Analysis of Boolean Functions and Inapproximability

Dimitris Tsipras

CoReLab, ECE, NTUA

February 5, 2015

Outline

Introduction to Boolean Functions

We will study boolean functions

$$f : \{-1, 1\}^n \to \{-1, 1\}$$

Introduction to Boolean Functions

We will study boolean functions

$$f : \{-1, 1\}^n \to \{-1, 1\}$$

For examply consider the Majority Function

$$\begin{aligned} Maj_3(-1,-1,-1) &= -1, & Maj_3(-1,-1,+1) &= -1 \\ Maj_3(-1,+1,-1) &= -1, & Maj_3(+1,-1,-1) &= -1 \\ Maj_3(-1,+1,+1) &= +1, & Maj_3(+1,-1,+1) &= +1 \\ Maj_3(+1,+1,-1) &= +1, & Maj_3(+1,+1,+1) &= +1 \end{aligned}$$

Interpolating Boolean Functions

We can interpolate any boolean function with a polynomial

$$
\begin{aligned}
Maj_3(x) = {} & \left(\frac{1+x_1}{2}\right)\left(\frac{1+x_2}{2}\right)\left(\frac{1+x_3}{2}\right)(+1) \\
& + \left(\frac{1+x_1}{2}\right)\left(\frac{1+x_2}{2}\right)\left(\frac{1-x_3}{2}\right)(+1) \\
& + \qquad \ldots \\
& + \left(\frac{1-x_1}{2}\right)\left(\frac{1-x_2}{2}\right)\left(\frac{1-x_3}{2}\right)(-1)
\end{aligned}
$$

Interpolating Boolean Functions

We can interpolate any boolean function with a polynomial

$$\begin{aligned} Maj_3(x) = & \left(\frac{1+x_1}{2}\right)\left(\frac{1+x_2}{2}\right)\left(\frac{1+x_3}{2}\right)(+1) \\ & + \left(\frac{1+x_1}{2}\right)\left(\frac{1+x_2}{2}\right)\left(\frac{1-x_3}{2}\right)(+1) \\ & + \quad \dots \\ & + \left(\frac{1-x_1}{2}\right)\left(\frac{1-x_2}{2}\right)\left(\frac{1-x_3}{2}\right)(-1) \end{aligned}$$

and then expand and simplify to get

$$Maj_3(x) = \frac{1}{2}x_1 + \frac{1}{2}x_2 + \frac{1}{2}x_3 - \frac{1}{2}x_1x_2x_3$$

"Fourier Expansion" of Boolean Functions

Theorem

Every function $f : \{-1,1\}^n \to \{-1,1\}$ *can be expressed as*

$$f(x) = \sum_{S \subseteq [n]} \widehat{f}(S) x_S(x)$$

where $x_S(x) = \prod_{i \in S} x_i$

"Fourier Expansion" of Boolean Functions

Theorem

Every function $f : \{-1,1\}^n \to \{-1,1\}$ *can be expressed as*

$$f(x) = \sum_{S \subseteq [n]} \widehat{f}(S) \chi_S(x)$$

where $\chi_S(x) = \prod_{i \in S} x_i$

Example: $Maj_3(x) = \frac{1}{2}x_1 + \frac{1}{2}x_2 + \frac{1}{2}x_3 - \frac{1}{2}x_1x_2x_3$

$$\widehat{Maj_3}(\emptyset) = 0$$

$$\widehat{Maj_3}(\{1\}) = \widehat{Maj_3}(\{2\}) = \widehat{Maj_3}(\{3\}) = \frac{1}{2}$$

$$\widehat{Maj_3}(\{1,2\}) = \widehat{Maj_3}(\{1,3\}) = \widehat{Maj_3}(\{2,3\}) = 0$$

$$\widehat{Maj_3}(\{1,2,3\}) = \frac{1}{2}$$

Plancheler Theorem

We will study the behavior of functions on uniformly random strings

$$\mathbf{x} \sim \{-1, 1\}^n$$

Plancheler Theorem

We will study the behavior of functions on uniformly random strings

$$\mathbf{x} \sim \{-1, 1\}^n$$

Theorem (Plancheler)

