Unique Games hardness of Quantum Max-Cut, and a conjectured vector-valued Borell's inequality

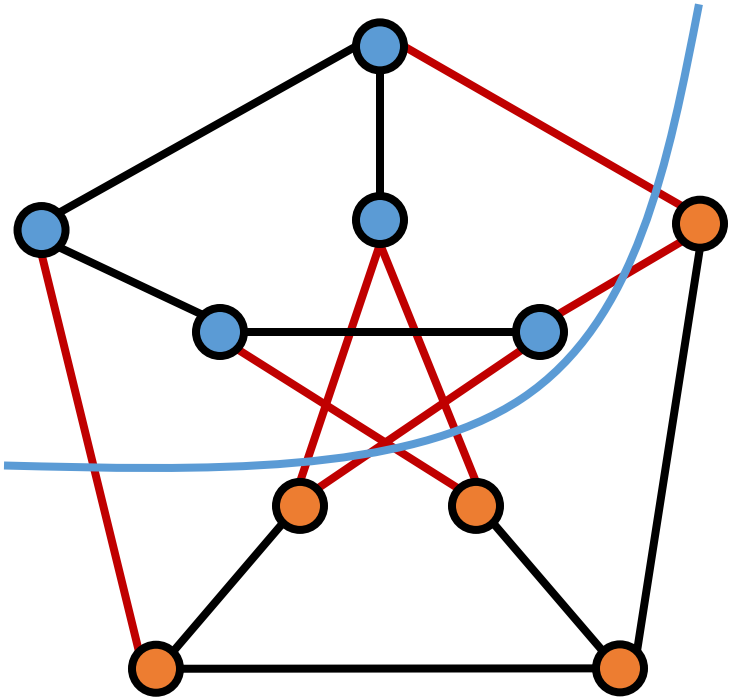
Yeongwoo Hwang
Harvard

Joe Neeman
UT Austin

Ojas Parekh
Kevin Thompson
Sandia

John Wright
UC Berkeley

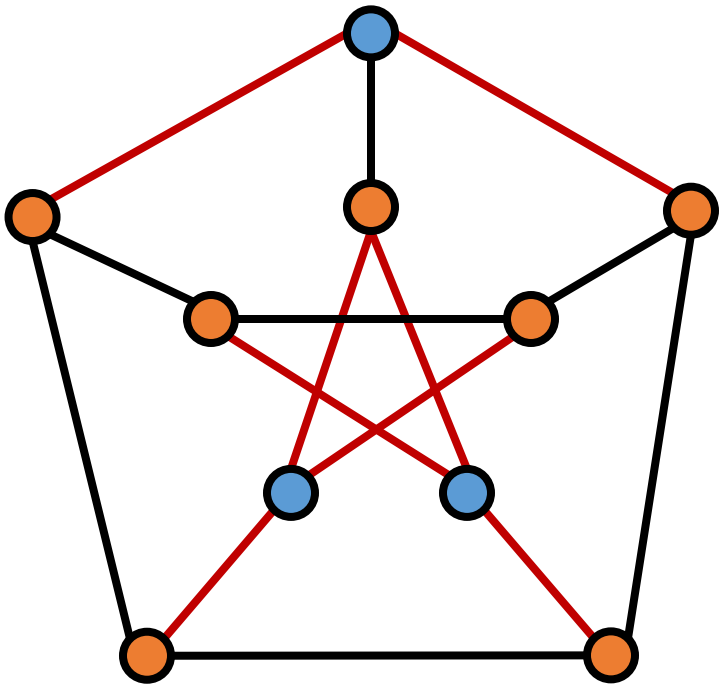
(Classical) Max-Cut



7 edges cut

$$\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$$

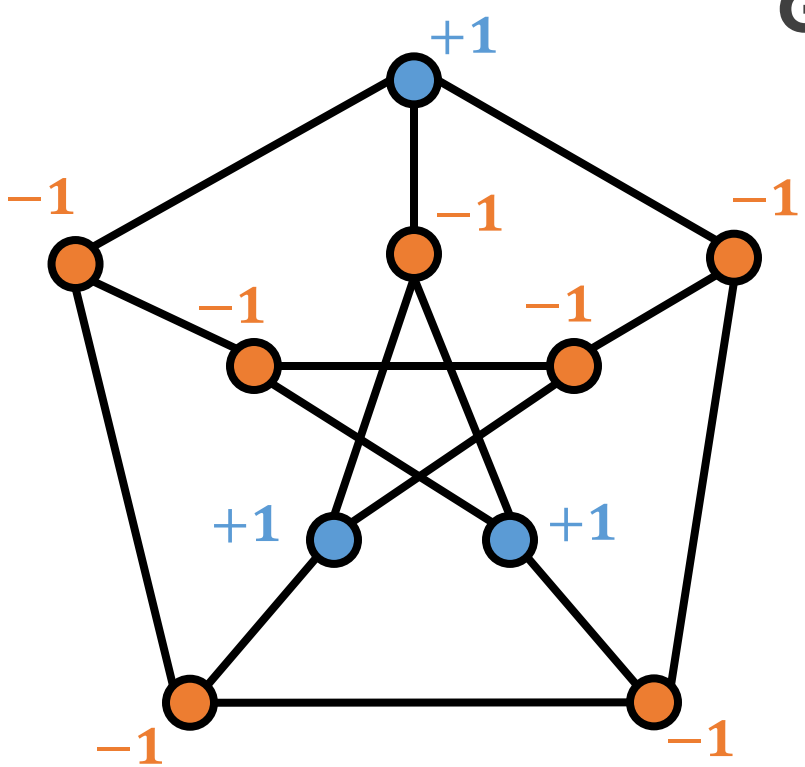
(Classical) Max-Cut



8 edges cut
(the **max cut**)

$G = (V, E)$

(Classical) Max-Cut



$G = (V, E)$

Goal: find partition $f: V \to \{+1, -1\}$ maximizing

$$\sum_{(u,v) \in E} \left(\frac{1 - f(u) \cdot f(v)}{2} \right)$$

NP-hard to solve exactly!

So instead look for **approximation algorithms**.

An algorithm achieves an **α-approximation** if it outputs a partition f such that

$$\#\text{ of edges } f \text{ cuts} \geq \alpha \cdot \textbf{Max-Cut}(G).$$

Approximating Max-Cut

[Goemans-Williamson]:
Poly-time algorithm achieving a $\mathbf{0.878}$ ... approximation

Based on a technique known as
semidefinite programming (SDP)

Powerful & versatile framework for designing algorithms

Will describe later!

[KKMO, MOO]: **GW** is the **optimal** poly-time algorithm.
(assuming Unique Games Conjecture (UGC))

Constraint satisfaction problems

Max-Cut is e.g. of a **constraint satisfaction problem** (CSP)

CSP: set of variables with constraints on them

Examples: Max-3Sat, Max-3Coloring, Max-3XOR

For every CSP, there is a **canonical SDP**
aka the **basic SDP**

[Raghavendra]: The **basic SDP** is the **optimal** poly-time alg

(assuming UGC)

Complete understanding of approximability of CSPs.

