# Lecture 15: Semidefinite Programming, Duality & SDP Relaxations

Rafael Oliveira

University of Waterloo Cheriton School of Computer Science

rafael.oliveira.teaching@gmail.com

June 29, 2023

イロト 不得 トイヨト イヨト 二日

1/67

### Overview

- Duality Theory
- Why Relax & Round?
- Conclusion
- Acknowledgements

#### Definition (Frobenius Inner Product)

 $A, B \in \mathcal{S}^m$ , define the *Frobenius inner product* as

$$\langle A,B
angle:= {\sf tr}[AB] = \sum_{i,j} A_{ij}B_{ij}$$

#### Definition (Frobenius Inner Product)

 $A,B\in\mathcal{S}^m$ , define the *Frobenius inner product* as

$$\langle A,B
angle:= {\sf tr}[AB] = \sum_{i,j} A_{ij}B_{ij}$$

• This is the "usual inner product" if you think of the matrices as vectors

#### Definition (Frobenius Inner Product)

 $A, B \in \mathcal{S}^m$ , define the *Frobenius inner product* as

$$\langle A,B
angle:= {\sf tr}[AB] = \sum_{i,j} A_{ij}B_{ij}$$

- This is the "usual inner product" if you think of the matrices as vectors
- Thus, have the norm

$$\|A\|_{F} = \sqrt{\langle A, A \rangle} = \sqrt{\sum_{i,j} A_{ij}^{2}}$$

#### Definition (Frobenius Inner Product)

 $A, B \in \mathcal{S}^m$ , define the *Frobenius inner product* as

$$\langle A,B
angle:= {\sf tr}[AB] = \sum_{i,j} A_{ij}B_{ij}$$

- This is the "usual inner product" if you think of the matrices as vectors
- Thus, have the norm

$$\|A\|_F = \sqrt{\langle A, A 
angle} = \sqrt{\sum_{i,j} A_{ij}^2}$$

With this norm, can talk about the *polar dual* to a given spectrahedron S ⊆ S<sup>m</sup>:

$$S^{\circ} = \{Y \in \mathcal{S}^m \mid \langle Y, X \rangle \leq 1, \ \forall X \in S\}$$

イロト 不得 トイヨト イヨト 二日

Just like in Linear Programming, we can represent SDPs in standard form:

$$\begin{array}{ll} \text{minimize} & \langle C, X \rangle \\ \text{subject to} & \langle A_i, X \rangle = b_i \\ & X \succeq 0 \end{array}$$

Just like in Linear Programming, we can represent SDPs in standard form:

$$\begin{array}{ll} \text{minimize} & \langle C, X \rangle \\ \text{subject to} & \langle A_i, X \rangle = b_i \\ & X \succeq 0 \end{array}$$

Where now:

• the variables are encoded in a positive semidefinite matrix X,

Just like in Linear Programming, we can represent SDPs in standard form:

$$\begin{array}{ll} \text{minimize} & \langle C, X \rangle \\ \text{subject to} & \langle A_i, X \rangle = b_i \\ & X \succeq 0 \end{array}$$

- the variables are encoded in a positive semidefinite matrix X,
- each constraint is given by an inner product  $\langle A_i, X \rangle = b_i$

Just like in Linear Programming, we can represent SDPs in standard form:

$$\begin{array}{ll} \text{minimize} & \langle C, X \rangle \\ \text{subject to} & \langle A_i, X \rangle = b_i \\ & X \succeq 0 \end{array}$$

- the variables are encoded in a positive semidefinite matrix X,
- each constraint is given by an inner product  $\langle A_i, X \rangle = b_i$
- Note the similarity with LP standard primal. Can obtain LP standard form by making X and A<sub>i</sub>'s to be diagonal

Just like in Linear Programming, we can represent SDPs in standard form:

$$\begin{array}{ll} \text{minimize} & \langle C, X \rangle \\ \text{subject to} & \langle A_i, X \rangle = b_i \\ & X \succeq 0 \end{array}$$

- the variables are encoded in a positive semidefinite matrix X,
- each constraint is given by an inner product  $\langle A_i, X \rangle = b_i$
- Note the similarity with LP standard primal. Can obtain LP standard form by making X and A<sub>i</sub>'s to be diagonal
- How is that an LMI though?

