Lecture 16: Semidefinite Programming Relaxation and MAX-CUT

Rafael Oliveira

University of Waterloo
Cheriton School of Computer Science

rafael.oliveira.teaching@gmail.com

November 4, 2021

Overview

- Max-Cut SDP Relaxation
- Max-Cut SDP Rounding
- Conclusion
- Acknowledgements

Relax... & Round!

In our quest to get efficient (exact or approximate) algorithms for problems of interest, the following strategy is very useful:

[1] Even more general mathematical program, so long as derive SDP from it.

Relax... & Round!

In our quest to get efficient (exact or approximate) algorithms for problems of interest, the following strategy is very useful:

1. Formulate optimization problem as QP[1]

[1] Even more general mathematical program, so long as derive SDP from it.

Relax... & Round!

In our quest to get efficient (exact or approximate) algorithms for problems of interest, the following strategy is very useful:

1. Formulate optimization problem as QP[1]
2. Derive SDP from the QP by going to higher dimensions and imposing PSD constraint

This is called an *SDP relaxation*.

[1] Even more general mathematical program, so long as derive SDP from it.

Relax... & Round!

In our quest to get efficient (exact or approximate) algorithms for problems of interest, the following strategy is very useful:

1. Formulate optimization problem as QP[1]
2. Derive SDP from the QP by going to higher dimensions and imposing PSD constraint

   This is called an *SDP relaxation*.
3. We are still maximizing the same objective function, but over a (potentially) larger set of solutions.

$$OPT(SDP) \geq OPT(QP)$$

↑ if our QP is maximization problem

because SDP is over a larger domain

[1] Even more general mathematical program, so long as derive SDP from it.

Relax... & Round!

In our quest to get efficient (exact or approximate) algorithms for problems of interest, the following strategy is very useful:

1. Formulate optimization problem as QP[1]
2. Derive SDP from the QP by going to higher dimensions and imposing PSD constraint

   This is called an *SDP relaxation*.
3. We are still maximizing the same objective function, but over a (potentially) larger set of solutions.

$$OPT(SDP) \geq OPT(QP)$$

4. Solve SDP (approximately) optimally using efficient algorithm.

[1] Even more general mathematical program, so long as derive SDP from it.

Relax... & Round!

In our quest to get efficient (exact or approximate) algorithms for problems of interest, the following strategy is very useful:

1. Formulate optimization problem as QP[1]
2. Derive SDP from the QP by going to higher dimensions and imposing PSD constraint

   This is called an *SDP relaxation*.
3. We are still maximizing the same objective function, but over a (potentially) larger set of solutions.

   $$OPT(SDP) \geq OPT(QP)$$
4. Solve SDP (approximately) optimally using efficient algorithm.
   1. If solution to SDP is *integral* and *one-dimensional*, then it is a solution to QP and we are done

[1] Even more general mathematical program, so long as derive SDP from it.

Relax... & Round!

In our quest to get efficient (exact or approximate) algorithms for problems of interest, the following strategy is very useful:

1. Formulate optimization problem as QP[1]
2. Derive SDP from the QP by going to higher dimensions and imposing PSD constraint

   This is called an *SDP relaxation*.
3. We are still maximizing the same objective function, but over a (potentially) larger set of solutions.

$$OPT(SDP) \geq OPT(QP)$$

4. Solve SDP (approximately) optimally using efficient algorithm.
   1. If solution to SDP is *integral* and *one-dimensional*, then it is a solution to QP and we are done
   2. If solution has *higher dimension*, then we have to devise *rounding procedure* that transforms

      high dimensional solutions $\rightarrow$ integral & 1D solutions

      rounded SDP solution value $\geq c \cdot OPT(QP)$

[1] Even more general mathematical program, so long as derive SDP from it.

Max-Cut

Maximum Cut (Max-Cut):

$G(V, E)$ graph.