For any functions $f, g : \{-1, 1\}^n \to \mathbb{R}$

$$\mathop{\mathbb{E}}_{\mathbf{x}}[f(\mathbf{x})g(\mathbf{x})] = \sum_{S \subseteq [n]} \widehat{f}(S)\widehat{g}(S)$$

Plancheler Theorem

Proof.

$$\mathop{\mathbb{E}}_{\mathbf{x}}[x_S(x)] = \begin{cases} 0, & \text{if } S \neq \emptyset \\ 1, & \text{otherwise} \end{cases}$$

Plancheler Theorem

Proof.

$$\mathop{\mathbb{E}}_{\mathbf{x}}[\chi_S(x)] = \begin{cases} 0, & \text{if } S \neq \emptyset \\ 1, & \text{otherwise} \end{cases}$$

$$\begin{aligned}
\mathop{\mathbb{E}}_{x}[f(x)g(x)] &= \mathop{\mathbb{E}}_{\mathbf{x}}\left[\sum_{S\subseteq[n]} \widehat{f}(S)\chi_S(\mathbf{x}) \cdot \sum_{T\subseteq[n]} \widehat{g}(T)\chi_T(\mathbf{x})\right] \\
&= \sum_{S,T\subseteq[n]} \widehat{f}(S)\widehat{g}(T) \mathop{\mathbb{E}}_{\mathbf{x}}[\chi_S(\mathbf{x})\chi_T(\mathbf{x})] \\
&= \sum_{S,T\subseteq[n]} \widehat{f}(S)\widehat{g}(T) \mathop{\mathbb{E}}_{\mathbf{x}}[\chi_{S\oplus T}(\mathbf{x})] \\
&= \sum_{S\subseteq[n]} \widehat{f}(S)\widehat{g}(S)
\end{aligned}$$

□

Parseval Theorem

Corollary (Parseval's Theorem)

For any functions $f, g : \{-1,1\}^n \to \mathbb{R}$

$$\mathop{\mathbb{E}}_{\mathbf{x}}[f^2(\mathbf{x})] = \sum_{S \subseteq [n]} \widehat{f}(S)^2$$

Parseval Theorem

Corollary (Parseval's Theorem)

For any functions $f, g : \{-1,1\}^n \to \mathbb{R}$

$$\mathop{\mathbb{E}}_{\mathbf{x}}[f^2(\mathbf{x})] = \sum_{S \subseteq [n]} \widehat{f}(S)^2$$

And therefore for functions $f : \{-1,1\}^n \to \{-1,1\}$

$$\sum_{S \subseteq [n]} \widehat{f}(S)^2 = 1$$

Formula for Fourier Coefficients

Corollary

For any functions $f : \{-1, 1\}^n \to \mathbb{R}$

$$\widehat{f}(S) = \mathop{\mathbb{E}}_{\mathbf{x}}[f(\mathbf{x})x_S(\mathbf{x})]$$

Formula for Fourier Coefficients

Corollary

For any functions $f : \{-1,1\}^n \to \mathbb{R}$

$$\widehat{f}(S) = \mathop{\mathbb{E}}_{\mathbf{x}}[f(\mathbf{x})\chi_S(\mathbf{x})]$$

Proof.

