Introduction to
Approximation Algorithms

Mong-Jen Kao (高孟駿)

Friday 13:20 – 15:10

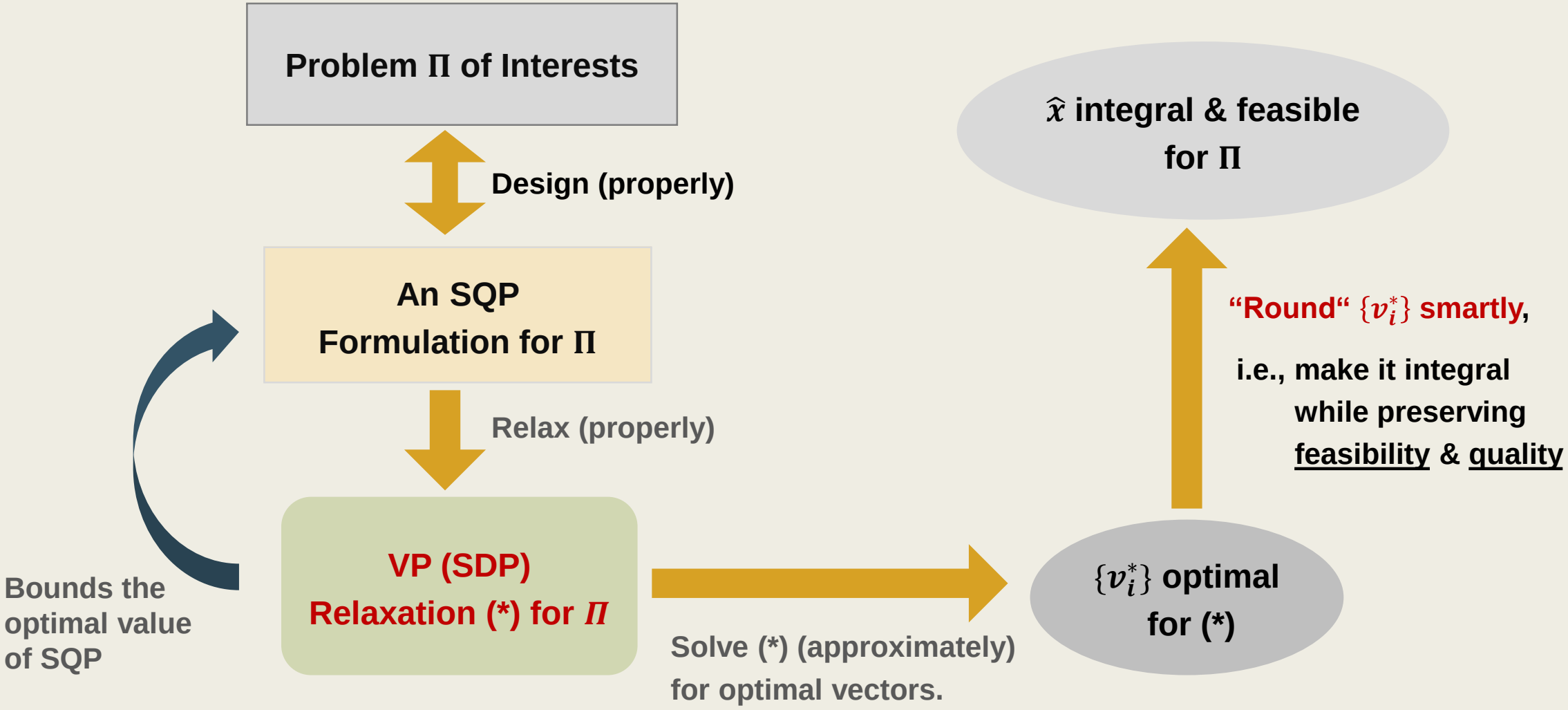
Strict Quadratic Program (SQP)

&

Vector Programming Relaxation

When ILP & LP relaxation do not work well,
higher-degree programs can be a possibility.

The Flowchart for VP (SDP) rounding



Strict Quadratic Programs (SQP)

- A quadratic program (QP) is the problem of
  - Optimizing a ***quadratic function*** subject to ***quadratic constraints***.
- When each monomial of a QP is ***of degree 0 or 2***,
  it is called a strict QP.

$$\begin{aligned} &\text{optimize} && \sum_{1\le i\le j\le n} a_{i,j}\cdot(x_i\cdot x_j)+c && (\boldsymbol{SQP})\\ &\text{s.t.} && \sum_{1\le i\le j\le n} c_{i,j}^{(k)}\cdot(x_i\cdot x_j)\le b_k, && \forall 1\le k\le m,\\ & && x_i\in\mathbb{R}, && \forall 1\le i\le n. \end{aligned}$$

Vector Programs (VP)

- A vector program (VP) is the problem of
  - Optimizing a *linear function* over *linear constraints* consisting of ***inner products of vector variables***.

$$\begin{aligned} \text{optimize} \quad & \sum_{1 \le i \le j \le n} a_{i,j} \cdot (v_i \cdot v_j) && (\boldsymbol{VP}) \\ \text{s.t.} \quad & \sum_{1 \le i \le j \le n} c_{i,j}^{(k)} \cdot (v_i \cdot v_j) \le b_k, && \forall 1 \le k \le m, \\ & v_i \in \mathbb{R}^n, && \forall 1 \le i \le n. \end{aligned}$$

VPs are Polynomial-time Solvable.