Cut $S \subseteq V$ and size of cut is

$$|E(S, \overline{S})| = |\{(u, v) \in E \ | \ u \in S, v \notin S\}|.$$

Goal: find cut of maximum size.

edges across the cut

Example - Weighted Variant

Maximum Cut (Max-Cut):

$G(V, E, w)$ weighted graph. $\sum_{e \in E} w_e = 1$

Cut $S \subseteq V$ and weight of cut is the sum of weights of edges crossing cut.
Goal: find cut of maximum weight.

Max-Cut

$G(V, E, w)$ weighted graph. $\sum_{e \in E} w_e = 1$

Quadratic Program:

$$\text{maximize} \quad \sum_{\{u,v\} \in E} \frac{1}{2} \cdot w_{u,v} \cdot (1 - x_u x_v)$$

$$\text{subject to} \quad x_v^2 = 1 \quad \text{for } v \in V$$

value of the cut

$x_v^2 = 1$, $x_v \in \mathbb{R}$ then $x_v \in \{\pm 1\}$

$$1 - x_u x_v = \begin{cases} 2 & \text{if } x_u \neq x_v \\ 0 & \text{otherwise} \end{cases}$$

SDP Relaxation [Delorme, Poljak 1993]

$G(V, E, w)$ weighted graph, $|V| = n$ and $\sum_{e \in E} w_e = 1$

Semidefinite Program:

$$\begin{aligned} \text{maximize} \quad & \sum_{\{u,v\} \in E} \frac{1}{2} \cdot w_{u,v} \cdot \left(1 - y_u^T y_v\right) \\ \text{subject to} \quad & \|y_v\|_2^2 = 1 \quad \text{for } v \in V \\ & y_v \in \mathbb{R}^d \quad \text{for } v \in V \end{aligned}$$

$x_v \in \mathbb{R} \longrightarrow y_v \in \mathbb{R}^d$

$x_v^2 = 1 \longrightarrow \|y_v\|^2 = 1$

$x_u x_v \longrightarrow y_u \cdot y_v = y_u^T y_v$

what value of d should we take ?

$d = n$ suffices

$Y = \begin{pmatrix} | & | & & | \\ y_1 & y_2 & \cdots & y_n \\ | & | & & | \end{pmatrix}$

$Y^T Y = X$ ← (X must be PSD)

X n×n symmetric matrix

$X_{ij} = y_i \cdot y_j$

and

$X_{ii} = \|y_i\|^2$

- Max-Cut SDP Relaxation
- Max-Cut SDP Rounding
- Conclusion
- Acknowledgements

SDP Relaxation [Delorme, Poljak 1993]

$G(V, E, w)$ weighted graph, $|V| = n$ and $\sum_{e \in E} w_e = 1$

Semidefinite Program:

$$\begin{aligned} \text{maximize} \quad & \sum_{\{u,v\} \in E} \frac{1}{2} \cdot w_{u,v} \cdot \left(1 - y_u^T y_v\right) \\ \text{subject to} \quad & \|y_v\|_2^2 = 1 \quad \text{for } v \in V \\ & y_v \in \mathbb{R}^d \quad \text{for } v \in V \end{aligned}$$

in unit sphere of $\mathbb{R}^d$

$$\text{maximize} \sum_{i,j \in E} \frac{1}{2} w_{ij} (1 - x_{ij})$$

$$\text{s.t.} \quad x_{ii} = 1 \quad \forall i \in [n]$$

$$\text{and } X \succcurlyeq 0$$

What is this SDP doing?

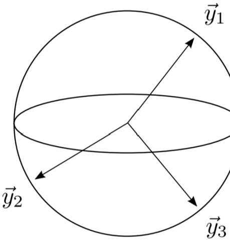


Figure 10.1: Vectors $\vec{y}_v$ embedded onto a unit sphere in $\mathbb{R}^d$.

What is this SDP doing?

