CS 598: Spectral Graph Theory. Lecture 14

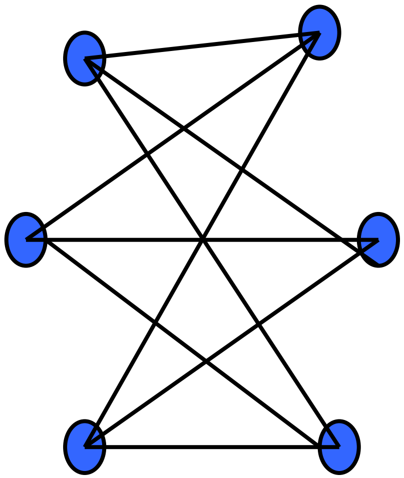
Spectral Algorithms for Unique Games

Alexandra Kolla

The MAX CUT Problem

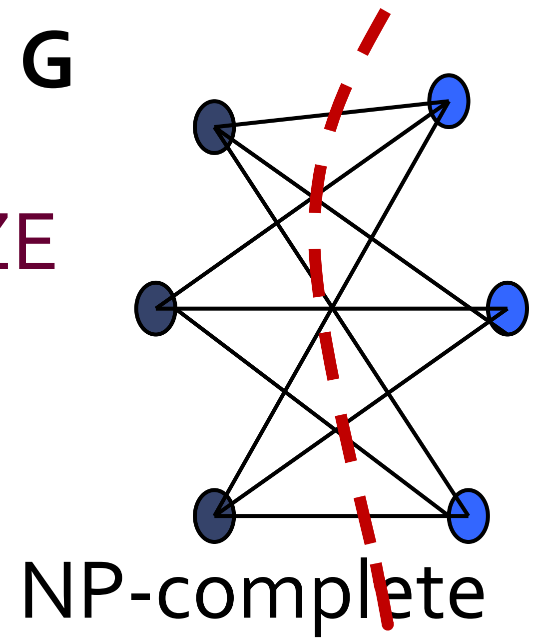
- **Input:** G = (V,E)

G



The MAX CUT Problem

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

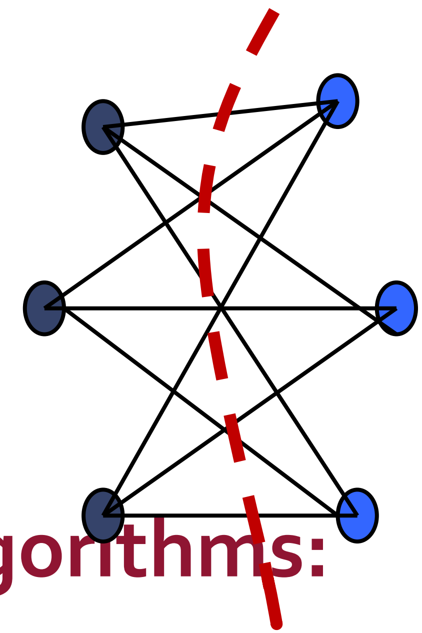


- **[Karp '72]:** MAX CUT is NP-complete
- What about approximating MAX CUT?

The MAX CUT Problem

- **Input:** G = (V,E)

G



- **Objective :** Partition G in (S,S’) as to MAXIMIZE number of edges cut

Approximation algorithms:

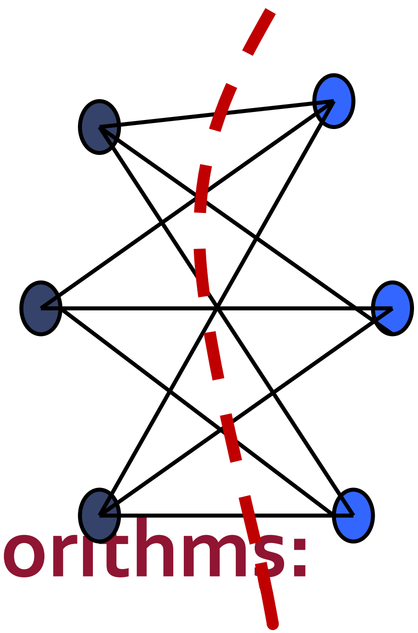
- **Random cut (trivial):** half of optimal
- **[GW’94]:** α_{GW}=0.878 approximation algorithm of MAX CUT

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

G



Approximation algorithms:

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

If UGC True, then it is the best!

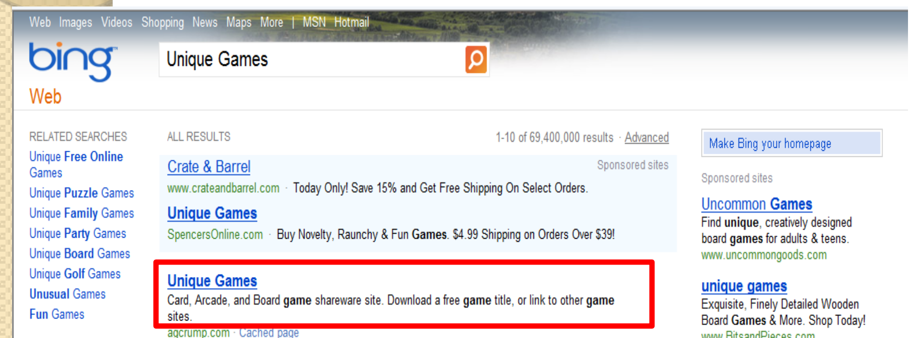
Can We Hope for Better Approximation Algorithms in P?

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

Problem	Best Approximation Algorithm Known	UGC-Hardness
MaxCut	0.878[GW94]	0.878 [KKMO07]
Vertex Cover	2	2-ε [KR06]
Max k-CSP	$\Omega(k/2^k)$[CMM07]	$O(k/2^k)$[ST,AM,GR]

What are Unique Games?

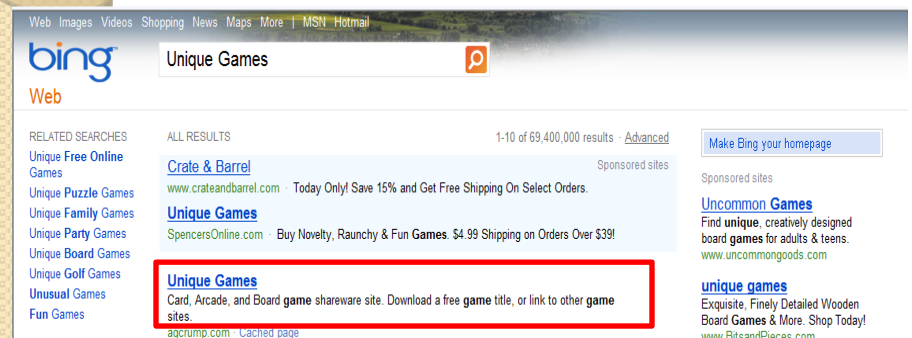
1. Unique Games are popular not only among computer scientist!

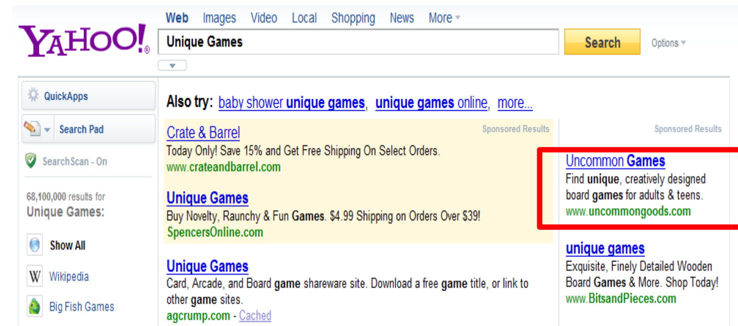


bing: 70 million pages

What are Unique Games?

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





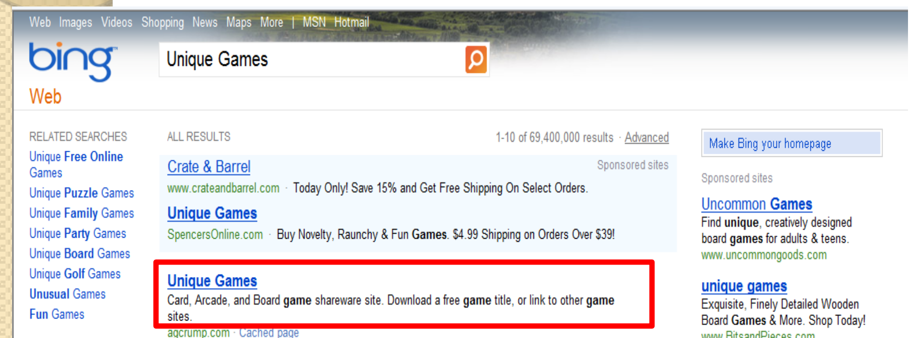
Yahoo!: 69 million pages

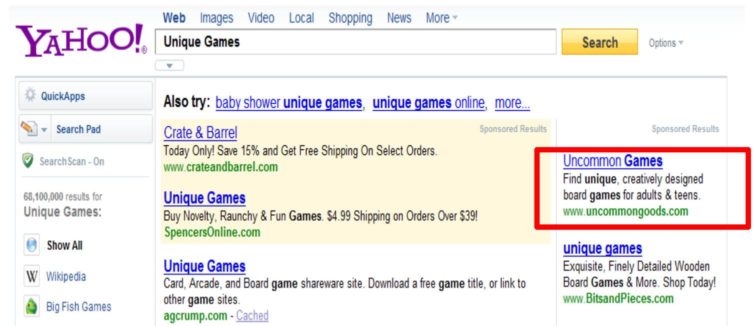
2. We can purchase Unique Games on-line!

bing: 70 million pages

What are Unique Games?

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





Yahoo!: 69 million pages

2. We can purchase Unique Games on-line!

bing: 70 million pages

3. Unique Games are related to the Unique Games Conjecture…

Google: 178 million pages

Unique Games = Unique Label Cover Problem

Given: set of constraints

Linear Equations mod k :

