Spectral Algorithms for Unique Games

Alexandra Kolla
The MAX CUT Problem

• **Input:** $G = (V,E)$
The MAX CUT Problem

- **Input:** $G = (V, E)$
- **Objective:** Partition $G$ in $(S, S')$ as to **MAXIMIZE** number of edges cut

- **[Karp ’72]:** MAX CUT is NP-complete
- What about approximating MAX CUT?
The MAX CUT Problem

- **Input:** $G = (V,E)$
- **Objective:** Partition $G$ in $(S,S')$ as to MAXIMIZE number of edges cut

Approximation algorithms:

- **Random cut (trivial):** half of optimal
- **[GW’94]:** $\alpha_{GW}=0.878$ approximation algorithm of MAX CUT

How many of you bet this is best we can do?
The MAX CUT Problem

- **Input:** \( G = (V,E) \)
- **Objective:** Partition \( G \) in \( (S,S') \) as to MAXIMIZE number of edges cut

Approximation algorithms:

- **Random cut (trivial):** half of optimal
- **[GW’94]:** \( \alpha_{GW} = 0.878 \) approximation algorithm of MAX CUT

If UGC True, then it is the best!
Can We Hope for Better Approximation Algorithms in P?

Previous inapproximability not a coincidence! Unique Games Conjecture (UGC) captures **exact** inapproximability of many more problems.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Best Approximation Algorithm Known</th>
<th>UGC-Hardness</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxCut</td>
<td>0.878 [GW94]</td>
<td>0.878 [KKMO07]</td>
</tr>
<tr>
<td>Vertex Cover</td>
<td>2</td>
<td>2-(\varepsilon) [KR06]</td>
</tr>
<tr>
<td>Max k-CSP</td>
<td>(\Omega(k/2^k)) [CMM07]</td>
<td>(O(k/2^k)) [ST,AM,GR]</td>
</tr>
</tbody>
</table>
What are Unique Games?

1. Unique Games are popular not only among computer scientists!
What are Unique Games?

1. Unique Games are popular not only among computer scientists!

2. We can purchase Unique Games on-line!
What are Unique Games?

1. Unique Games are popular not only among computer scientists!

2. We can purchase Unique Games online!

3. Unique Games are related to the Unique Games Conjecture…
Unique Games = Unique Label Cover Problem

Given: set of constraints

Linear Equations mod $k$:
$x_i - x_j = c_{ij} \mod k$

**GOAL**

$k =$ “alphabet” size

Find labeling that satisfies maximum number of constraints.

**EXAMPLE**

\[
\begin{align*}
x_1 - x_2 &= 0 \pmod{3} \\
x_2 - x_3 &= 0 \pmod{3} \\
x_1 - x_3 &= 1 \pmod{3}
\end{align*}
\]

The constraint graph

- $x_1 - x_3 = 1 \pmod{3}$
- $x_1 - x_2 = 0 \pmod{3}$
- $x_2 - x_3 = 0 \pmod{3}$
Unique Games, an Example

Given: set of constraints

Linear Equations mod k:
\[ x_i - x_j = c_{ij} \mod k \]

**GOAL**

Find labeling that satisfies *maximum* number of constraints.

**EXAMPLE**

The constraint graph

\[ x_1 - x_3 = 1 \mod 3 \]
\[ x_1 - x_2 = 0 \mod 3 \]
\[ x_2 - x_3 = 0 \mod 3 \]

Satisfy 2/3 constraints
Unique Games, an Example

Given: set of constraints

Linear Equations mod \( k \):

\[ x_i - x_j = c_{ij} \mod k \]

**GOAL**

\( k \) = “alphabet” size

Find labeling that satisfies **maximum** number of constraints.

**EXAMPLE**

\[
\begin{align*}
    x_1 - x_2 &= 0 \mod 3 & \checkmark \\
    x_2 - x_3 &= 0 \mod 3 & \checkmark \\
    x_1 - x_3 &= 1 \mod 3 & \times \\
\end{align*}
\]

The constraint graph

\[
\begin{align*}
    x_1 - x_3 &= 1 \mod 3 \\
    x_1 - x_2 &= 0 \mod 3 \\
    x_2 - x_3 &= 0 \mod 3 \\
\end{align*}
\]

Rest of the talk: d-regular graphs
Unique Games Conjecture

- [Khot’02] For every positive $\varepsilon$ and $\delta$ there is a large enough $k$ s.t. for some instance of Unique Games with alphabet size $k$ and $\text{OPT} > 1 - \varepsilon$, it is NP hard to satisfy a $\delta$ fraction of all constraints.

