consider the function $f_{\mathbf{u}}(\mathbf{x}) = \mathbf{x} \odot \mathbf{u}$.
Here for $\mathbf{x}, \mathbf{y} \in \{0,1\}^n, \mathbf{x} \odot \mathbf{y} = \sum_i \mathbf{x}_i \mathbf{y}_i \ (mod\ 2)$.

Note that $f_{\mathbf{u}}$ is equivalent to choosing a subset (given by $\mathbf{u}$) of elements from $[n]$ and evaluating parity over this subset.

Random Subsum Principle

For $\mathbf{x}, \mathbf{y} \in \{0,1\}^n$ with $\mathbf{y} \neq 0^n$, $Pr_{\mathbf{x} \in \{0,1\}^n}[\mathbf{x} \odot \mathbf{y} = 1] = \frac{1}{2}$

Walsh-Hadamard Code

Main Idea: Bit strings $\mathbf{u} \in \{0, 1\}^n$ are encoded as the truth table of a *linear* function over $\mathbf{F}_2$

Walsh-Hadamard Code

Main Idea: Bit strings $\mathbf{u} \in \{0,1\}^n$ are encoded as the truth table of a *linear* function over $\mathbf{F}_2$

WH encoding

For an n-bit string $\mathbf{u} \in \{0,1\}^n$, $\mathrm{WH}(\mathbf{u})$ is the 2^n bit string representing the truth table of the function $f(\mathbf{x}) = \mathbf{x} \odot \mathbf{u}$ for $\mathbf{x} \in \{0,1\}^n$.

Walsh-Hadamard Code

Main Idea: Bit strings $\mathbf{u} \in \{0,1\}^n$ are encoded as the truth table of a *linear* function over $\mathbf{F}_2$

WH encoding

For an n-bit string $\mathbf{u} \in \{0,1\}^n$, $\mathrm{WH}(\mathbf{u})$ is the 2^n bit string representing the truth table of the function $f(\mathbf{x}) = \mathbf{x} \odot \mathbf{u}$ for $\mathbf{x} \in \{0,1\}^n$.

Walsh-Hadamard codeword

$f \in \{0,1\}^{2^n}$ such that $f = \mathrm{WH}(\mathbf{u})$ for some $\mathbf{u} \in \{0,1\}^n$.

Properties of WH

EC *Error correcting* with minimum distance $\frac{1}{2}$.
This means that for $\mathbf{x}, \mathbf{y} \in_R \{0,1\}^n$ with $\mathbf{x} \neq \mathbf{y}$, $\mathrm{WH}(\mathbf{x})$ and $\mathrm{WH}(\mathbf{y})$ differ in $1/2$ the bits.

Properties of WH

EC *Error correcting* with minimum distance $\frac{1}{2}$.
This means that for $\mathbf{x}, \mathbf{y} \in_R \{0,1\}^n$ with $\mathbf{x} \neq \mathbf{y}$, $\mathrm{WH}(\mathbf{x})$ and $\mathrm{WH}(\mathbf{y})$ differ in $1/2$ the bits.

LIN *Linearity* of $\mathrm{WH}(\mathbf{u})$
$f = \mathrm{WH}(\mathbf{u})$ when viewed as a function from $\{0,1\}^n$ to $\{0,1\}$ is in fact *linear* (over $\mathbf{F}_2$)

Properties of WH

EC *Error correcting* with minimum distance $\frac{1}{2}$.
This means that for $\mathbf{x}, \mathbf{y} \in_R \{0,1\}^n$ with $\mathbf{x} \neq \mathbf{y}$, $\mathrm{WH}(\mathbf{x})$ and $\mathrm{WH}(\mathbf{y})$ differ in $1/2$ the bits.

LIN *Linearity* of $\mathrm{WH}(\mathbf{u})$
$f = \mathrm{WH}(\mathbf{u})$ when viewed as a function from $\{0,1\}^n$ to $\{0,1\}$ is in fact *linear* (over $\mathbf{F}_2$)

LT *Locally Testable*
Given access to a function $f : \{0,1\}^n \to \{0,1\}$, we can check whether it is a *Walsh-Hadamard* code-word by querying a *constant* number of places.