(modulo the UGC)

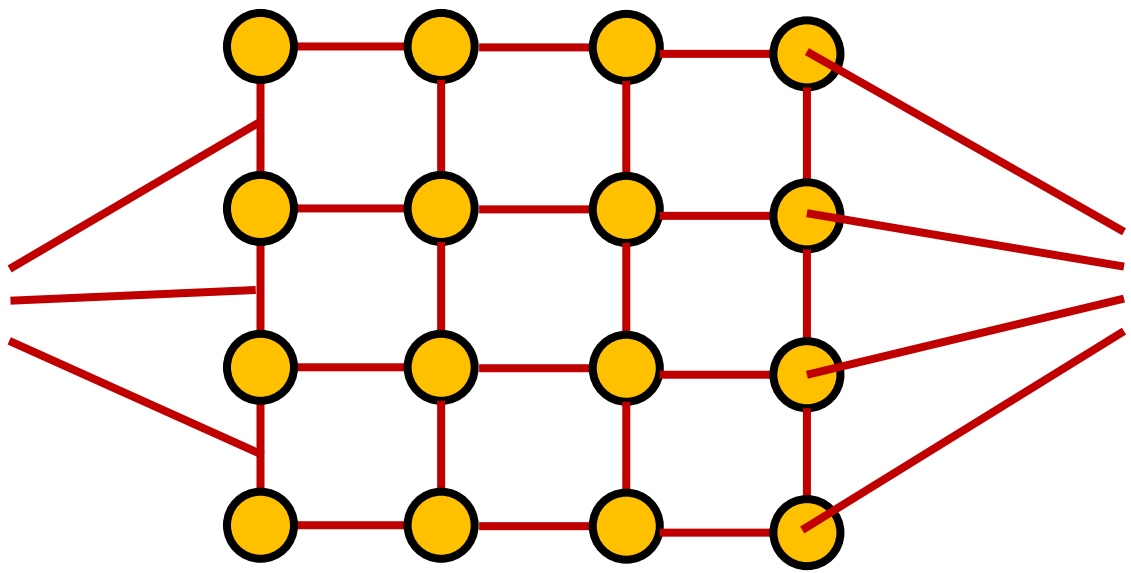
Beautiful theory!

The local Hamiltonian problem

Like **quantum** version of CSPs

Input: a physical system that looks like

like quantum **constraints**



like quantum **variables**

Output: the "**ground state**" of the system

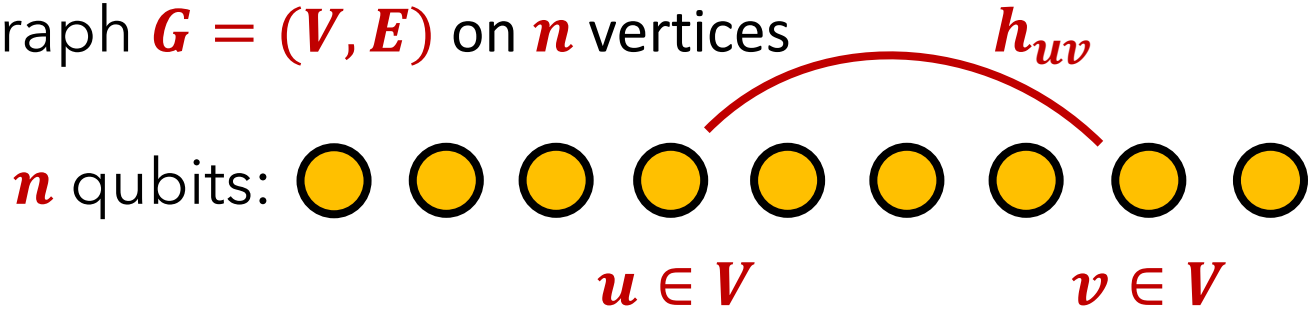
or the "ground state **energy**" of the system

Very important problem in physics!

The local Hamiltonian problem

Like **quantum** version of CSPs

Input: 1. A graph $G = (V, E)$ on n vertices



2. A Hamiltonian $H = \sum_{(u,v)\in E} h_{uv} \otimes I_{V\setminus\{u,v\}}$

Output: a quantum state $|\psi\rangle$ on n qubits
with the **maximum** energy $\lambda_{\max}(H)$

energy = $\langle\psi|H|\psi\rangle$

Problems with the LHP

Much less understood than classical CSPs!

We do **not** have:

- a theory of optimal algorithms
- a good understanding of the power of SDPs

like classical CSPs, there is a **basic SDP**

how well does it do?

is it also the optimal algorithm?

- a quantum version of the PCP theorem

(quantum PCP conjecture is still a conjecture)

Problems with the LHP part 2: the ansatz

Supposed to output quantum state $|\psi\rangle$.
What does that mean?

If algorithm is classical,
should output classical **description** of $|\psi\rangle$.

$|\psi\rangle$ must be efficiently describable!

E.g. product states $|\psi\rangle = |\psi_1\rangle \otimes \cdots \otimes |\psi_n\rangle$

Def: An **ansatz** is a family of quantum states which is efficiently describable.

Question: What is the **optimal** ansatz for approximation algorithms?

Quantum Max-Cut

Special case of 2-local Hamiltonian:

$$H_G = \sum_{(u,v)\in E} \frac{1}{4} \cdot (I - X_u X_v - Y_u Y_v - Z_u Z_v)$$

Goal: Output the **maximum energy state** of $\boldsymbol{H_G}$

Note: max energy state of $\boldsymbol{H_G}$

= **min** energy state of $\underline{\sum_{(u,v)\in E} (X_u X_v + Y_u Y_v + Z_u Z_v)}$

(antiferromagnetic) Heisenberg model

Dates back to **[Heisenberg 1928]**

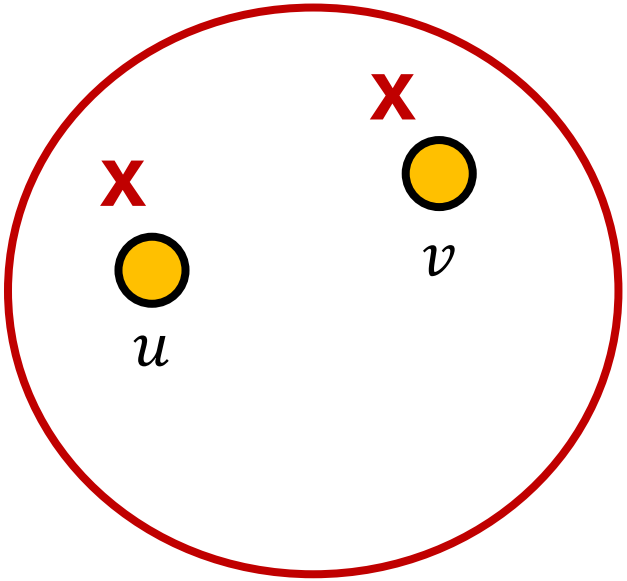
Well-studied class of Hamiltonians

Quantum Max-Cut

Special case of 2-local Hamiltonian:

$$H_G = \sum_{(u,v)\in E} \frac{1}{4} \cdot (I \boxed{- X_u X_v} - Y_u Y_v - Z_u Z_v)$$

Intuition



$|\psi\rangle$ ($\boldsymbol{n}$ qubits)

Term 1: Does nothing

Term 2: Measure in **X** basis

- $\mathbf{-1}$ if same ($+\,+$ or $-\,-$)
- $\mathbf{+1}$ if different ($+\,-$ or $-\,+$)

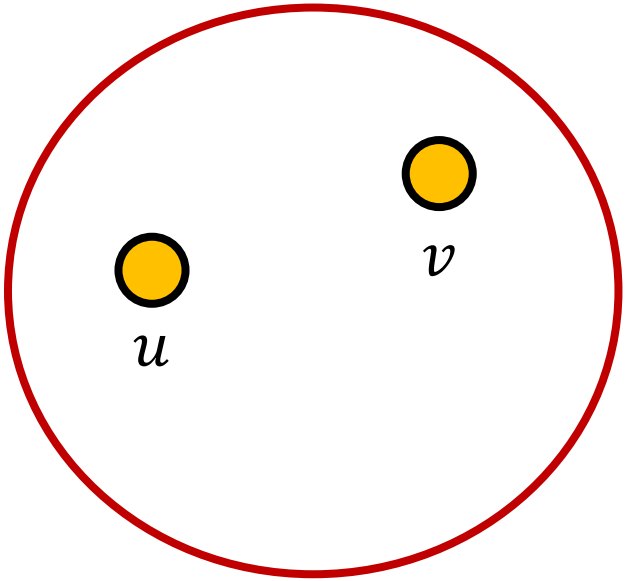
$\longrightarrow$ want both different!