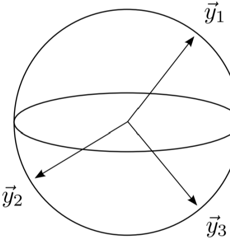
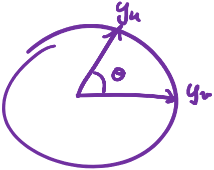


Figure 10.1: Vectors $\vec{y}_v$ embedded onto a unit sphere in $\mathbb{R}^d$.



- Let $\gamma_{u,v} = y_u^T y_v = \cos(y_u, y_v)$

inner product

$$\langle y_u, y_v \rangle = \|y_u\| \cdot \|y_v\| \cdot \cos(y_u, y_v)$$

$$= 1 \cdot 1 \cdot \cos\theta$$

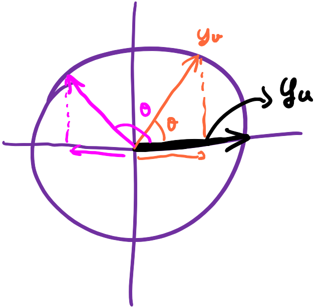
What is this SDP doing?

$\theta \in [0, \pi]$

$\theta \to$ small $\leftrightarrow$ large cosine

$\theta > \pi/2 \leftrightarrow$ cosine negative

$\theta = \pi \leftrightarrow$ cosine $= -1$



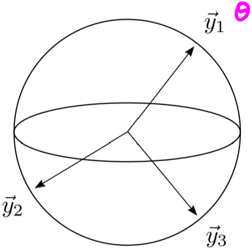
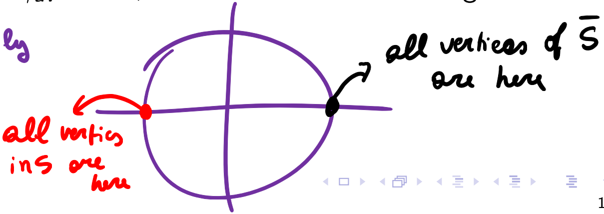


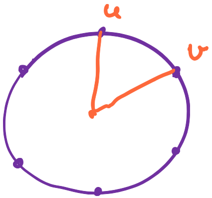
Figure 10.1: Vectors $\vec{y}_v$ embedded onto a unit sphere in $\mathbb{R}^d$.

- Let $\gamma_{u,v} = y_u^T y_v = \cos(y_u, y_v)$
- for any edge, want $\gamma_{uv} \approx -1$, as this maximizes our weight

a cut is exactly



What is this SDP doing?



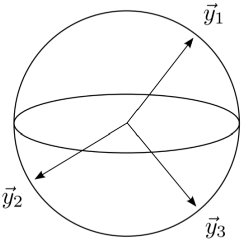


Figure 10.1: Vectors $\vec{y}_v$ embedded onto a unit sphere in $\mathbb{R}^d$.

SDP is pushing apart vertices which have an edge of high weight between them

- Let $\gamma_{u,v} = y_u^T y_v = \cos(y_u, y_v)$
- for any edge, want $\gamma_{uv} \approx -1$, as this maximizes our weight
- Geometrically, want vertices from our max-cut S to be as far away from the complement $\overline{S}$ as possible

What is this SDP doing?

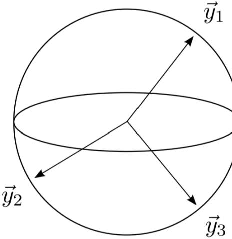


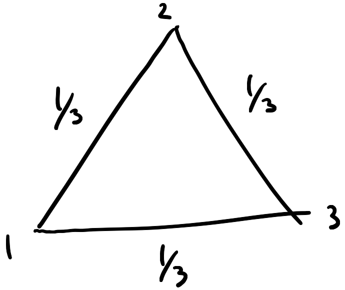
Figure 10.1: Vectors $\vec{y}_v$ embedded onto a unit sphere in $\mathbb{R}^d$.