Properties of WH

EC *Error correcting* with minimum distance $\frac{1}{2}$.
This means that for $\mathbf{x}, \mathbf{y} \in_R \{0,1\}^n$ with $\mathbf{x} \neq \mathbf{y}$, $\mathrm{WH}(\mathbf{x})$ and $\mathrm{WH}(\mathbf{y})$ differ in $1/2$ the bits.

LIN *Linearity* of $\mathrm{WH}(\mathbf{u})$
$f = \mathrm{WH}(\mathbf{u})$ when viewed as a function from $\{0,1\}^n$ to $\{0,1\}$ is in fact *linear* (over $\mathbf{F}_2$)

LT *Locally Testable*
Given access to a function $f : \{0,1\}^n \to \{0,1\}$, we can check whether it is a *Walsh-Hadamard* code-word by querying a *constant* number of places.

LD *Locally Decodable*
Given f and an $\mathbf{x} \in \{0,1\}^n$, we can find $\tilde{f}(\mathbf{x})$ in constant queries to f, where $\tilde{f}$ is the true codeword.

LT: Local Testability

LIN: $\mathrm{WH}(\mathbf{u})$ for $\mathbf{u} \in \{0,1\}^n$ captures *all* n-bit linear functions on $\mathbf{F}_2$.

LT: Local Testability

LIN: $\mathrm{WH}(\mathbf{u})$ for $\mathbf{u} \in \{0,1\}^n$ captures *all* n-bit linear functions on $\mathbf{F}_2$.

ρ-closeness of functions

Functions f, g are ρ-close if $Pr_{\mathbf{x} \in_R \{0,1\}^n}[f(\mathbf{x}) = g(\mathbf{x})] \geq \rho$.

A function f is ρ-close to a linear function if there exists a linear function g such that f and g are ρ-close

LIN: $\mathrm{WH}(\mathbf{u})$ for $\mathbf{u} \in \{0,1\}^n$ captures *all* n-bit linear functions on $\mathbf{F}_2$.

ρ-closeness of functions

Functions f, g are ρ-close if $Pr_{\mathbf{x} \in_R \{0,1\}^n}[f(\mathbf{x}) = g(\mathbf{x})] \geq \rho$.

A function f is ρ-close to a linear function if there exists a linear function g such that f and g are ρ-close

Let f be such that

$$Pr_{\mathbf{x},\mathbf{y} \in_R \{0,1\}^n}[f(\mathbf{x}+\mathbf{y}) = f(\mathbf{x}) + f(\mathbf{y})] \geq \rho$$

for some $\rho > \frac{1}{2}$. Then f is ρ-close to a linear function.

LD: Local Decodability

EC: For $\mathbf{x} \neq \mathbf{y}$, $\mathrm{WH}(\mathbf{x})$ and $\mathrm{WH}(\mathbf{y})$ differ in $1/2$ the bits

LD: Local Decodability

EC: For $\mathbf{x} \neq \mathbf{y}$, $\mathrm{WH}(\mathbf{x})$ and $\mathrm{WH}(\mathbf{y})$ differ in $1/2$ the bits

Let f be $(1-\delta)$-close to a linear function $\tilde{f}$ for some $\delta < 1/4$. Then by EC, f uniquely determines $\tilde{f}$.

LD: Local Decodability

EC: For $\mathbf{x} \neq \mathbf{y}$, $\mathrm{WH}(\mathbf{x})$ and $\mathrm{WH}(\mathbf{y})$ differ in $1/2$ the bits

Let f be $(1-\delta)$-close to a linear function $\tilde{f}$ for some $\delta < 1/4$. Then by EC, f uniquely determines $\tilde{f}$.

So, given a (possibly illegal) f, having a corresponding $\tilde{f}$, we want to find $\tilde{f}(\mathbf{x})$. Here we have oracle access only to f. The idea is to once again use randomness.

LD: Local Decodability

Objective: With oracle access only to f, given an $\mathbf{x} \in \{0,1\}^n$, find $\tilde{f}(\mathbf{x})$.
The idea is to use randomness and linearity.

LD: Local Decodability

Objective: With oracle access only to f, given an $\mathbf{x} \in \{0,1\}^n$, find $\tilde{f}(\mathbf{x})$. The idea is to use randomness and linearity.