Quantum Max-Cut

Special case of 2-local Hamiltonian:

$$H_G = \sum_{(u,v)\in E} \frac{1}{4} \cdot (I - X_u X_v - Y_u Y_v - Z_u Z_v)$$

Intuition



$|\psi\rangle$ ($\boldsymbol{n}$ qubits)

Term 1: Does nothing

Term 2: Should be different in **X** basis

Term 3: Should be different in **Y** basis

Term 4: Should be different in **Z** basis

Like **(classical)** Max-Cut in **X**, **Y**, and **Z** bases!

Approximating Quantum Max-Cut

[CM16,PM17]: Given a Quantum Max-Cut instance $\boldsymbol{H_G}$, estimating $\boldsymbol{\lambda_{\max}(H_G)}$ to error $\boldsymbol{\pm 1/\mathrm{poly}(n)}$ is **QMA-hard**.

[Gharibian-Parekh 2019]: **0.498**-approximation

Ansatz: product states $|\psi\rangle = |\psi_1\rangle \otimes \cdots \otimes |\psi_n\rangle$

[Anshu-Gosset-Morenz 2020]: **0.531**-approximation

Ansatz: tensor products of one- and two-qubit states

$$|\psi\rangle = |\psi_{12}\rangle \otimes |\psi_3\rangle \otimes \cdots \otimes |\psi_n\rangle$$

[Parekh-Thompson 2020]: **0.533**-approximation

Ansatz: same as **[AGM20]**

Approximating Quantum Max-Cut

[CM16,PM17]: Given a Quantum Max-Cut instance H_G, estimating $\lambda_{\max}(H_G)$ to error $\pm 1/\mathrm{poly}(n)$ is **QMA-hard**.

[Gharibian-Parekh 2019]: **0.498**-approximation

[Anshu-Gosset-Morenz 2020]: **0.531**-approximation

[Parekh-Thompson 2020]: **0.533**-approximation

[Lee 2022]: **0.562**-approximation

[King 2022]: **0.582**-approximation*

Ansatz: something more complicated

*only works for triangle-free graphs

Approximating Quantum Max-Cut

[CM16,PM17]: Given a Quantum Max-Cut instance H_G, estimating $\lambda_{\max}(H_G)$ to error $\pm 1/\mathrm{poly}(n)$ is **QMA-hard**.

[Gharibian-Parekh 2019]: **0.498**-approximation

uses the **basic SDP** for Quantum Max-Cut

[Anshu-Gosset-Morenz 2020]: **0.531**-approximation

[Parekh-Thompson 2020]: **0.533**-approximation

[Lee 2022]: **0.562**-approximation

[King 2022]: **0.582**-approximation*

do **not** use the basic SDP

Our motivation

[GP] uses **basic SDP** for Quantum Max-Cut

But **[AGM]**, **[PT]**, **[L]**, and **[K]** outperform it

Classically, the basic SDP is **always** optimal

Originally, I figured we could
just design a **better** SDP algorithm

Maybe product states are a **bad** ansatz?

So we spent some time working on it…

Our results

Thm: The Gharibian Parekh algorithm is the **optimal** algorithm using the basic SDP. *

Aka: Any alg using basic SDP has approx ratio $\leq \mathbf{0.498}$ ...

(show this via an integrality gap)

Thm: It is **NP**-hard to achieve a $\mathbf{0.956} + \epsilon$ approximation for Quantum Max-Cut. * **(assuming UGC)**

Far from the best algorithm ($\mathbf{0.562}$)

Seems to be the limit of current techniques

Can we improve either bound?

Our results

Thm: The Gharibian Parekh algorithm is the **optimal** algorithm using the basic SDP. *

Aka: Any alg using basic SDP has approx ratio $\leq 0.498\ldots$

(show this via an integrality gap)

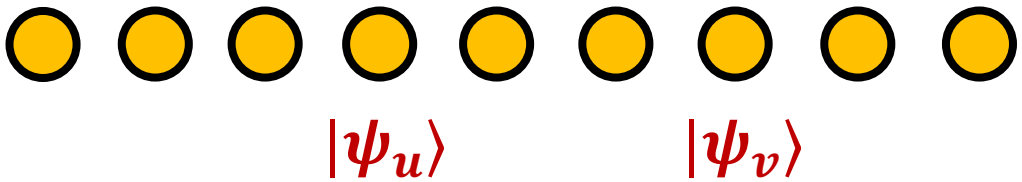
Thm: It is **NP**-hard to achieve a $0.956 + \epsilon$ approximation for Quantum Max-Cut. * **(assuming UGC)**

* Assuming a conjecture in Gaussian geometry
The "**vector-valued Borell's conjecture**"
We don't know how to prove it!
Will explain later…

Product states for QMax-Cut

States of the form $|\psi\rangle = \bigotimes_{u \in V} |\psi_u\rangle$

n qubits:



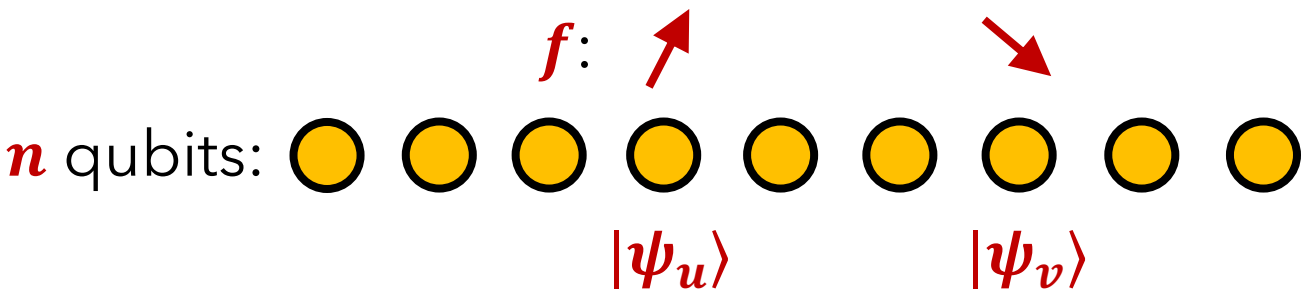
Product states possess no entanglement

But they can often be
close to the ground state!

[Brandao Harrow 2016]: The ground state is close to product if G is **high degree**.

Product states for QMax-Cut

States of the form $|\psi\rangle = \bigotimes_{u \in V} |\psi_u\rangle$



Useful to look at **Bloch sphere** representation:

$$|\psi_u\rangle\langle\psi_u| = \frac{1}{2} \cdot (I + c_X \cdot X + c_Y \cdot Y + c_z \cdot Z)$$

$|\psi_u\rangle$ is pure $\Rightarrow c_X^2 + c_Y^2 + c_Z^2 = 1$

Set $f(u) = (c_X, c_Y, c_Z)$. Then $f: V \to S^2$.

unit sphere in $\mathbb{R}^3$

Product states for QMax-Cut

States of the form $|\psi\rangle = \bigotimes_{u \in V} |\psi_u\rangle$

Bloch sphere representation: $f: V \to S^2$.

Product state energy formula:

$$\langle \psi | H_G | \psi \rangle = \sum_{(u,v) \in E} \left(\frac{1 - \langle f(u), f(v) \rangle}{4} \right)$$

"Want" neighboring $f(u)$ and $f(v)$
to point in opposite directions.

Like **(classical)** Max-Cut! There, $f: V \to \{\pm 1\} = S^0$.

Product state results

Also consider the problem of finding the **best** product state

[BOV]: Algorithm outputting a product state $|\psi\rangle$ with energy $\geq \mathbf{0.956} \times$ **(best product state energy)**.

(uses basic SDP)

[Us]:

1. This is the **optimal** alg using the basic SDP. *
2. This is the **optimal** poly-time alg. **(assuming UGC)**
   Aka it **NP**-hard to achieve a $\mathbf{0.956} + \epsilon$ approx. *

Same **NP**-hardness
as for **general** QMax-Cut!