- Vector programs can be solved within *an additive error of ϵ* in time ***polynomial*** **in n and $\log(1/\epsilon)$**.

$$\begin{aligned} \text{optimize} \quad & \sum_{1 \le i \le j \le n} a_{i,j} \cdot (v_i \cdot v_j) && (\boldsymbol{VP}) \\ \text{s.t.} \quad & \sum_{1 \le i \le j \le n} c_{i,j}^{(k)} \cdot (v_i \cdot v_j) \le b_k, && \forall 1 \le k \le m, \\ & v_i \in \mathbb{R}^n, && \forall 1 \le i \le n. \end{aligned}$$

- VP is equivalent to semidefinite programming (SDP).

We will introduce
the concept of SDP later!

VPs as relaxations for Integer SQP

- Since VPs are approximately poly-time solvable,
  they yield good relaxations for integer SQPs.
  - **Replace** scalar variables $\boldsymbol{x_i}$ with **vector variables** $\boldsymbol{v_i} \in \mathbb{R}^{\boldsymbol{n}}$
  - **Replace** each quadratic monomial $\boldsymbol{x_i} \cdot \boldsymbol{x_j}$ with **inner product** $\boldsymbol{v_i} \cdot \boldsymbol{v_j}$

Note that, *the number of dimension must be at least* n, the number of scalar variables.

$$\begin{aligned} \text{optimize} \quad & \sum_{1 \le i \le j \le n} a_{i,j} \cdot (x_i \cdot x_j) + c && (\boldsymbol{SQP}) \\ \text{s.t.} \quad & \sum_{1 \le i \le j \le n} c_{i,j}^{(k)} \cdot (x_i \cdot x_j) \le b_k, && \forall 1 \le k \le m, \\ & x_i \in \mathbb{R}, && \forall 1 \le i \le n. \end{aligned}$$

$$\begin{aligned} \text{optimize} \quad & \sum_{1 \le i \le j \le n} a_{i,j} \cdot (v_i \cdot v_j) && (\boldsymbol{VP}) \\ \text{s.t.} \quad & \sum_{1 \le i \le j \le n} c_{i,j}^{(k)} \cdot (v_i \cdot v_j) \le b_k, && \forall 1 \le k \le m, \\ & v_i \in \mathbb{R}^n, && \forall 1 \le i \le n. \end{aligned}$$

VPs as relaxations for Integer SQP

$$\begin{aligned} \text{optimize} \quad & \sum_{1 \le i \le j \le n} a_{i,j} \cdot (x_i \cdot x_j) + c && (\boldsymbol{SQP}) \\ \text{s.t.} \quad & \sum_{1 \le i \le j \le n} c_{i,j}^{(k)} \cdot (x_i \cdot x_j) \le b_k, && \forall 1 \le k \le m, \\ & x_i \in \mathbb{R}, && \forall 1 \le i \le n. \end{aligned}$$

$$\begin{aligned} \text{optimize} \quad & \sum_{1 \le i \le j \le n} a_{i,j} \cdot (v_i \cdot v_j) && (\boldsymbol{VP}) \\ \text{s.t.} \quad & \sum_{1 \le i \le j \le n} c_{i,j}^{(k)} \cdot (v_i \cdot v_j) \le b_k, && \forall 1 \le k \le m, \\ & v_i \in \mathbb{R}^n, && \forall 1 \le i \le n. \end{aligned}$$

- It may seem that the total number of variables increases (from n to n^2),
  they are actually used to model the n^2 quadratic monomials.

The Maximum Cut Problem

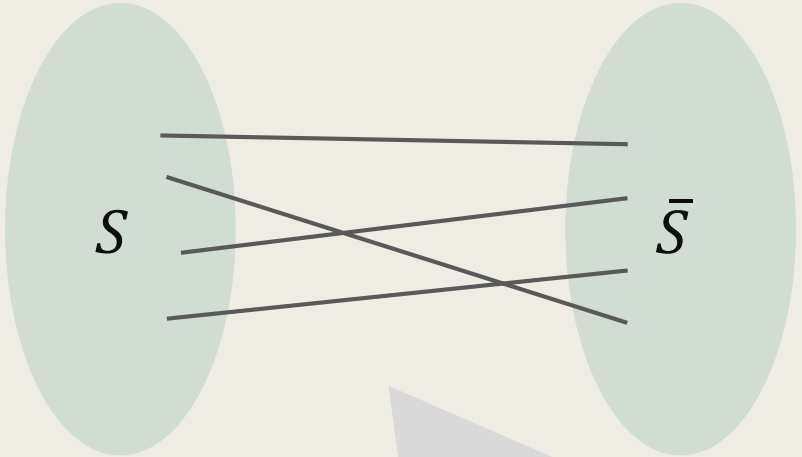
A bipartite classification problem.