- Given UG instance where 99% of constraints are satisfiable, it is NP-hard to even satisfy 0.1%
Unique Games Conjecture

- Embarrassing not to know, since solving systems of linear equations is easy.

- How? (Gaussian elimination, propagation…)
Where to begin if we want to refute UGC?

• Several attempts in recent years to refute or prove UGC.
• Lot of progress but still no consensus.

Plan of attack: start ruling out cases.

- Classify graphs according to their “spectral profile” (eigenvalues)
- Expanders [AKKTSV’08, KT’08],
- Local expanders, graphs with relatively few large eigenvalues [AIMS’09, SR’09, K’10]

- Find distributions that are hard?
  - Random Instances: NO! Follows from expander result.
  - Quasi-Random Instances? [KMM’10] NO!
### Summary: Algorithmic Results for UG

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>On 1-(\varepsilon) instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Khot</td>
<td>1 - (O(k^2 \varepsilon^{1/5}\sqrt{\log(1/\varepsilon)}))</td>
</tr>
<tr>
<td>Trevisan</td>
<td>1 - (O(3\sqrt{\varepsilon \log n}))</td>
</tr>
<tr>
<td>Gupta-Talwar</td>
<td>1 - (O(\varepsilon \log n))</td>
</tr>
<tr>
<td>CMM1</td>
<td>(k^{-\varepsilon/2})</td>
</tr>
<tr>
<td>CMM2</td>
<td>1 - (O(\varepsilon \sqrt{\log n \sqrt{\log k}}))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Expander</th>
<th>Constant, depends on conductance</th>
</tr>
</thead>
<tbody>
<tr>
<td>AKKTSV’08, KT’08, MM’10</td>
<td>Constant, depends on conductance</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Local expander</th>
<th>Constant, depends on local expansion</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIMS’09, SR’09</td>
<td></td>
</tr>
</tbody>
</table>

Almost all above approaches were LP or SDP based.

SDP/LP based

Tight for SDP, there is counterexample
Summary: Algorithmic Results for UG

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>On 1-(\varepsilon) instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Khot</td>
<td>1 - O((k^2 \varepsilon^{1/5} \sqrt{\log(1/\varepsilon)}))</td>
</tr>
<tr>
<td>Trevisan</td>
<td>1 - O((3\sqrt{\varepsilon \log n}))</td>
</tr>
<tr>
<td>Gupta-Talwar</td>
<td>1 - O((\varepsilon \log n))</td>
</tr>
<tr>
<td>CMM1</td>
<td>(k^{-\varepsilon/2-\varepsilon})</td>
</tr>
<tr>
<td>CMM2</td>
<td>1 - O((\varepsilon \sqrt{\log n \sqrt{\log k}}))</td>
</tr>
</tbody>
</table>

**General Graphs**
- SDP/LP based
- Tight for SDP, there is counterexample

**Special Graphs**
- Purely SPECTRAL Approach “beats” SDP

**Expander**
- AKKTSV’08
- KT’08, MM’10
- Constant, depends on conductance

**Local expander**
- AIMS’09, SR’09
- Constant, depends on local expansion

**Few large eigenvalues**
- K’10
- Quality and running time depends on eigenspace
### Summary: Algorithmic Results for UG

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>On 1-(\varepsilon) instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Khot</td>
<td>1-(O(k^2 \varepsilon^{1/5} \sqrt{\log(1/\varepsilon)}))</td>
</tr>
<tr>
<td>Trevisan</td>
<td>1-(O(3\sqrt{(\varepsilon \log n)}))</td>
</tr>
<tr>
<td>Gupta-Talwar</td>
<td>1-(O(\varepsilon \log n))</td>
</tr>
<tr>
<td>CMM1</td>
<td>(k^{-\varepsilon/2})</td>
</tr>
<tr>
<td>CMM2</td>
<td>1-(O(\varepsilon \sqrt{\log n} \sqrt{\log k}))</td>
</tr>
</tbody>
</table>

