

# ISEN 629: Engineering Optimization

## Lecture 27

Sergiy Butenko

Industrial and Systems Engineering  
Texas A&M University

Fall 2007

1/19

## Convex relaxations in discrete optimization

In discrete optimization we are dealing with (finite) discrete feasible sets.

Many of practically important discrete optimization problems are NP-hard, meaning that no polynomial-time algorithm have been developed for these problems.

Traditional exact algorithms for such problems are based on branch-and-bound procedures, which, at each step, fix values of some variables and generate lower and upper bounds on the objective function value for the resulting subproblem.

Lower bounds (for minimization problem) are typically obtained by solving a convex relaxation of the subproblem.

Good quality lower bounds are also crucial in estimating quality of the solutions obtained using heuristics or approximation algorithms.

2/19

## MAX-CUT Problem

Next we discuss an example of an approximation algorithm for the classical MAX-CUT problem based on a semidefinite programming relaxation.

Given a complete undirected graph

$$G = (V, E), V = \{1, 2, \dots, n\}, E = V \times V$$

with weights  $w_{ij}, i, j = 1, \dots, n$  on the edges, find a vertex partition

$$S, \bar{S} \subseteq V, S \cup \bar{S} = V, S \cap \bar{S} = \emptyset$$

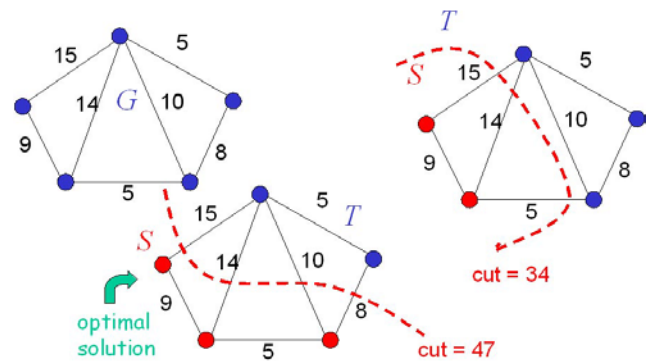
such that the sum of the weights in the cut  $(S, \bar{S})$ ,

$$\sum_{i \in S, j \in \bar{S}} w_{ij}$$

is maximized.

3/19

## MAX-CUT Problem: An Example



4/19

## MAX-CUT Problem

We can formulate the MAX-CUT problem as the following **strict binary quadratic program** (*strict* since no monomials of degree 1 are involved):

$$\begin{aligned} \max \quad & \frac{1}{2} \sum_{1 \leq i < j \leq n} w_{ij}(1 - x_i x_j) \\ \text{s.t.} \quad & x_i^2 = 1, \quad i \in V. \end{aligned}$$

We will first relax this program to a vector program and then will show that our relaxation is equivalent to a certain semidefinite program.

5/19

## Vector programs

A **vector program** is the problem of optimizing a linear function of the inner products  $v_i^T v_j$ ,  $1 \leq i < j \leq n$ , where  $v_1, \dots, v_n$  are vector variables in  $\mathbb{R}^n$ , subject to linear constraints of these inner products.

We can obtain a vector program corresponding to a strict binary quadratic program as follows:

- ▶ Define  $n$  vector variables corresponding to the  $n$  binary variables in the quadratic program (e.g.,  $x_i \rightarrow v_i$ );
- ▶ Replace each degree 2 term with the corresponding inner product (i.e.,  $x_i x_j \rightarrow v_i^T v_j$ ).

6/19

## Vector program for MAX-CUT

For the MAX-CUT formulation,

$$\begin{aligned} \max \quad & \frac{1}{2} \sum_{1 \leq i < j \leq n} w_{ij}(1 - x_i x_j) \\ \text{s.t.} \quad & x_i^2 = 1, \quad i \in V, \end{aligned}$$

we obtain the following vector program:

$$\begin{aligned} \max \quad & \frac{1}{2} \sum_{1 \leq i < j \leq n} w_{ij}(1 - v_i^T v_j) \\ \text{s.t.} \quad & v_i^T v_i = 1, \quad i \in V. \end{aligned}$$

Note that for any feasible solution of the original formulation, the vector  $v_i = (x_i, 0, \dots, 0)^T \in \mathbb{R}^n$  has the same objective function value in the vector program. Therefore, the vector program is a relaxation of the binary quadratic program.