Cut & Cut edges

- Let $G = (V, E)$ be an undirected graph with edge weight function $w : E \to Q^+$.
  - A ***cut*** $(S, \bar{S})$ is a partition of V.
  - The ***cut edges*** of a cut $(S, \bar{S})$ is the set of edges that have one endpoint in S and one endpoint in $\bar{S}$.
  - For the sake of convenience, the term “cut” can be used to indicate either the vertex partition or the edges crossing the cut.

The Maximum Cut Problem

- Given an undirected graph $G = (V, E)$ with edge weight function $w : E \to Q^+$, the maximum cut (max-cut) problem is to compute a cut $(S, \bar{S})$ with the maximum total weight.



Find a cut with the maximum the total weight.

Status of Max-Cut

- From HW#1, we know that $1/2$-approximation can be obtained by simple greedy algorithm.
- Natural LP for max-cut has an integrality gap of $1/2$.
- It is known that, $(16/17 \approx 0.941 + \epsilon)$-approx. is NP-hard.
  - Assuming the unique game conjecture (UGC), $(0.87856 + \epsilon)$-approx. is NP-hard.
- In this lecture,
  we will use SDP technique to obtain a 0.87856-approximation.

SQP-formulation for Max-Cut

- For each $v \in V$, let $y_v \in \{1, -1\}$ denote the partition decision for v.

  Then it follows that, $y_u \cdot y_v \in \{1, -1\}$ for all $u, v \in V$, and

  the edge (u, v) is a cut-edge if and only if $y_u \cdot y_v = -1$.

$$\begin{aligned} \max \quad & \frac{1}{2} \cdot \sum_{u,v \in V} w_{u,v} \cdot (1 - y_u \cdot y_v) && (\boldsymbol{SQP}\ *) \\ \text{s.t.} \quad & y_u^2 = 1, && \forall u \in V, \\ & y_u \in \mathbb{R}, && \forall u \in V. \end{aligned}$$

By setting $w_{u,v} = 0$ for all $(u, v) \notin E$,

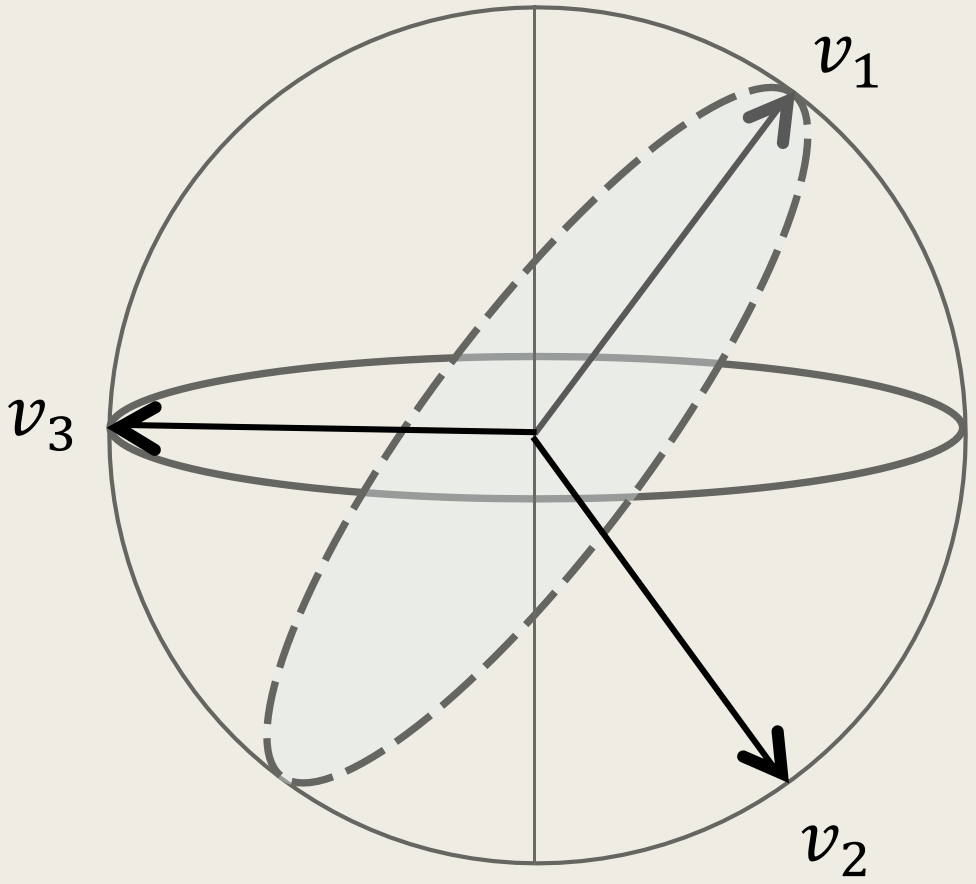
we may assume that G is a complete graph.