* Assuming the same conjecture as before

Outline for rest of talk

Goal: explain this result:

Any alg for Quantum Max-Cut
using basic SDP has approx ratio $\leq \mathbf{0.498\ldots}$

Outline:

1. Goemans-Williamson algorithm
2. Gharibian-Parekh algorithm
3. Why GW is optimal alg for basic SDP

   (uses Borell's isoperimetric inequality)
4. Why GP is optimal alg for basic SDP

   (assuming vector-valued Borell's conjecture)

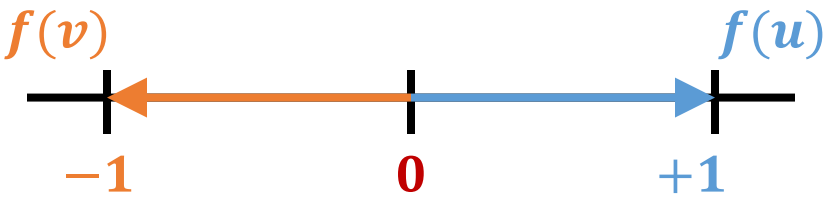
Goemans-Williamson algorithm

Goemans-Williamson algorithm

Goal: maximize

$$\sum_{(u,v) \in E} \left(\frac{1 - f(u) \cdot f(v)}{2} \right)$$

where $f(u) \in \{\pm 1\}$



Can't optimize over this!

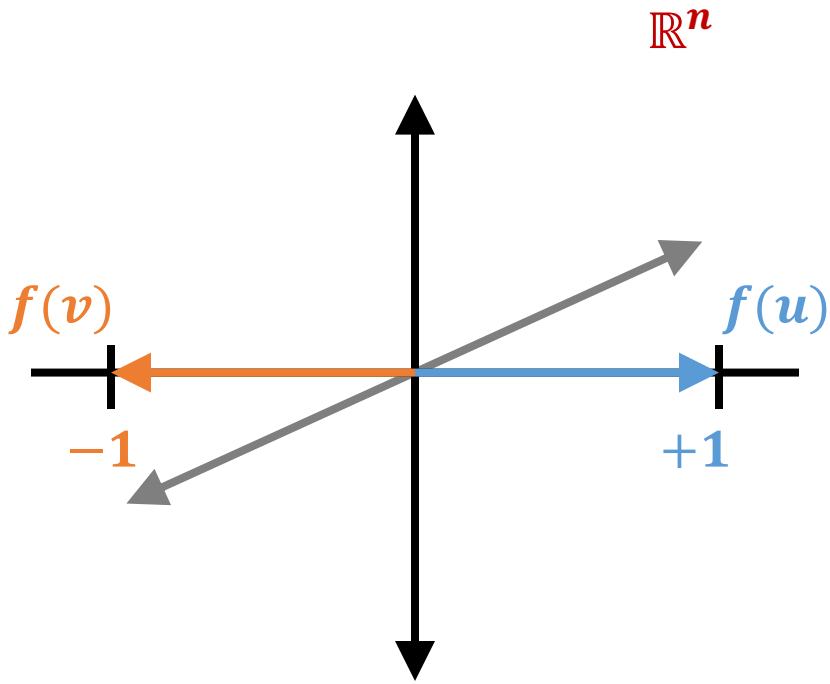
(**NP-hard**)

Goemans-Williamson algorithm

Goal: maximize

$$\sum_{(u,v)\in E}\left(\frac{1-\langle f(u), f(v)\rangle}{2}\right)$$

where $f(u) \in (\pm 1, 0, \ldots, 0)$



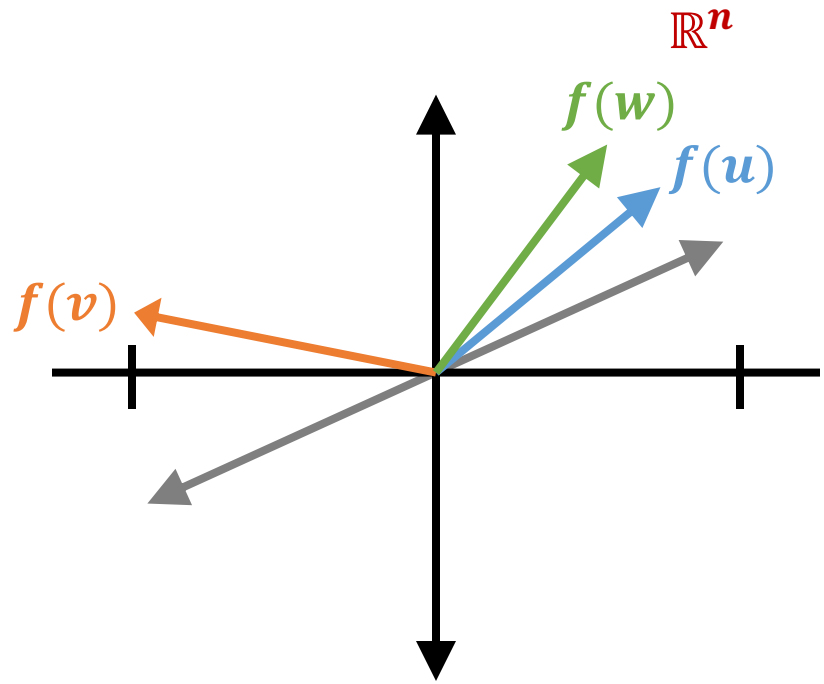
Still can't optimize over this!

Goemans-Williamson algorithm

Goal: maximize

$$\sum_{(u,v)\in E}\left(\frac{1-\langle f(u), f(v)\rangle}{2}\right)$$

where $f(u)$ unit vector $\in \mathbb{R}^n$



Can optimize over this!

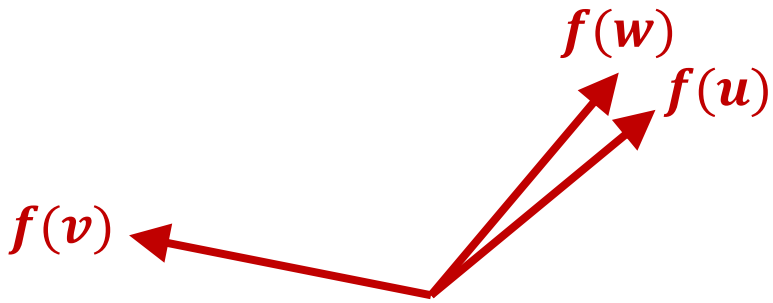
Known as the Max-Cut **semidefinite program** (SDP)