$x_i - x_j = c_{ij} \bmod k$

GOAL

k ="alphabet" size

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

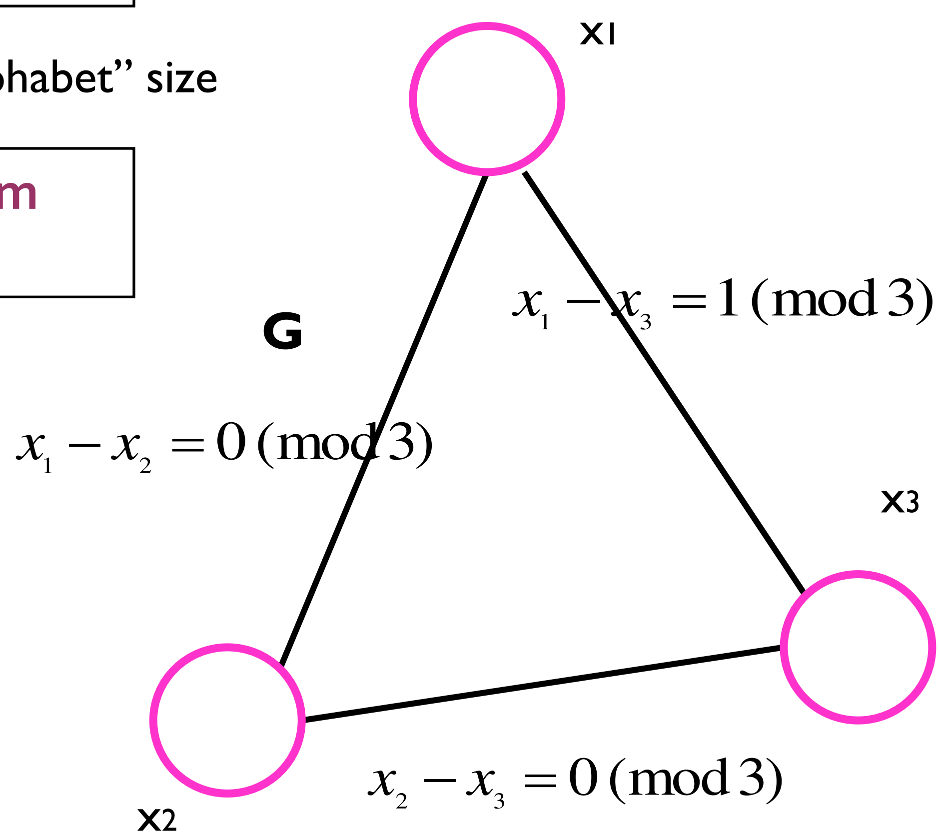
EXAMPLE

$$x_1 - x_2 = 0 \pmod 3$$

$$x_2 - x_3 = 0 \pmod 3$$

$$x_1 - x_3 = 1 \pmod 3$$

The constraint graph



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.**

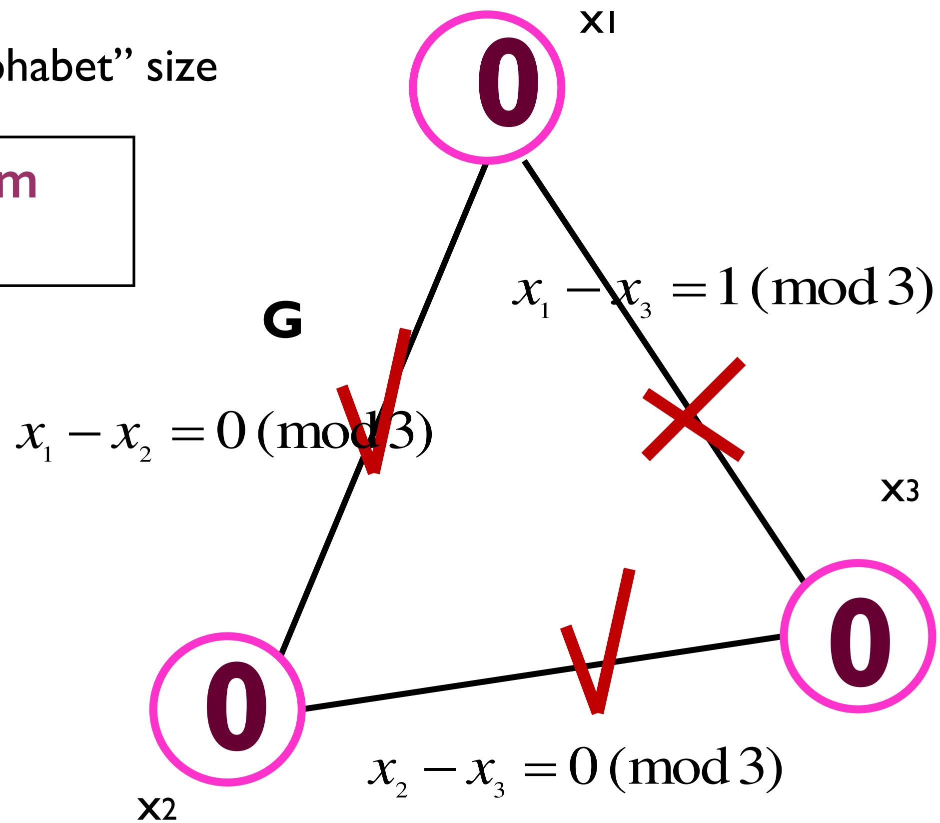
EXAMPLE

$x_1 - x_2 = 0 \pmod 3$ ✓

$x_2 - x_3 = 0 \pmod 3$ ✓

$x_1 - x_3 = 1 \pmod 3$ ✗

The constraint graph



Satisfy 2/3 constraints

Unique Games , an Example

Given: set of constraints

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

GOAL

k ="alphabet" size

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

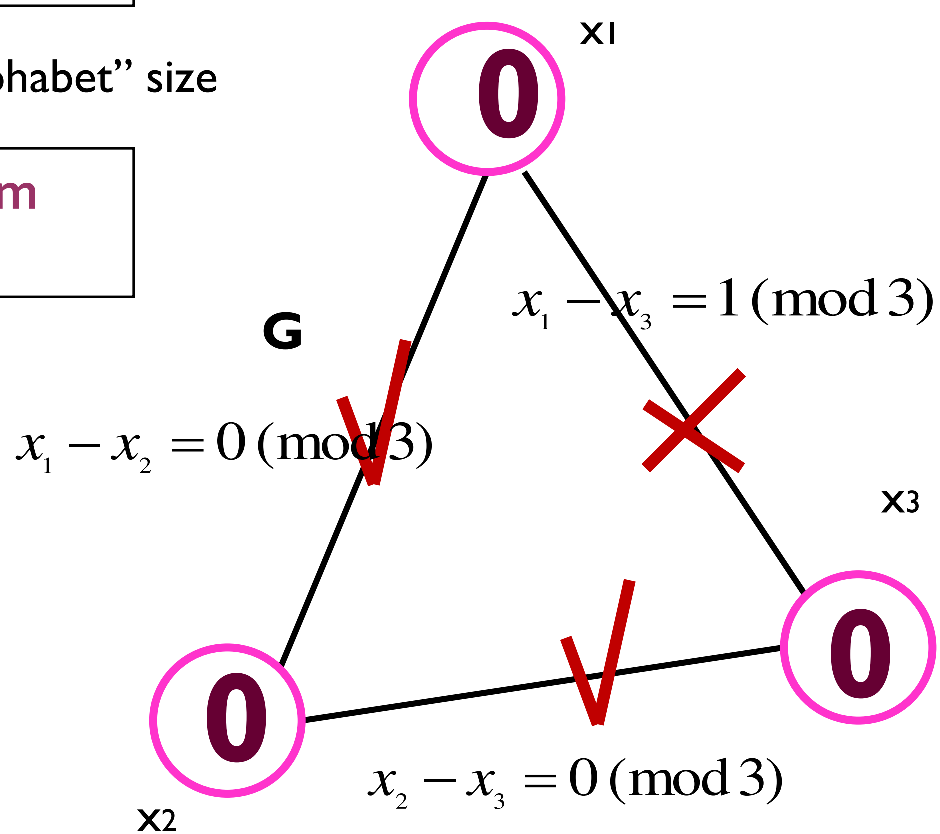
EXAMPLE

$x_1 - x_2 = 0 \pmod 3$ ✓

$x_2 - x_3 = 0 \pmod 3$ ✓

$x_1 - x_3 = 1 \pmod 3$ ✗

The constraint graph



Rest of the talk: d-regular graphs

Unique Games Conjecture

- [Khot'02] For every positive ε and δ there is a large enough k s.t. for some instance of Unique Games with alphabet size k and OPT > 1 – ε, it is NP hard to satisfy a δ 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.

Easy Instances

- 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]

Easy Distributions

- Find distributions that are hard?
  - Random Instances : NO! Follows from expander result.
  - Quasi-Random Instances? [KMM'10] NO!

Summary: Algorithmic Results for UG

General Graphs

Algorithm	On 1-ε instances
Khot	$1-O(k^2 \varepsilon^{1/5}\sqrt{\log(1/\varepsilon)})$
Trevisan	$1-O(\sqrt[3]{\varepsilon \log n})$
Gupta-Talwar	$1-O(\varepsilon \log n)$
CMM1	$k^{-\varepsilon/2-\varepsilon}$
CMM2	$1-O(\varepsilon \sqrt{\log n} \sqrt{\log k})$

SDP/LP based

Tight for SDP, there is counterexample

Special Graphs

Expander

AKKTSV'08 KT'08, MM'10	Constant, depends on conductance

Local expander

AIMS'09, SR'09	Constant, depends on local expansion

Almost all above approaches were LP or SDP based

Summary: Algorithmic Results for UG

General Graphs

Algorithm	On 1-ε instances
Khot	$1-O(k^2 \varepsilon^{1/5}\sqrt{\log(1/\varepsilon)})$
Trevisan	$1-O(\sqrt[3]{\varepsilon \log n})$
Gupta-Talwar	$1-O(\varepsilon \log n)$
CMM1	$k^{-\varepsilon/2-\varepsilon}$
CMM2	$1-O(\varepsilon \sqrt{\log n} \sqrt{\log k})$

SDP/LP based

Tight for SDP, there is counterexample

Special Graphs

Special graph	Algorithm	On 1-ε instances
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

Purely SPECTRAL Approach "beats" SDP

Summary: Algorithmic Results for UG

General Graphs

Algorithm	On $1-\varepsilon$ instances
Khot	$1-O(k^2\, \varepsilon^{1/5}\sqrt{\log(1/\varepsilon)})$
Trevisan	$1-O(\sqrt[3]{\varepsilon \log n})$
Gupta-Talwar	$1-O(\varepsilon \log n)$
CMM1	$k^{-\varepsilon/2-\varepsilon}$
CMM2	$1-O(\varepsilon \sqrt{\log n} \sqrt{\log k})$

Special Graphs

Special Graphs	Algorithm	Result
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

ABS'10: Subexponential time algorithm for ANY instance

Summary: Algorithmic Results for UG

Graph class	Algorithm	On 1-ε instances
General Graphs	Khot	$1-O(k^2 \varepsilon^{1/5}\sqrt{\log(1/\varepsilon)})$
	Trevisan	$1-O(\sqrt[3]{\varepsilon \log n})$
	Gupta-Talwar	$1-O(\varepsilon \log n)$
	CMM1	$k^{-\varepsilon/2-\varepsilon}$
	CMM2	$1-O(\varepsilon \sqrt{\log n} \sqrt{\log k})$

Special Graphs

Graph class	Algorithm	On 1-ε instances
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

ABS'10: Subexponential time algorithm for ANY instance

Summary: Algorithmic Results for UG

General Graphs

Algorithm	On 1-ε instances
Khot	$1-O(k^2\ \varepsilon^{1/5}\sqrt{\log(1/\varepsilon)})$
Trevisan	$1-O(\sqrt[3]{(\varepsilon \log n)})$
Gupta-Talwar	$1-O(\varepsilon \log n)$
CMM1	$k^{-\varepsilon/2-\varepsilon}$
CMM2	$1-O(\varepsilon \sqrt{\log n} \sqrt{\log k})$

Special Graphs

Graphs	Algorithm	Result
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

KMM'10: Semi-Random instances are easy

instance

Unique Games = Unique Label Cover Problem

Given: set of constraints

Linear Equations mod k :