- Choose $\mathbf{x}' \in_R \{0,1\}^n$
- Set $\mathbf{x}'' \leftarrow \mathbf{x} + \mathbf{x}'$
- Let $\mathbf{y}' = f(\mathbf{x}')$ and $\mathbf{y}'' = f(\mathbf{x}'')$
- Output $\mathbf{y}' + \mathbf{y}''$

LD: Local Decodability

Objective: With oracle access only to f, given an $\mathbf{x} \in \{0,1\}^n$, find $\tilde{f}(\mathbf{x})$.
The idea is to use randomness and linearity.

- Choose $\mathbf{x}' \in_R \{0,1\}^n$
- Set $\mathbf{x}'' \leftarrow \mathbf{x} + \mathbf{x}'$
- Let $\mathbf{y}' = f(\mathbf{x}')$ and $\mathbf{y}'' = f(\mathbf{x}'')$
- Output $\mathbf{y}' + \mathbf{y}''$

With probability atleast $1 - 2\delta$ we have $\mathbf{y}' = f(\mathbf{x}')$ and $\mathbf{y}'' = f(\mathbf{x}'')$ and hence $\tilde{f}(\mathbf{x}) = \mathbf{y}' + \mathbf{y}''$.

Trajectory

Proof of the PCP Theorem [Weaker]

PCP Theorem [Weaker]

$NP = PCP_{1,\frac{1}{2}}[poly(n), 1]$

Proof of the PCP Theorem [Weaker]

PCP Theorem [Weaker]

$NP = PCP_{1,\frac{1}{2}}[poly(n), 1]$

We show that QUADEQ - the satisfiability problem for quadratic equations over $\mathbf{F}_2$ - has a $PCP[poly(n), 1]$ proof system

Proof of the PCP Theorem [Weaker]

PCP Theorem [Weaker]

$NP = PCP_{1,\frac{1}{2}}[poly(n), 1]$

We show that QUADEQ - the satisfiability problem for quadratic equations over $\mathbf{F}_2$ - has a $PCP[poly(n), 1]$ proof system

QUADEQ over variables $u_1, u_2, \cdots, u_n$ is is of the form $AU = b$, where A is an $m \times n^2$ matrix and $b \in \{0, 1\}^m$. $U = \mathbf{u} \otimes \mathbf{u}$ is the tensor product (or the Hadamard product).

Proof of the PCP Theorem [Weaker]

PCP Theorem [Weaker]

$NP = PCP_{1,\frac{1}{2}}[poly(n), 1]$

We show that QUADEQ - the satisfiability problem for quadratic equations over $\mathbf{F}_2$ - has a $PCP[poly(n), 1]$ proof system

QUADEQ over variables $u_1, u_2, \cdots, u_n$ is is of the form $AU = b$, where A is an $m \times n^2$ matrix and $b \in \{0,1\}^m$. $U = \mathbf{u} \otimes \mathbf{u}$ is the tensor product (or the Hadamard product).

Claim

QUADEQ, the language of all satisfiable instances is NP-complete

π and $\mathcal{V}$

What is the proof π and what does the verifier $\mathcal{V}$ do?

π and $\mathcal{V}$

What is the proof π and what does the verifier $\mathcal{V}$ do?

π The proof is $\langle \mathrm{WH}(\mathbf{u}), \mathrm{WH}(\mathbf{u} \otimes \mathbf{u}) \rangle$

π and $\mathcal{V}$

What is the proof π and what does the verifier $\mathcal{V}$ do?

- π The proof is $\langle \mathrm{WH}(\mathbf{u}), \mathrm{WH}(\mathbf{u} \otimes \mathbf{u}) \rangle$
- $\mathcal{V}$ Denote the proof by $f = \mathrm{WH}(\mathbf{u})$ and $g = \mathrm{WH}(\mathbf{u} \otimes \mathbf{u})$.
  The verifier does the following
  1) Check linearity of f and g
  2) Verify that g encodes $\mathbf{u} \otimes \mathbf{u}$
  3) Verify that f encodes a satisfying assignment

Check linearity of f and g

Note that $f = \mathrm{WH}(\mathbf{u})$ and $g = \mathrm{WH}(\mathbf{u} \otimes \mathbf{u})$

Check linearity of f and g

> Note that $f = \mathrm{WH}(\mathbf{u})$ and $g = \mathrm{WH}(\mathbf{u} \otimes \mathbf{u})$

$\mathcal{V}$ performs a 0.99-close (high-probability) linearity test on both f and g. This is done by the LT property described earlier.