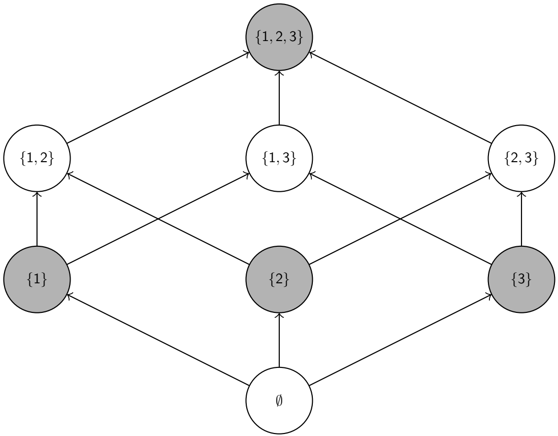
$$\begin{aligned}
\mathop{\mathbb{E}}_{\mathbf{x}}[f(\mathbf{x})\chi_S(\mathbf{x})] &= \mathop{\mathbb{E}}_{\mathbf{x}}\left[\left(\sum_T \widehat{f}(T)\chi_T(\mathbf{x})\right)\chi_S(\mathbf{x})\right] \\
&= \sum_T \widehat{f}(T) \mathop{\mathbb{E}}_{\mathbf{x}}[\chi_S(\mathbf{x})\chi_T(\mathbf{x})] \\
&= \widehat{f}(S)
\end{aligned}$$

□

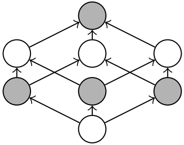
Fourier Coefficients as Weights

Definition

The "(Fourier) weight" of f on S is $\widehat{f}(S)^2$.

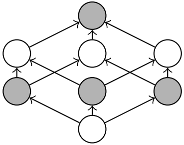


Some Illustrative Examples

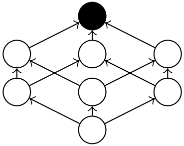


- Majority: $Maj_3(x)$

Some Illustrative Examples



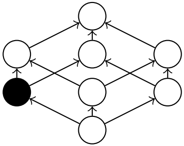
- Majority: $Maj_3(x)$



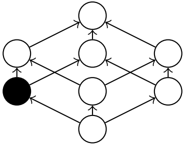
- Parity: $Par_3(x) = x_1 x_2 x_3$

Some Illustrative Examples

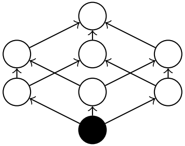
- Dictatorship: $Dict_1(x) = x_1$



Some Illustrative Examples



- Dictatorship: $Dict_1(x) = x_1$



- Constants: $Const_{-1}(x) = -1$, $Const_1(x) = 1$

Outline

Linearity Testing

Definition

A function $f : \{-1,1\}^n \to \{-1,1\}$ is linear if for some $S \subseteq [n]$

$$f(x) = \chi_S(x) = \prod_{i \in S} x_i$$

Linearity Testing

Definition

A function $f : \{-1, 1\}^n \to \{-1, 1\}$ is linear if for some $S \subseteq [n]$

$$f(x) = x_S(x) = \prod_{i \in S} x_i$$

Suppose we have **black-box** access to an unknown function f and want to test if it is linear. Specifically we want to design a test such that

Linearity Testing

Definition

A function $f : \{-1, 1\}^n \to \{-1, 1\}$ is linear if for some $S \subseteq [n]$

$$f(x) = x_S(x) = \prod_{i \in S} x_i$$

Suppose we have **black-box** access to an unknown function f and want to test if it is linear. Specifically we want to design a test such that

- If f is linear, it passes the test with probability $1 - \epsilon$.

Linearity Testing

Definition

A function $f : \{-1, 1\}^n \to \{-1, 1\}$ is linear if for some $S \subseteq [n]$

$$f(x) = x_S(x) = \prod_{i \in S} x_i$$

Suppose we have **black-box** access to an unknown function f and want to test if it is linear. Specifically we want to design a test such that

- If f is linear, it passes the test with probability $1 - \epsilon$.
- If f passes the test with probability $1 - \epsilon$, then f is ϵ-close to some linear function.