$x_i - x_j = c_{ij} \bmod k$

GOAL

k =“alphabet” size

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

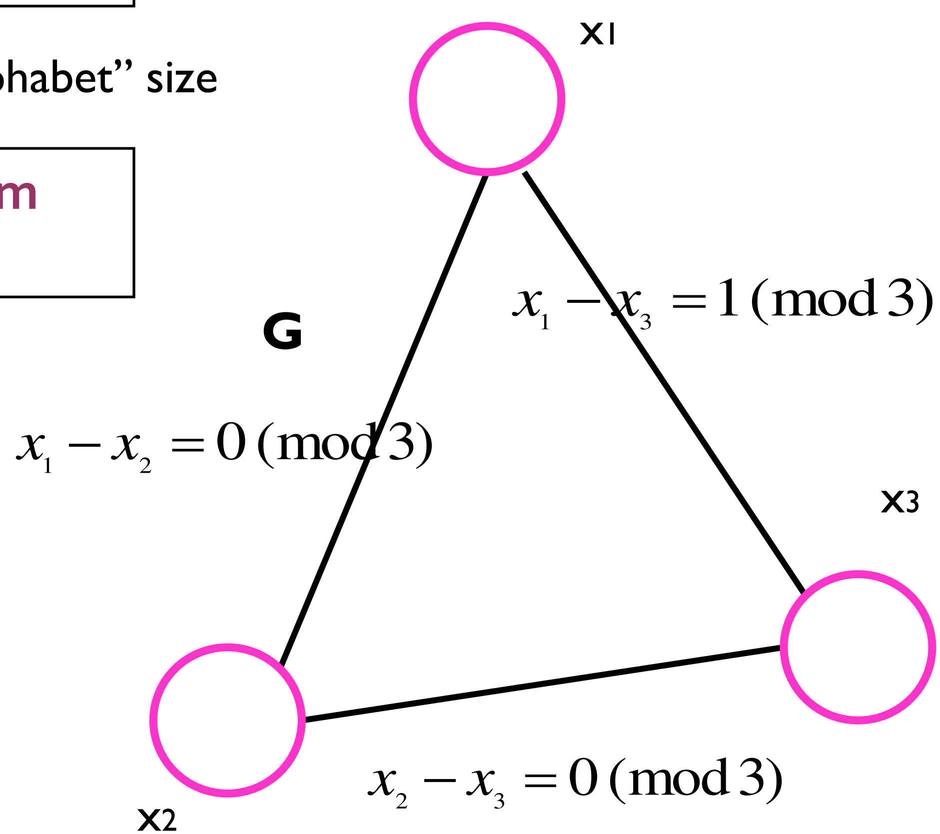
EXAMPLE

$$x_1 - x_2 = 0 \pmod 3$$

$$x_2 - x_3 = 0 \pmod 3$$

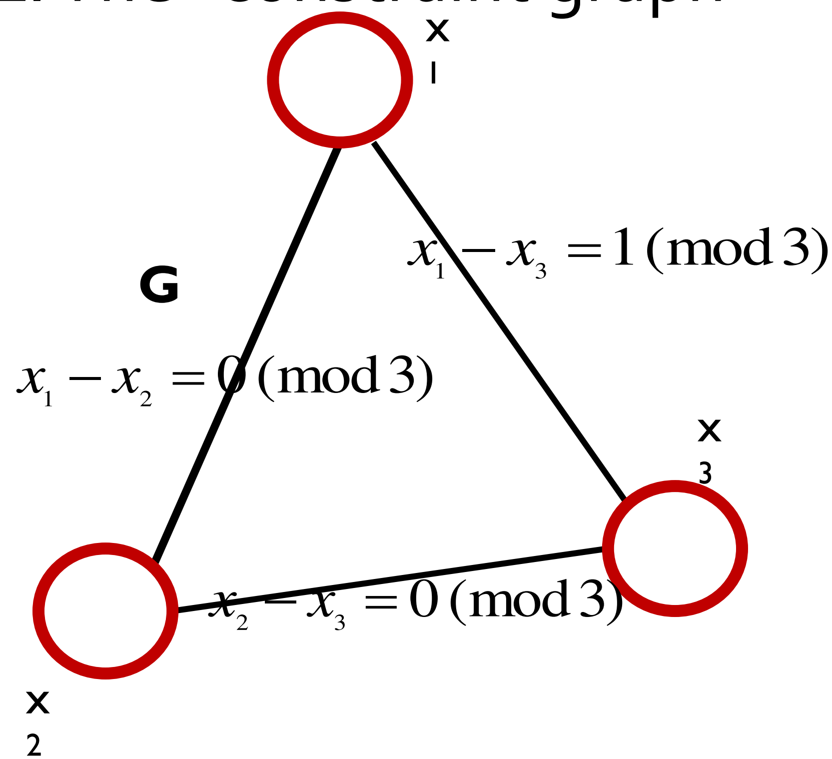
$$x_1 - x_3 = 1 \pmod 3$$

The constraint graph

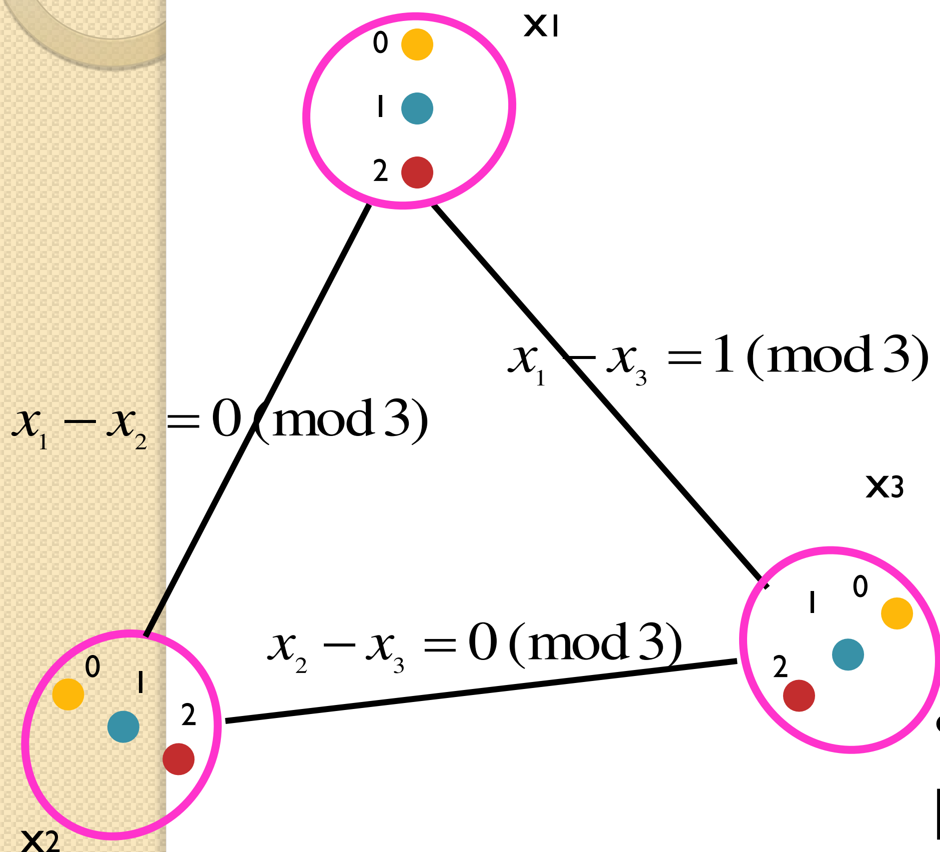


Unique Games and Graphs

1. The "constraint graph"



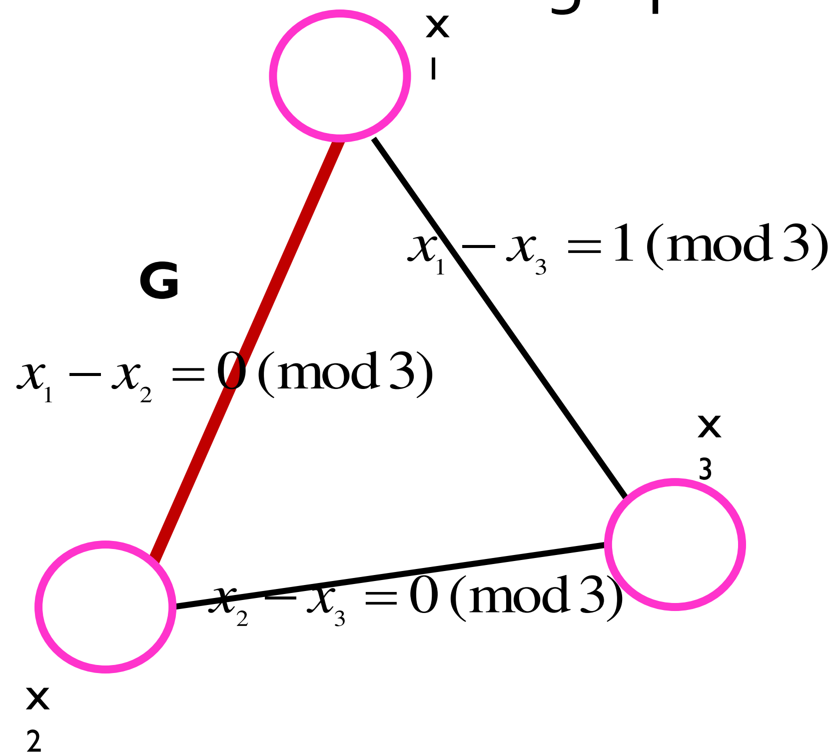
2. The "label-extended" graph



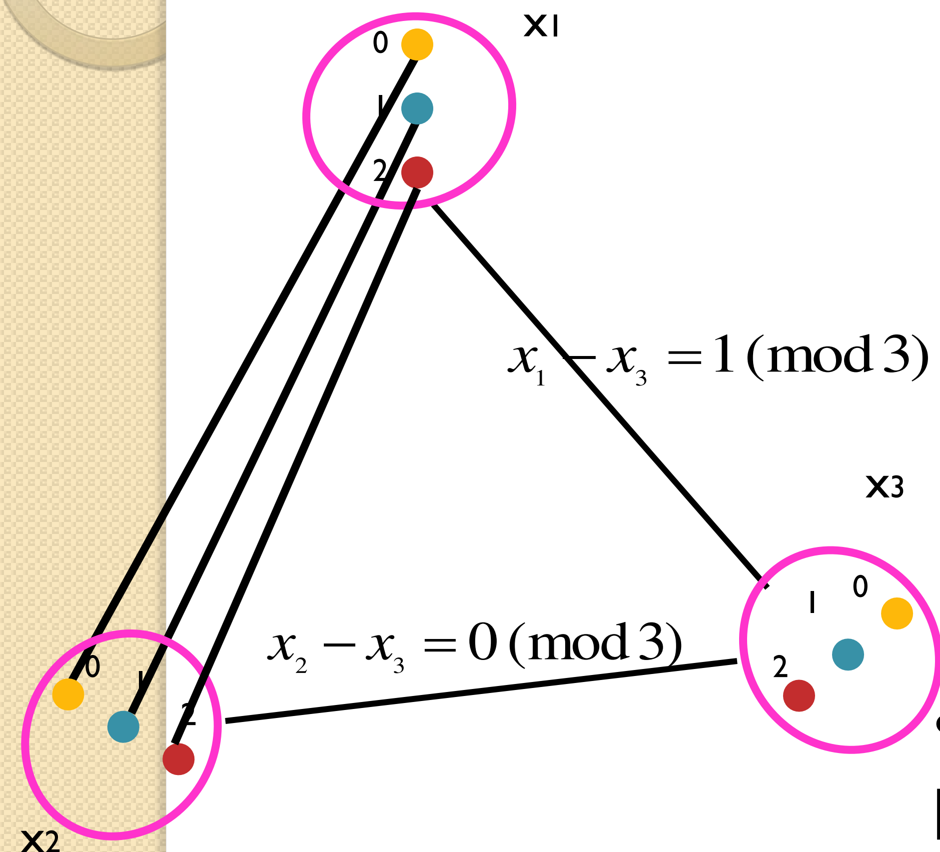
- Replace each vertex with k vertices- one for each label

Unique Games and Graphs

1. The "constraint graph"



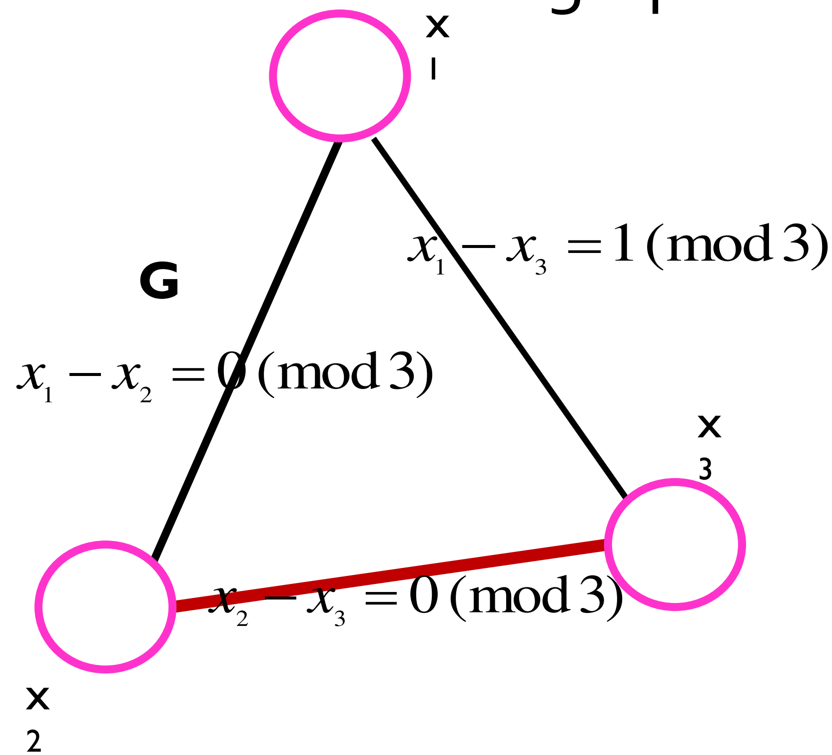
2. The "label-extended" graph



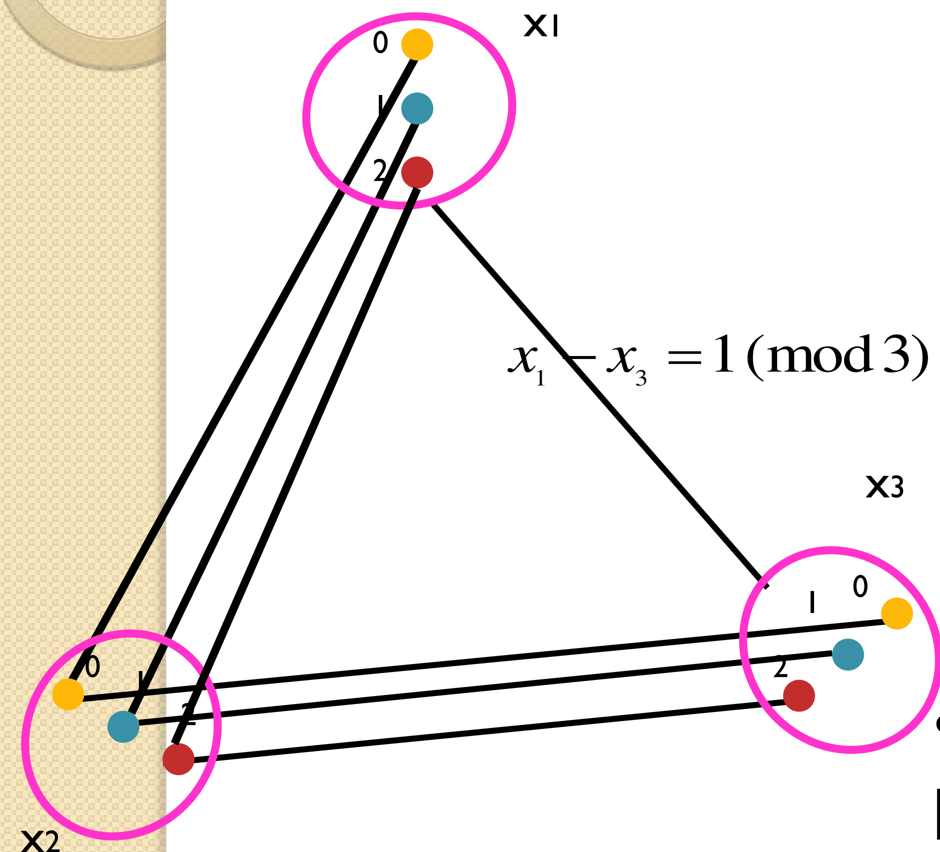
- Replace each vertex with k vertices- one for each label
- Replace each edge with the "permutation matching"