**General Graphs**

**Special Graphs**

**Expander**

**Local expander**

**Few large eigenvalues**

**AKKTSV’08**

**KT’08,MM’10**

Constant, depends on conductance

**AIMS’09, SR’09**

Constant, depends on local expansion

**K’10**

Quality and running time depends on eigenspace

**ABS’10**: Subexponential time algorithm for ANY instance
## Summary: Algorithmic Results for UG

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>On $1-\varepsilon$ instances</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>General Graphs</strong></td>
<td></td>
</tr>
<tr>
<td>Khot</td>
<td>$1 - O(k^2 \varepsilon^{1/5}\sqrt{\log(1/\varepsilon)})$</td>
</tr>
<tr>
<td>Trevisan</td>
<td>$1 - O(3\sqrt{\varepsilon \log n})$</td>
</tr>
<tr>
<td>Gupta-Talwar</td>
<td>$1 - O(\varepsilon \log n)$</td>
</tr>
<tr>
<td>CMM1</td>
<td>$k^{-\varepsilon/2}$</td>
</tr>
<tr>
<td>CMM2</td>
<td>$1 - O(\varepsilon \sqrt{\log n \sqrt{\log k}})$</td>
</tr>
<tr>
<td><strong>Special Graphs</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Expander</strong></td>
<td></td>
</tr>
<tr>
<td>AKKTSV’08, KT’08, MM’10</td>
<td>Constant, depends on conductance</td>
</tr>
<tr>
<td><strong>Local expander</strong></td>
<td></td>
</tr>
<tr>
<td>AIMS’09, SR’09</td>
<td>Constant, depends on local expansion</td>
</tr>
<tr>
<td><strong>Few large eigenvalues</strong></td>
<td></td>
</tr>
<tr>
<td>K’10</td>
<td>Quality and running time depends on eigenspace</td>
</tr>
</tbody>
</table>

ABS’10: Subexponential time algorithm for ANY instance
### Summary: Algorithmic Results for UG

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>On $1-\varepsilon$ instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Khot</td>
<td>$1 - O(k^2 \varepsilon^{1/5} \sqrt{\log(1/\varepsilon)})$</td>
</tr>
<tr>
<td>Trevisan</td>
<td>$1 - O(3\sqrt{\varepsilon \log n})$</td>
</tr>
<tr>
<td>Gupta-Talwar</td>
<td>$1 - O(\varepsilon \log n)$</td>
</tr>
<tr>
<td>CMM1</td>
<td>$k^{-\varepsilon/2}$</td>
</tr>
<tr>
<td>CMM2</td>
<td>$1 - O(\varepsilon \sqrt{\log n} \sqrt{\log k})$</td>
</tr>
</tbody>
</table>

**General Graphs**

**Special Graphs**

- **Expander**
  - AKKTSV’08, KT’08, MM’10: Constant, depends on conductance
  - AIMS’09, SR’09: Constant, depends on local expansion
  - K’10: Quality and running time depends on eigenspace

**AKKTSV’08:** Subexponential time algorithm for **ANY** instance

**KMM’10:** Semi-Random instances are easy
Unique Games = Unique Label Cover Problem

Given: set of constraints

\[
\begin{align*}
\text{Linear Equations mod } k : \\
x_i - x_j &= c_{ij} \mod k
\end{align*}
\]

**GOAL**

\[k = \text{“alphabet” size}\]

Find labeling that satisfies **maximum** number of constraints.

**EXAMPLE**

\[
\begin{align*}
x_1 - x_2 &= 0 \mod 3 \\
x_2 - x_3 &= 0 \mod 3 \\
x_1 - x_3 &= 1 \mod 3
\end{align*}
\]

The constraint graph

\[
\begin{align*}
x_1 - x_3 &= 1 \mod 3 \\
x_1 - x_2 &= 0 \mod 3 \\
x_2 - x_3 &= 0 \mod 3
\end{align*}
\]
Unique Games and Graphs

2. The “label-extended” graph

1. The “constraint graph”

- Replace each vertex with \( k \) vertices - one for each label.
Unique Games and Graphs

2. The “label-extended” graph

1. The “constraint graph”

- Replace each vertex with \( k \) vertices - one for each label
- Replace each edge with the “permutation matching”
Unique Games and Graphs