The BLR Test

The Blum-Luby-Rubinfield Linearity Test:

The BLR Test

The Blum-Luby-Rubinfield Linearity Test:

- Pick $\mathbf{x} \sim \{-1, 1\}^n$ and $\mathbf{y} \sim \{-1, 1\}^n$ independently

The BLR Test

The Blum-Luby-Rubinfield Linearity Test:

- Pick $\mathbf{x} \sim \{-1, 1\}^n$ and $\mathbf{y} \sim \{-1, 1\}^n$ independently
- Query f at $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{x} \cdot \mathbf{y}$ (where $\cdot$ the pointwise product of $\mathbf{x}$,$\mathbf{y}$)

The BLR Test

The Blum-Luby-Rubinfield Linearity Test:

- Pick $\mathbf{x} \sim \{-1,1\}^n$ and $\mathbf{y} \sim \{-1,1\}^n$ independently
- Query f at $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{x} \cdot \mathbf{y}$ (where $\cdot$ the pointwise product of $\mathbf{x}$,$\mathbf{y}$)
- Accept if $f(\mathbf{x}) \cdot f(\mathbf{y}) = f(\mathbf{x} \cdot \mathbf{y})$

The BLR Test

The Blum-Luby-Rubinfield Linearity Test:

- Pick $\mathbf{x} \sim \{-1,1\}^n$ and $\mathbf{y} \sim \{-1,1\}^n$ independently
- Query f at $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{x} \cdot \mathbf{y}$ (where $\cdot$ the pointwise product of $\mathbf{x}$,$\mathbf{y}$)
- Accept if $f(\mathbf{x}) \cdot f(\mathbf{y}) = f(\mathbf{x} \cdot \mathbf{y})$

Claim 1 (obvious)

If f is a linear (or ϵ-close to a linear) function, then it passes the test with probability 1 (or at least $1 - \epsilon$).

The BLR Test

Claim 2

If f is accepted with probability $1 - \epsilon$, then there exists some S such that $\mathbf{Pr}_{\mathbf{x}}[f(\mathbf{x}) \neq x_S(\mathbf{x})] \geq 1 - \epsilon$

The BLR Test

Claim 2

If f is accepted with probability $1-\epsilon$, then there exists some S such that $\mathbf{Pr}_{\mathbf{x}}[f(\mathbf{x}) \neq x_S(\mathbf{x})] \geq 1-\epsilon$

Proof.

$$
\begin{aligned}
1-\epsilon = \mathbf{Pr}[\text{BLR accepts}] &= \mathop{\mathbb{E}}_{x,y}\left[\frac{1}{2}+\frac{1}{2}f(\mathbf{x})f(\mathbf{y})f(\mathbf{x}\cdot\mathbf{y})\right] \\
&= \frac{1}{2}+\frac{1}{2}\mathop{\mathbb{E}}_{\mathbf{x}}[f(\mathbf{x})\mathop{\mathbb{E}}_{\mathbf{y}}[f(\mathbf{y})f(\mathbf{x}\cdot\mathbf{y})]] \\
&= \frac{1}{2}+\frac{1}{2}\mathop{\mathbb{E}}_{\mathbf{x}}[f(\mathbf{x})g(\mathbf{x})] \\
&= \frac{1}{2}+\frac{1}{2}\sum_{S}\widehat{f}(S)\widehat{g}(S)
\end{aligned}
$$

where $g(x) = E_{\mathbf{y}}[f(\mathbf{y})f(x\cdot\mathbf{y})]$

The BLR Test

Proof.

For $g(x)$

$$\begin{aligned}\widehat{g}(S) &= \mathop{\mathbb{E}}_{\mathbf{x}}[\mathop{\mathbb{E}}_{\mathbf{y}}[f(\mathbf{y})f(\mathbf{x}\cdot\mathbf{y})]\chi_S(\mathbf{x})] \\ &= \mathop{\mathbb{E}}_{\mathbf{x},\mathbf{z}}[f(\mathbf{y})f(\mathbf{z})\chi_S(\mathbf{y}\cdot\mathbf{z})] \\ &= \mathop{\mathbb{E}}_{\mathbf{x},\mathbf{z}}[f(\mathbf{y})\chi_S(\mathbf{y})f(\mathbf{z})\chi_S(\mathbf{z})] \\ &= \widehat{f}(S)^2\end{aligned}$$

Therefore,

$$\begin{aligned}1-\epsilon = \mathbf{Pr}[\text{BLR accepts}] &= \frac{1}{2}+\frac{1}{2}\sum_S \widehat{f}(S)^3 \\ &\le \frac{1}{2}+\frac{1}{2}\max_S\{\widehat{f}(S)\}\end{aligned}$$

The BLR Test

Proof.