# Standard Primal Form as LMI

$$\begin{array}{ll} \text{minimize} & \langle C, X \rangle \\ \text{subject to} & \langle A_i, X \rangle = b_i \\ & X \succ 0 \end{array}$$

# Example

minimize 
$$2x_{11} + 2x_{12}$$
  
subject to 
$$x_{11} + x_{22} = 1$$
$$\begin{pmatrix} x_{11} & x_{12} \\ x_{12} & x_{22} \end{pmatrix} \succeq 0$$

 $\begin{array}{ll} \text{minimize} & \langle C, X \rangle \\ \text{subject to} & \langle A_i, X \rangle = b_i \\ & X \succeq 0 \end{array}$ 

$$\begin{array}{ll} \text{minimize} & \langle C, X \rangle \\ \text{subject to} & \langle A_i, X \rangle = b_i \\ & X \succeq 0 \end{array}$$

• If we look at what happens when we multiply *i*<sup>th</sup> equality by a variable *y<sub>i</sub>*:

$$\sum_{i=1}^{t} y_i \cdot \langle A_i, X \rangle = \sum_{i=1}^{t} y_i \cdot b_i \quad \Rightarrow \quad \left\langle \sum_{i=1}^{t} y_i A_i, X \right\rangle = y^T b$$

 $\begin{array}{ll} \text{minimize} & \langle C, X \rangle \\ \text{subject to} & \langle A_i, X \rangle = b_i \\ & X \succeq 0 \end{array}$ 

If we look at what happens when we multiply *i<sup>th</sup>* equality by a variable *y<sub>i</sub>*:

$$\sum_{i=1}^{t} y_i \cdot \langle A_i, X \rangle = \sum_{i=1}^{t} y_i \cdot b_i \quad \Rightarrow \quad \left\langle \sum_{i=1}^{t} y_i A_i , X \right\rangle = y^T b$$
  
• Thus, if  $\sum_{i=1}^{t} y_i A_i \leq C$ , then we have:  

$$y^T b = \left\langle \sum_{i=1}^{t} y_i A_i , X \right\rangle \leq \langle C, X \rangle$$

 $\begin{array}{ll} \text{minimize} & \langle C, X \rangle \\ \text{subject to} & \langle A_i, X \rangle = b_i \\ & X \succeq 0 \end{array}$ 

• If we look at what happens when we multiply *i*<sup>th</sup> equality by a variable *y<sub>i</sub>*:

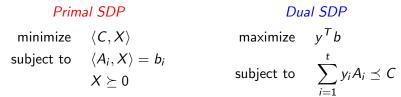
$$\sum_{i=1}^{t} y_i \cdot \langle A_i, X \rangle = \sum_{i=1}^{t} y_i \cdot b_i \quad \Rightarrow \quad \left\langle \sum_{i=1}^{t} y_i A_i , X \right\rangle = y^T b$$
  
• Thus, if  $\sum_{i=1}^{t} y_i A_i \leq C$ , then we have:  

$$y^T b = \left\langle \sum_{i=1}^{t} y_i A_i , X \right\rangle \leq \langle C, X \rangle$$

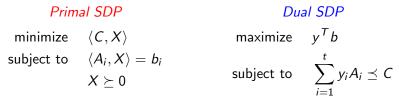
•  $y^T b$  is a *lower bound* on the solution to our SDP!

17 / 67

Consider the following SDPs:



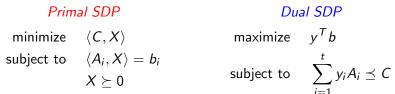
Consider the following SDPs:



• From previous slide

 $\sum_{i=1}^{t} y_i A_i \preceq C \Rightarrow y^T b \text{ is a lower bound on value of Primal}$ 

Consider the following SDPs:

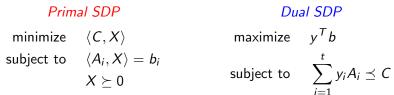


• From previous slide

 $\sum_{i=1}^{t} y_i A_i \preceq C \Rightarrow y^T b \text{ is a lower bound on value of Primal}$ 

• Thus, the optimal (maximum) value of *dual LP* lower bounds the optimal (minimum) value of the *Primal LP*!

Consider the following SDPs:



• From previous slide

 $\sum_{i=1}^{t} y_i A_i \preceq C \Rightarrow y^T b \text{ is a lower bound on value of Primal}$ 

• Thus, the optimal (maximum) value of *dual LP* lower bounds the optimal (minimum) value of the *Primal LP*!

