Complexity Theory

Jörg Kreiker

Chair for Theoretical Computer Science
Prof. Esparza
TU München

Summer term 2010

Lecture 21

$\mathbf{NP} \subseteq \mathbf{PCP}[poly(n), 1]$

Recap: Two views of the PCP theorem

prob. checkable proofs		hardness of approximation
PCP verifier V	$\leftrightarrow$	CSP instance
proof π	$\leftrightarrow$	variable assignment
$\lvert\pi\rvert$	$\leftrightarrow$	number of vars in CSP
number of queries	$\leftrightarrow$	arity of constraints
number of random bits	$\leftrightarrow$	$\log m$, where m is number of clauses

Goal and plan

Goal

- proof a weaker PCP theorem
- learn interesing encoding/decoding schemes useful in such proofs

Plan

- questionnaire
- proof
    - an **NP**-complete language: Quadeq
    - Walsh-Hadamard encodings
    - a **PCP**$[poly, 1]$ system for Quadeq
- summary: PCP and hardness of approximation

Weak PCP

Theorem

$\mathbf{NP} \subseteq \mathbf{PCP}[\mathit{poly}, 1]$

Proof: It suffices to come up with a PCP system for one **NP**-complete language, where the verifier

- uses polynomially many random bits (exponentially long proofs)
- makes a constant number of queries to that proof

Plan:

- an **NP**-complete language: Quadeq
- Walsh-Hadamard encodings
- a $\mathbf{PCP}[\mathit{poly}, 1]$ system for Quadeq

Disclaimer

All arithmetic today will be modulo 2, that is, over the field $\{0, 1\}$!

- $1 + 1 = 0$
- $x^2 = x$
- $x + y = x - y$

Quadeq

- satisfiable quadratic equations over $\{0, 1\}$
- n variables/m equations
- no purely linear terms
- **NP**-complete (exercise!)

Example (Running example)

$$\begin{aligned} xy + xz &= 1 \\ y^2 + yz + z^2 &= 1 \\ x^2 + yx + z^2 &= 0 \end{aligned}$$

Quadeq

- satisfiable quadratic equations over $\{0, 1\}$
- n variables/m equations
- no purely linear terms
- **NP**-complete (exercise!)

Example (Running example)

$$\begin{aligned} xy + xz &= 1 \\ y^2 + yz + z^2 &= 1 \\ x^2 + yx + z^2 &= 0 \end{aligned}$$

Solution: $x = 1,\ y = 0,\ z = 1$
as a vector: $\mathbf{s} = (1\ 0\ 1)$

Be smart, use vector notation

$$\begin{aligned} xy + xz &= 1 \\ y^2 + yz + z^2 &= 1 \\ x^2 + yx + z^2 &= 0 \end{aligned}$$

$$\mathbf{s} = (1\ 0\ 1)$$

Be smart, use vector notation

$$\begin{aligned} xy + xz &= 1 \\ y^2 + yz + z^2 &= 1 \\ x^2 + yx + z^2 &= 0 \end{aligned}$$

$$\mathbf{s} = (1\ 0\ 1)$$

vector notation: for a given $m \times n^2$ matrix A and m vector $\mathbf{b}$ find solution $\mathbf{u} = (x\ y\ z)$ such that

$$A(\mathbf{u} \otimes \mathbf{u}) = \mathbf{b}$$

$\mathbf{u} \otimes \mathbf{u}$	x^2	xy	xz	yx	y^2	yz	zx	zy	z^2	
$\mathbf{s} \otimes \mathbf{s}$	1	0	1	0	0	0	1	0	1	$\mathbf{b}$
A	0	1	1	0	0	0	0	0	0	1
	0	0	0	0	1	1	0	0	1	1
	1	0	0	1	0	0	0	0	1	0

Overview

- Quadeq is the language of satisfiable systems of quadratic equations over $\{0, 1\}$
- natural PCP system expects a solution $\mathbf{u}$ and checks whether it is valid
- but this yields superconstant number of queries!
- how can we encode a solution such that a constant number of queries suffices?