Unique Games and Graphs

1. The "constraint graph"



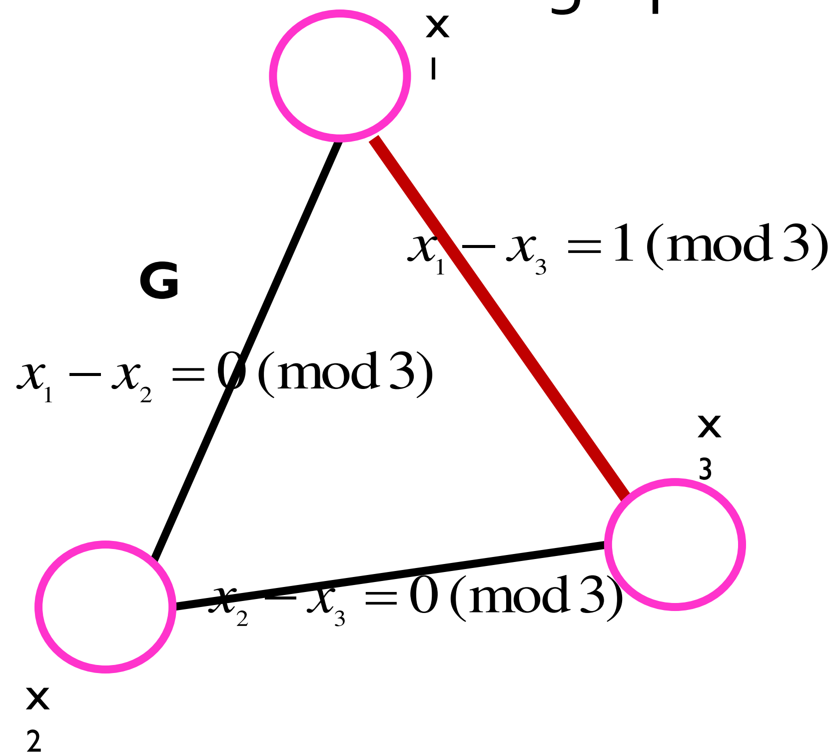
2. The "label-extended" graph



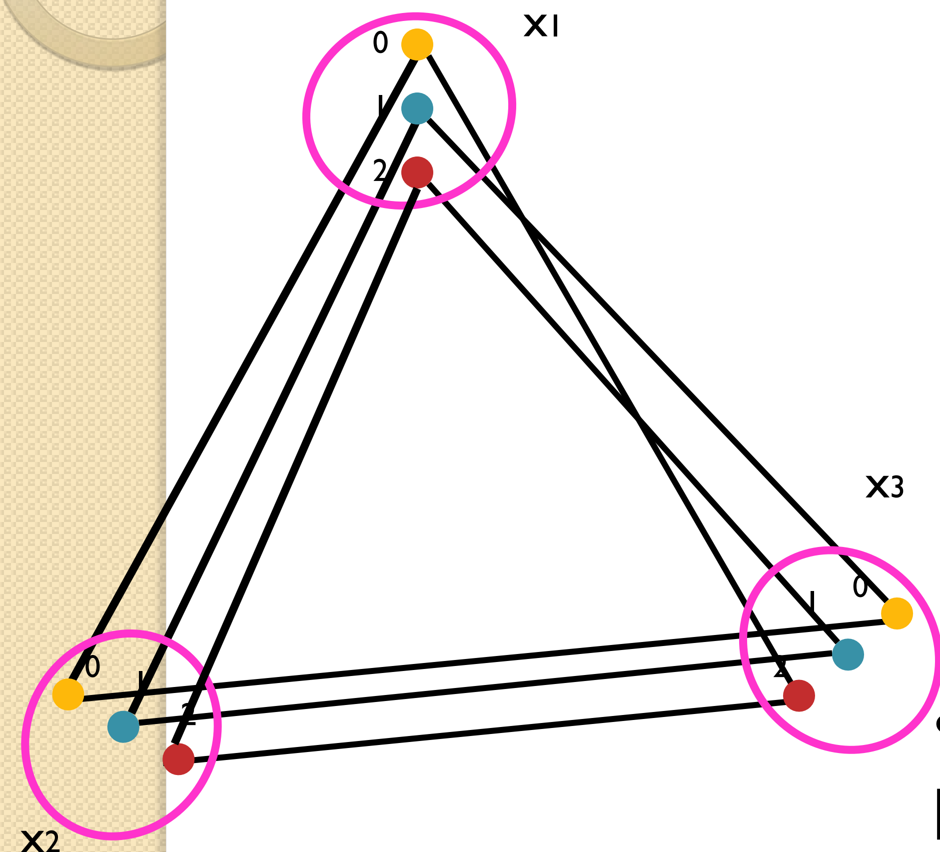
- Replace each vertex with k vertices- one for each label
- Replace each edge with the "permutation matching"

Unique Games and Graphs

1. The "constraint graph"



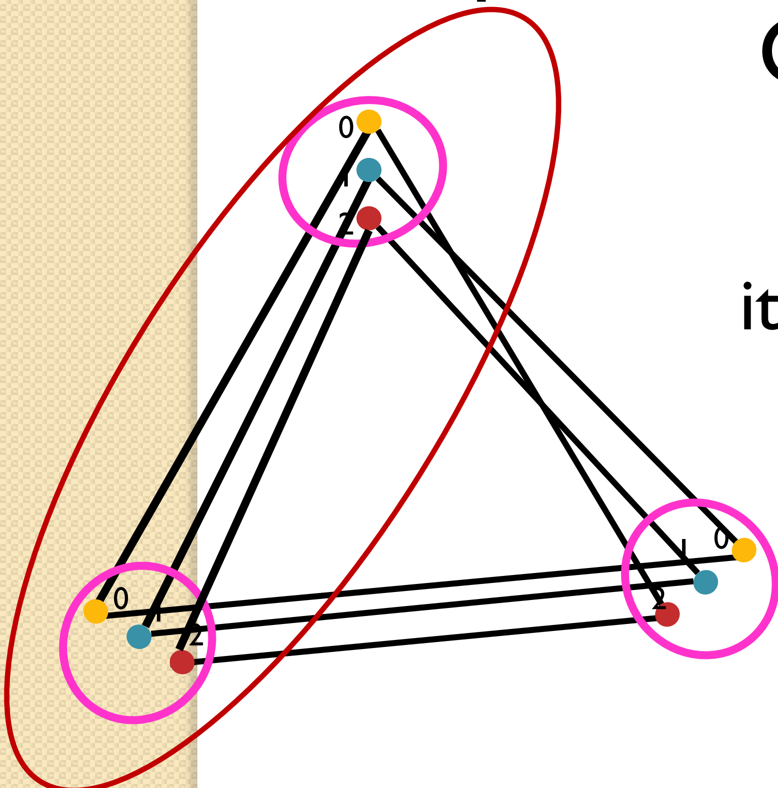
2. The "label-extended" graph



•Replace each vertex with k vertices- one for each label

•Replace each edge with the "permutation matching"

More Graph Theory: The Label-Extended Graph



GRAPH THEORY?

it's a graph, it has adjacency matrix!

0	0	0	1	0	0			
0	0	0	0	1	0			
0	0	0	0	0	1			

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

Sketch UGC False on Expanders

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

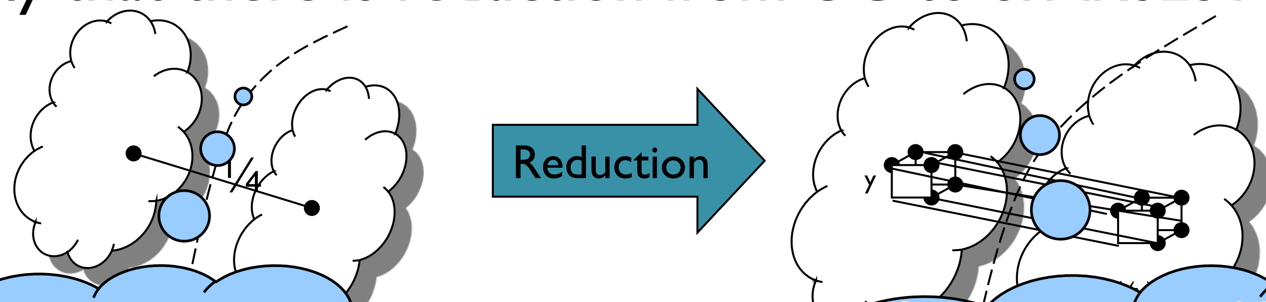
Problem	Best Approximation Algorithm Known	UGC-Hardness
MaxCut	0.878[GW94]	0.878 [KKMO07]
Vertex Cover	2	2-ε [KR06]
Max k-CSP	Ω(k/2^k)[CMM07]	O(k/2^k)[ST,AM,GR]
Uniform Sparsest		

No hardness even assuming UGC unless expansion

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

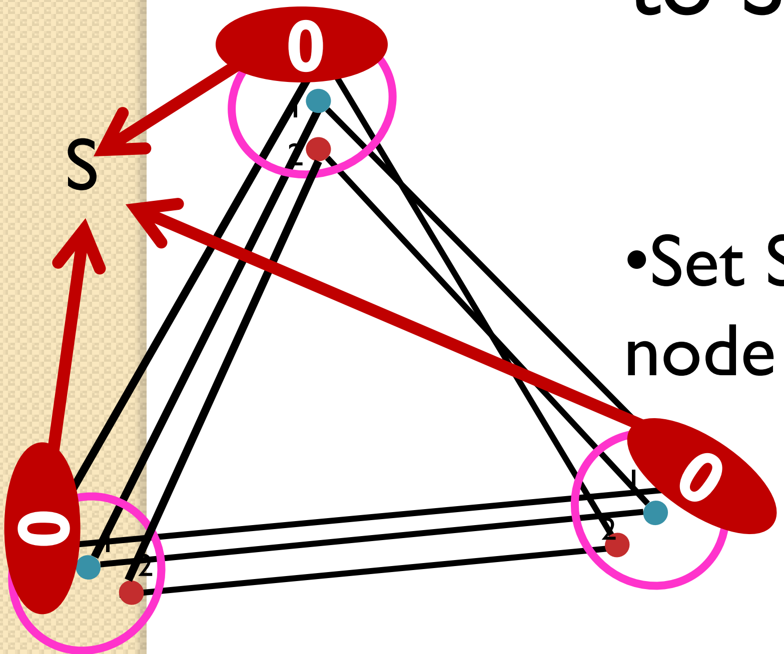
Get an instance of SC that has a sparse cut irrespective of UG being YES or NO instance.

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

Because then any sparse cut would correspond to a good labeling

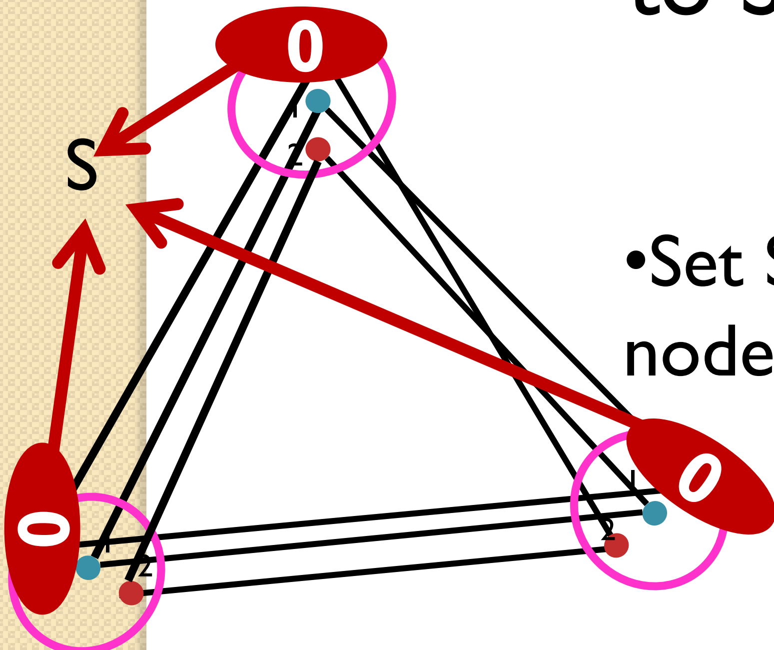
Off-the-record belief that expanders were hardest instances

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



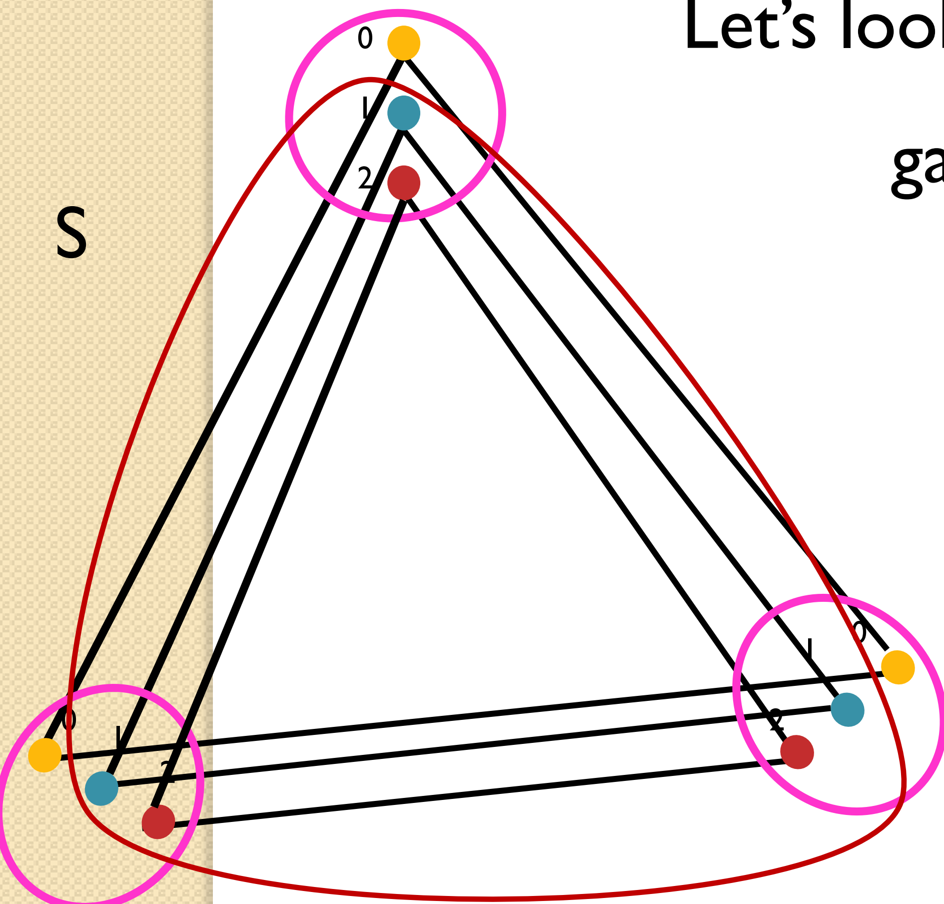
- Set S that contains exactly one "small" node from each node group = labeling

- Corresponds to a cut (S,S').

- Corresponds to a "characteristic vector".