#### Theorem (Weak Duality)

Let X be a feasible solution of the primal SDP and y be a feasible solution of the dual SDP. Then

 $y^T b \leq \langle C, X \rangle.$ 

## **Complementary Slackness**

# Primal SDPDual SDPminimize $\langle C, X \rangle$ maximize $y^T b$ subject to $\langle A_i, X \rangle = b_i$ subject to $\sum_{i=1}^t y_i A_i \preceq C$

## **Complementary Slackness**

# Primal SDPDual SDPminimize $\langle C, X \rangle$ maximize $y^T b$ subject to $\langle A_i, X \rangle = b_i$ subject to $\sum_{i=1}^{t} y_i A_i \preceq C$

#### Theorem (Complementary Slackness)

Let X be a feasible solution of the primal SDP and y be a feasible solution of the dual SDP. If (X, y) satisfy the complementary slackness condition

$$\left(C-\sum_{i=1}^t y_i A_i\right)X=0$$

Then (X, y) are primal and dual optimum solutions of the SDP problem.

## **Complementary Slackness**

# Primal SDPDual SDPminimize $\langle C, X \rangle$ maximize $y^T b$ subject to $\langle A_i, X \rangle = b_i$ subject to $\sum_{i=1}^{t} y_i A_i \preceq C$

#### Theorem (Complementary Slackness)

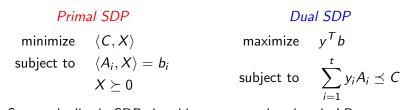
Let X be a feasible solution of the primal SDP and y be a feasible solution of the dual SDP. If (X, y) satisfy the complementary slackness condition

$$\left(C-\sum_{i=1}^t y_i A_i\right)X=0$$

Then (X, y) are primal and dual optimum solutions of the SDP problem.

Complementary slackness gives us *sufficient* conditions to check optimality of our solutions. (24/67)

# Primal SDPDual SDPminimize $\langle C, X \rangle$ maximize $y^T b$ subject to $\langle A_i, X \rangle = b_i$ subject to $\sum_{i=1}^t y_i A_i \preceq C$



Strong duality in SDPs is a bit more complex than in LPs.

Primal SDP		Dual SDP	
minimize	$\langle C, X \rangle$	maximize	у <sup>т</sup> b
subject to	$\langle A_i, X \rangle = b_i$ $X \succeq 0$	subject to	$\sum_{i=1}^t y_i A_i \preceq C$

- Strong duality in SDPs is a bit more complex than in LPs.
- Both primal and dual may be feasible, and yet strong duality may not hold!

Primal SDP		Dual SDP	
minimize	$\langle C, X \rangle$	maximize	у <sup>т</sup> b
subject to	$\langle A_i, X \rangle = b_i$ $X \succeq 0$	subject to	$\sum_{i=1}^t y_i A_i \preceq C$

- Strong duality in SDPs is a bit more complex than in LPs.
- Both primal and dual may be feasible, and yet strong duality may not hold!
- But under mild conditions, strong duality holds!

Primal SDP		Dual SDP	
minimize	$\langle C, X \rangle$	maximize	у <sup>Т</sup> Ь
subject to	$\langle A_i, X \rangle = b_i$ $X \succeq 0$	subject to	$\sum_{i=1}^t y_i A_i \preceq C$

- Strong duality in SDPs is a bit more complex than in LPs.
- Both primal and dual may be feasible, and yet strong duality may not hold!
- But under mild conditions, strong duality holds!
- Primal SDP is *strictly feasible* if there is feasible solution  $X \succ 0$ .
- Dual SDP is *strictly feasible* if there is feasible  $\sum_{i=1}^{t} y_i A_i \prec C$ .

Primal SDP		Dual SDP	
minimize	$\langle C, X \rangle$	maximize	у <sup>т</sup> b
subject to	$egin{array}{l} \langle {\sf A}_i, X  angle = b_i \ X \succeq 0 \end{array}$	subject to	$\sum_{i=1}^t y_i A_i \preceq C$

- Strong duality in SDPs is a bit more complex than in LPs.
- Both primal and dual may be feasible, and yet strong duality may not hold!
- But under mild conditions, strong duality holds!
- Primal SDP is *strictly feasible* if there is feasible solution  $X \succ 0$ .
- Dual SDP is *strictly feasible* if there is feasible  $\sum_{i=1}^{t} y_i A_i \prec C$ .

#### Theorem (Strong Duality under Slater Conditions)

If primal SDP and dual SDP are both strictly feasible, then

max dual = min of primal.

#### • Duality Theory

• Why Relax & Round?

Conclusion

Acknowledgements

- Many important problems are NP-hard to solve.
- What do we do when we see one?