Overview

- Quadeq is the language of satisfiable systems of quadratic equations over $\{0, 1\}$
- natural PCP system expects a solution **u** and checks whether it is valid
- but this yields superconstant number of queries!
- how can we encode a solution such that a constant number of queries suffices?
- use longer proofs!

- an **NP**-complete language: Quadeq ✓
- Walsh-Hadamard encodings
- a **PCP**[$poly$, 1] system for Quadeq

PCP for Quadeq

Input: $m \times n^2$ matrix A, m vector $\mathbf{b}$

Verifier	Proof π
1. check that f, g are linear functions 2. check that $g = WH(\mathbf{u} \otimes \mathbf{u})$ where $f = WH(\mathbf{u})$ 3. check that g encodes a satisfying assignment	• $\pi \in \{0,1\}^{2^n + 2^{n^2}}$ • π is a pair of linear functions $\langle f, g \rangle$, i.e. strings from $\{0,1\}^{2^n}$ and $\{0,1\}^{2^{n^2}}$, resp. • if $\mathbf{u}$ satisfies $A(\mathbf{u} \otimes \mathbf{u}) = \mathbf{b}$ then $f = WH(\mathbf{u})$ and $g = WH(\mathbf{u} \otimes \mathbf{u})$ are Walsh-Hadamard encodings

Walsh-Hadamard encoding

Definition (WH)

Let $\mathbf{u} \in \{0,1\}^n$ be a vector. The Walsh-Hadamard encoding of $\mathbf{u}$ written $WH(\mathbf{u})$ is the truth table of the linear function $f : \{0,1\}^n \rightarrow \{0,1\}$ where $f(\mathbf{x}) = \mathbf{u} \odot \mathbf{x}$. Furthermore $(u_1 \ldots u_n) \odot (x_1 \ldots x_n) = \Sigma_{i=1}^n u_i x_i$

Walsh-Hadamard encoding

Definition (WH)

Let $\mathbf{u} \in \{0,1\}^n$ be a vector. The Walsh-Hadamard encoding of $\mathbf{u}$ written $WH(\mathbf{u})$ is the truth table of the linear function $f : \{0,1\}^n \to \{0,1\}$ where $f(\mathbf{x}) = \mathbf{u} \odot \mathbf{x}$. Furthermore $(u_1 \ldots u_n) \odot (x_1 \ldots x_n) = \Sigma_{i=1}^n u_i x_i$

Example

The solution to our running example is $\mathbf{s} = (1\ 0\ 1)$. We have

$$WH(\mathbf{s}) = (0\ 1\ 0\ 1\ 1\ 0\ 1\ 0)$$

Note: $|WH(\mathbf{u})| = 2^{|\mathbf{u}|}$

Properties (without proof)

Random subsum principle

- if $\mathbf{u} \neq \mathbf{v}$ then for 1/2 of the choices of $\mathbf{x}$ we have $\mathbf{u} \odot \mathbf{x} \neq \mathbf{v} \odot \mathbf{x}$
- if $\mathbf{u} \neq \mathbf{v}$ then $WH(\mathbf{u})$ and $WH(\mathbf{v})$ differ on at least half their bits

Properties (without proof)

Random subsum principle

- if $\mathbf{u} \neq \mathbf{v}$ then for 1/2 of the choices of $\mathbf{x}$ we have $\mathbf{u} \odot \mathbf{x} \neq \mathbf{v} \odot \mathbf{x}$
- if $\mathbf{u} \neq \mathbf{v}$ then $WH(\mathbf{u})$ and $WH(\mathbf{v})$ differ on at least half their bits

Local linearity testing

- we say that $f, g : \{0,1\}^n \to \{0,1\}$ are ρ-close if

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

- if there exists a $\rho > 1/2$ s.t.