The VP (SDP)-relaxation for Max-Cut

$$\begin{aligned} \max \quad & \frac{1}{2} \cdot \sum_{i,j \in V} w_{i,j} \cdot (1 - v_i \cdot v_j) && (\boldsymbol{VP}*) \\ \text{s.t.} \quad & v_i \cdot v_i = 1, && \forall i \in V, \\ & v_i \in \mathbb{R}^{|V|}, && \forall i \in V. \end{aligned}$$

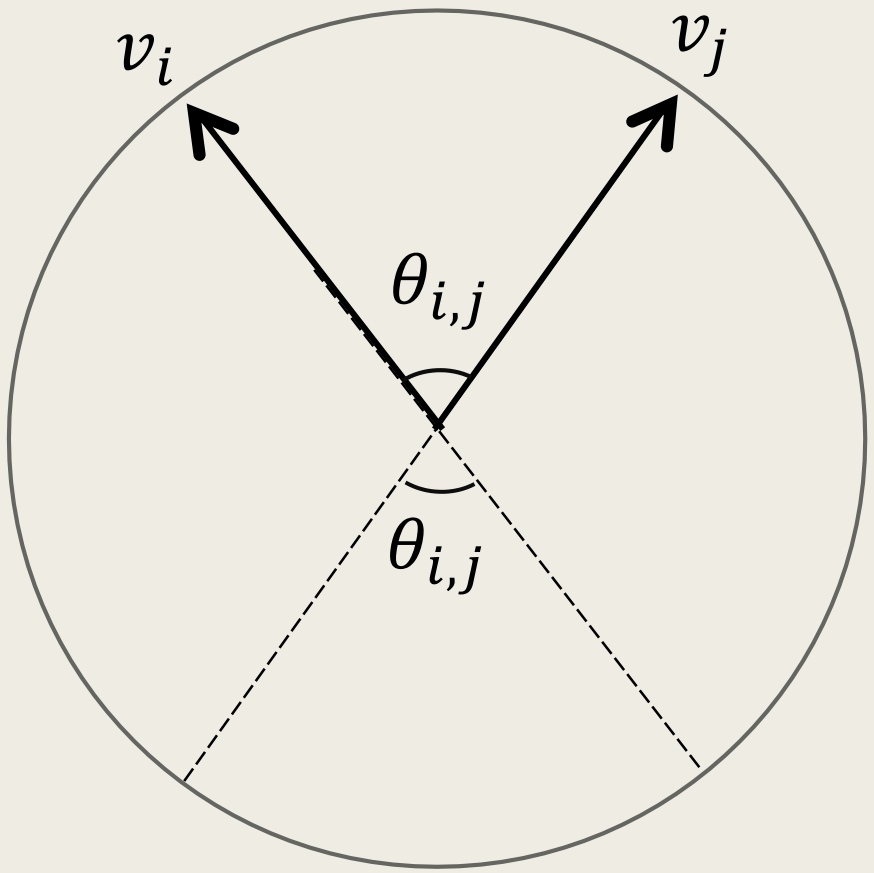
- From the constraint that $v_i \cdot v_i = 1$, we know that $\|v_i\| = 1$.
  - v_i lies on the $|V|$-dimensional unit sphere.

An interpretation of the SDP-relaxation



- From the constraint that $v_i \cdot v_i = 1$, we know that $\|v_i\| = 1$.
  - v_i lies on the $|V|$-dimensional unit sphere.
- Intuitively, the directions to which the vectors point indicate how they should be clustered.

An interpretation of the SDP-relaxation



- Consider the contribution of any v_i, v_j in the objective function.
  - We have

    $$1 - v_i \cdot v_j = 1 - \cos\theta_{i,j}.$$

- Intuitively, **the larger $\theta_{i,j}$ is**, ***the more likely*** i and j should be separated in the final cut.

A simple randomized rounding

Use a random hyperplane to classify the vectors.