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

Proof Intuition: a Perfect Game



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

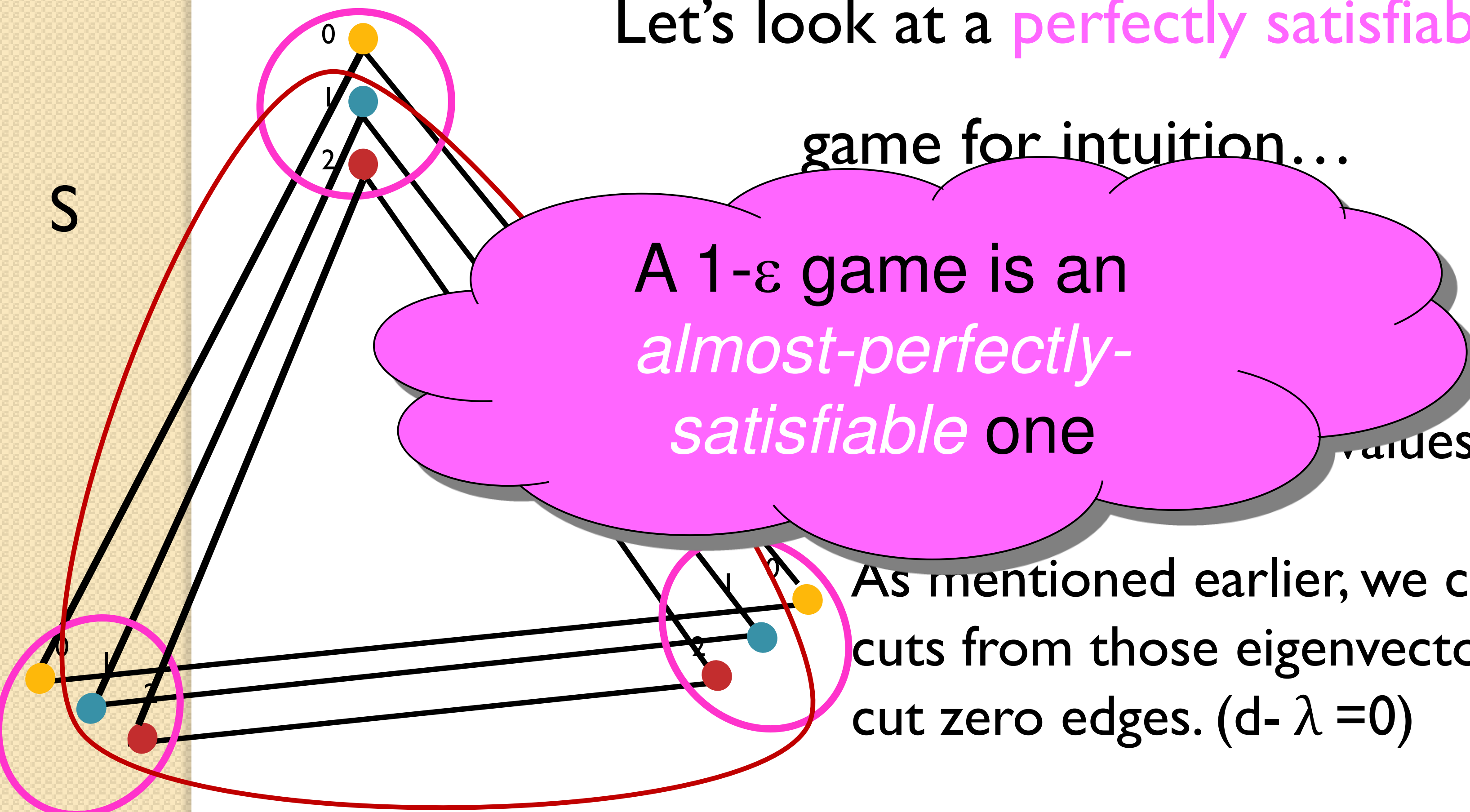
Graph is disconnected,
it has second eigenvalue λ = d
(in fact, it has k eigenvalues = d)

As mentioned earlier, we can find cuts from those eigenvectors that cut zero edges. (d- λ =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…



= d

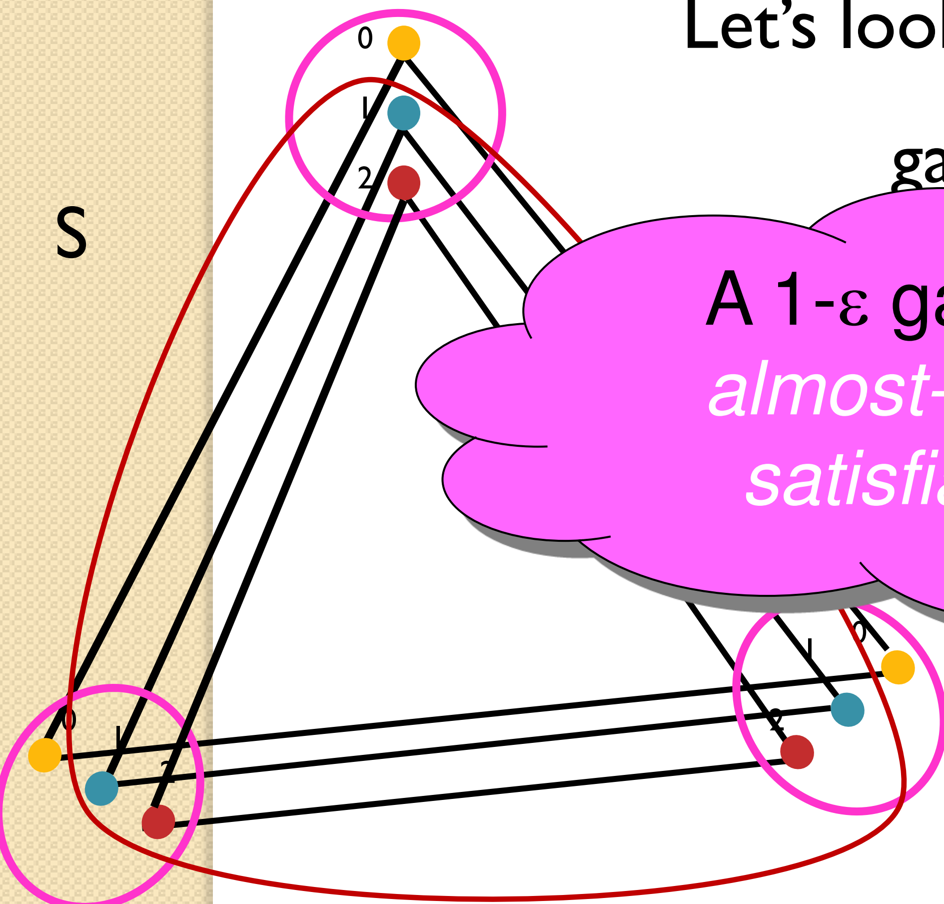
values = d)

As mentioned earlier, we can find cuts from those eigenvectors that cut zero edges. (d- λ =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-ε game is an *almost-perfectly-satisfiable* one

= d

values = d)

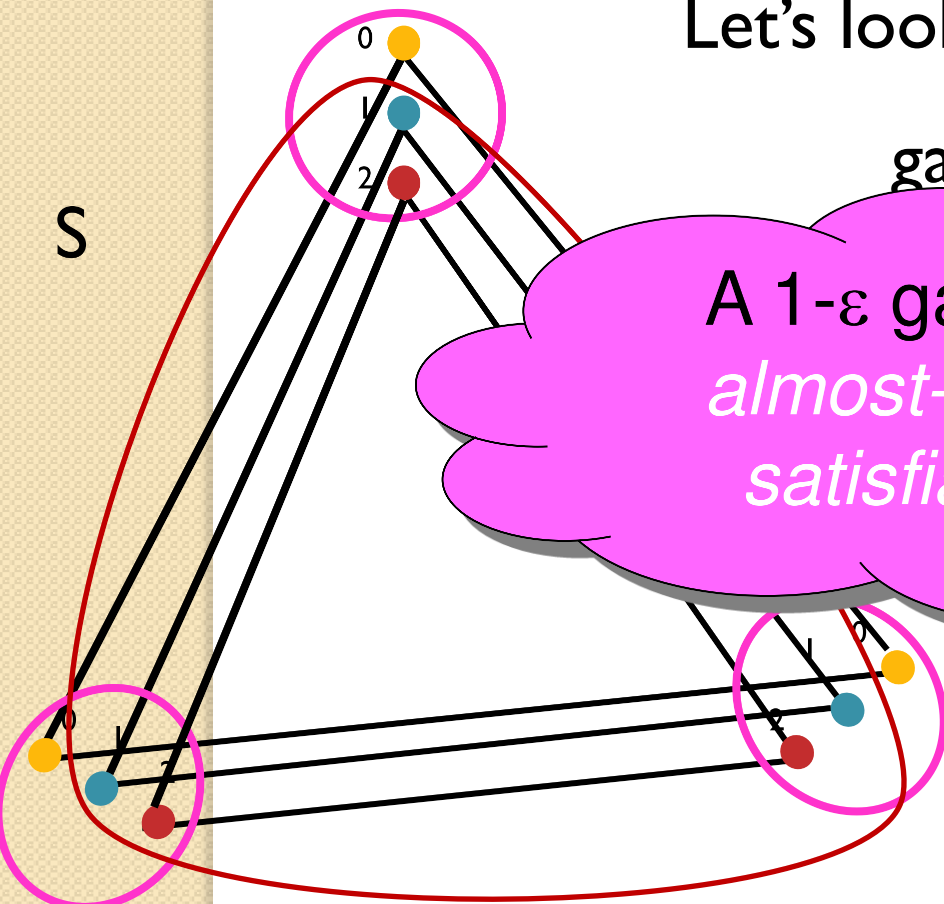
As mentioned earlier, we can find cuts from those eigenvectors that cut zero edges. (d- λ =0)

If graph G was originally ~~connected~~ expander, 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-ε game is an *almost-perfectly-satisfiable* one

= d

alues = d)

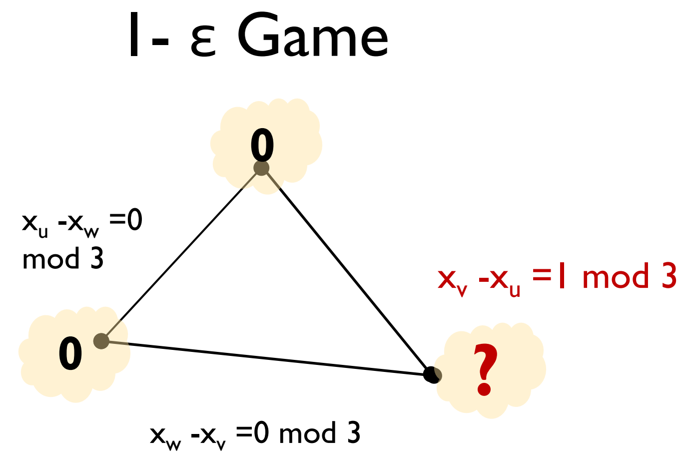
As mentioned earlier, we can find cuts from those eigenvectors that cut zero edges. (d- λ =0)

expander

If graph G was originally ~~connected~~, those are the only "sparsest cuts".

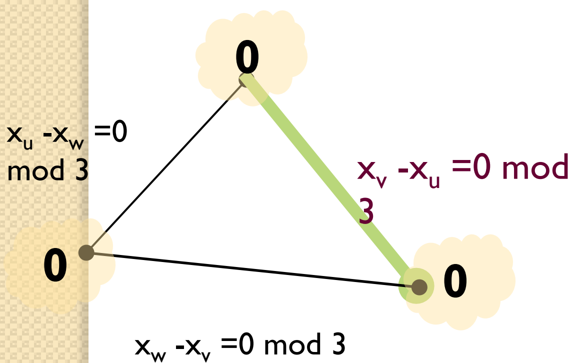
They correspond to almost-perfect labelings

Proof: Reverse Engineering + Graph Spectra

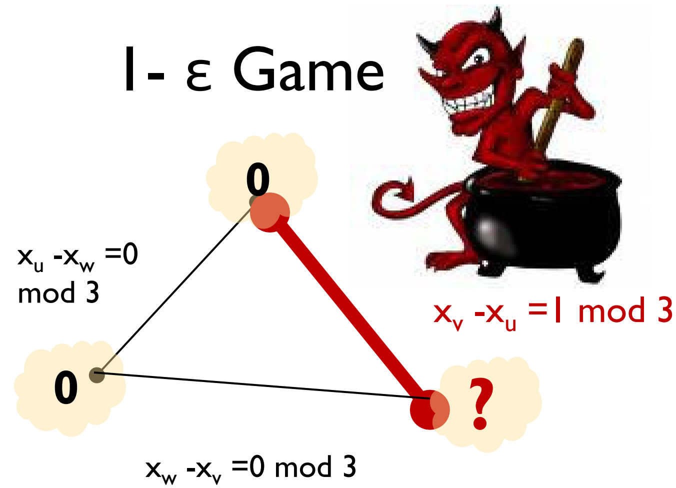


Proof: Reverse Engineering + Graph Spectra

Perfect Game:



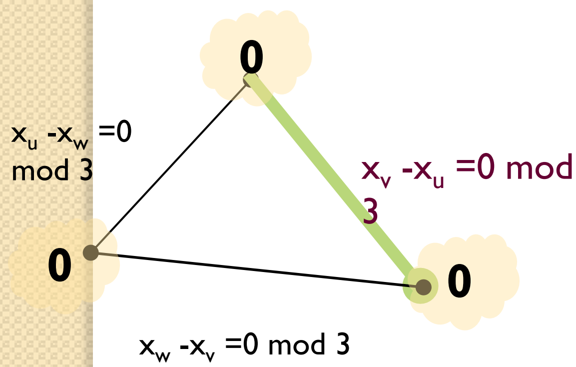
1- ε Game



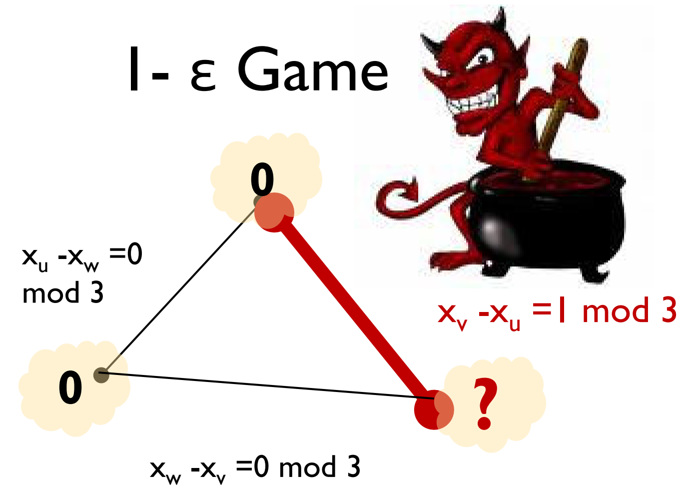
Think of it as "coming from" adversarialy perturbed completely satisfiable game