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

  then f is ρ-close to a linear function

PCP for Quadeq

Input: $m \times n^2$ matrix A, m vector $\mathbf{b}$

Verifier	Proof π
1. check that f, g are linear functions 2. check that $g = WH(\mathbf{u} \otimes \mathbf{u})$ where $f = WH(\mathbf{u})$ 3. check that g encodes a satisfying assignment	• $\pi \in \{0,1\}^{2^n + 2^{n^2}}$ • π is a pair of linear functions $\langle f, g \rangle$, i.e. strings from $\{0,1\}^{2^n}$ and $\{0,1\}^{2^{n^2}}$, resp. • if $\mathbf{u}$ satisfies $A(\mathbf{u} \otimes \mathbf{u}) = \mathbf{b}$ then $f = WH(\mathbf{u})$ and $g = WH(\mathbf{u} \otimes \mathbf{u})$ are Walsh-Hadamard encodings

Local linearity testing

- we test the linearity condition ($f(x+y) = f(x) + f(y)$) independently $1/\delta > 2$ times, and accept if all tests pass
- we accept a linear function with probability 1
- if f is not $1-\delta$-close to a linear function
  - all tests are passed with probability at most $(1-\delta)^{(1/\delta)}$

  $\Rightarrow$ such a function is rejected with probability at least $1 - 1/e > 1/2$
- for instance, we could make a 0.999 linearity test using 1000 trials

Local decoding

- it might happen, that we accept non-linear functions that are very close to linear functions
- in this case we treat them as if they were linear
- if we want to query $f(\mathbf{x})$
  1. we choose $\mathbf{x}' \in \{0,1\}^n$ at random
  2. set $\mathbf{x}'' = \mathbf{x} + \mathbf{x}'$
  3. let $\mathbf{y}' = f(\mathbf{x}')$ and $\mathbf{y}'' = f(\mathbf{x}'')$
  4. output $\mathbf{y}' + \mathbf{y}''$
- this makes two queries instead of one
- and recovers the value of the closest linear function with high probability

PCP for Quadeq

Input: $m \times n^2$ matrix A, m vector $\mathbf{b}$

Verifier	Proof π

Verifier

1. check that f, g are linear functions ✓
2. check that $g = WH(\mathbf{u} \otimes \mathbf{u})$ where $f = WH(\mathbf{u})$
3. check that g encodes a satisfying assignment

Proof π

- $\pi \in \{0,1\}^{2^n + 2^{n^2}}$
- π is a pair of linear functions $\langle f, g \rangle$, i.e. strings from $\{0,1\}^{2^n}$ and $\{0,1\}^{2^{n^2}}$, resp.
- if $\mathbf{u}$ satisfies $A(\mathbf{u} \otimes \mathbf{u}) = \mathbf{b}$ then $f = WH(\mathbf{u})$ and $g = WH(\mathbf{u} \otimes \mathbf{u})$ are Walsh-Hadamard encodings

Check WH encodings

Test 10 times for random $\mathbf{r}, \mathbf{r}' \in \{0,1\}^n$

$$f(\mathbf{r})f(\mathbf{r}') = g(\mathbf{r} \otimes \mathbf{r}')$$

Check WH encodings

Test 10 times for random $\mathbf{r}, \mathbf{r}' \in \{0,1\}^n$

$$f(\mathbf{r})f(\mathbf{r}') = g(\mathbf{r} \otimes \mathbf{r}')$$

If the proof is correct we always accept:

$$\begin{aligned} f(\mathbf{r})f(\mathbf{r}') &= \left(\Sigma_{i\in[n]} u_i r_i\right)\left(\Sigma_{j\in[n]} u_j r'_j\right) \\ &= \Sigma_{i,j\in[n]} u_i u_j r_i r'_j \\ &= ((\mathbf{u} \otimes \mathbf{u}) \odot (\mathbf{r} \otimes \mathbf{r}')) \\ &= g(\mathbf{r} \otimes \mathbf{r}') \end{aligned}$$