A simple randomized rounding for SDP-(*)

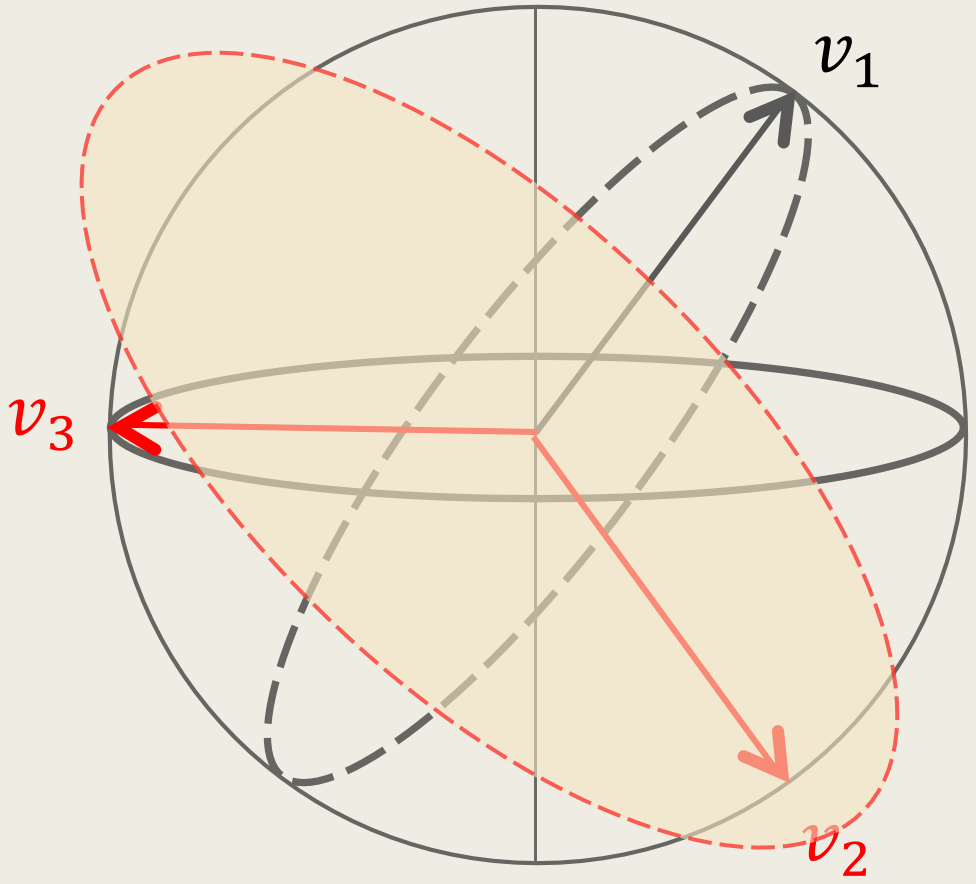
1. Solve (approximately) VP-(*) for an optimal $\{v_i^*\}_{i \in V}$.
2. Pick a vector r on the $|V|$-dimensional unit sphere uniformly at random. Let

$$S \coloneqq \{\, i \in V \;:\; v_i^* \cdot r \geq 0. \,\}.$$

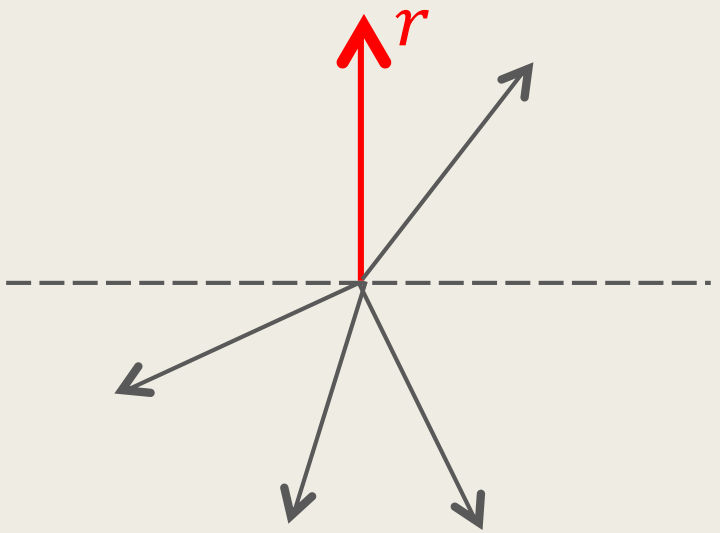
3. Output $(S, \bar{S})$ as the approximate cut.

$$\begin{aligned} \max \quad & \frac{1}{2} \cdot \sum_{i,j \in V} w_{i,j} \cdot (1 - v_i \cdot v_j) && (\boldsymbol{VP}\ *) \\ \text{s.t.} \quad & v_i \cdot v_i = 1, && \forall i \in V, \\ & v_i \in \mathbb{R}^{|V|}, && \forall i \in V. \end{aligned}$$

A simple randomized rounding for SDP-(*)



- Intuitively, we pick a random hyperplane, defined by r, to classify the vectors.



Uniform distribution on the sphere.

- Let $X = (X_1, X_2, \dots, X_n)$, where
  - $X_i \sim N(0,1)$ are i.i.d. random variables and
  - $N(0,1)$ is the normal distribution with mean 0 and standard deviation 1.