- Many important problems are NP-hard to solve.
- What do we do when we see one?
  - Find approximate solutions in polynomial time!
  - Sometimes we even do that for problems in P (but we want much much faster solutions)

- Many important problems are NP-hard to solve.
- What do we do when we see one?
  - Ind approximate solutions in polynomial time!
  - Sometimes we even do that for problems in P (but we want much much faster solutions)

#### • Integer Linear Program (ILP):

minimize  $c^T x$ subject to  $Ax \leq b$  $x \in \mathbb{N}^n$ 

- Many important problems are NP-hard to solve.
- What do we do when we see one?
  - Ind approximate solutions in polynomial time!
  - Sometimes we even do that for problems in P (but we want much much faster solutions)
- Integer Linear Program (ILP):

minimize  $c^T x$ subject to  $Ax \leq b$  $x \in \mathbb{N}^n$ 

- Advantage of ILPs: very expressive language to formulate optimization problems (capture many combinatorial optimization problems)
- Disadvantage of ILPs: capture even NP-hard problems (thus NP-hard)
- But we know how to solve LPs. Can we get partial credit in life?

• Quadratic Program (QP):

minimize g(x)subject to  $q_i(x) \ge 0$ 

where each  $q_i(x)$  and g(x) are quadratic functions on x.

• Quadratic Program (QP):

 $\begin{array}{ll} \mbox{minimize} & g(x) \\ \mbox{subject to} & q_i(x) \geq 0 \end{array}$ 

where each  $q_i(x)$  and g(x) are quadratic functions on x.

 Advantage of QPs: very expressive language to formulate optimization problems

• Quadratic Program (QP):

 $\begin{array}{ll} \mbox{minimize} & g(x) \\ \mbox{subject to} & q_i(x) \geq 0 \end{array}$ 

- Advantage of QPs: very expressive language to formulate optimization problems
- Disadvantage of QPs: capture even NP-hard problems (ILPs for instance)

• Quadratic Program (QP):

 $\begin{array}{ll} \mbox{minimize} & g(x) \\ \mbox{subject to} & q_i(x) \geq 0 \end{array}$ 

- Advantage of QPs: very expressive language to formulate optimization problems
- Disadvantage of QPs: capture even NP-hard problems (ILPs for instance)
- Can relax quadratic programs with SDPs

• Quadratic Program (QP):

 $\begin{array}{ll} \mbox{minimize} & g(x) \\ \mbox{subject to} & q_i(x) \geq 0 \end{array}$ 

- Advantage of QPs: very expressive language to formulate optimization problems
- Disadvantage of QPs: capture even NP-hard problems (ILPs for instance)
- Can relax quadratic programs with SDPs
  - Can we get better approximations using SDPs instead of ILPs?

• Quadratic Program (QP):

 $\begin{array}{ll} \mbox{minimize} & g(x) \\ \mbox{subject to} & q_i(x) \geq 0 \end{array}$ 

- Advantage of QPs: very expressive language to formulate optimization problems
- Disadvantage of QPs: capture even NP-hard problems (ILPs for instance)
- Can relax quadratic programs with SDPs
  - Can we get better approximations using SDPs instead of ILPs?
- Yes. Today and next lecture we will see Max-Cut (more generally constraint satisfaction relaxations)

• Quadratic Program (QP):

 $\begin{array}{ll} \mbox{minimize} & g(x) \\ \mbox{subject to} & q_i(x) \geq 0 \end{array}$ 

- Advantage of QPs: very expressive language to formulate optimization problems
- Disadvantage of QPs: capture even NP-hard problems (ILPs for instance)
- Can relax quadratic programs with SDPs
  - Can we get better approximations using SDPs instead of ILPs?
- Yes. Today and next lecture we will see Max-Cut (more generally constraint satisfaction relaxations)
- Very impressive recent theoretical developments! Unique Games Conjecture, Sum-of-Squares, and more!

## Example

Maximum Cut (Max-Cut):

G(V, E) graph.

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

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

Goal: find cut of maximum size.

#### Example

Maximum Cut (Max-Cut):

G(V, E) graph.

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

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

Goal: find cut of maximum size.

