Semidefinite Programming and Approximate Pure Nash Equilibria for Cut Games

Ethan Mader

Purdue University

Fall 2024

SDP Introduction

Constraint solver which uses real vectors as variables (developed in 1990s)
Encapsulates linear programming (developed in 1946)
Still poly time solvable, but harder in practice

Definition

An SDP is any problem of the form

$$\min_{v_1,\ldots,v_n \in \mathbb{R}^n} \sum_{i,j \in [n]} c_{i,j}(v_i \cdot v_j)$$

$$\text{subject to} \sum_{i,j \in [n]} a_{i,j,k}(v_i \cdot v_j) \leq b_k \text{ for all } k$$

where $c_{i,j}, a_{i,j,k}, b_k \in \mathbb{R}$, and $v_i \cdot v_j$ is the dot product

Max Cut

SDP is useful for poly time approximations of NP-hard problems
Canonical example is the 0.87856-approx Max Cut algorithm by Goemans and Williamson from 1995 (still the best known)

Max Cut Problem

Input: An undirected graph $G = (V, E)$ with nonnegative weights $w_{i,j}$ on edges $(i, j) \in E$ (let $w_{i,j} = 0$ for $(i, j) \notin E$)
Output: A set of vertices S that maximizes the weight of the edges in the cut $(S, \bar{S})$, denoted $w(S, \bar{S}) = \sum_{i \in S, j \in \bar{S}} w_{i,j}$

NP-complete when formulated as a decision problem

Max Cut (Continued)

Key idea: represent nodes as unit vectors in $\mathbb{R}^n$
Dot product close to 1 represents same side of the cut, and close to -1 represents opposite sides of the cut

SDP Formulation

$$\min_{v_1,\ldots,v_n \in \mathbb{R}^n} \sum_{(i,j)\in E} -w_{i,j}\left(\frac{1 - v_i \cdot v_j}{2}\right) \text{ subject to } v_i \cdot v_i = 1 \text{ for all } i \in [n]$$

Note that the returned SDP objective value is at least OPT for Max Cut
How to convert the vector solution to a set of vertices without losing too much objective value?

Max Cut (Continued)

The solution: randomized hyperplane rounding!
Choose a random unit vector $r \in \mathbb{R}^n$
Let vertices i corresponding to vectors v_i such that $v_i \cdot r \geq 0$ be in S, and the rest be in $\bar{S}$

Intuitively, vectors which were split by the SDP are likely to be split by hyperplane rounding
With some pretty simple analysis, the expected solution is 0.87856 of OPT

Assuming the Unique Games Conjecture, this is the best approximation factor for Max Cut (Khot et al 2005)

Cut Game

Undirected n-node graph $G = (V, E)$ with nonnegative weights $w_{i,j}$ on edges $(i, j) \in E$
A set $\mathcal{N}$ of n agents, each representing one node in G
A state is $S = (s_1, \ldots, s_m)$, where action s_i has two possible values indicating the node is on the left or right side of the cut

Let $\text{CUT}(S)$ be the set of edges across the cut (between left and right)
Let $\text{CUT}_B(S)$ be the set of edges across the cut which are incident to a node controlled by a player in $B \subseteq N$
The utility for agent i is $u_i(S) = w(\text{CUT}_i(S))$, or the weight of edges incident to i that cross the cut

Cut Game (Continued)

Define a potential function $\Phi(S) = w(\text{CUT}(S))$
The game is an exact potential game, meaning the change in potential for $S' = (S_{-i}, s_i')$ is the same as the change in utility for i
Also guarantees existence of PNE

The stretch of S for a set of agents $R \subseteq N$ is $\frac{\Phi_R(S_{-R}, S_R')}{\Phi_R(S)}$
A ρ-move for player i means that $u_j(S_{-i}, s_i') > \rho \cdot u_j(S)$
A state S is a ρ-approximate pure Nash equilibrium if no agent has a ρ-move

Caragiannis, Jiang (STOC 2022) showed a poly time algorithm for a 2.7371-approx pure Nash equilibrium for cut games (improving on the previous best 3-approx)

The Meta-algorithm

Approximation factor ρ and tolerance parameter ϵ
Agents split into m blocks, polynomially related by the maximum utility each of them could achieve
LOCAL() makes all possible $(\rho + 2\epsilon/3)$-moves for agents within a block
GLOBAL() uses an SDP and subsequent rounding to find a subset $R \subseteq B$ for the current state S with stretch at least $\sigma = \rho + \epsilon/3$, or proves there is none (negative objective value)