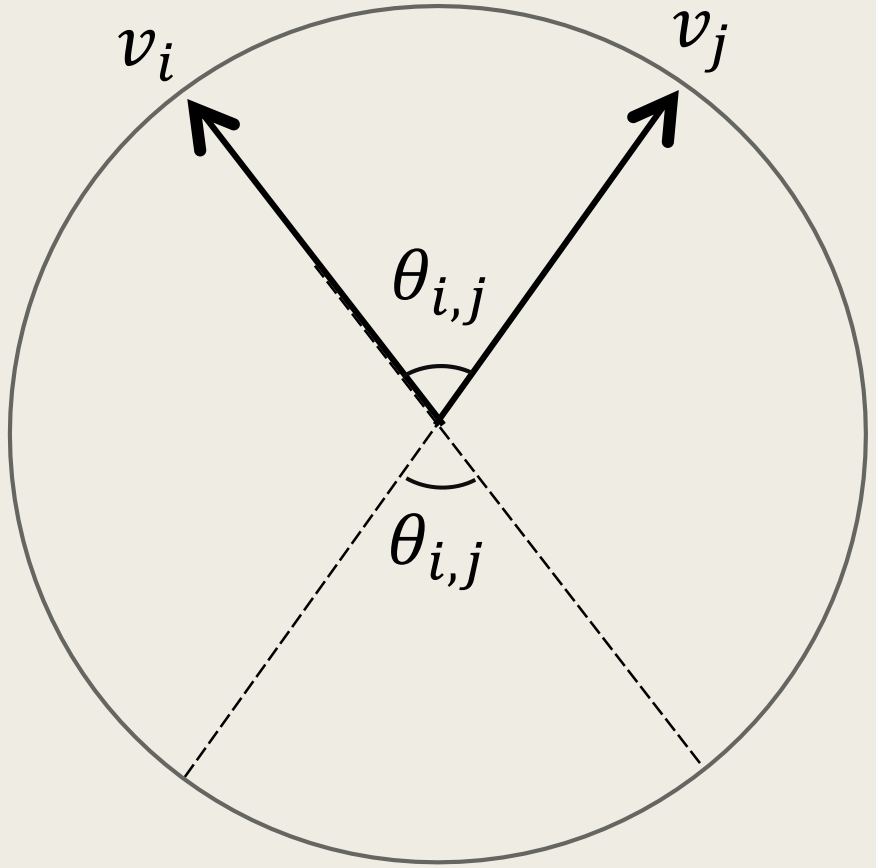
- Then $X/\|X\|$ is a uniform distribution on the unit sphere,
  since $\|X\|$ has pdf

$$f(x_1, x_2, \dots, x_n) = \prod_{1 \le i \le n} \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{1}{2} \cdot x_i^2} = (2\pi)^{-\frac{n}{2}} \cdot e^{-\frac{1}{2} \cdot \sum_{1 \le i \le n} x_i^2}$$

  which depends only on $\sum_{1 \le i \le n} x_i^2 = \|X\|$.

Analysis

- The following lemma follows from the fact that $\boldsymbol{r}$ is sampled from a uniform distribution on the sphere.
  - The probability that the hyperplane falls between v_i and v_j is exactly $\theta_{i,j}/\pi$.



Lemma 1.

For any $i, j \in V$,

$$\Pr\left[\, v_i \text{ and } v_j \text{ are separated by } r \,\right] = \frac{\theta_{i,j}}{\pi} \ .$$

- We have

$$OPT_f = \frac{1}{2} \cdot \sum_{i,j \in V} w_{i,j} \cdot \left(1 - v_i^* \cdot v_j^*\right) = \frac{1}{2} \cdot \sum_{i,j \in V} w_{i,j} \cdot \left(1 - \cos\theta_{i,j}\right)$$

and

$$E\left[\sum_{\substack{i \in S,\\ j \in \bar{S}}} w_{i,j}\right] = \sum_{i,j \in V} w_{i,j} \cdot \frac{\theta_{i,j}}{\pi} \geq \sum_{i,j \in V} w_{i,j} \cdot \alpha \cdot \frac{1 - \cos\theta_{i,j}}{2} = \alpha \cdot OPT_f ,$$

By Lemma 1.

Just compare the coefficients with OPT_f, α is the smallest ratio.

where

$$\alpha := \frac{2}{\pi} \cdot \min_{0 \leq \theta \leq \pi} \frac{\theta}{1 - \cos\theta} .$$

Analysis

- The following lemma can be verify by elementary calculus.

Lemma 2.

$$\alpha := \frac{2}{\pi} \cdot \min_{0 \le \theta \le \pi} \frac{\theta}{1 - \cos\theta} > 0.87856 .$$

Positive Semidefinite (PSD) Matrices

Positive Semidefinite (PSD) Matrices

- Let $A \in \mathbb{R}^{n \times n}$ be a real symmetric matrix. Then,
  - All the eigenvalues of A are real, and
  - The eigenvectors of A span the entire $\mathbb{R}^n$.

- A real symmetric matrix $A \in \mathbb{R}^{n \times n}$ is said to be positive semidefinite (PSD), denoted $A \succcurlyeq 0$, if all of its eigenvalues are non-negative.
  - Intuitively, it means that,
    A as a linear transformation *never reverses the direction of a vector*,
    and *the orientation* of the vectors *is kept*.