Let $S^* = \text{argmax}_S\{\widehat{f}(S)\}$, then

$$\begin{aligned}
1-\epsilon &\leq \frac{1}{2} + \frac{1}{2}\widehat{f}(S^*) \\
&= \frac{1}{2} + \frac{1}{2}\mathop{\mathbb{E}}_{\mathbf{x}}[f(\mathbf{x})x_{S^*}(\mathbf{x})] \\
&= \frac{1}{2} + \frac{1}{2}(\mathop{\mathbf{Pr}}_{\mathbf{x}}[f(\mathbf{x}) = x_{S^*}(\mathbf{x})] - \mathop{\mathbf{Pr}}_{\mathbf{x}}[f(\mathbf{x}) \neq x_{S^*}(\mathbf{x})]) \\
&= 1 - \mathop{\mathbf{Pr}}_{\mathbf{x}}[f(\mathbf{x}) \neq x_{S^*}(\mathbf{x})]
\end{aligned}$$

And therefore

$$\mathop{\mathbf{Pr}}_{\mathbf{x}}[f(\mathbf{x}) \neq x_{S^*}(\mathbf{x})] \leq \epsilon$$

□

The BLR Test

We have therefore constructed $1 - \epsilon$ vs. 1 Linearity with 3 queries, which uses a linear predicate for acceptance.

The BLR Test

We have therefore constructed $1 - \epsilon$ vs. 1 Linearity with 3 queries, which uses a linear predicate for acceptance.

- Any linear function passes with probability 1 (Completeness).

The BLR Test

We have therefore constructed $1-\epsilon$ vs. 1 Linearity with 3 queries, which uses a linear predicate for acceptance.

- Any linear function passes with probability 1 (Completeness).
- Any function the is ϵ-far from a linear function passes with probability at most $1-\epsilon$ (Soundness).

The BLR Test

We have therefore constructed $1 - \epsilon$ vs. 1 Linearity with 3 queries, which uses a linear predicate for acceptance.

- Any linear function passes with probability 1 (Completeness).
- Any function the is ϵ-far from a linear function passes with probability at most $1 - \epsilon$ (Soundness).

One can create similar tests for a variety of function properties, this is a huge field known as Property Testing.

Outline

The Long Code

Bellare, Goldreich, Sudan: Let $i \in [q]$ that is to be coded in a PCP proof. Then instead of representing it with $\log q$ bits we will represent i by writing down the truth table of the i-th dictatorship function 2^q bits.

The Long Code

Bellare, Goldreich, Sudan: Let $i \in [q]$ that is to be coded in a PCP proof. Then instead of representing it with $\log q$ bits we will represent i by writing down the truth table of the i-th dictatorship function 2^q bits.
If $q = 3$ and $i = 1$ the instead of

$$0\ 1$$

we code i as

$$0\ 0\ 0\ 0\ 1\ 1\ 1\ 1$$

The Long Code

The framework incorporates

- An outer PCP making **non-boolean** queries to the proof

The Long Code

The framework incorporates

- An outer PCP making **non-boolean** queries to the proof
- An inner PCP translating these queries to **boolean** queries through dictatorship testing and Fourier Analysis tools.

Many (non-tight) inapproximability bounds were estabilished this way.

Håstad's optimized PCPs

Håstad:

Håstad's optimized PCPs

Håstad:

- a full dictatorship test is not needed for these Long Code reductions

Håstad's optimized PCPs

Håstad:

- a full dictatorship test is not needed for these Long Code reductions
- we only need to distinguish dictators from functions that are **far** from dictators

Håstad's optimized PCPs

Håstad:

- a full dictatorship test is not needed for these Long Code reductions
- we only need to distinguish dictators from functions that are **far** from dictators

Which functions are far from dictatorships?