Check linearity of f and g

Note that $f = \mathrm{WH}(\mathbf{u})$ and $g = \mathrm{WH}(\mathbf{u} \otimes \mathbf{u})$

$\mathcal{V}$ performs a 0.99-close (high-probability) linearity test on both f and g. This is done by the LT property described earlier.

Note crucially that a high but nevertheless constant closeness suffices. This is since we eventually plan to query π at only a small constant number of points.

Verify that g encodes $\mathbf{u} \otimes \mathbf{u}$

$\mathcal{V}$ chooses $\mathbf{r}$, $\mathbf{r}'$ independently at random from $\{0,1\}^n$ and assert that $f(\mathbf{r})f(\mathbf{r}') = g(\mathbf{r} \otimes \mathbf{r}')$.

Verify that g encodes $\mathbf{u} \otimes \mathbf{u}$

$\mathcal{V}$ chooses $\mathbf{r}$, $\mathbf{r}'$ independently at random from $\{0,1\}^n$ and assert that $f(\mathbf{r})f(\mathbf{r}') = g(\mathbf{r} \otimes \mathbf{r}')$.

Let W be an $n \times n$ matrix representing the entries of $\mathbf{w}$ and U be such a matrix for $\mathbf{u} \otimes \mathbf{u}$. Then

1. $g(\mathbf{r} \oplus \mathbf{r}') = \mathbf{w} \odot (\mathbf{r} \otimes \mathbf{r}') = \mathbf{r}W\mathbf{r}'$
2. $f(\mathbf{r})f(\mathbf{r}') = (\mathbf{u} \odot \mathbf{r})(\mathbf{u} \odot \mathbf{r}') = \mathbf{r}U\mathbf{r}'$

Verify that g encodes $\mathbf{u} \otimes \mathbf{u}$

$\mathcal{V}$ chooses $\mathbf{r}$, $\mathbf{r}'$ independently at random from $\{0,1\}^n$ and assert that $f(\mathbf{r})f(\mathbf{r}') = g(\mathbf{r} \otimes \mathbf{r}')$.

Let W be an $n \times n$ matrix representing the entries of $\mathbf{w}$ and U be such a matrix for $\mathbf{u} \otimes \mathbf{u}$. Then

1. $g(\mathbf{r} \oplus \mathbf{r}') = \mathbf{w} \odot (\mathbf{r} \otimes \mathbf{r}') = \mathbf{r}W\mathbf{r}'$
2. $f(\mathbf{r})f(\mathbf{r}') = (\mathbf{u} \odot \mathbf{r})(\mathbf{u} \odot \mathbf{r}') = \mathbf{r}U\mathbf{r}'$

By the random subsum principle, we claim this test rejects atleast $1/4$ of the time on instances where $\mathbf{w} \neq \mathbf{u} \otimes \mathbf{u}$. Repeating this 3 times, we get probability of rejection as $37/64$.

Verify that f encodes a satisfying assignment

Now we are assured that the form of π is $\langle \mathrm{WH}(\mathbf{u}), \mathrm{WH}(\mathbf{u} \otimes \mathbf{u}) \rangle$ for some $\mathbf{u} \in \{0,1\}^n$.

Verify that f encodes a satisfying assignment

Now we are assured that the form of π is $\langle \mathrm{WH}(\mathbf{u}), \mathrm{WH}(\mathbf{u} \otimes \mathbf{u}) \rangle$ for some $\mathbf{u} \in \{0,1\}^n$.

1. All that remains is to check that $\mathbf{u}$ is a satisfying assignment
2. Wonderfully, we also have $O(1)$ access to $A_i \cdot (\mathbf{u} \otimes \mathbf{u})$, the value of the i^{th} equation in the QUADEQ instance and can match it with b_i.

Verify that f encodes a satisfying assignment

Now we are assured that the form of π is $\langle \mathrm{WH}(\mathbf{u}), \mathrm{WH}(\mathbf{u} \otimes \mathbf{u}) \rangle$ for some $\mathbf{u} \in \{0, 1\}^n$.