2. The “label-extended” graph

1. The “constraint graph”

- Replace each vertex with $k$ vertices— one for each label
- Replace each edge with the “permutation matching”
Unique Games and Graphs

2. The “label-extended” graph

1. The “constraint graph”

- Replace each edge with the “permutation matching”

- Replace each vertex with \( k \) vertices - one for each label
More Graph Theory: The Label-Extended Graph

M has each non-zero entry \((u,w)\) replaced by a block corresponding to the permutation on edge

\[
egin{array}{ccc|ccc}
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
\end{array}
\]

GRAPH THEORY?

it’s a graph, it has adjacency matrix!
UGC FALSE on expanders [AKKTSV’08, KT’08 MM’10]: When UG instance highly satisfiable and graph is expander, ptime algorithm finds labeling that satisfies 99% of the constraints.
Why Expanders? Expansion of Unique Games and Sparsest Cut

<table>
<thead>
<tr>
<th>Problem</th>
<th>Best Approximation Algorithm Known</th>
<th>UGC-Hardness</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxCut</td>
<td>0.878 [GW94]</td>
<td>0.878 [KKMO07]</td>
</tr>
<tr>
<td>Vertex Cover</td>
<td>2</td>
<td>2-ε [KR06]</td>
</tr>
<tr>
<td>Max k-CSP</td>
<td>Ω(k/2^k) [CMM07]</td>
<td>O(k/2^k) [ST, AM, GR]</td>
</tr>
</tbody>
</table>

No hardness even assuming UGC unless expansion

Uniform
Sparsest Cut
Why Expanders? Expansion of Unique Games and Sparsest Cut

No hardness for Sparsest Cut even assuming UGC!

Unlikely that there is reduction from UG to SPARSEST CUT

In known reductions, if I start with an instance of UG that has a sparse cut

Get an instance of SC that has a sparse cut irrespective of UG being YES or NO instance.

…unless UG instance has expansion! [KV,manuscript]

Because then any sparse cut would correspond to a good labeling

Off-the-record belief that expanders were hardest instances
Proof with Graph Theory: From Labelings to Spectra

• Set $S$ that contains exactly one “small” node from each node group = labeling
Proof with Graph Theory: From Labelings to Spectra

• Set $S$ that contains exactly one “small” node from each node group = labeling

• Corresponds to a cut $(S, S')$.

• Corresponds to a “characteristic vector”.

\[ \chi_{(0,0,0)} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \]
Proof Intuition: a Perfect Game

Let’s look at a perfectly satisfiable game for intuition...

Graph is disconnected, it has second eigenvalue $\lambda = d$ (in fact, it has $k$ eigenvalues = $d$)

As mentioned earlier, we can find cuts from those eigenvectors that cut zero edges. ($d - \lambda = 0$)

If graph $G$ was originally connected, those are the only “sparsest cuts”. They correspond to perfect labelings.
Proof Intuition: a Perfect Game

Let’s look at a perfectly satisfiable game for intuition…

A 1-\(\varepsilon\) game is an almost-perfectly-satisfiable one

As mentioned earlier, we can find cuts from those eigenvectors that cut zero edges. (\(d - \lambda = 0\))

If graph G was originally connected, those are the only “sparsest cuts”. They correspond to perfect labelings.
Proof Intuition: a Perfect Game

Let’s look at a perfectly satisfiable game for intuition…

A $1-\varepsilon$ game is an almost-perfectly-satisfiable one

As mentioned earlier, we can find cuts from those eigenvectors that cut zero edges. ($d - \lambda = 0$)

If graph $G$ was originally connected, those are the only “sparsest cuts”. They correspond to perfect labelings.
Proof Intuition: a Perfect Game

Let’s look at a perfectly satisfiable game for intuition...

A $1-\varepsilon$ game is an almost-perfectly-satisfiable one

As mentioned earlier, we can find cuts from those eigenvectors that cut zero edges. ($d - \lambda = 0$)

If graph $G$ was originally connected, those are the only “sparsest cuts”.