The Dictatorship vs. No-Notables Test

Definition

An **r**-query, **s** vs. **c** Dictatorship vs. No-Notables Test using predicate $\boldsymbol{\Psi}$, is a randomized algorithm the queries a function f at r points and accepts if

$$\Psi(f(\mathbf{x}_1), \ldots, f(\mathbf{x}_r)) = 1$$

such that

The Dictatorship vs. No-Notables Test

Definition

An **r**-query, **s** vs. **c** Dictatorship vs. No-Notables Test using predicate $\boldsymbol{\Psi}$, is a randomized algorithm the queries a function f at r points and accepts if

$$\Psi(f(\mathbf{x}_1), \ldots, f(\mathbf{x}_r)) = 1$$

such that

- if f is a dictator, the test accepts w.p. at least c

The Dictatorship vs. No-Notables Test

Definition

An $\mathbf{r}$-query, $\mathbf{s}$ vs. $\mathbf{c}$ Dictatorship vs. No-Notables Test using predicate $\boldsymbol{\Psi}$, is a randomized algorithm the queries a function f at r points and accepts if

$$\Psi(f(\mathbf{x}_1), \ldots, f(\mathbf{x}_r)) = 1$$

such that

- if f is a dictator, the test accepts w.p. at least c
- if f has no notable coordinates, then the test accepts w.p. at most s

Constraint Satisfaction Problems

Constraint Satisfaction Problems

- a set of n variables, $x_1, \ldots, x_n$

Constraint Satisfaction Problems

- a set of n variables, $x_1, \ldots, x_n$
- a domain Ω, e.g. $\{-1, 1\}$

Constraint Satisfaction Problems

- a set of n variables, $x_1, \ldots, x_n$
- a domain Ω, e.g. $\{-1, 1\}$
- a (multi)set of constraints, which we try to satisfy

Examples:

Constraint Satisfaction Problems

- a set of n variables, $x_1, \ldots, x_n$
- a domain Ω, e.g. $\{-1, 1\}$
- a (multi)set of constraints, which we try to satisfy

Examples:

Max-E3-Sat

$$\begin{gathered}(x_1 \vee x_2 \vee \neg x_5)\\(x_2 \vee x_4 \vee \neg x_3)\\\ldots\\(\neg x_{10} \vee \neg x_{21} \vee x_{50})\end{gathered}$$

Constraint Satisfaction Problems

- a set of n variables, $x_1, \ldots, x_n$
- a domain Ω, e.g. $\{-1, 1\}$
- a (multi)set of constraints, which we try to satisfy

Examples:

Max-E3-Sat

$$\begin{gathered}(x_1 \vee x_2 \vee \neg x_5)\\(x_2 \vee x_4 \vee \neg x_3)\\\ldots\\(\neg x_{10} \vee \neg x_{21} \vee x_{50})\end{gathered}$$

Max-E3-Lin

$$\begin{gathered}x_1 + x_2 + x_5 = 0\\x_6 + x_7 + x_9 = 1\\\ldots\\x_1 + x_{20} + x_{50} = 0\end{gathered}$$

Constraint Satisfaction Problems

- a set of n variables, $x_1, \ldots, x_n$
- a domain Ω, e.g. $\{-1, 1\}$
- a (multi)set of constraints, which we try to satisfy

Examples:

Max-E3-Sat

$$\begin{gathered}(x_1 \vee x_2 \vee \neg x_5)\\(x_2 \vee x_4 \vee \neg x_3)\\\ldots\\(\neg x_{10} \vee \neg x_{21} \vee x_{50})\end{gathered}$$

Max-E3-Lin

$$\begin{gathered}x_1 + x_2 + x_5 = 0\\x_6 + x_7 + x_9 = 1\\\ldots\\x_1 + x_{20} + x_{50} = 0\end{gathered}$$

Max-Cut

$$\begin{gathered}x_1 \neq x_5\\x_2 \neq x_3\\\ldots\\x_{10} \neq x_{42}\end{gathered}$$