Algorithm:
- Start with arbitrary state S
- Create blocks $B_1, \ldots, B_m$
- Perform GLOBAL() and then LOCAL() for block B_1
- Repeat for all blocks $B_2, \ldots, B_m$
- Output final state S

The Meta-algorithm (Continued)

Making improving moves always increases the potential
When the algorithm terminates, there is no ρ-move, so we have a ρ-approximate PNE

The SDP is different than the Max Cut one in that there are OR terms in addition to XOR terms
Dot product with special vector $\hat{v}$ used to determine whether j is in R
Only want to consider $j \in B$ for inclusion, so $v_j = -\hat{v}$ for all $j \in \mathcal{N} \setminus B$

Simple hyperplane rounding will not be enough - we have to use a rounding function first

The SDP

SDP Formulation

$$\max_{v_1,\ldots,v_n,\hat{v}\in\mathbb{R}^n} \sum_{(j,k)\in E\setminus \mathrm{CUT}_B(S)} w_{j,k}\left(\frac{1-v_i\cdot v_j}{2}\right) -$$

$$\sum_{(j,k)\in \mathrm{CUT}_B(S)} w_{j,k}\left(\frac{3\sigma-1+(\sigma-1)v_j\cdot\hat{v}+(\sigma-1)v_k\cdot\hat{v}-(\sigma-1)v_j\cdot v_k}{4}\right)$$

$$\text{subject to} \sum_{(j,k)\in \mathrm{CUT}_B(S)} w_{j,k}(3+v_j\cdot\hat{v}+v_k\cdot\hat{v}-v_j\cdot v_k) \geq \min_{j\in B} U_j$$

$$\hat{v}\cdot\hat{v}=1$$

$$v_j\cdot v_j = 1 \text{ for all } j\in B$$

$$v_j = -\hat{v} \text{ for all } j\in \mathcal{N}\setminus B$$

SDP Rounding

The paper uses rounding function $f(\theta) = \frac{\pi}{2}(1 - \cos\theta)$
For each $v_j \in B$ from the output, produce v_j' by changing the angle with $\hat{v}$ according to $f(\theta)$ in the plane defined by v_j and $\hat{v}$

Then, perform hyperplane rounding
Choose random unit vector $r \in \mathbb{R}^n$, and include j in R if $v_j' \cdot r$ and $\hat{v} \cdot r$ have the same sign

Intuitively, $f(\theta)$ moves vectors close to $\hat{v}$ even closer and vectors far from $\hat{v}$ even further to emphasize the trend from the SDP

Proving Correctness

The only part of the proof that requires the value of ρ is in verifying some properties of the SDP rounding

Lemma 4.2

For all unit vectors $v_j, v_k \in R^n$ with v_j', v_k' obtained by rounding with respect to unit vector $\hat{v} \in \mathbb{R}^n$, the following hold:

$$Pr[\text{XOR}(j,k)] \geq \frac{1}{4}(1 - v_j \cdot v_k)$$

$$Pr[\text{XOR}(j,k)] \leq \frac{1}{8}(3\rho - 1 + (\rho - 1)v_j \cdot \hat{v} + (\rho - 1)v_k \cdot \hat{v} - (\rho + 1)v_j \cdot v_k)$$

Where $Pr[\text{XOR}(j,k)] = \frac{\arccos(v_j' \cdot v_k')}{\pi}$

We can represent things trigonometrically by rewriting vectors and dot products in terms of angles $\theta_j, \theta_k, \phi \in [0, \pi]$