- Let $\gamma_{u,v} = y_u^T y_v = \cos(y_u, y_v)$
- for any edge, want $\gamma_{uv} \approx -1$, as this maximizes our weight
- Geometrically, want vertices from our max-cut S to be as far away from the complement $\overline{S}$ as possible
- If all y_v's are in a one-dimensional space, then we get original quadratic program

$$OPT(SDP) \geq \text{Weight of Maximum Cut}$$

Example

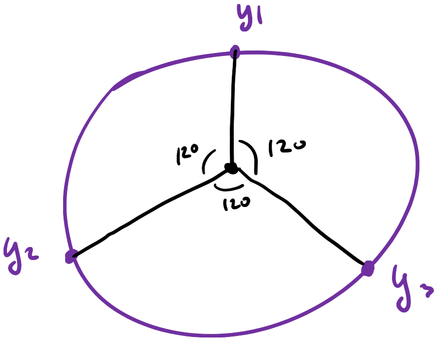
Let's consider $G = K_3$ with equal weights on edges.



Example

Let's consider $G = K_3$ with equal weights on edges.

- Embed $y_1, y_2, y_3 \in \mathbb{R}^2$ 120 degrees apart in unit circle



Example

Let's consider $G = K_3$ with equal weights on edges.

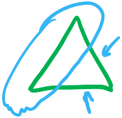
- Embed $y_1, y_2, y_3 \in \mathbb{R}^2$ 120 degrees apart in unit circle
- We get:

Example

exercise

Let's consider $G = K_3$ with equal weights on edges.

- Embed $y_1, y_2, y_3 \in \mathbb{R}^2$ 120 degrees apart in unit circle
- We get:
- $OPT_{SDP}(K_3) = 3/4$
- $OPT_{\text{max-cut}}(K_3) = 2/3$



$$\max \quad \frac{1}{2} \cdot \frac{1}{2} \cdot \Big[\big(1 - \cos(y_1, y_2)\big) + \big(1 - \cos(y_1, y_3)\big) + \big(1 - \cos(y_2, y_3)\big) \Big]$$

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

if not $y_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$ $y_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$ $y_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$

$OPT_{SDP} \stackrel{?}{=} \frac{1}{4} \cdot 3 = 3/4$

Example

Let's consider $G = K_3$ with equal weights on edges.

- Embed $y_1, y_2, y_3 \in \mathbb{R}^2$ 120 degrees apart in unit circle
- We get:
- $OPT_{SDP}(K_3) = 3/4$
- $OPT_{\text{max-cut}}(K_3) = 2/3$
- So we get approximation $8/9$ (better than the LP relaxation)

Example

Let's consider $G = K_3$ with equal weights on edges.

- Embed $y_1, y_2, y_3 \in \mathbb{R}^2$ 120 degrees apart in unit circle
- We get:
- $OPT_{SDP}(K_3) = 3/4$
- $OPT_{\text{max-cut}}(K_3) = 2/3$
- So we get approximation $8/9$ (better than the LP relaxation)
- **Practice problem:** try this with C_5.

Max-Cut - Rounding

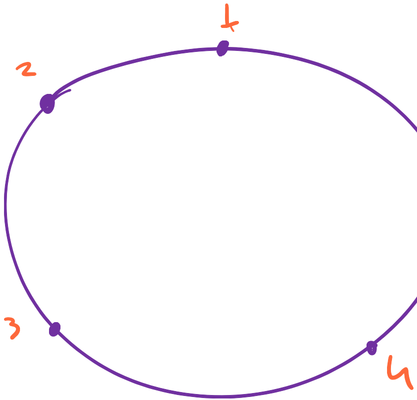
1. Let $z_u \in \mathbb{R}^n$ be an optimal solution to our SDP