Positive Semidefinite (PSD) Matrices

Theorem 3.

Let $A \in \mathbb{R}^{n \times n}$ be a real symmetric matrix. The followings are equivalent.

1. For any $v \in \mathbb{R}^n$, $v^T A v \geq 0$.
2. All eigenvalues of A are nonnegative.
3. There is a matrix $W \in \mathbb{R}^{n \times n}$ such that $A = W^T W$.

- We will see later that, condition 3 establishes the equivalence between vector programs (VPs) and semidefinite programs (SDPs).

Theorem 3.

Let $A \in \mathbb{R}^{n\times n}$ be a real symmetric matrix. The followings are equivalent.

1. For any $v \in \mathbb{R}^n$, $v^T A v \geq 0$.
2. All eigenvalues of A are nonnegative.
3. There is a matrix $W \in \mathbb{R}^{n\times n}$ such that $A = W^T W$.

- $(3 \Rightarrow 1)$: For any $v \in \mathbb{R}^n$, we have

$$v^T A v = v^T \left(W^T W\right) v = (Wv)^T (Wv) \geq 0.$$

- $(1 \Rightarrow 2)$: Let (λ, v) be an eigen-pair of A. Then, by assumption,

$$\lambda \cdot v^T v = v^T (Av) = v^T A v \geq 0.$$

Since $v^T v \geq 0$ for any $v \in \mathbb{R}^n$, it follows that $\lambda \geq 0$.

2. All eigenvalues of A are nonnegative.
3. There is a matrix $W \in \mathbb{R}^{n \times n}$ such that $A = W^T W$.

$(2 \Rightarrow 3)$: Let $\lambda_1, \lambda_2, \ldots, \lambda_n$ be eigenvalues of A and $v_1, v_2, \ldots, v_n$ be the corresponding eigenvectors that form an orthonormal basis of $\mathbb{R}^n$.

- Let $U = [\, v_1 \;\; v_2 \;\; \cdots \;\; v_n \,]$ be the matrix with column vectors $v_1, v_2, \ldots, v_n$ and $D = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ be the diagonal matrix consisting of $\lambda_1, \lambda_2, \ldots, \lambda_n$.

  Then, $AU = UD$, $UU^T = I$ and hence $U^T = U^{-1}$.

  Let $Q = \mathrm{diag}\left(\sqrt{\lambda_1}, \sqrt{\lambda_2}, \ldots, \sqrt{\lambda_n}\right)$. Since $\lambda_i \geq 0$ for all i, Q is real. We get

$$A = UDU^{-1} = U \cdot \left(QQ^T\right) \cdot U^T = (UQ) \cdot (UQ)^T.$$

Some Remarks.

- Positive semidefiniteness of a matrix A can be verified in polynomial time.
  - By applying the Cholesky decomposition to obtain $A = UDU^T$ where D is a diagonal matrix consisting of eigenvalues of A.
- The decomposition $A = W^T W$ involves square-root computation, which can be approximated to any desirable degree.
  Hence, we can assume that it can be obtained in polynomial time.
  (and let the error be absorbed in the error for solving SDPs.)

Semidefinite Programming (SDP)

Some definition

- Let $A, B \in \mathbb{R}^{n \times n}$ be a real matrix.

  The Frobenius inner product, or, component-wise inner product, of A and B is defined as

$$A \bullet B \coloneqq tr\left(A^T B\right) = \sum_{1 \le i,j \le n} a_{i,j} \cdot b_{i,j} \ .$$

- A more intuitive way to understand it is that,….
  - It's just the component-wise product of A and B.

Use entities in A as the coefficients of the entities in B.

Semidefinite Programming (SDP)

- Let $Y \in \mathbb{R}^{n \times n}$ be a real matrix.

  A ***semidefinite program (SDP)*** is a *linear program with variables* $y_{i,j}$ and the additional constraints that *all coefficient matrices* are ***symmetric*** and Y are both ***symmetric*** and ***positive semidefinite***.

- Let $M_n \subset \mathbb{R}^{n \times n}$ be the set of real symmetric matrices, $C, D_1, D_2, \ldots, D_k \in M_n$ and $d_1, d_2, \ldots, d_k \in \mathbb{R}$. SDP has the following form.

$$
\begin{array}{lll}
\text{optimize} & C \bullet Y & (\boldsymbol{SDP}) \\
\text{s.t.} & D_i \bullet Y = d_i, & \forall 1 \le i \le k, \\
 & Y \succcurlyeq 0, & \\
 & Y \in M^n. &
\end{array}
$$

LP with symmetric variables $y_{i,j}$ and symmetric coefficient matrices C, D_i.

Y is *PSD*.