Constraint Satisfaction Problems

Definition

An algorithm (α, β)-approximates a CSP if for every instance

Constraint Satisfaction Problems

Definition

An algorithm (α, β)-approximates a CSP if for every instance

- the β**est** assignment satisfies a fraction β of the constraints

Constraint Satisfaction Problems

Definition

An algorithm (α, β)-approximates a CSP if for every instance

- the β**est** assignment satisfies a fraction β of the constraints
- the αlgorithms satisfies at least a fraction α

Facts:

Constraint Satisfaction Problems

Definition

An algorithm (α, β)-approximates a CSP if for every instance

- the β**est** assignment satisfies a fraction β of the constraints
- the αlgorithms satisfies at least a fraction α

Facts:

- (β, β)-approximating most CSPs is *NP*-Hard

Constraint Satisfaction Problems

Definition

An algorithm (α, β)-approximates a CSP if for every instance

- the β**est** assignment satisfies a fraction β of the constraints
- the αlgorithms satisfies at least a fraction α

Facts:

- (β, β)-approximating most CSPs is *NP*-Hard
- $(1, 1)$-approximating Max-E3-Lin is easy

Constraint Satisfaction Problems

Definition

An algorithm (α, β)-approximates a CSP if for every instance

- the β**est** assignment satisfies a fraction β of the constraints
- the αlgorithms satisfies at least a fraction α

Facts:

- (β, β)-approximating most CSPs is *NP*-Hard
- $(1, 1)$-approximating Max-E3-Lin is easy
- $(\frac{1}{2}, \beta)$-approximating Max-3-Lin is easy

Constraint Satisfaction Problems

Definition

An algorithm (α, β)-approximates a CSP if for every instance

- the β**est** assignment satisfies a fraction β of the constraints
- the αlgorithms satisfies at least a fraction α

Facts:

- (β, β)-approximating most CSPs is *NP*-Hard
- $(1, 1)$-approximating Max-E3-Lin is easy
- $(\frac{1}{2}, \beta)$-approximating Max-3-Lin is easy
- $(\frac{7}{8}, \beta)$-approximating Max-3-Sat is easy

A black box reduction

Theorem

A black box reduction

Theorem

- *Fix any CSP over domain* $\{-1, 1\}$ *with predicate set* Ψ.

A black box reduction

Theorem

- *Fix any CSP over domain* $\{-1, 1\}$ *with predicate set* Ψ.
- *Suppose there exists some* r*-query,* s *vs.* c *Dictatorship vs. No-Notables Test using predicate* Ψ

A black box reduction

Theorem

- *Fix any CSP over domain $\{-1,1\}$ with predicate set Ψ.*
- *Suppose there exists some r-query, s vs. c Dictatorship vs. No-Notables Test using predicate Ψ*
- *Then for any $\delta > 0$ it is* **UG-hard** *to $(s+\delta, c-\delta)$-approximate Max-CSP$_r(\Psi)$.*

A $\frac{1}{2}$ vs. $1 - \delta$ Dictator vs. No-Notables Test

The BLR Linearity tests whether a function is a parity or not

A $\frac{1}{2}$ vs. $1-\delta$ Dictator vs. No-Notables Test

The BLR Linearity tests whether a function is a parity or not

- Dictators pass w.p. 1 (small parities too but this is ok)

A $\frac{1}{2}$ vs. $1-\delta$ Dictator vs. No-Notables Test

The BLR Linearity tests whether a function is a parity or not

- Dictators pass w.p. 1 (small parities too but this is ok)
- we need to reject large parities (Par_n):
  Add a little ϵ-noise to $\mathbf{x} \cdot \mathbf{y}$

A $\frac{1}{2}$ vs. $1 - \delta$ Dictator vs. No-Notables Test

The BLR Linearity tests whether a function is a parity or not

- Dictators pass w.p. 1 (small parities too but this is ok)
- we need to reject large parities (Par_n):
  Add a little ϵ-noise to $\mathbf{x} \cdot \mathbf{y}$
  - dictators still pass w.p. $1 - \epsilon$
  - large parities fail with large probability