Integer Linear Program:

$$\begin{array}{ll} \text{maximize} & \sum_{e \in E} z_e \\ \text{subject to} & x_u + x_v \ge z_e \quad \text{for } e = \{u, v\} \in E \\ & 2 - x_u - x_v \ge z_e \quad \text{for } e = \{u, v\} \in E \\ & x_v \in \{0, 1\} \quad \text{for } v \in V \end{array}$$

#### 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.

Integer Linear Program:

$$\begin{array}{ll} \text{maximize} & \sum_{e \in E} z_e \cdot w_e \\ \text{subject to} & x_u + x_v \geq z_e \ \text{ for } e = \{u, v\} \in E \\ & 2 - x_u - x_v \geq z_e \ \text{ for } e = \{u, v\} \in E \\ & x_v \in \{0, 1\} \ \text{ for } v \in V \end{array}$$

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  $\rightarrow$   $_{2}$  -  $\odot$   $\sim$ 

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

Formulate optimization problem as QP<sup>1</sup>

<sup>&</sup>lt;sup>1</sup>Even more general mathematical program, so long as derive SDP from it. = -20 (or = -20)

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

- Formulate optimization problem as QP<sup>1</sup>
- Oerive SDP from the QP by going to higher dimensions and imposing PSD constraint

This is called an *SDP relaxation*.

 $<sup>^1</sup>Even$  more general mathematical program, so long as derive SDP from it  $\rightarrow$ 

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

- Formulate optimization problem as QP<sup>1</sup>
- Oerive SDP from the QP by going to higher dimensions and imposing PSD constraint

This is called an *SDP relaxation*.

We are still maximizing the same objective function, but over a (potentially) larger set of solutions.

 $OPT(SDP) \ge OPT(QP)$ 

<sup>&</sup>lt;sup>1</sup>Even more general mathematical program, so long as derive SDP from it.  $\sim = -9$  and  $\sim = -9$ 

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

- Formulate optimization problem as QP<sup>1</sup>
- Oerive SDP from the QP by going to higher dimensions and imposing PSD constraint

This is called an *SDP relaxation*.

We are still maximizing the same objective function, but over a (potentially) larger set of solutions.

 $OPT(SDP) \ge OPT(QP)$ 

Solve SDP (approximately) optimally using efficient algorithm.

<sup>&</sup>lt;sup>1</sup>Even more general mathematical program, so long as derive SDP from it  $\rightarrow$   $\equiv$   $\Im$   $\Im$ 

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

- Formulate optimization problem as QP<sup>1</sup>
- Oerive SDP from the QP by going to higher dimensions and imposing PSD constraint

This is called an *SDP relaxation*.

We are still maximizing the same objective function, but over a (potentially) larger set of solutions.

 $OPT(SDP) \ge OPT(QP)$ 

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

<sup>&</sup>lt;sup>1</sup>Even more general mathematical program, so long as derive SDP from it.  $\sim = -9$  and

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

- Formulate optimization problem as QP<sup>1</sup>
- Oerive SDP from the QP by going to higher dimensions and imposing PSD constraint

This is called an *SDP relaxation*.

We are still maximizing the same objective function, but over a (potentially) larger set of solutions.

 $OPT(SDP) \ge OPT(QP)$ 

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

<sup>1</sup>Even more general mathematical program, so long as derive SDP from it,  $\sim 2$ 

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

Integer Linear Program:

$$\begin{array}{ll} \text{maximize} & \sum_{e \in E} z_e \cdot w_e \\ \text{subject to} & x_u + x_v \geq z_e \ \text{ for } e = \{u, v\} \in E \\ & 2 - x_u - x_v \geq z_e \ \text{ for } e = \{u, v\} \in E \\ & x_v \in \{0, 1\} \ \text{ for } v \in V \end{array}$$

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

Integer Linear Program:

$$\begin{array}{ll} \text{maximize} & \sum_{e \in E} z_e \cdot w_e \\ \text{subject to} & x_u + x_v \geq z_e \ \text{ for } e = \{u, v\} \in E \\ & 2 - x_u - x_v \geq z_e \ \text{ for } e = \{u, v\} \in E \\ & x_v \in \{0, 1\} \ \text{ for } v \in V \end{array}$$

•  $OPT(ILP) = 1 \Leftrightarrow G$  is bipartite

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

Integer Linear Program:

$$\begin{array}{ll} \text{maximize} & \sum_{e \in E} z_e \cdot w_e \\ \text{subject to} & x_u + x_v \geq z_e \ \text{ for } e = \{u, v\} \in E \\ & 2 - x_u - x_v \geq z_e \ \text{ for } e = \{u, v\} \in E \\ & x_v \in \{0, 1\} \ \text{ for } v \in V \end{array}$$