Let $\mathbf{SDP_{MC}}(G)$ be the value of the best vectors

Note: $\mathbf{SDP_{MC}}(G) \geq \mathbf{Max\text{-}Cut}(G)$

Goemans-Williamson algorithm

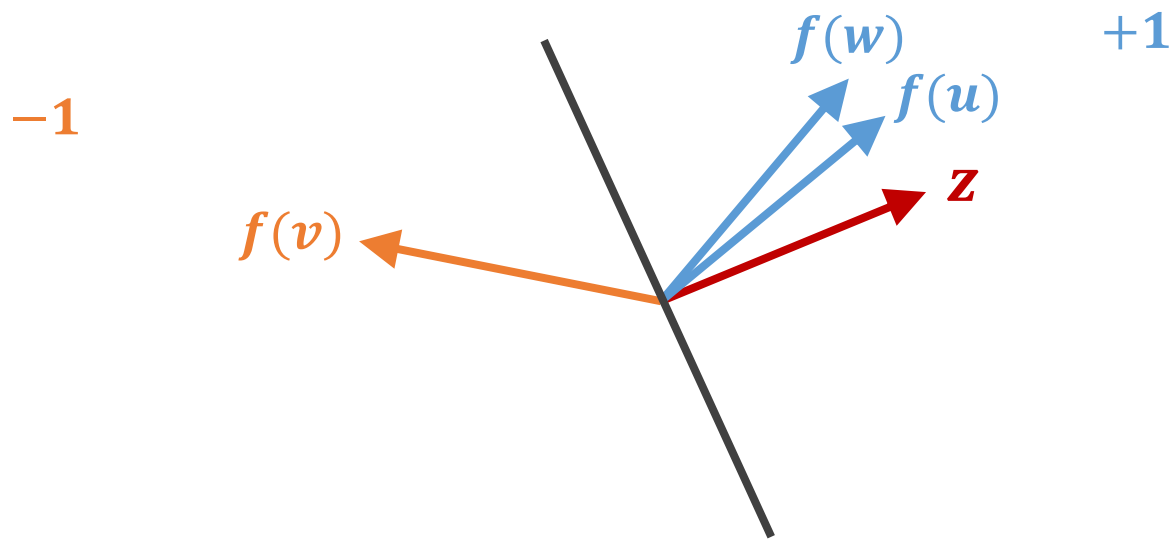
Step 1: Compute the optimal SDP vectors



Step 2: "Round" $f: V \to S^{n-1}$ into a partition $g: V \to \{\pm 1\}$

Goemans-Williamson algorithm

Step 1: Compute the optimal SDP vectors



Step 2: "Round" $f: V \to S^{n-1}$ into a partition $g: V \to \{\pm 1\}$

(a) Pick a random vector $z \in \mathbb{R}^n$

(b) Set $g(u) = \mathbf{sign}(\langle z, f(u) \rangle)$, for all $u \in V$

[GW]: Achieves a $0.878\ldots$ approximation

Analyzing the GW algorithm

Step 1: Compute the optimal SDP vectors

Step 2: "Round" $\boldsymbol{f}: \boldsymbol{V} \to \boldsymbol{S}^{\boldsymbol{n}-\mathbf{1}}$ into a partition $\boldsymbol{g}: \boldsymbol{V} \to \{\pm\mathbf{1}\}$

Goal: $\mathbf{E}[\#$ edges cut by $\boldsymbol{g}] \geq \mathbf{0.878} \cdot$ (SDP value of $\boldsymbol{f}$)

(recall) $\mathbf{0.878} \cdot \mathbf{Max\text{–}Cut}(\boldsymbol{G})$ $\square$

Analyzing the GW algorithm

Step 1: Compute the optimal SDP vectors

Step 2: "Round" $f: V \to S^{n-1}$ into a partition $g: V \to \{\pm 1\}$

Goal: $\mathbf{E}[\#\text{ edges cut by } g] \geq 0.878 \cdot (\text{SDP value of } f)$

$$\sum_{(u,v) \in E} \left(\frac{1 - g(u) \cdot g(v)}{2} \right) \qquad \sum_{(u,v) \in E} \left(\frac{1 - \langle f(u), f(v) \rangle}{2} \right)$$

Goal inequality is true for **each** edge!

Key lemma:

$$\forall\, u, v\colon\ \mathbf{E}[1 - g(u) \cdot g(v)] \geq 0.878 \cdot (1 - \langle f(u), f(v) \rangle)$$

Analyzing the GW algorithm

Key lemma:

$\forall\, u, v$: $\mathbf{E}[1 - g(u) \cdot g(v)] \geq 0.878 \cdot (1 - \rho_{uv})$

Pf: Set $\rho_{uv} = \langle f(u), f(v) \rangle$.

LHS is function only of ρ_{uv} too.

Want to compute

$$\min_{-1 \leq \rho_{uv} \leq 1} \frac{\mathbf{E}[1 - g(u) \cdot g(v)]}{(1 - \rho_{uv})}$$

Equal to 0.878 (when $\rho_{uv} = -0.69\ldots$). □

Note: g's value on edge (u, v)

is only $0.878 \times$ (f's value)

if ρ_{uv} is the "bad angle" $-0.69\ldots$

Gharibian-Parekh algorithm

Back to **Quantum** Max-Cut

Goal: Gharibian-Parekh **0.498**-approximation algorithm

Recall:

1. Based on **semidefinite programming**
2. Outputs product state $|\psi\rangle = |\psi_1\rangle \otimes \cdots \otimes |\psi_n\rangle$

   (aka a Bloch sphere assignment $g: V \to S^2$)

Back to **Quantum** Max-Cut

Goal: Gharibian-Parekh **0.498**-approximation algorithm

Recall: 1. Based on **semidefinite programming**

For every optimization problem,
there is a **canonical** SDP

a) Gives an upper bound to the true optimum
b) Can be efficiently solved
c) Probably involves some vectors

For Quantum Max-Cut, a bit complicated

But you can simplify it…

Back to Quantum Max-Cut

Goal: Gharibian-Parekh **0.498**-approximation algorithm

Recall: 1. Based on **semidefinite programming**

Given a graph G and Hamiltonian H_G, the SDP is:

$$\mathbf{SDP_{QMC}}(G) = \max \sum_{(u,v) \in E} \left(\frac{1 - 3 \cdot \langle f(u), f(v) \rangle}{4} \right)$$

$$\text{s.t. } f: V \to S^{n-1}$$

Very similar to Max-Cut SDP!

Optimizing $f: V \to S^{n-1}$ is same in both!

Back to Quantum Max-Cut

Goal: Gharibian-Parekh **0.498**-approximation algorithm

Recall: 1. Based on **semidefinite programming**

Given a graph G and Hamiltonian H_G, the SDP is:

$$\mathbf{SDP_{QMC}}(G) = \max \sum_{(u,v) \in E} \left(\frac{1 - 3 \cdot \langle f(u), f(v) \rangle}{4} \right)$$

$$\text{s.t. } f: V \to S^{n-1}$$

Gives an upper bound to the true optimum:

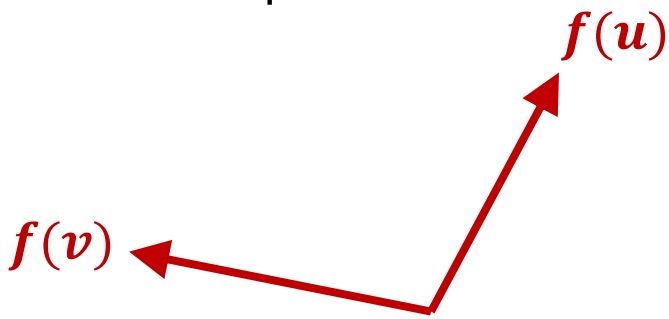
$$\mathbf{QMax\text{-}Cut}(G) \leq \mathbf{SDP_{QMC}}(G)$$

Next step: rounding this into a
Bloch sphere assignment $g: V \to S^2$.

How to do this???

Gharibian-Parekh algorithm

Step 1: Compute the optimal SDP vectors



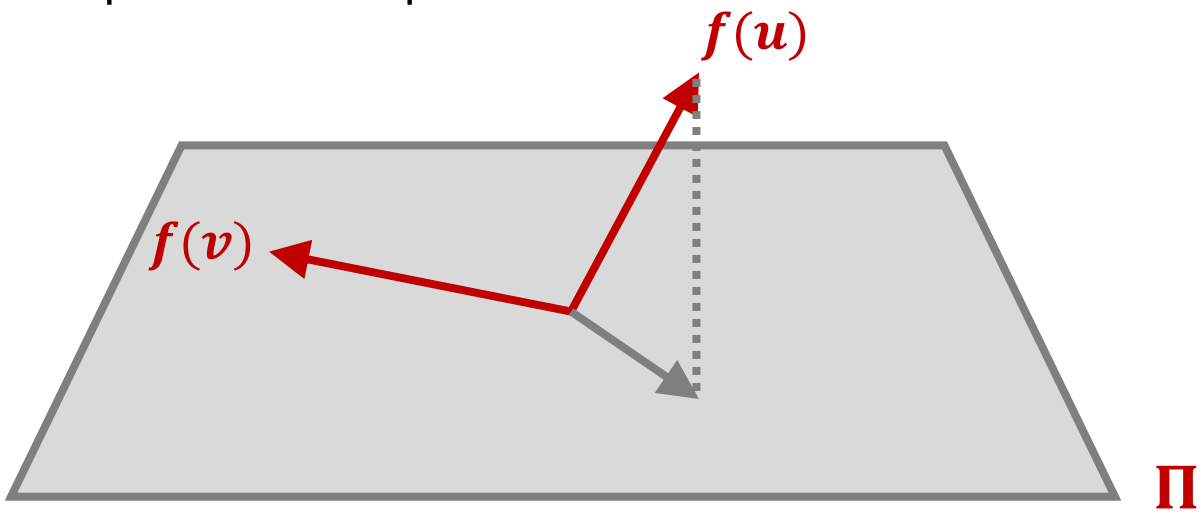
Step 2: "Round" f into a Bloch sphere assignment $g: V \to S^2$