Semidefinite Programming (SDP)

- Note that,

$$\begin{aligned} \text{optimize} \quad & C \bullet Y && (\boldsymbol{SDP}) \\ \text{s.t.} \quad & D_i \bullet Y = d_i, && \forall 1 \le i \le k, \\ & Y \succcurlyeq 0, \\ & Y \in M^n. \end{aligned}$$

is just the concise format of

$$\begin{aligned} \text{optimize} \quad & \sum_{1 \le i,j \le n} c_{i,j} \cdot y_{i,j} && (\boldsymbol{SDP}) \\ \text{s.t.} \quad & \sum_{1 \le i,j \le n} d_{i,j}^{\ell} \cdot y_{i,j} = d_{\ell}, && \forall 1 \le \ell \le k, \\ & Y = \{y_{i,j}\}_{1 \le i,j \le n} \succcurlyeq 0, && Y \in M^n. \end{aligned}$$

Equivalence between SDP and VP

- Consider the following two programs.

$$\begin{aligned} \text{optimize} \quad & C \bullet Y && (\boldsymbol{SDP}) \\ \text{s.t.} \quad & D_i \bullet Y = d_i, && \forall 1 \le i \le k, \\ & Y \succcurlyeq 0, \\ & Y \in M^n. \end{aligned}$$

$$\begin{aligned} \text{optimize} \quad & \sum_{1 \le i \le j \le n} c_{i,j} \cdot (v_i \cdot v_j) && (\boldsymbol{VP}) \\ \text{s.t.} \quad & \sum_{1 \le i \le j \le n} d_{i,j}^{(k)} \cdot (v_i \cdot v_j) = d_k, && \forall 1 \le k \le m, \\ & v_i \in \mathbb{R}^n, && \forall 1 \le i \le n. \end{aligned}$$

- Let Y^* be a feasible solution for (SDP) and

  W be the matrix given by Theorem 3 with $Y = W^T W$.

  Then, the column vectors of W is feasible for (VP) with the same value.

Equivalence between SDP and VP

- Consider the following two programs.

$$\begin{aligned} \text{optimize} \quad & C \bullet Y && (\boldsymbol{SDP}) \\ \text{s.t.} \quad & D_i \bullet Y = d_i, && \forall 1 \le i \le k, \\ & Y \succcurlyeq 0, \\ & Y \in M^n. \end{aligned}$$

$$\begin{aligned} \text{optimize} \quad & \sum_{1 \le i \le j \le n} c_{i,j} \cdot (v_i \cdot v_j) && (\boldsymbol{VP}) \\ \text{s.t.} \quad & \sum_{1 \le i \le j \le n} d_{i,j}^{(k)} \cdot (v_i \cdot v_j) = d_k, && \forall 1 \le k \le m, \\ & v_i \in \mathbb{R}^n, && \forall 1 \le i \le n. \end{aligned}$$

- Conversely, let $\{v_i^*\}_{1 \le i \le n}$ be a feasible vectors for (VP) and $W = [\, v_1^* \, v_2^* \cdots v_n^* \,]$ be the matrix consisting of $\{v_i^*\}_{1 \le i \le n}$ as column vectors.

  Then, the matrix $Y^* \coloneqq W^T W$ is PSD by Theorem 3 and also feasible for (SDP) with the same value.

SDPs are Polynomial-Time Solvable

- The separation problem of SDPs can be answered in polynomial time.

Theorem 4.

For any $A \in \mathbb{R}^{n \times n}$, we can determine in polynomial-time whether or not A is feasible for (SDP) and, if not, output a separating hyperplane.

- Hence, for any $\epsilon > 0$, SDPs can be solved approximately within an additive error of ϵ in time polynomial in n and $\log 1/\epsilon$ by the Ellipsoid method.

- The separation problem of SDPs can be answered in polynomial time.

Theorem 4.

For any $A \in \mathbb{R}^{n \times n}$, we can determine in polynomial-time whether or not A is feasible for (SDP) and, if not, output a separating hyperplane.

- If A is not symmetric, then $a_{i,j} > a_{j,i}$ for some i, j.
  Then $y_{i,j} \leq y_{j,i}$ is a separating hyperplane.
- If A is not PSD, then it has an eigen-pair (λ, v) with $\lambda < 0$.
  Then $v^T Y v = \sum_{1 \leq i,j \leq n} (v_i v_j) \cdot y_{i,j} = (vv^T) \cdot Y \geq 0$ is a separating hyperplane.
- If any of the linear constraints is violated,
  then it directly yields a separating hyperplane.

An SDP Formulation for Max-Cut

- As an example, the following is the SDP formulation for Max-Cut.

$$\begin{aligned} \max \quad & \frac{1}{2} \cdot \sum_{i,j \in V} w_{i,j} \cdot (1 - y_{i,j}) && (\boldsymbol{SDP}*) \\ \text{s.t.} \quad & y_{i,i} = 1, && \forall i \in V, \\ & Y = \{y_{i,j}\}_{1 \le i,j \le n} \succcurlyeq 0, \\ & Y \in M^n. \end{aligned}$$