Proof: Reverse Engineering + Graph Spectra

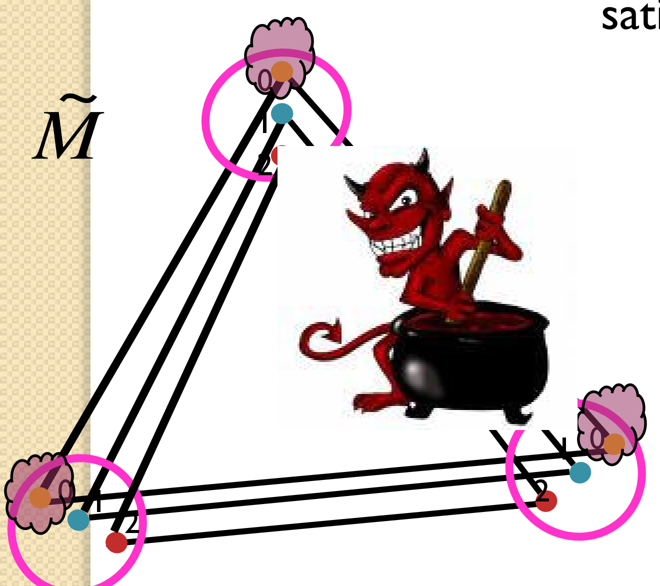
Perfect Game:

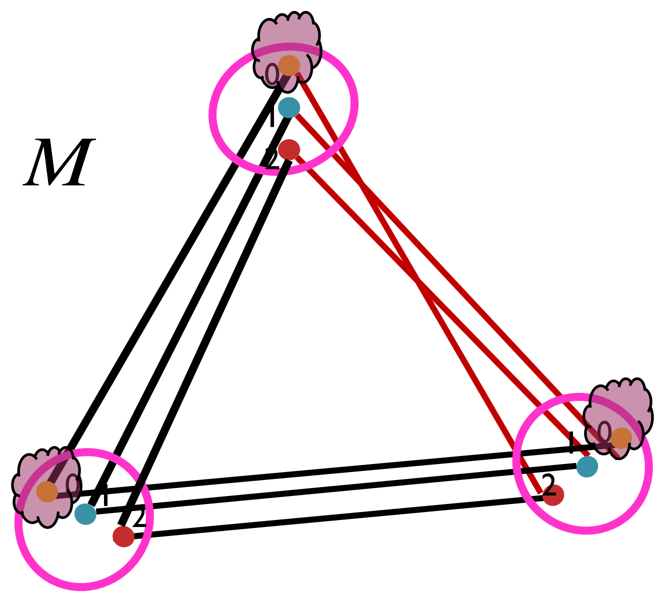


1- ε Game



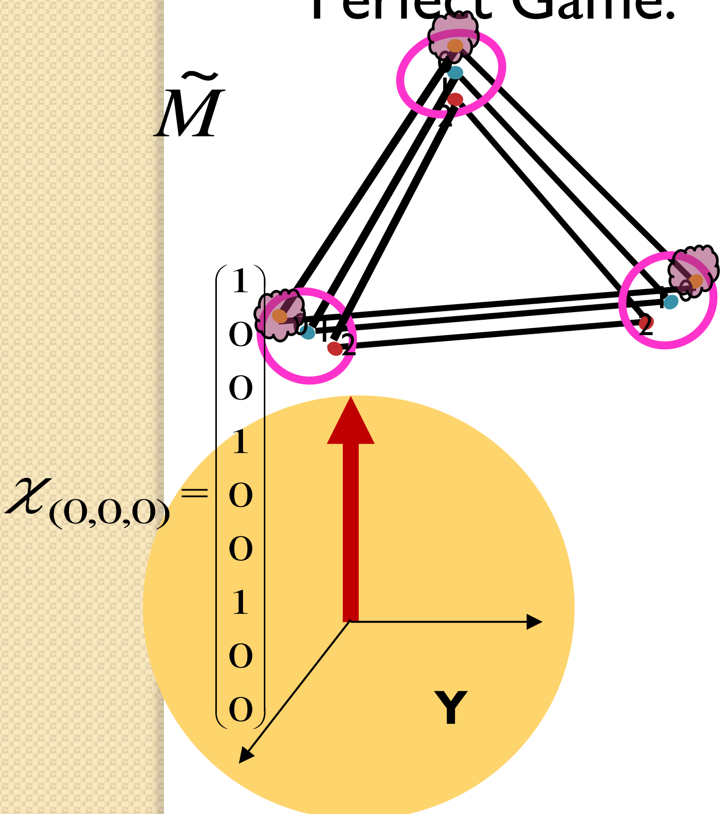
Think of it as "coming from" adversarialy perturbed completely satisfiable game





Proof: Reverse Engineering + Graph Spectra

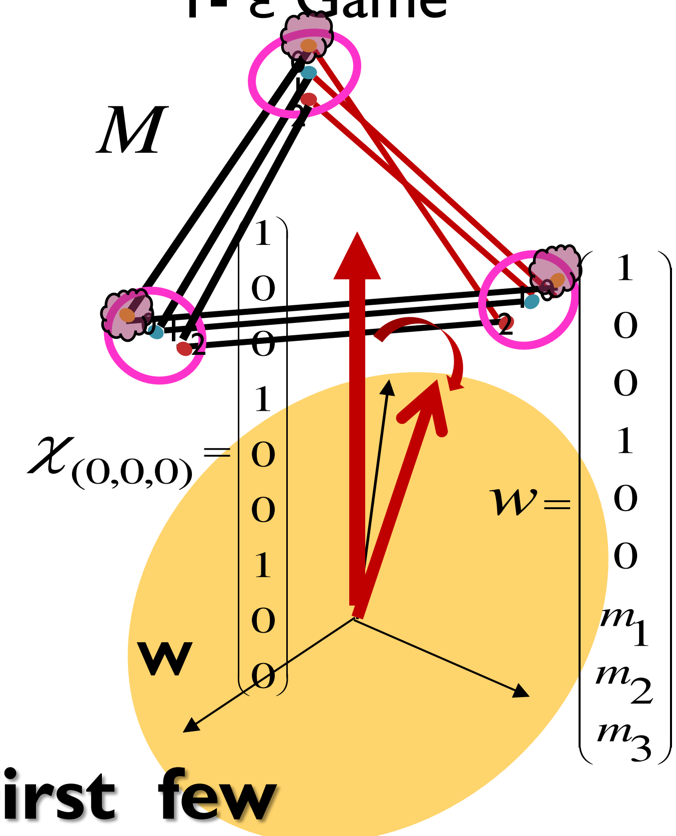
Perfect Game:



"Labeling" eigenvectors:

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

1- ε Game

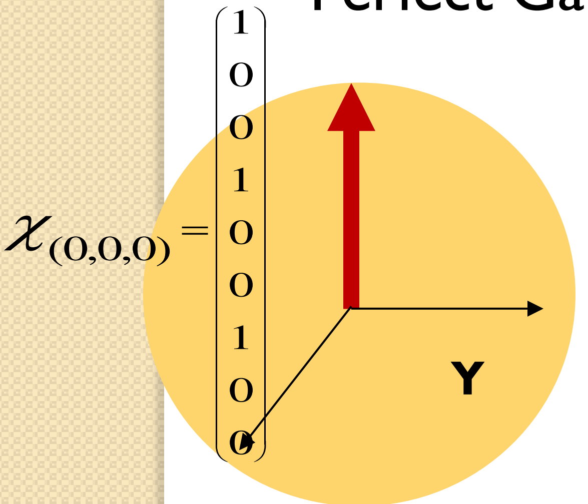


First few eigenvectors:

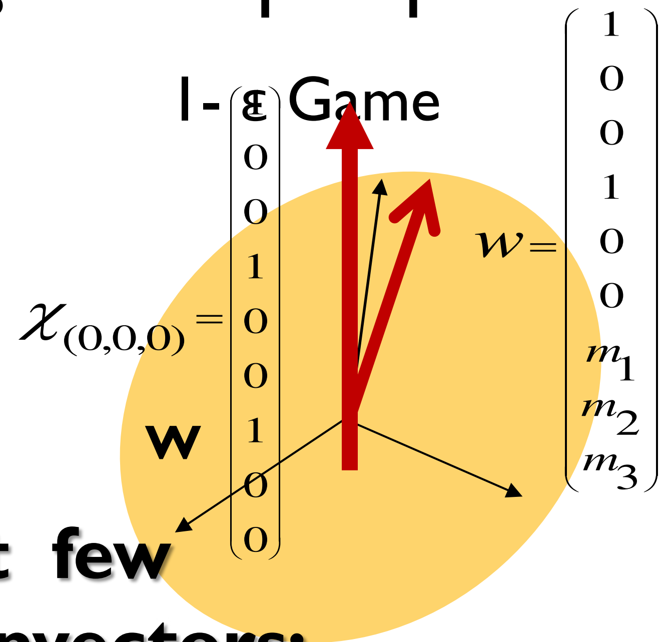
The k "labeling vectors" have large projection onto espace W with evalues >(1- 200ε)d

Proof: Reverse Engineering + Graph Spectra

Perfect Game:



1-ε Game



"Labeling" eigenvectors:

First few eigenvectors:

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

The k "labeling vectors" have large projection onto espace W with evalues >(1- 200ε)d

$for \|\chi\| = 1, \quad \chi^T \tilde{M} \chi = d$

$\chi^T M \chi \geq (1-2\varepsilon)d$

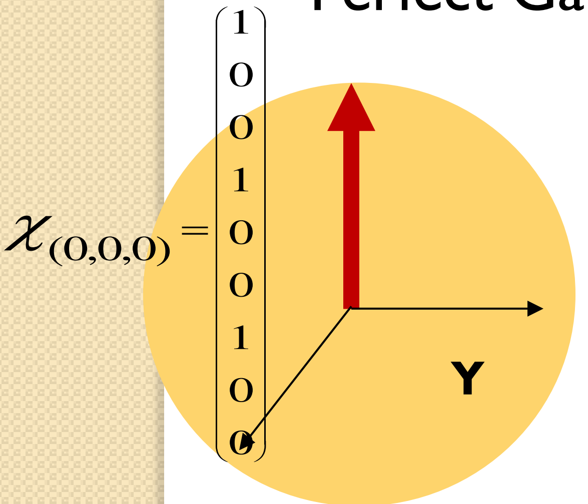
$$(1-2\varepsilon)d \leq \chi^T M \chi = a^2 w^T M w + \beta^2 w_\perp^T M w_\perp$$

Write: $\chi = \alpha w + \beta w_\perp$

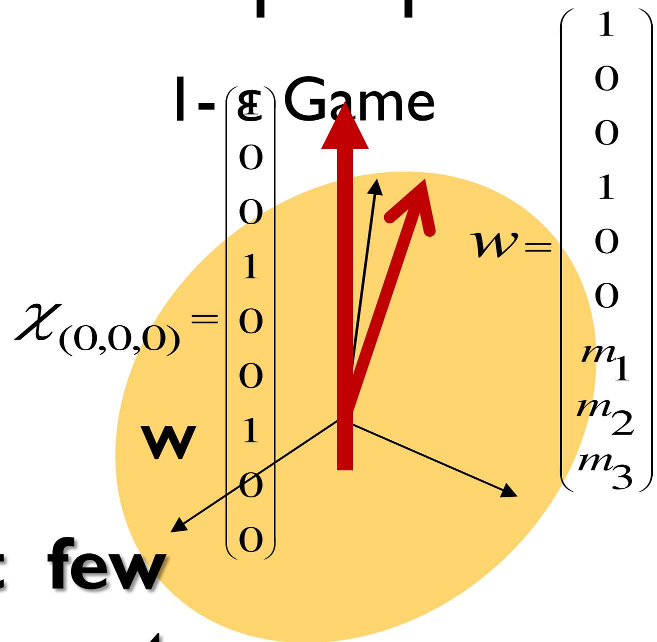
$$\leq a^2 d + \beta^2 (1-200\varepsilon)d \Rightarrow |\beta| \leq \frac{1}{10}$$

Proof: Reverse Engineering + Graph Spectra

Perfect Game:



1- ε Game



"Labeling" eigenvectors:

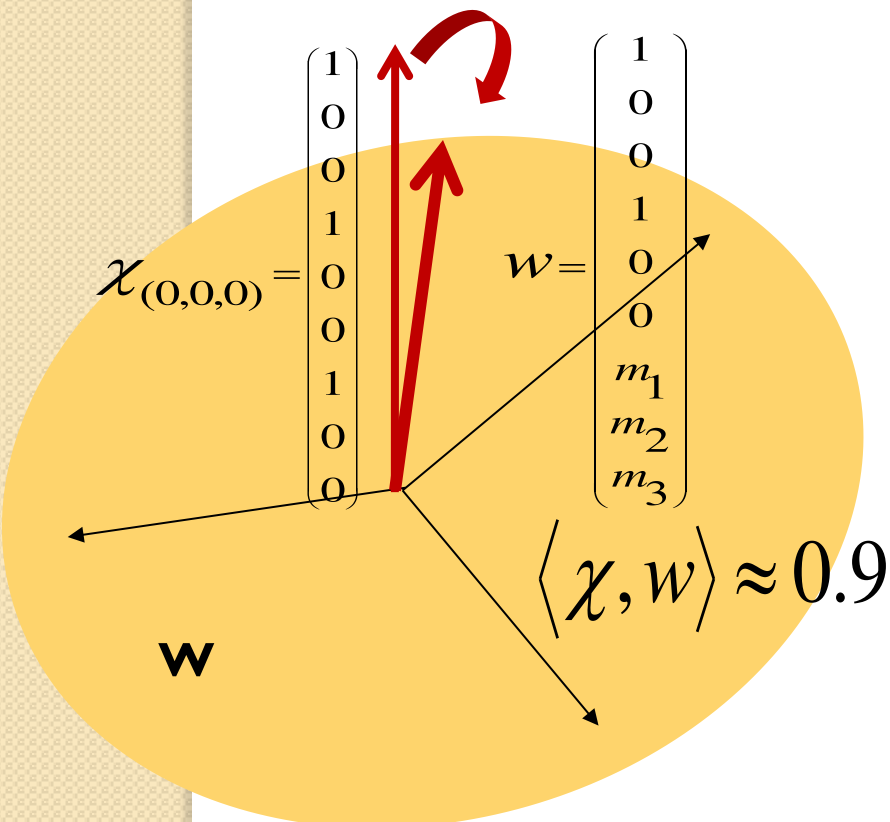
The k-dimensional espace Y of evalues equal to d contains all the information for the best labeling