(a) Pick random 3-dimensional projector $\Pi \in \mathbb{R}^{3\times n}$

$$\Pi = \begin{bmatrix} - & x & - \\ - & y & - \\ - & z & - \end{bmatrix}, \qquad x, y, z \in \mathbb{R}^n \text{ random, orthonormal}$$

Gharibian-Parekh algorithm

Step 1: Compute the optimal SDP vectors



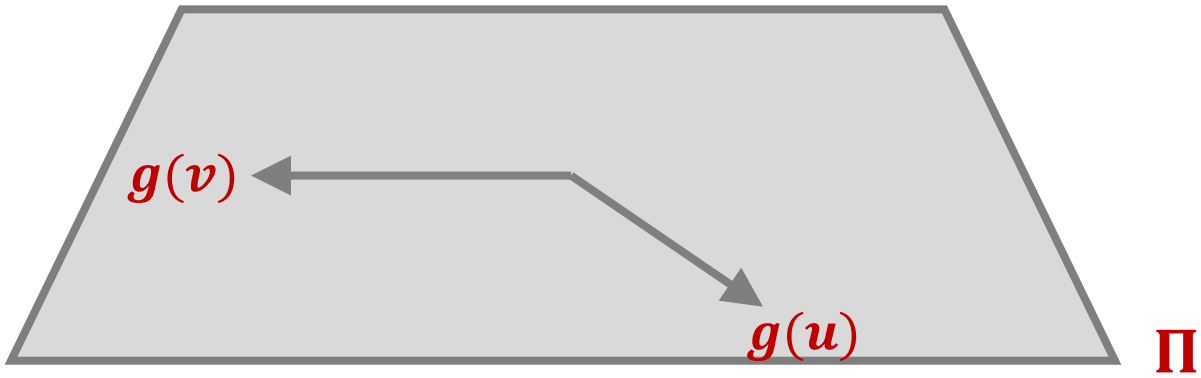
Step 2: "Round" f into a Bloch sphere assignment $g: V \to S^2$

(a) Pick random 3-dimensional projector $\Pi \in \mathbb{R}^{3\times n}$

(b) For all $u \in V$, set $g(u) = \Pi \cdot f(u)/\|\Pi \cdot f(u)\|_2$.

Gharibian-Parekh algorithm

Step 1: Compute the optimal SDP vectors



Step 2: "Round" f into a Bloch sphere assignment $g: V \to S^2$

(a) Pick random 3-dimensional projector $\Pi \in \mathbb{R}^{3\times n}$

(b) For all $u \in V$, set $g(u) = \Pi \cdot f(u)/\|\Pi \cdot f(u)\|_2$.

[GP]: Achieves a $0.498\ldots$ approximation

Analyzing the GP algorithm

Similar to the **GW** analysis

Boils down to a per edge inequality:

Key lemma:

$\forall\, \boldsymbol{u}, \boldsymbol{v}$: $\mathbf{E}[\mathbf{1} - \langle \boldsymbol{g}(\boldsymbol{u}), \boldsymbol{g}(\boldsymbol{v}) \rangle] \geq \mathbf{0.498} \cdot (\mathbf{1} - \mathbf{3} \cdot \boldsymbol{\rho}_{\boldsymbol{uv}})$

inequality is tight if $\boldsymbol{\rho}_{\boldsymbol{uv}} = \mathbf{-0.97} \ldots$

this is the "bad angle" for GP algorithm

for these edges, $\boldsymbol{g}$'s value is only $\mathbf{0.498} \times$ ($\boldsymbol{f}$'s value)

Why GW is optimal alg for the basic SDP

GW analysis

Wanted to show: $\alpha \cdot \mathbf{Max\text{-}Cut}(G) \leq \mathbf{E}[\#\text{ edges cut by } g]$

$(\alpha = 0.878\ldots)$

Actually showed: $\alpha \cdot \mathbf{SDP_{MC}}(G) \leq \mathbf{E}[\#\text{ edges cut by } g]$

$\leq \mathbf{Max\text{–}Cut}(G)$

Stronger statement!

"**Standard analysis**": comparing g to SDP value

Note: $\alpha \leq \boxed{\min_G \left\{ \frac{\mathbf{Max\text{–}Cut}(G)}{\mathbf{SDP_{MC}}(G)} \right\}}$ → "**Integrality gap**" of SDP

Integrality gap provides **limitation** on any algorithm analyzed via "**standard analysis**"

Integrality gap for Max-Cut

[FS]: The basic SDP for Max-Cut has integrality gap $\mathbf{0.878\ldots}$

$\therefore$ Any algorithm which rounds basic SDP has approx. ratio $\leq \mathbf{0.878\ldots}$

(according to standard analysis)

[Raghavendra]: For any CSP,

optimal poly-time approx. ratio = **integrality gap** of basic SDP.

(assuming UGC)

Can't beat the standard analysis!

Integrality gap for Max-Cut

[FS]: The basic SDP for Max-Cut has integrality gap $\mathbf{0.878}$ ...

Pf: Want a graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ satisfying the following:

1. $\mathbf{Max\text{–}Cut}(\boldsymbol{G}) = \mathbf{0.878} \cdot \mathbf{SDP_{MC}}(\boldsymbol{G})$
2. $\mathbf{Max\text{–}Cut}(\boldsymbol{G}) = \mathbf{E}[\text{\# edges cut by } \boldsymbol{g}]$

   (Since $\mathbf{E}[\text{\# edges cut by } \boldsymbol{g}] \geq \mathbf{0.878} \cdot \mathbf{SDP_{MC}}(\boldsymbol{G})$)

Integrality gap for Max-Cut

[FS]: The basic SDP for Max-Cut has integrality gap $\mathbf{0.878}$ ...

Pf: Want a graph $G = (V, E)$ satisfying the following:

1. $\mathbf{Max\text{–}Cut}(G) = \mathbf{0.878} \cdot \mathbf{SDP_{MC}}(G)$
2. $\mathbf{Max\text{–}Cut}(G) = \mathbf{E}[\#\text{ edges cut by } g]$

   GW algorithm always outputs optimal cut!