Max-Cut - Rounding

1. Let $z_u \in \mathbb{R}^n$ be an optimal solution to our SDP
2. How do we convert it into a cut?

$\{1,2\} \in E$

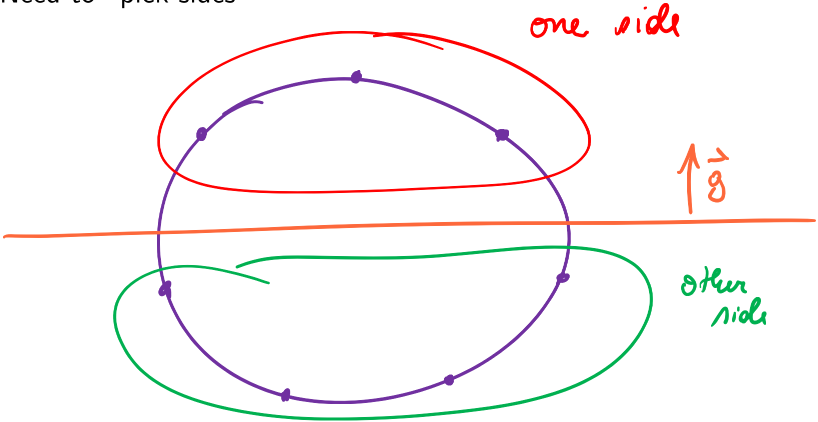
$\{1,3\}$



intuitively $w_{13} > w_{12}$

Max-Cut - Rounding

1. Let $z_u \in \mathbb{R}^n$ be an optimal solution to our SDP
2. How do we convert it into a cut?
3. Need to "pick sides"



Max-Cut - Rounding

1. Let $z_u \in \mathbb{R}^n$ be an optimal solution to our SDP
2. How do we convert it into a cut?
3. Need to "pick sides"
4. **[Goemans, Williamson 1994]:** Choose a random hyperplane though origin!

hope that SDP separated a lot of edges with high weight
(intuition)

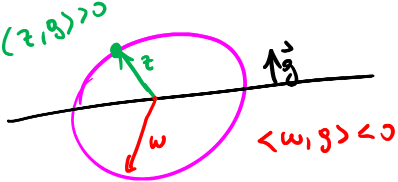
and that a random hyperplane will catch a good cut

Max-Cut - Rounding

1. Let $z_u \in \mathbb{R}^n$ be an optimal solution to our SDP
2. How do we convert it into a cut?
3. Need to "pick sides"
4. **[Goemans, Williamson 1994]:** Choose a random hyperplane though origin!
5. Choose normal vector $g \in \mathbb{R}^n$ from a Gaussian distribution.
6. Set $x_u = \text{sign}(g^T z_u)$ as our solution

$$H = \{ x \in \mathbb{R}^n \mid \langle \vec{g}, x \rangle = 0 \}$$

normal vector



Max-Cut - Rounding

1. Let $z_u \in \mathbb{R}^n$ be an optimal solution to our SDP
2. How do we convert it into a cut?
3. Need to "pick sides"
4. **[Goemans, Williamson 1994]:** Choose a random hyperplane though origin!
5. Choose normal vector $g \in \mathbb{R}^n$ from a Gaussian distribution.
6. Set $x_u = \text{sign}(g^T z_u)$ as our solution

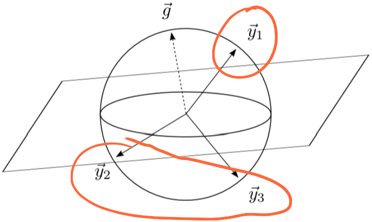


Figure 10.2: Vectors being separated by a hyperplane with normal $\vec{g}$.

Facts we need

- We can pick a random hyperplane through origin in polynomial time.

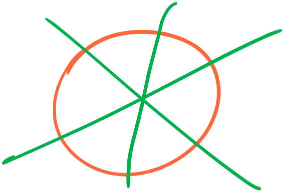
  sample vector $g = (g_1, \ldots, g_n)$ by drawing $g_i \in \mathcal{N}(0,1)$

Normal or Gaussian distribution

Facts we need

- We can pick a random hyperplane through origin in polynomial time.