First few eigenvectors:

The k "labeling vectors" have large projection onto espace W with evalues >(1- 200ε)d

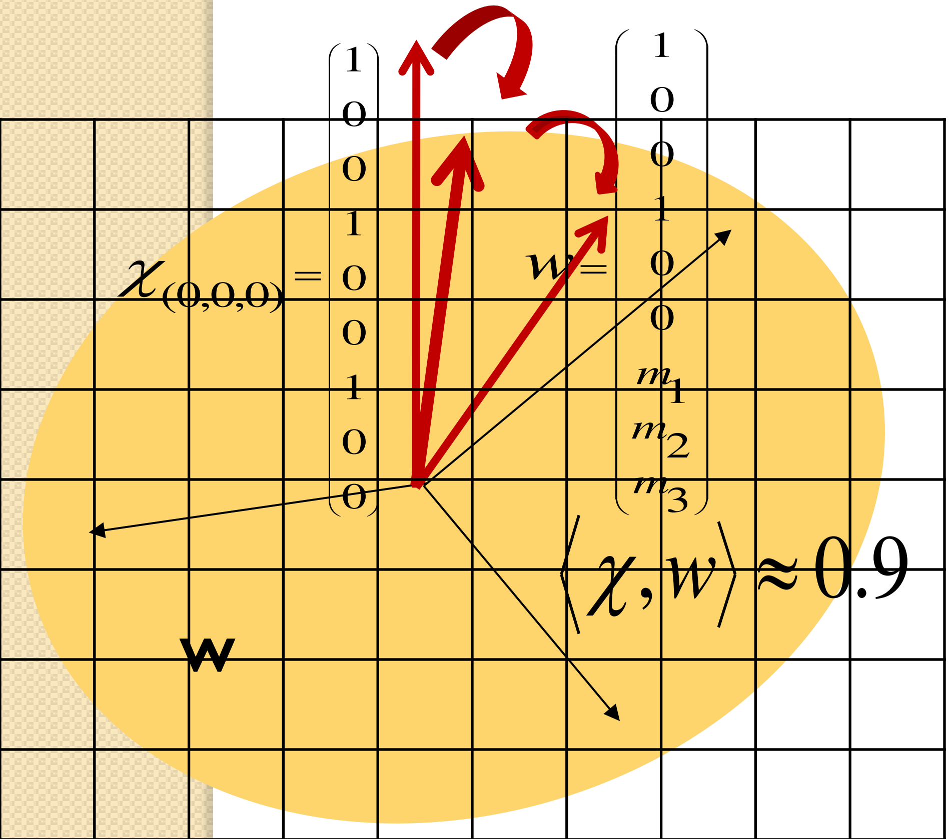
If we knew the projection w of χ then we could just "read off" a good labeling

Searching for a Needle in a Haystack?



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

Searching for a Needle, but "Efficiently"



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

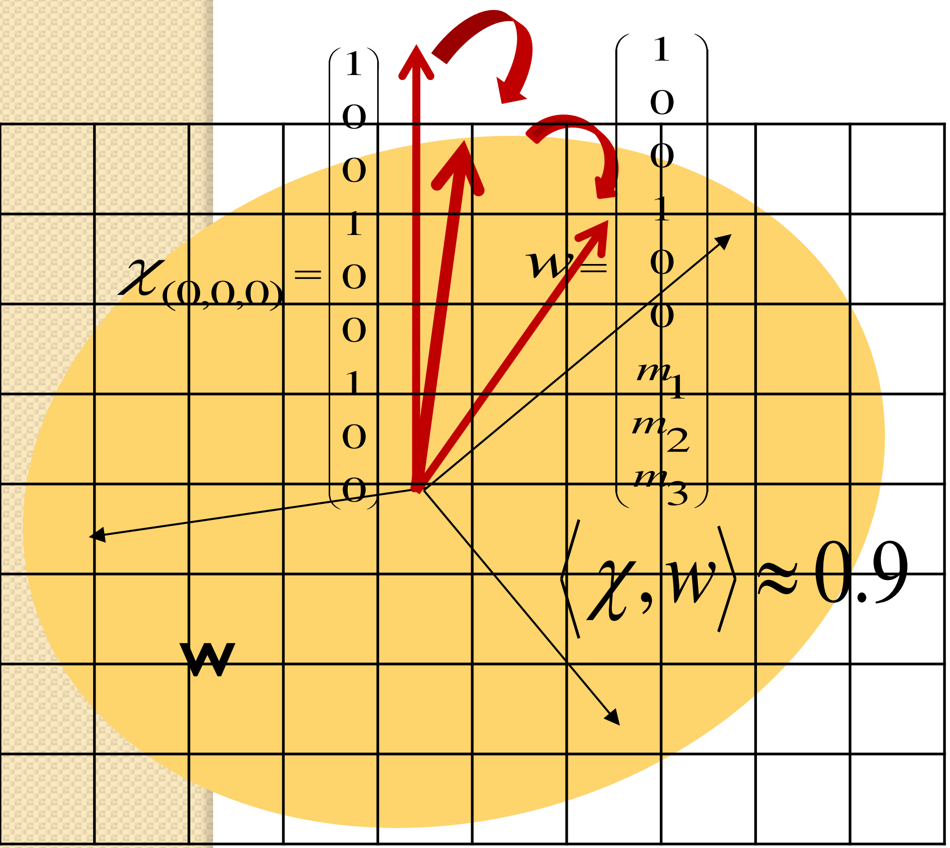
Idea:
Discretize the space by net!

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

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!

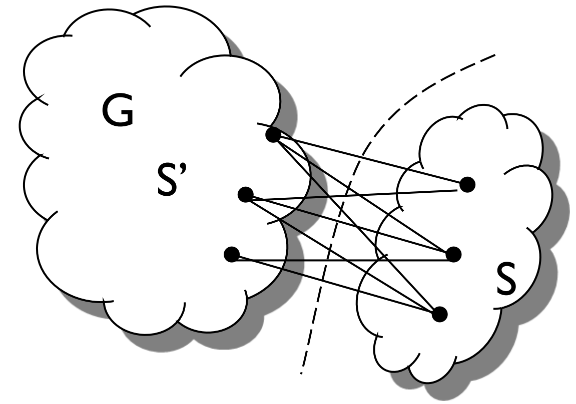
Algorithm runs in time ~ #points in the net

=

exponential in the dimension of eigenspace W

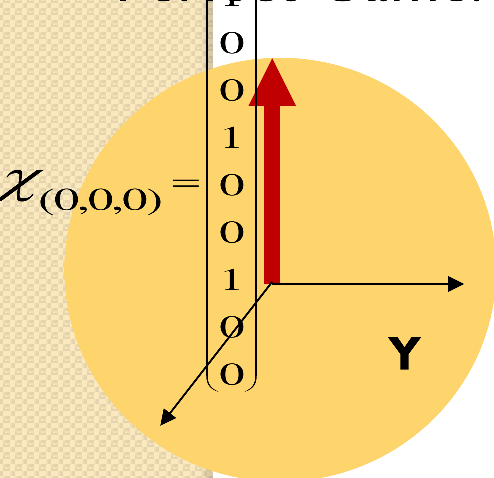
The Dimension of W for Expanders

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

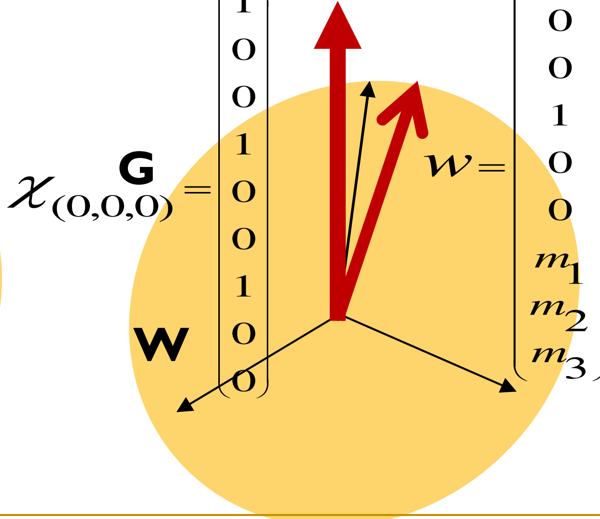


The Dimension of W for Expanders

Perfect Game:

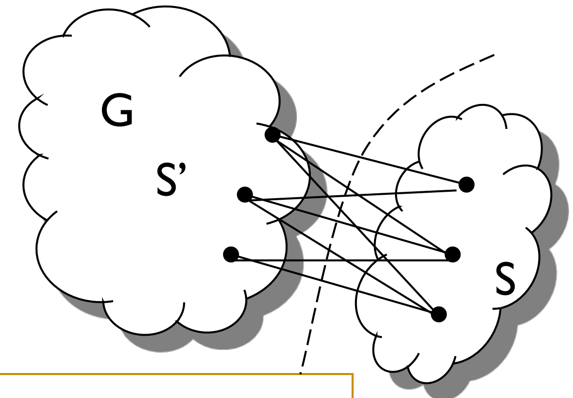


1- ε Game:



(Spectral Gap)=

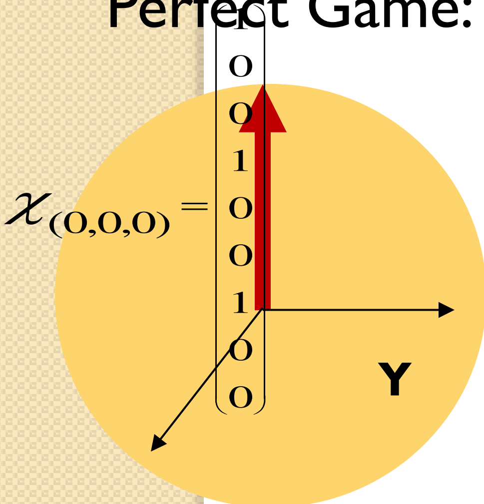
$d - \lambda = \gamma d$



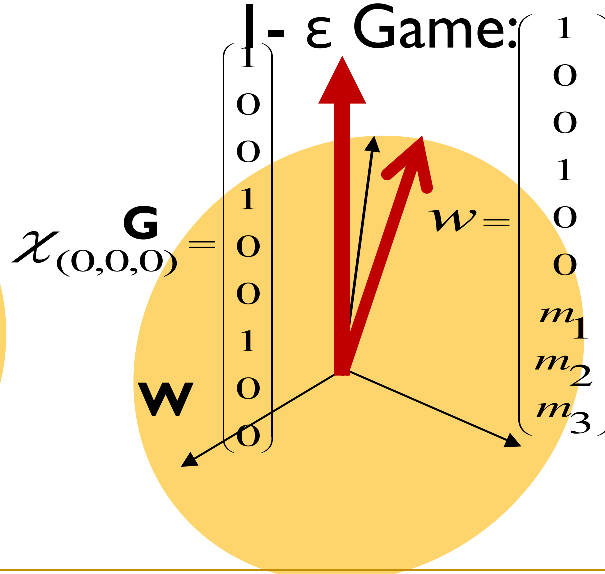
(Spectral gap between Y,$Y_\perp$)= absgap= γd

The Dimension of W for Expanders

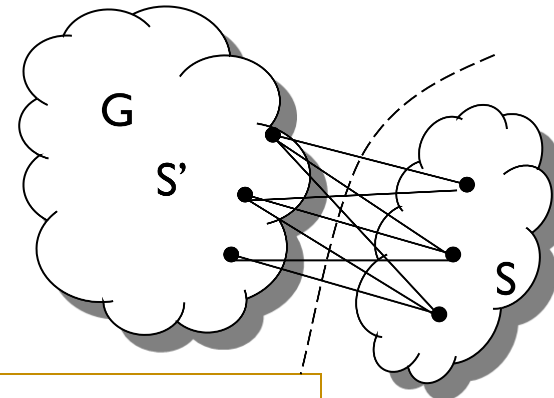
Perfect Game:



1- ε Game:



(Spectral Gap)=
d -λ = γ d



(Spectral gap between Y,Y⊥)= absgap= γd

W is "perturbed analog" of Y

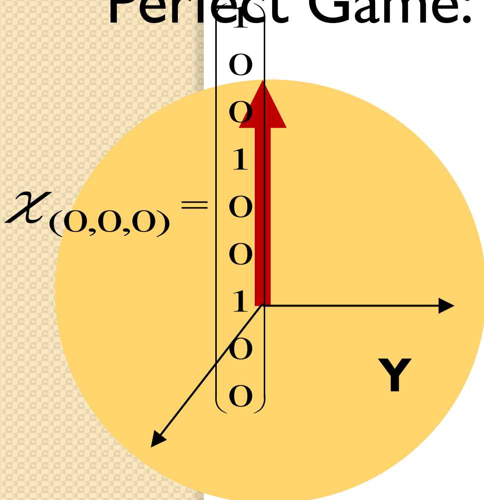
"The sin μ" Theorem [DK'70] : Angle between Y and "perturbed analog of Y" small