3. For each edge (u, v),

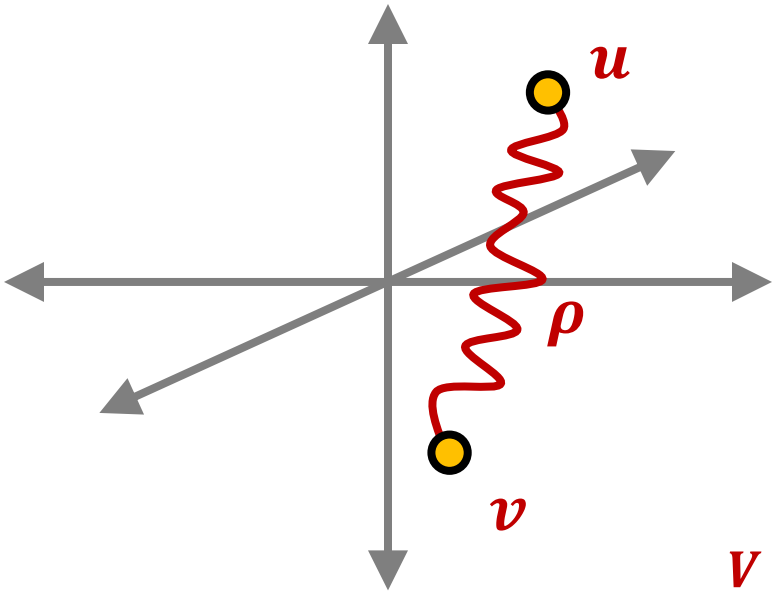
   $$\rho_{uv} = \langle f(u), f(v) \rangle$$

   equals the "bad angle" $\rho_{uv} = -0.69$

(Necessary for GW to lose factor of $\mathbf{0.878}$ ... on each edge)

Gaussian graph

Vertex set:



$V = \mathbb{R}^n$

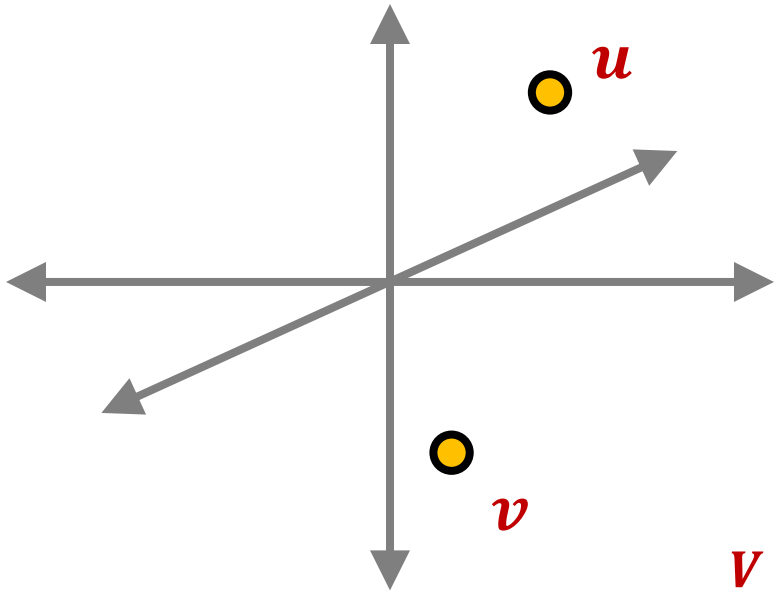
Random edge:

(u, v), where $u \sim_\rho v$ are ρ-correlated Gaussians

$(\rho = -0.69 \ldots)$

Gaussian graph

Vertex set:



$V = \mathbb{R}^n$

Random edge:

(u, v), where $u \sim_\rho v$ are ρ-correlated Gaussians

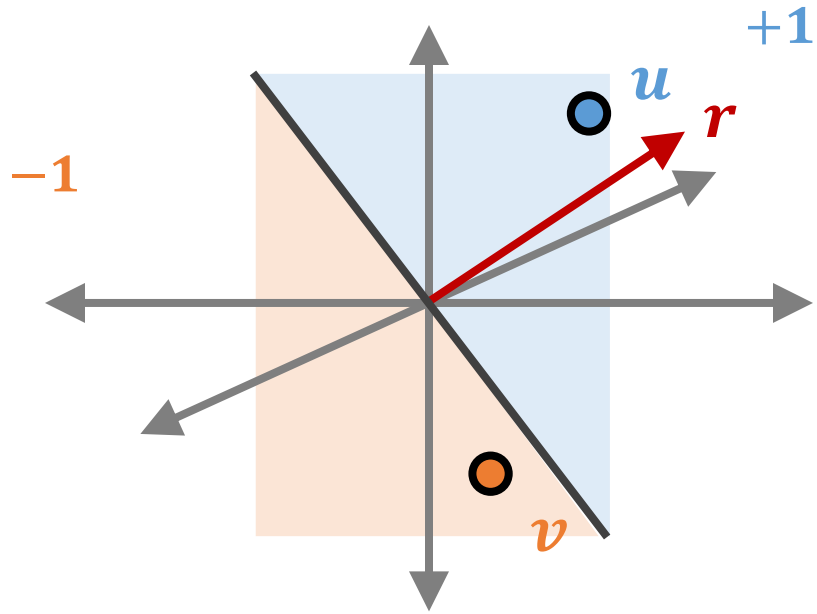
$(\rho = -0.69 \ldots)$

SDP solution $f: V \to \mathbb{R}^n$:

1. $f(u) = u/\|u\|_2$
2. $\langle f(u), f(v) \rangle \approx \rho$

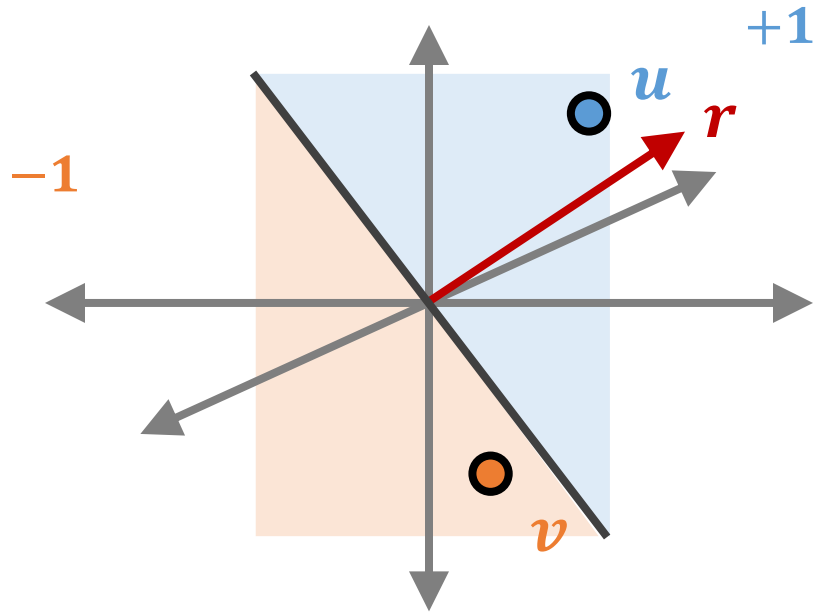
- **GW loses $0.878 \ldots$ on each edge**
- **Need to show GW is optimal**

Optimality of GW



Value of solution: $\mathbf{E}_{u \sim_\rho v}\left[\left(\frac{1 - g(u) \cdot g(v)}{2}\right)\right]$

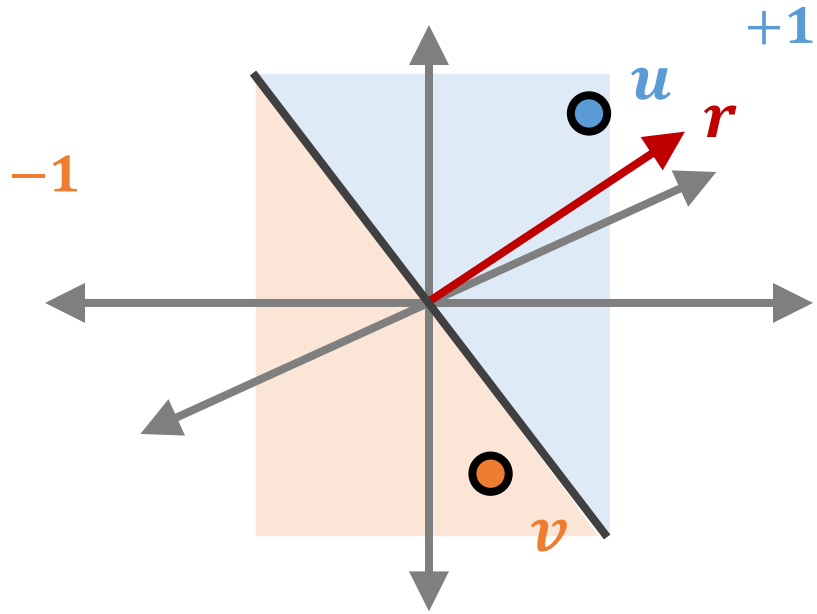
Optimality of GW



Value of solution: $\mathbf{E}_{u \sim_\rho v}\left[\left(\frac{1 - \mathrm{sign}(\langle u, r\rangle) \cdot \mathrm{sign}(\langle v, r\rangle)}{2}\right)\right]$