Check WH encodings

Test 10 times for random $\mathbf{r}, \mathbf{r}' \in \{0,1\}^n$

$$f(\mathbf{r})f(\mathbf{r}') = g(\mathbf{r} \otimes \mathbf{r}')$$

If the proof is correct we always accept:

$$\begin{aligned} f(\mathbf{r})f(\mathbf{r}') &= \Big(\Sigma_{i\in[n]} u_i r_i\Big)\Big(\Sigma_{j\in[n]} u_j r'_j\Big) \\ &= \Sigma_{i,j\in[n]} u_i u_j r_i r'_j \\ &= ((\mathbf{u}\otimes\mathbf{u}) \odot (\mathbf{r}\otimes\mathbf{r}')) \\ &= g(\mathbf{r}\otimes\mathbf{r}') \end{aligned}$$

If the proof is wrong we reject with probability at least 1/4 by applying the random subsum principle twice, because in esence we compute $\mathbf{r}U\mathbf{r}'$ and $\mathbf{r}W\mathbf{r}'$ for different matrices U and W.

PCP for Quadeq

Input: $m \times n^2$ matrix A, m vector $\mathbf{b}$

Verifier	Proof π

Verifier

1. check that f, g are linear functions ✓
2. check that $g = WH(\mathbf{u} \otimes \mathbf{u})$ where $f = WH(\mathbf{u})$ ✓
3. check that g encodes a satisfying assignment

Proof π

- $\pi \in \{0,1\}^{2^n + 2^{n^2}}$
- π is a pair of linear functions $\langle f, g \rangle$, i.e. strings from $\{0,1\}^{2^n}$ and $\{0,1\}^{2^{n^2}}$, resp.
- if $\mathbf{u}$ satisfies $A(\mathbf{u} \otimes \mathbf{u}) = \mathbf{b}$ then $f = WH(\mathbf{u})$ and $g = WH(\mathbf{u} \otimes \mathbf{u})$ are Walsh-Hadamard encodings

Is the assignment satisfying?

- for each of m equations we can check $g(\mathbf{z})$ at some place $\mathbf{z}$ corresponding to the coefficients in matrix A
- but this is not constant queries!

Is the assignment satisfying?

- for each of m equations we can check $g(\mathbf{z})$ at some place $\mathbf{z}$ corresponding to the coefficients in matrix A
- but this is not constant queries!
- instead multiply each equation by a random bit and take the sum of all equations
- if g encodes a solution, we will always have a solution to the sum
- otherwise, we have a solution with probability 1/2 only

Is the system in PCP[$poly(n), 1$]?

1. $\pi \in \{0,1\}^{2^n + 2^{n^2}}$
2. check that f, g are linear functions
   - $2(1-\delta) \cdot n$ random bits, $2(1-\delta)$ queries
3. check that $g = WH(\mathbf{u} \otimes \mathbf{u})$ where $f = WH(\mathbf{u})$
   - $20n$ random bits, 20 queries
4. check that g encodes a satisfying assignment
   - m random bits (one per equation), 1 query

Is the system in PCP$[poly(n), 1]$?

1. $\pi \in \{0,1\}^{2^n + 2^{n^2}}$
2. check that f, g are linear functions
   - $2(1-\delta) \cdot n$ random bits, $2(1-\delta)$ queries
3. check that $g = WH(\mathbf{u} \otimes \mathbf{u})$ where $f = WH(\mathbf{u})$
   - $20n$ random bits, 20 queries
4. check that g encodes a satisfying assignment
   - m random bits (one per equation), 1 query

Yes!

Conclusion

PCP and hardness of approximation

- computing approximate solutions to **NP**-hard problems is important
- the classical Cook-Levin reduction does not rule out efficient approximations
- many nontrivial approximation algorithms exist (2-app for metric TSP, knapsack, 2-app for vertex cover)
- PCP theorem shows hardness of approximating max3SAT to within any constant factor if **P** $\neq$ **NP**
- we showed hardness of approximation for Indset as well
- this is equivalent to having a probabilistically checkable proof system with logarithmic randomness and constant queries
- PCP proofs involve intricate encoding schemes like Walsh-Hadamard

Further Reading *Luca Trevisan*, Inapproximability of Combinatorial Optimization Problems, available from
`http://www.cs.berkeley.edu/~luca/pubs/inapprox.pdf`

Next and final topic: Parallelism