They correspond to almost-perfect labelings
Proof: Reverse Engineering + Graph Spectra

1 - \( \varepsilon \) Game

\[ x_v - x_u = 1 \pmod{3} \]

\[ x_u - x_w = 0 \pmod{3} \]

\[ x_w - x_v = 0 \pmod{3} \]
Think of it as “coming from” adversarially perturbed completely satisfiable game.

Perfect Game:

\[ x_u - x_w = 0 \mod 3 \]
\[ x_v - x_u = 0 \mod 3 \]
\[ x_w - x_v = 0 \mod 3 \]

\[ x_u - x_w = 0 \mod 3 \]
\[ x_v - x_u = 1 \mod 3 \]
\[ x_w - x_v = 0 \mod 3 \]

1-ε Game

Proof: Reverse Engineering + Graph Spectra
Think of it as “coming from” adversarialy perturbed completely satisfiable game

Perfect Game:

\[ x_w - x_v = 0 \mod 3 \]
\[ x_v - x_u = 0 \mod 3 \]
\[ x_u - x_w = 0 \mod 3 \]

\[ M \]

\[ \tilde{M} \]

1- \( \varepsilon \) Game

\[ x_v - x_u = 1 \mod 3 \]
\[ x_w - x_v = 0 \mod 3 \]

Proof: Reverse Engineering + Graph Spectra
Proof: Reverse Engineering + Graph Spectra

"Labeling" eigenvectors:

The k-dimensional espace $Y$ of eigenvalues equal to $d$ contains all the information for the best labeling.

Perfect Game:

\[ \tilde{M} \]

\[ \chi_{(0,0,0)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \]

$Y$

First few eigenvectors:

The k "labeling vectors" have large projection onto espace $W$ with eigenvalues $>(1 - 200\varepsilon)d$.

$M$

$\chi_{(0,0,0)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

$W = \begin{pmatrix} m_1 \\ m_2 \\ m_3 \end{pmatrix}$

1 - $\varepsilon$ Game
Proof: Reverse Engineering + Graph Spectra

Perfect Game:

\[ \chi_{(0,0,0)} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \]

First few eigenvectors:

The k-dimensional space \( Y \) of eigenvalues equal to \( d \) contains all the information for the best labeling.

\[ \text{for } \| \chi \| = 1, \quad \chi^T \tilde{M} \chi = d \]
\[ \chi^T M \chi \geq (1 - 2\varepsilon)d \]

Write: \( \chi = \alpha w + \beta w_\perp \)

\[(1 - 2\varepsilon)d \leq \chi^T M \chi = a^2 w^T M w + \beta^2 w^T M w \perp \]

\[ \leq a^2 d + \beta^2 (1 - 200\varepsilon)d \Rightarrow |\beta| \leq \frac{1}{10} \]

I-Game:

\[ \chi_{(0,0,0)} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ m_1 \\ m_2 \\ m_3 \end{pmatrix} \]

The k “labeling vectors” have large projection onto space \( W \) with eigenvalues \( > (1 - 200\varepsilon)d \)

"Labeling" eigenvectors:
Proof: Reverse Engineering + Graph Spectra

“Labeling” eigenvectors:

The k-dimensional space Y of eigenvalues equal to d contains all the information for the best labeling.

Perfect Game:

\[ \chi_{(0,0,0)} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 0 \end{pmatrix} \]

First few eigenvectors:

The k “labeling vectors” have large projection onto space W with eigenvalues \( > (1 - 200\epsilon)d \).

If we knew the projection \( w \) of \( \chi \) then we could just “read off” a good labeling.

1-\((\epsilon)\) Game:

\[ \chi_{(0,0,0)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \]

\[ w = \begin{pmatrix} m_1 \\ m_2 \\ m_3 \end{pmatrix} \]
Searching for a Needle in a Haystack?

But we need to find a particular vector in this whole space $W$!

$\chi_{(0,0,0)} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$

$w = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ m_1 \\ m_2 \\ m_3 \end{pmatrix}$

$\langle \chi, w \rangle \approx 0.9$
Searching for a Needle, but “Efficiently”

But we need to find a particular vector in this whole space $W$!

Idea: Discretize the space by net!

One point of the net is close to the vector we want.
We find this vector and then “read off” the coordinates.