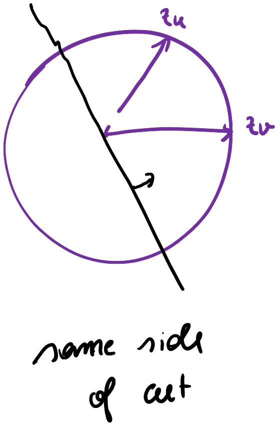
  sample vector $g = (g_1, \ldots, g_n)$ by drawing $g_i \in \mathcal{N}(0, 1)$
- If g' is the projection of g onto a two dimensional plane, then $g'/\|g'\|_2$ is *uniformly distributed* over the unit circle in this plane.

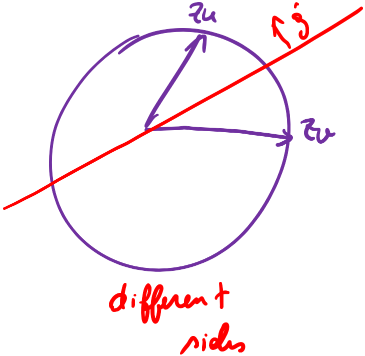


Analysis of Rounding

- Probability that edge $\{u, v\}$ crosses the cut is same as probability that z_u, z_v fall in different sides of hyperplane

$$\Pr[\{u, v\} \text{ crosses cut }] = \Pr[g \text{ splits } z_u, z_v]$$



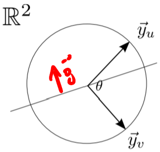


Analysis of Rounding

- Probability that edge $\{u, v\}$ crosses the cut is same as probability that z_u, z_v fall in different sides of hyperplane

$$\Pr[\{u, v\} \text{ crosses cut }] = \Pr[g \text{ splits } z_u, z_v]$$

- Looking at the problem in the plane:



hyperplane we choose is random by fact 2

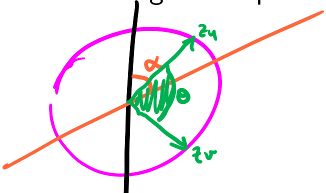
Figure 10.3: The plane of two vectors being cut by the hyperplane

Analysis of Rounding

- Probability that edge $\{u, v\}$ crosses the cut is same as probability that z_u, z_v fall in different sides of hyperplane

$$\Pr[\{u, v\} \text{ crosses cut }] = \Pr[g \text{ splits } z_u, z_v]$$

- Looking at the problem in the plane:



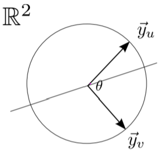


Figure 10.3: The plane of two vectors being cut by the hyperplane

- Probability of splitting z_u, z_v:

$$\Pr[\{u, v\} \text{ crosses cut}] = \frac{\theta}{\pi} = \frac{\cos^{-1}(z_u^T z_v)}{\pi} = \frac{\cos^{-1}(\gamma_{uv})}{\pi}$$

Analysis of Rounding

- Expected value of cut:

$$\mathbb{E}[\text{value of cut}] = \sum_{\{u,v\} \in E} w_{u,v} \cdot \frac{\cos^{-1}(\gamma_{uv})}{\pi}$$

weight of edge u,v

Pr[{u,v} is a cut edge]

E[value of u,v]

Analysis of Rounding

- Expected value of cut:

$$\mathbb{E}[\text{value of cut}] = \sum_{\{u,v\} \in E} w_{u,v} \cdot \frac{\cos^{-1}(\gamma_{uv})}{\pi}$$

- Recall that

$$OPT_{SDP} = \sum_{\{u,v\} \in E} \frac{1}{2} \cdot w_{u,v} \cdot \left(1 - z_u^T z_v\right) = \sum_{\{u,v\} \in E} \frac{1}{2} \cdot w_{u,v} \cdot (1 - \gamma_{uv})$$

because $\{z_u\}_{u \in V}$ is an optimum solution

Analysis of Rounding

- Expected value of cut:

$$\mathbb{E}[\text{value of cut}] = \sum_{\{u,v\}\in E} w_{u,v} \cdot \frac{\cos^{-1}(\gamma_{uv})}{\pi}$$