A $\frac{1}{2}$ vs. $1 - \delta$ Dictator vs. No-Notables Test

The BLR Linearity tests whether a function is a parity or not

- Dictators pass w.p. 1 (small parities too but this is ok)
- we need to reject large parities (Par_n):
  Add a little ϵ-noise to $\mathbf{x} \cdot \mathbf{y}$
  - dictators still pass w.p. $1 - \epsilon$
  - large parities fail with large probability
- we need to reject the constant 1

A $\frac{1}{2}$ vs. $1-\delta$ Dictator vs. No-Notables Test

The BLR Linearity tests whether a function is a parity or not

- Dictators pass w.p. 1 (small parities too but this is ok)
- we need to reject large parities (Par_n):
  Add a little ϵ-noise to $\mathbf{x}\cdot\mathbf{y}$
  - dictators still pass w.p. $1-\epsilon$
  - large parities fail with large probability
- we need to reject the constant 1 Instead of testing whether $f(\mathbf{x})f(\mathbf{y})f(\mathbf{x}\cdot\mathbf{y})=1$ we test w.p. $1/2$
  - if $f(\mathbf{x})f(\mathbf{y})f(\mathbf{x}\cdot\mathbf{y})=1$
  - if $f(\mathbf{x})f(\mathbf{y})f(\overline{\mathbf{x}\cdot\mathbf{y}})=-1$

Inapproximability Results

Corollary

It is UG-Hard to $(\frac{1}{2} + \delta, 1 - \delta)$*-approximate Max-E3-Lin.*

Inapproximability Results

Corollary

It is UG-Hard to $(\frac{1}{2} + \delta, 1 - \delta)$*-approximate Max-E3-Lin.*

With similar tests and Fourier Analysis

Corollary

It is UG-Hard to $(\frac{7}{8} + \delta, 1 - \delta)$*-approximate Max-E3-Sat.*

Inapproximability Results

Corollary

It is UG-Hard to $(\frac{1}{2} + \delta, 1 - \delta)$-approximate Max-E3-Lin.

With similar tests and Fourier Analysis

Corollary

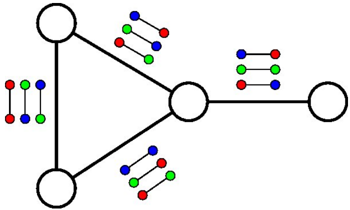
It is UG-Hard to $(\frac{7}{8} + \delta, 1 - \delta)$-approximate Max-E3-Sat.

Corollary

It is UG-Hard to $((0.878 + \delta)\beta, \beta)$-approximate Max-Cut.

Unique Games

- a set of variables (nodes)
- a domain Ω (colors)
- a set of **bijective** constraints



Unique Games Conjecture

Conjecture [Khot '02]

For every $\delta > 0$, $(\delta,\ 1 - \delta)$-approximating UG is NP-Hard.

Unique Games Conjecture

Conjecture [Khot '02]

For every $\delta > 0$, $(\delta,\ 1 - \delta)$-approximating UG is NP-Hard.

Problem	Best Known	NP-Hardness	UGC-Hardness
Max-2-Sat	0.940	0.954+ϵ	0.940+ϵ
Max-Cut	0.878	0.941+ϵ	0.878+ϵ
Min-Vertex-Cover	2	1.360-ϵ	2 - ϵ

Further Reading

- O'Donnell, Ryan. Analysis of boolean functions. Cambridge University Press, 2014.
- Khot, Subhash. "Guest column: Inapproximability results via long code based PCPs." ACM SIGACT News 36.2 (2005): 25-42.
- Khot, Subhash. "Inapproximability of np-complete problems, discrete fourier analysis, and geometry." International Congress of Mathematics. Vol. 5. 2010.
- O'Donnell, Ryan. "Some topics in analysis of Boolean functions." Proceedings of the fortieth annual ACM symposium on Theory of computing. ACM, 2008. S. Jemand.

THANK YOU!