Most blocks have (unique) maximum entry in the position that corresponds to the original value of node $u$. 

$$
\chi_{(0,0,0)} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \quad w = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ m_1 \\ m_2 \\ m_3 \end{pmatrix}
$$

$$
\langle \chi, w \rangle \approx 0.9
$$
Searching for a Needle, but “Efficiently”

But we need to find a particular vector in this whole space $W$!

Idea:
Discretize the space by net!

Algorithm runs in time $\sim \#\text{points in the net} = \text{exponential}$ in the dimension of eigenspace $W$
The Dimension of $\mathcal{W}$ for Expanders

(Spectral Gap) = $d - \lambda = \gamma d$
The Dimension of $W$ for Expanders

Perfect Game: $\chi_{(0,0,0)} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$

$G$ (Spectral gap between $Y, Y_\perp$) = $\text{absgap} = \gamma d$

$\mathbf{w} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ m_1 \\ m_2 \\ m_3 \end{pmatrix}$

$(\text{Spectral Gap}) = d - \lambda = \gamma d$
The Dimension of W for Expanders

Perfect Game: \( \chi_{(0,0,0)} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \)

I- \( \varepsilon \) Game:

\[
\begin{bmatrix}
1 \\
0 \\
0 \\
0 \\
\end{bmatrix} = \begin{bmatrix}
1 \\
0 \\
0 \\
0 \\
\end{bmatrix} + \begin{bmatrix}
m_1 \\
m_2 \\
m_3 \\
0 \\
\end{bmatrix}
\]

(Spectral Gap) = \( d - \lambda = \gamma d \)

\( W \) is “perturbed analog” of \( Y \)

“The sin \( \mu \)” Theorem [DK’70]: Angle between \( Y \) and “perturbed analog of \( Y \)” small

Equivalently, we can write every vector \( w \) in \( W \) as \( w = \alpha y + \beta y \perp, y \in Y \)

\[
|\beta| \leq \frac{\| (M - M_\varepsilon) w \|}{absgap} \leq O(\sqrt{\frac{\varepsilon}{\gamma^3}})
\]
The Dimension of $W$ for Expanders

Perfect Game:

$\chi_{(0,0,0)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

$Y$

$I - \varepsilon$ Game:

$G$

$W$

$w = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ m_1 & m_2 & m_3 \end{bmatrix}$

(Spectral Gap) = $d - \lambda = \gamma d$

(Spectral gap between $Y, Y_{\perp}$) = $\text{absgap} = \gamma d$

$W$ is “perturbed analog” of $Y$

“The sin $\mu$” Theorem [DK’70]: Angle between $Y$ and “perturbed analog of $Y$” small

$W$ is close to $Y$ so $\text{dim}(W) \leq \text{dim}(Y) = k$
A General Algorithm

For expanders, \( W \) is close to \( Y \) so
\[ \dim(W) \leq \dim(Y) = k \]
Running time is
\[ 2^k \approx 2^{\log n} \approx \text{poly}(n) \]

Algorithm runs in time \( \sim \) #points in the net
= exponential in the dimension of eigenspace \( W \)
A General Algorithm

\[ \chi_{(0,0,0)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \]

\[ \langle \chi, w \rangle \approx 0.9 \]

Algorithm runs in time ~ #points in the net = exponential in the dimension of eigenspace \( W \)
Another Special Case: The “Khot-Vishnoi” Graph

Graph that “cheats” a canonical semidefinite program for UG

We show: Eigenspace in question has polylogarithmic dimension

Algorithm runs in time \( \sim \# \text{points in the net} = \text{exponential} \) in the dimension of eigenspace
Another Special Case: The “Khot-Vishnoi” Graph

Algorithm runs in time ~ #points in the net = quasi-polynomial

We show: Eigenspace in question has polylogarithmic dimension
After expanders, we realized that other constraint graphs are easy for UGC.

How “easy” the graph is, depends on the number of large (close to d) eigenvalues of the adjacency matrix of the label-extended graph.

Could solve previously “hardest” cases, where all other techniques failed.

Essentially only one case left, reflected by the Boolean Hypercube!! (?)
Open Questions

Disprove the Unique Games Conjecture

• Can we argue about UGC on the cube?
• About 2 years ago a group of Quantum Computing Theorists came together and tried to find a quantum algorithm…
• Proved Maximal Inequality on the Cube, failed for UGC.
• What is the quantum complexity of UGC?
THANK YOU!