OPT(ILP) = 1 ⇔ G is bipartite
OPT(ILP) ≥ 1/2

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

Integer Linear Program:

$$\begin{array}{ll} \text{maximize} & \sum_{e \in E} z_e \cdot w_e \\ \text{subject to} & x_u + x_v \geq z_e \ \text{ for } e = \{u, v\} \in E \\ & 2 - x_u - x_v \geq z_e \ \text{ for } e = \{u, v\} \in E \\ & x_v \in \{0, 1\} \ \text{ for } v \in V \end{array}$$

• 
$$OPT(ILP) = 1 \Leftrightarrow G$$
 is bipartite

•  $OPT(ILP) \geq 1/2$ 

• G complete graph 
$$\Rightarrow OPT = \frac{1}{2} + \frac{1}{2(n-1)}$$

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

Integer Linear Program:

$$\begin{array}{ll} \text{maximize} & \sum_{e \in E} z_e \cdot w_e \\ \text{subject to} & x_u + x_v \geq z_e \ \text{ for } e = \{u, v\} \in E \\ & 2 - x_u - x_v \geq z_e \ \text{ for } e = \{u, v\} \in E \\ & x_v \in \{0, 1\} \ \text{ for } v \in V \end{array}$$

• 
$$OPT(ILP) = 1 \Leftrightarrow G$$
 is bipartite

•  $OPT(ILP) \ge 1/2$ 

• G complete graph 
$$\Rightarrow OPT = \frac{1}{2} + \frac{1}{2(n-1)}$$

Max-Cut NP-hard

# Proof that $OPT(ILP) \ge 1/2$

<ロト < 回 ト < 言 ト < 言 ト 注 の < で 58 / 67

# Rounding Max-Cut ILP

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

Linear Program Relaxation:

$$\begin{array}{ll} \text{maximize} & \sum_{e \in E} z_e \cdot w_e \\ \text{subject to} & x_u + x_v \ge z_e \quad \text{for } e = \{u, v\} \in E \\ & 2 - x_u - x_v \ge z_e \quad \text{for } e = \{u, v\} \in E \\ & 0 \le x_v \le 1 \quad \text{for } v \in V \\ & 0 \le z_e \le 1 \quad \text{for } e \in E \end{array}$$

#### Rounding Max-Cut ILP

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

Linear Program Relaxation:

$$\begin{array}{ll} \text{maximize} & \sum_{e \in E} z_e \cdot w_e \\ \text{subject to} & x_u + x_v \geq z_e \ \text{ for } e = \{u, v\} \in E \\ & 2 - x_u - x_v \geq z_e \ \text{ for } e = \{u, v\} \in E \\ & 0 \leq x_v \leq 1 \ \text{ for } v \in V \\ & 0 \leq z_e \leq 1 \ \text{ for } e \in E \end{array}$$

• Setting  $x_v = 1/2$ ,  $z_e = 1$  we get OPT(LP) always = 1

## Rounding Max-Cut ILP

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

Linear Program Relaxation:

$$\begin{array}{ll} \text{maximize} & \sum_{e \in E} z_e \cdot w_e \\ \text{subject to} & x_u + x_v \geq z_e \quad \text{for } e = \{u, v\} \in E \\ & 2 - x_u - x_v \geq z_e \quad \text{for } e = \{u, v\} \in E \\ & 0 \leq x_v \leq 1 \quad \text{for } v \in V \\ & 0 \leq z_e \leq 1 \quad \text{for } e \in E \end{array}$$

• Setting  $x_v = 1/2$ ,  $z_e = 1$  we get OPT(LP) always = 1

• This relaxation is not helpful! :(

# Max-Cut

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

Quadratic Program:

maximize 
$$\sum_{\{u,v\}\in E} \frac{1}{2} \cdot w_{u,v} \cdot (1 - x_u x_v)$$
  
subject to  $x_v^2 = 1$  for  $v \in V$ 

SDP Relaxation [Delorme, Poljak 1993] G(V, E, w) weighted graph, |V| = n and  $\sum_{e \in E} w_e = 1$ 

Semidefinite Program:

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

SDP Relaxation [Delorme, Poljak 1993] G(V, E, w) weighted graph, |V| = n and  $\sum_{e \in E} w_e = 1$ 

Semidefinite Program:

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

• How is that an SDP?

# Conclusion

- Mathematical programming very general, and pervasive in (combinatorial) algorithmic life
- Mathematical Programming hard in general
- Sometimes can get SDP rounding!

Next lecture Max-Cut 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/
- 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