Rounding Functions

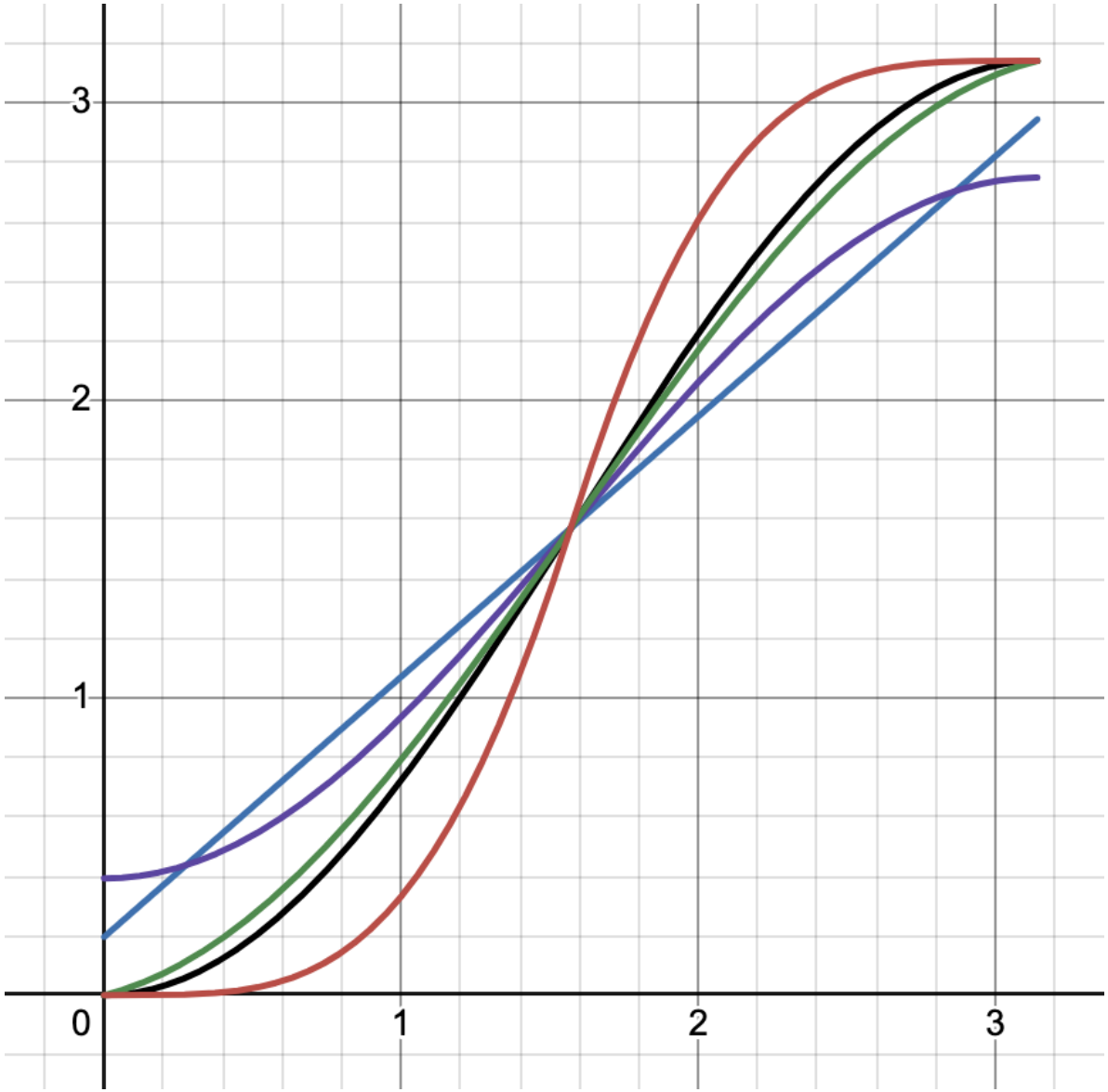
Proving the two inequalities requires a couple pages of clever trig analysis, supplemented by computer verification

Goal: See if changing $f(\theta)$ can allow us to decrease the value of ρ and maintain the analysis, hence improving the approximation factor

Classes of functions with counterexamples:

- $f(\theta) = a\frac{\pi}{2}(1 - \cos\theta) + (1 - a)\theta$ where $0 \leq a \leq 1$
- $f(\theta) = a\theta + b$ where $0 \leq a \leq 1$, $0 \leq b \leq \pi/2$
- $f(\theta) = a\frac{\pi}{2}(1 - \cos\theta) + b$ where $a \leq 1$, $b \geq 0$
- $f(\theta) = \begin{cases} \frac{\pi}{2}(1 - \cos\theta)^2, & \theta < \frac{\pi}{2} \\ \pi - \frac{\pi}{2}(1 + \cos\theta)^2, & \theta \geq \frac{\pi}{2} \end{cases}$

Rounding Functions Visualized



Conclusion

Future directions:
Use a Mathematica program to test rounding functions efficiently
Scale down or linear combination of the piecewise continuous function
Rounding techniques which are not simple functions

Open questions:
Since most alternative rounding functions seem to fail, could $\rho = \frac{1+2\sqrt{13}}{3}$ actually be optimal with this SDP approach?
Perhaps there is a better way to use SDP, or a non-SDP way to improve the approximation factor for PNE in cut games
Unique Games Conjecture impossibility proof hiding somewhere...?