- Recall that

$$OPT_{SDP} = \sum_{\{u,v\}\in E} \frac{1}{2} \cdot w_{u,v} \cdot \left(1 - z_u^T z_v\right) = \sum_{\{u,v\}\in E} \frac{1}{2} \cdot w_{u,v} \cdot (1 - \gamma_{uv})$$

- If we find α such that

$$\frac{\cos^{-1}(\gamma_{uv})}{\pi} \geq \frac{\alpha}{2}(1 - \gamma_{uv}), \quad \text{for all } \gamma_{uv} \in [-1, 1]$$

Then we have an α-approximation algorithm!

$\Rightarrow$ $\mathbb{E}[\text{value of your cut}] \geq \alpha \cdot OPT_{SDP}$

$\| $

$\sum w_{uv} \cdot \frac{\cos^{-1}(\gamma_{uv})}{\pi} \geq \sum w_{uv} \cdot \frac{\alpha}{2}(1-\gamma_{uv}) = \alpha \cdot \sum \frac{1}{2} w_{uv}(1-\gamma_{uv})$

Analysis of Rounding

- Expected value of cut:

$$\mathbb{E}[\text{value of cut}] = \sum_{\{u,v\}\in E} w_{u,v} \cdot \frac{\cos^{-1}(\gamma_{uv})}{\pi}$$

- Recall that

$$OPT_{SDP} = \sum_{\{u,v\}\in E} \frac{1}{2} \cdot w_{u,v} \cdot \left(1 - z_u^T z_v\right) = \sum_{\{u,v\}\in E} \frac{1}{2} \cdot w_{u,v} \cdot (1 - \gamma_{uv})$$

- If we find α such that

$$\frac{\cos^{-1}(\gamma_{uv})}{\pi} \geq \frac{\alpha}{2}(1 - \gamma_{uv}), \quad \text{for all } \gamma_{uv} \in [-1, 1]$$

  Then we have an α-approximation algorithm!

- For $x \in [-1, 1]$, we have

$$\frac{\cos^{-1}(x)}{\pi} \geq \underbrace{0.878}_{\alpha =} \cdot \frac{1 - x}{2}$$

  proof by elementary calculus.

Conclusion of rounding algorithm

with constant probability we get a cut
with $(0.878 - \epsilon) \cdot OPT$

(can amplify this probability)

Putting Everything Together

1. Formulate Max-Cut problem as Quadratic Programming

Putting Everything Together

1. Formulate Max-Cut problem as Quadratic Programming
2. Derive SDP from the quadratic program *SDP relaxation*

Putting Everything Together

1. Formulate Max-Cut problem as Quadratic Programming
2. Derive SDP from the quadratic program *SDP relaxation*
3. We are still maximizing the same objective function (weight of cut), but over a (potentially) larger (*higher-dimensional*) set of solutions.

$$OPT(SDP) \geq OPT(\text{Max-Cut})$$

Putting Everything Together

1. Formulate Max-Cut problem as Quadratic Programming
2. Derive SDP from the quadratic program *SDP relaxation*
3. We are still maximizing the same objective function (weight of cut), but over a (potentially) larger (*higher-dimensional*) set of solutions.

$$OPT(SDP) \geq OPT(\text{Max-Cut})$$

4. Solve SDP optimally using efficient algorithm.

Putting Everything Together

1. Formulate Max-Cut problem as Quadratic Programming
2. Derive SDP from the quadratic program *SDP relaxation*
3. We are still maximizing the same objective function (weight of cut), but over a (potentially) larger (*higher-dimensional*) set of solutions.

$$OPT(SDP) \geq OPT(\text{Max-Cut})$$

4. Solve SDP optimally using efficient algorithm.
    1. If solution to SDP is *integral* and *one dimensional*, then it is a solution to Max-Cut and we are done