Equivalently, we can write every vector w in W as w = α y + β y⊥, y in Y

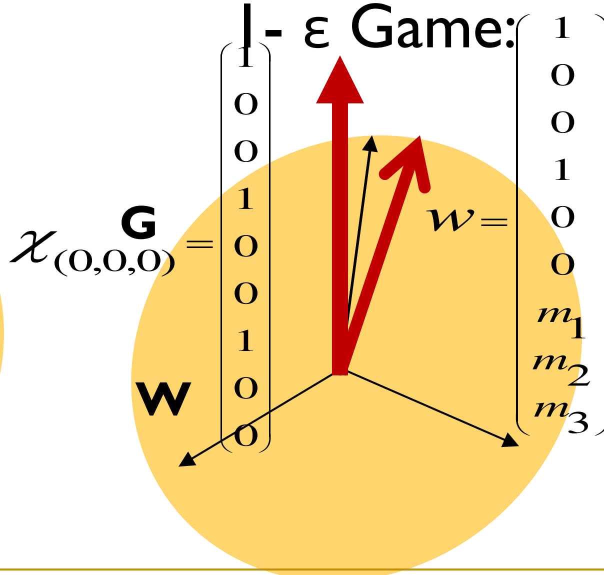
$$|\beta| \le \frac{\|(M - M_\epsilon)w\|}{absgap} \le O(\sqrt{\frac{\epsilon}{\gamma^3}})$$

The Dimension of W for Expanders

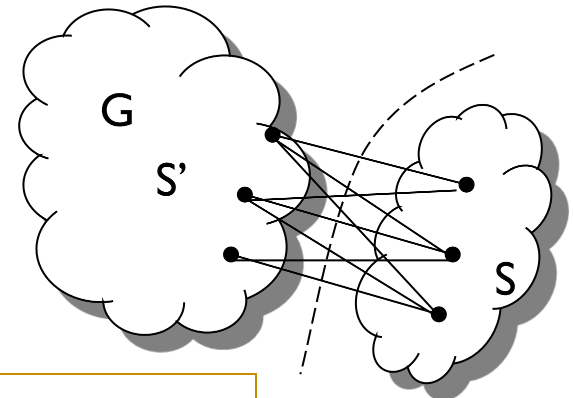
Perfect Game:



1- ε Game:



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



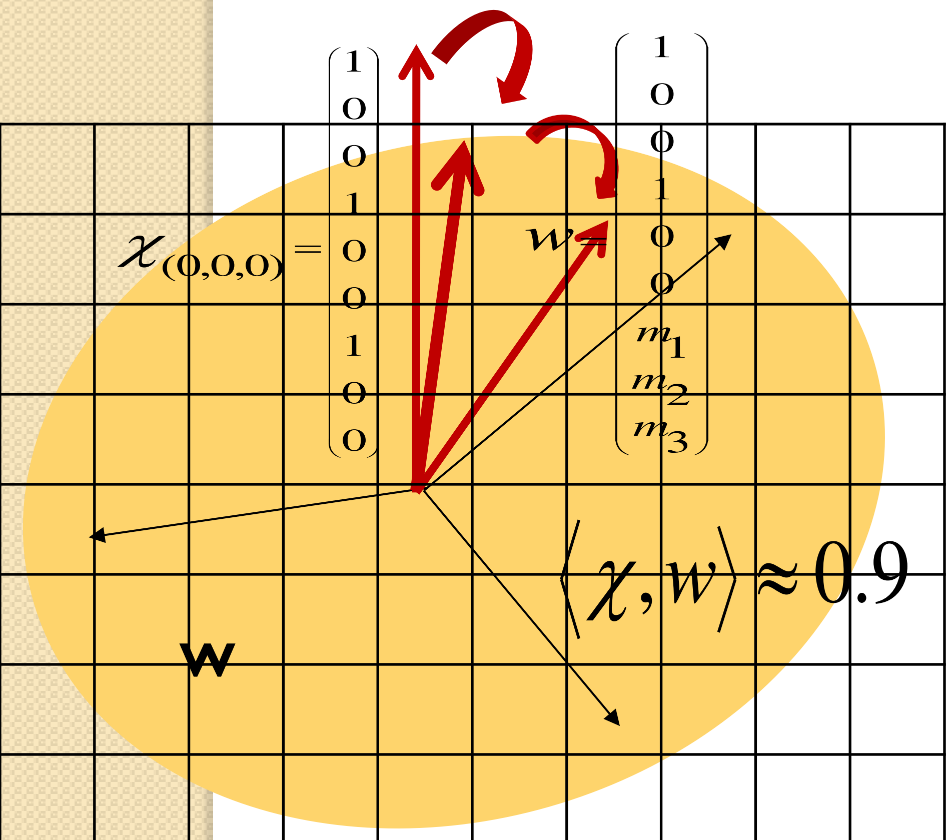
(Spectral gap between Y,$Y_\perp$)= absgap=
γd

W is "perturbed analog" of Y

"The sin μ" Theorem [DK'70] : Angle between Y and "perturbed analog of Y" small

W is close to Y so **dim(W) ≤dim(Y) =k**

A General Algorithm

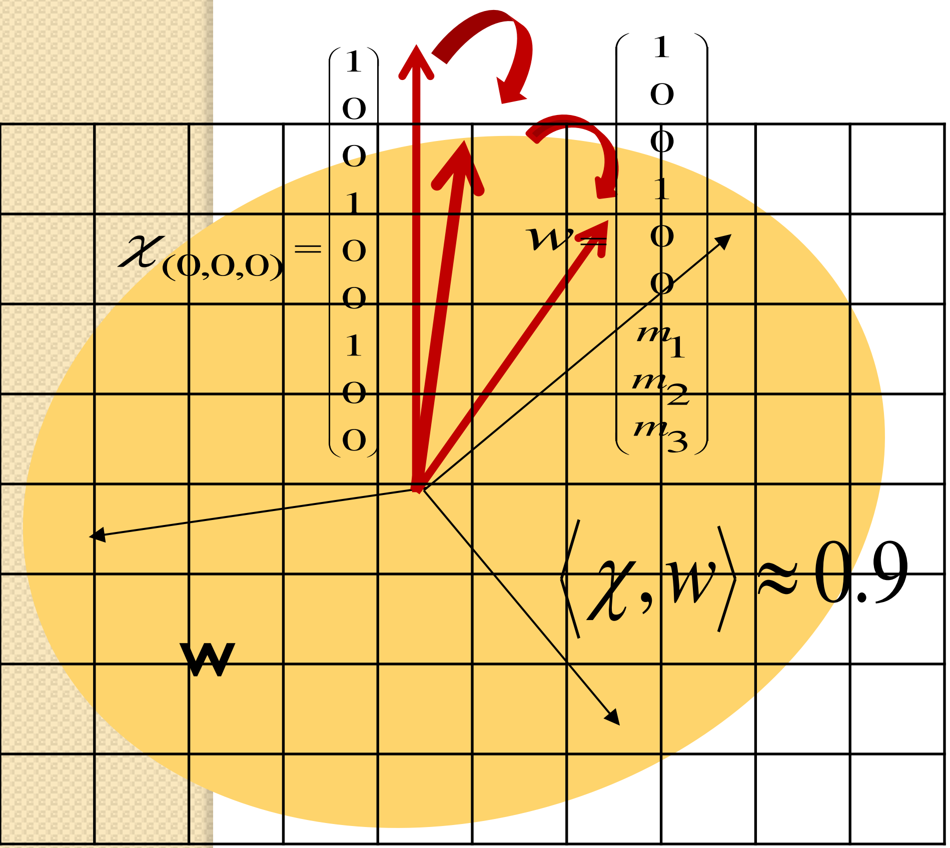


For expanders,
W is close to Y so
dim(W) ≤dim(Y) =k

Running time is
$\mathbf{2^k \approx 2^{\log n} \approx poly(n)}$

Algorithm runs in time ~ #points in the net
=
exponential in the dimension of eigenspace W

A General Algorithm

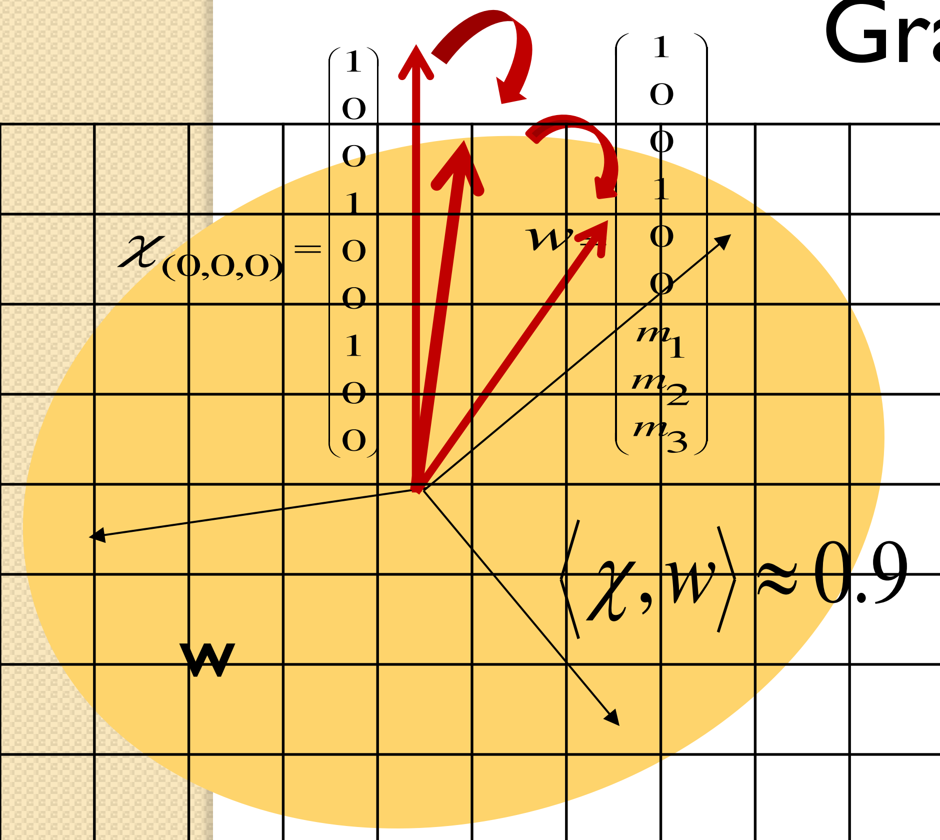


Algorithm runs in time ~ #points in the net

=

exponential in the dimension of eigenspace W

Another Special Case: The "Khot-Vishnoi" Graph



Graph that "cheats" a canonical semidefinite program for UG

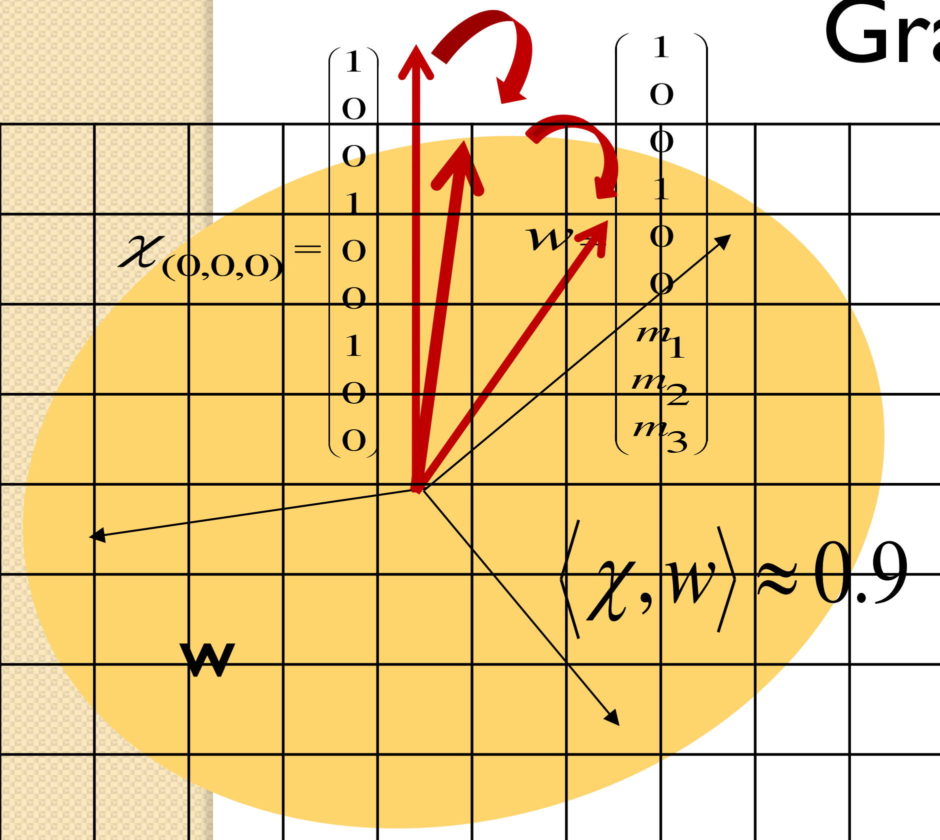
We show: Eigenspace in question has poly-logarithmic dimension

Algorithm runs in time ~ #points in the net

=

exponential in the dimension of eigenspace

Another Special Case: The "Khot-Vishnoi" Graph



Graph that "cheats" a canonical semidefinite program for UG

We show: Eigenspace in question has poly-logarithmic dimension

Algorithm runs in time ~ #points in the net

=

quasi-polynomial

UGC and the Spectrum of General Graphs

- 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!