- Rotationally symmetric
- Can pick $r = (1, 0, \ldots, 0)$ WOLOG

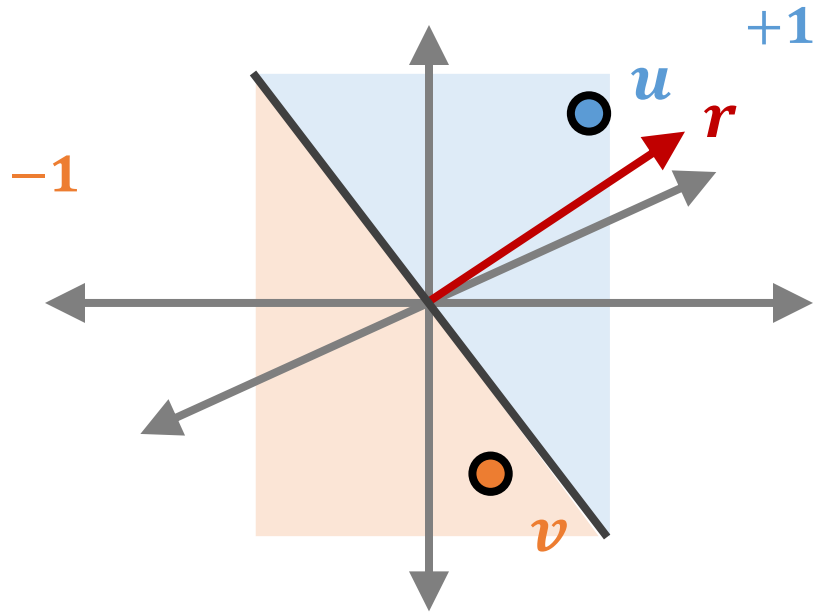
Optimality of GW



Value of solution: $\mathbf{E}_{u \sim_\rho v}\left[\left(\frac{1 - \mathrm{sign}(u_1) \cdot \mathrm{sign}(v_1)}{2}\right)\right]$

- Rotationally symmetric
- Can pick $r = (1, 0, \ldots, 0)$ WOLOG

Optimality of GW



Value of solution: $\mathbf{E}_{u \sim_\rho v}\left[\left(\frac{1 - \mathrm{sign}(u_1) \cdot \mathrm{sign}(v_1)}{2}\right)\right]$

Want this to be optimal partition.

Should be at least: $\mathbf{E}_{u \sim_\rho v}\left[\left(\frac{1 - h(u) \cdot h(v)}{2}\right)\right]$,

for all partitions $h: V \to \{\pm 1\}$

Borell's theorem

The quantity **(negative ρ)**

$$\mathbf{E}_{u \sim_\rho v}[\boldsymbol{h}(\boldsymbol{u}) \cdot \boldsymbol{h}(\boldsymbol{v})]$$

is minimized by $\boldsymbol{h}(\boldsymbol{u}) = \mathbf{sign}(\boldsymbol{u_1})$.

First proven in **[Borell 1985]**

Basic result in Gaussian geometry

Why GP is optimal alg for the basic SDP

Integrality gap for Quantum Max-Cut

[Us]: The basic SDP for Quantum Max-Cut has integrality gap $\mathbf{0.498\ldots}$

Same graph $\boldsymbol{G}$: $\boldsymbol{\rho}$-correlated Gaussian graph,

$\boldsymbol{\rho}$ is the "bad angle" $\boldsymbol{\rho} = \mathbf{-0.97\ldots}$

$\Rightarrow$ **GP** algorithm loses factor of $\mathbf{0.498}$

Goal: **GP** algorithm produces optimal solution

Optimality of GP

GP picks random 3-dimensional projector $\boldsymbol{\Pi}$

Sets $\boldsymbol{g}(\boldsymbol{u}) = \boldsymbol{\Pi} \cdot \boldsymbol{u} / \|\boldsymbol{\Pi} \cdot \boldsymbol{u}\|_2$ for each $\boldsymbol{u} \in \mathbb{R}^{\boldsymbol{n}}$

Value of solution:

$$\mathbf{E}_{\boldsymbol{u} \sim_{\rho} \boldsymbol{v}} \left[\left(\frac{\mathbf{1} - \langle \boldsymbol{g}(\boldsymbol{u}), \boldsymbol{g}(\boldsymbol{v}) \rangle}{\mathbf{4}} \right) \right]$$

- Rotationally symmetric
- Can pick $\boldsymbol{\Pi}$ = projector onto $(\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3)$

$$\Rightarrow \boldsymbol{g}(\boldsymbol{u}) = (\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3) / \|(\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3)\|_2$$

Optimality of GP

GP picks random 3-dimensional projector $\boldsymbol{\Pi}$

Sets $\boldsymbol{g}(\boldsymbol{u}) = \boldsymbol{\Pi} \cdot \boldsymbol{u} / \|\boldsymbol{\Pi} \cdot \boldsymbol{u}\|_2$ for each $\boldsymbol{u} \in \mathbb{R}^{\boldsymbol{n}}$

Value of solution:

$$\mathbf{E}_{\boldsymbol{u} \sim_{\rho} \boldsymbol{v}} \left[\left(\frac{\mathbf{1} - \langle \boldsymbol{g}(\boldsymbol{u}), \boldsymbol{g}(\boldsymbol{v}) \rangle}{\mathbf{4}} \right) \right]$$

Want this to be optimal quantum state.

This is a product state (Bloch sphere rep.)

Need to compare against **all** quantum states

However, $\boldsymbol{G}$ is high degree ⇒ ground state is **product**

[Brandao Harrow 2016]

Only need to compare to product states $\boldsymbol{h}: \mathbb{R}^{\boldsymbol{n}} \to \boldsymbol{S}^{\mathbf{2}}$

A vector-valued Borell's conjecture

For all $\boldsymbol{k} \geq \mathbf{1}$, the quantity **(negative $\boldsymbol{\rho}$)**

$$\mathbf{E}_{\boldsymbol{u} \sim_{\boldsymbol{\rho}} \boldsymbol{v}}[\langle \boldsymbol{h}(\boldsymbol{u}), \boldsymbol{h}(\boldsymbol{v}) \rangle]$$

is minimized by $\boldsymbol{h}(\boldsymbol{u}) = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k)/\|(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k)\|_2$

We **thought** we had proven this!

(but Steve Heilman found a **bug** in our proof)

What we **can** show:

1. Show that $\boldsymbol{h}$ is "essentially $\boldsymbol{k}$ dimensional" **(positive $\boldsymbol{\rho}$)**

   (via calculus of variations)

2. Prove it in the case of $\boldsymbol{n} = \boldsymbol{k}$ **(negative $\boldsymbol{\rho}$)**

   (via Fourier analysis)

Conclusion

Open problems

1. Prove the vector-valued Borell's conjecture!
2. Improve our UG-hardness past **0.956**
3. Improve the best approx. ratio past **0.562** **[Lee]**
4. The **[PT]** algorithm uses degree-4 Sum of Squares (SoS)

   (more powerful SDP than the basic SDP)

   Can we understand the power of SoS?
5. **Algorithmic gap** for the Gharibian-Parekh algorithm:

   Graph $\boldsymbol{G = (V, E)}$ s.t.

   - $\mathbf{QMax\text{-}Cut}(\boldsymbol{G}) = \boldsymbol{\nu}$
   - GP alg produces state with energy $\mathbf{0.498} \cdot \boldsymbol{\nu}$

   (Or maybe "standard analysis" is not tight?)

Thanks!