Putting Everything Together

1. Formulate Max-Cut problem as Quadratic Programming
2. Derive SDP from the quadratic program *SDP relaxation*
3. We are still maximizing the same objective function (weight of cut), but over a (potentially) larger (*higher-dimensional*) set of solutions.

   $$OPT(SDP) \geq OPT(\text{Max-Cut})$$

4. Solve SDP optimally using efficient algorithm.
   1. If solution to SDP is *integral* and *one dimensional*, then it is a solution to Max-Cut and we are done
   2. If have *higher dimensional* solutions, *rounding procedure*

      Random Hyperplane Cut algorithm, ~~with high probability~~ we get

      $$\mathbb{E}[\text{cost(rounded solution)}] \geq 0.878 \cdot OPT(SDP) \geq 0.878 \cdot OPT(\text{Max-Cut})$$

Remarks

1. SDPs are very powerful for solving (approximating) many hard problems

Remarks

1. SDPs are very powerful for solving (approximating) many hard problems
2. Recent and exciting work, driven by *Unique Games Conjecture* (UGC), shows that if UGC is true, then all these approximation algorithms are *tight*!

   https://www.cs.cmu.edu/~anupamg/adv-approx/lecture24.pdf

in particular UGC $\Rightarrow$ cannot obtain better than 0.878 approx. for MAX-cut

Remarks

1. SDPs are very powerful for solving (approximating) many hard problems
2. Recent and exciting work, driven by *Unique Games Conjecture* (UGC), shows that if UGC is true, then all these approximation algorithms are *tight*!

   https://www.cs.cmu.edu/~anupamg/adv-approx/lecture24.pdf

3. Other applications in robust statistics, via the SDP & Sum-of-Squares connection

   https://arxiv.org/abs/1711.11581

Remarks

1. SDPs are very powerful for solving (approximating) many hard problems
2. Recent and exciting work, driven by *Unique Games Conjecture* (UGC), shows that if UGC is true, then all these approximation algorithms are *tight*!

   `https://www.cs.cmu.edu/~anupamg/adv-approx/lecture24.pdf`
3. Other applications in robust statistics, via the SDP & Sum-of-Squares connection

   `https://arxiv.org/abs/1711.11581`
4. Connections to automated theorem proving

   https://eccc.weizmann.ac.il/report/2019/106/

Remarks

1. SDPs are very powerful for solving (approximating) many hard problems
2. Recent and exciting work, driven by *Unique Games Conjecture* (UGC), shows that if UGC is true, then all these approximation algorithms are *tight*!

   https://www.cs.cmu.edu/~anupamg/adv-approx/lecture24.pdf
3. Other applications in robust statistics, via the SDP & Sum-of-Squares connection

   https://arxiv.org/abs/1711.11581
4. Connections to automated theorem proving

   https://eccc.weizmann.ac.il/report/2019/106/

All of these are amazing final project topics!

Conclusion

- Mathematical programming - very general, and pervasive in (combinatorial) algorithmic life
- Mathematical Programming hard in general
- Sometimes can get SDP rounding!
- Solve SDP and round the solution
  - Deterministic rounding when solutions are nice
  - Randomized rounding when things a bit more complicated

Acknowledgement

- Lecture based largely on:
  - Lecture 14 of Anupam Gupta and Ryan O'Donnell's Optimization class

    `https://www.cs.cmu.edu/~anupamg/adv-approx/`
  - Chapter 6 of book [Williamson, Shmoys 2010]
- See their notes at

  `https://www.cs.cmu.edu/~anupamg/adv-approx/lecture14.pdf`

References I

Delorme, Charles, and Svatopluk Poljak (1993)
Laplacian eigenvalues and the maximum cut problem.
Mathematical Programming 62.1-3 (1993): 557-574.

Goemans, Michel and Williamson, David 1994
0.879-approximation algorithms for Max Cut and Max 2SAT.
Proceedings of the twenty-sixth annual ACM symposium on Theory of computing. 1994

Williamson, David and Shmoys, David 2010
Design of Approximation Algorithms
Cambridge University Press