1. All that remains is to check that $\mathbf{u}$ is a satisfying assignment
2. Wonderfully, we also have $O(1)$ access to $A_i \cdot (\mathbf{u} \otimes \mathbf{u})$, the value of the i^{th} equation in the QUADEQ instance and can match it with b_i.

But but but ... how do we check all the m equations of the QUADEQ instance in constant number of queries? ☹

Verify that f encodes a satisfying assignment

Now we are assured that the form of π is $\langle \mathrm{WH}(\mathbf{u}), \mathrm{WH}(\mathbf{u} \otimes \mathbf{u}) \rangle$ for some $\mathbf{u} \in \{0,1\}^n$.

1. All that remains is to check that $\mathbf{u}$ is a satisfying assignment
2. Wonderfully, we also have $O(1)$ access to $A_i \cdot (\mathbf{u} \otimes \mathbf{u})$, the value of the i^{th} equation in the QUADEQ instance and can match it with b_i.

But but but ... how do we check all the m equations of the QUADEQ instance in constant number of queries? ☹

Use the random subsum principle AGAIN! Choose a subset of equations randomly from $[k]$ and add them together to create a new quadratic equation. If $\mathbf{u}$ did not satisfy even one equation of the original system, it will not satisfy the new equation with probability at least $1/2$.

QED

With this, we have proved that $NP \subseteq PCP[poly(n), 1]$. The other direction is trivial.

The stronger theorem makes further observations regarding the form of the proof π given here. Then it uses further results such as gap amplification and alphabet reduction to prove the general statement $NP = PCP(log(n), 1)$.

Trajectory

Hardness of Approximation

A Sad Result

gap_α-$MAX3SAT$ is NP-hard

Just as SAT captured the essence of hardness of exact solution, gap_α-MAX3SAT, or it's more general formulation, qCSP, captures the essence of hardness of approximation

Proof Method

PCP Theorem $\implies$ gap_{α}-$MAX3SAT$ is NP-hard

Proof: Consider any $L \in PCP_{1,\frac{1}{2}}[c \cdot log(n), Q]$. The idea is to encode the Verifier's possible actions by a Boolean formula Ψ.

$$\Psi = \bigwedge_{coins\ R} h_R$$

But h_r is an arbitrary predicate over Q variables.

Hardness of Approximation

Fact

$\forall q$, $\exists l(q), k(q)$ such that any q-ary Boolean function h can be encoded by a 3-CNF formula ψ_h with $k(q)$ clauses over $q + l(q)$ variables $x_1, \ldots, x_q, z_1, \ldots, z_l(q)$ such that

$$h(x) = 1 \implies \exists z, \psi_h(x, z) = 1$$

$$h(x) = 0 \implies \forall z, \psi_h(x, z) = 0$$

$$\Psi = \bigwedge_{coins\ R} \psi_{h_R}$$

1. If $x \in L$ then $\exists$ proof π such that $\forall R, h_R(\pi) = 1$
2. If $x \notin L$ then at least $\frac{1}{2}$ choices of R accept make $h_R(\pi) = 0$. If the total number of clauses are M $(= 2^R k(q))$ then maximum fraction of clauses that can be satisfied is $(1 - \frac{1}{k})$. $\square$

This proves that PCP-theorem leads to the hardness of gap_α-MAX3SAT which in turn as presented earlier leads to NP-hardness of α-approximating MAX3SAT.

References

1 Arora, Sanjeev, and Boaz Barak. Computational complexity: a modern approach. Cambridge University Press, 2009.

2 Limits of approximation algorithms : PCPs and Unique Games - Spring Semester (2009-10) at TIFR, IMSc http://www.tcs.tifr.res.in/ prahladh/teaching/2009-10/limits/

Shukria


THANK
YOU
GRACIAS
ARIGATO
SHUKURIA
DANKSCHEEN
TASHAKKUR ATU
YAQHANYELAY
SUKSAMA
EKHMET
GOZAIMASHITA
EFCHARISTO
JUSPAXAR
KOMAPSUMNIDA
MAAKE
GRAZIE
MEHRBANI
PALDIES
BOLZİN
MERCI
BİYAN
SHUKRIA
TINGKI