7/19

## Semidefinite optimization

Recall that a semidefinite program is formulated as follows:

$$\begin{aligned} \min \quad & \langle C, X \rangle_F, \\ \text{s.t.} \quad & \langle A_i, X \rangle_F = b_i, \quad i = 1, \dots, m, \\ & X \in \mathcal{P}_n, \end{aligned} \tag{1}$$

where  $C$  and  $A_i$  are some matrices from  $S^{n \times n}$  and the Frobenius inner product on  $S^{n \times n}$  is defined by

$$\langle X, Y \rangle_F = \sum_{i=1}^n \sum_{j=1}^n X^{(i,j)} Y^{(i,j)} = \text{trace}(XY),$$

for any  $X, Y \in S^{n \times n}$ . Here  $S^{n \times n}$  is the linear space of symmetric  $n \times n$ -matrices and  $\mathcal{P}_n \subset S^{n \times n}$  is the cone of positive semidefinite  $n \times n$ -matrices.

8/19

## SDP relaxation for MAX-CUT

Next we show that vector programs are equivalent to SDP.

Let  $(V)$  be a vector program on  $n$   $n$ -dimensional vector variables  $v_1, \dots, v_n$ .

We define the corresponding SDP  $(S)$  as follows:

- ▶ replace each inner product  $v_i^T v_j$  in  $(V)$  with the variable  $y_{ij}$
- ▶ Require that the matrix  $Y = (y_{ij})_{i,j=1}^n$  is symmetric and positive semidefinite.

### Lemma

The vector program  $(V)$  and the SDP  $(S)$  are equivalent.

### Proof.

Correspondence between solutions of  $(V)$  and  $(S)$ :

- ▶  $w_1, \dots, w_n$  - a feasible solution of  $(V)$ ;
- ▶  $W$  - the matrix with columns  $w_1, \dots, w_n$ ;
- ▶  $Y = W^T W$  is a feasible solution of  $(S)$ .

□

9/19

## SDP relaxation for MAX-CUT

For the vector program relaxing the MAX-CUT formulation,

$$\begin{aligned} \max \quad & \frac{1}{2} \sum_{1 \leq i < j \leq n} w_{ij} (1 - v_i^T v_j) \\ \text{s.t.} \quad & v_i^T v_i = 1, \quad i \in V. \end{aligned}$$

we obtain the following equivalent SDP:

$$\begin{aligned} \max \quad & \frac{1}{2} \sum_{1 \leq i < j \leq n} w_{ij} (1 - y_{ij}) \\ \text{s.t.} \quad & y_{ii} = 1, \quad i \in V, \\ & Y \in \mathcal{P}_n. \end{aligned}$$

10/19

## Randomized rounding algorithm

Assume that we have an optimal solution  $v_1^*, \dots, v_n^*$  of our vector program for MAX-CUT with  $OPT_v$  being the corresponding objective function value.

Note that vectors  $v_1^*, \dots, v_n^*$  all lie on the  $n$ -dimensional unit sphere.

Our goal is to obtain a cut  $(S, \bar{S})$  whose weight is a large fraction of  $OPT_v$ .

Let  $\theta_{ij}$  denote the angle between  $v_i^*$  and  $v_j^*$ . Since  $v_i^{*T} v_j^* = \cos \theta_{ij}$ , the contribution of the pair  $v_i^*, v_j^*$  to  $OPT_v$  is given by

$$\frac{w_{ij}}{2} (1 - \cos \theta_{ij}).$$

We want to separate  $v_i^*$  and  $v_j^*$  if  $\theta_{ij}$  is large.

11/19

## Randomized rounding algorithm

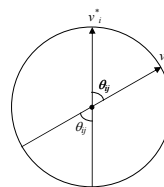
**Randomized rounding:** Let  $r$  be a uniformly distributed vector on the  $n$ -dimensional unit sphere. Set

$$S = \{i : v_i^{*T} r \geq 0\}.$$

### Lemma

$$\text{Prob}(i \text{ and } j \text{ are separated}) = \frac{\theta_{ij}}{\pi}.$$

### Proof.



Project  $r$  onto the plane defined by  $v_i^*$  and  $v_j^*$ . The vertices  $i$  and  $j$  will be separated if and only if the projection lies on one of the two arcs of angle  $\theta_{ij}$ . Since  $r$  has been picked from the uniform distribution over the sphere, its projection is a random direction on the plane.

□

12/19

## Randomized rounding algorithm

### Lemma

Let  $x_1, \dots, x_n$  be independent random variables with the standard normal distribution  $N(0, 1)$ . Then

$$\frac{1}{\sqrt{x_1^2 + \dots + x_n^2}}(x_1, \dots, x_n)^T$$

is a random vector on the unit sphere in  $\mathbb{R}^n$ .

### Proof.

Let  $r = (x_1, \dots, x_n)^T$ . Then the distribution function for  $r$  has density

$$f(y_1, \dots, y_n) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi}} e^{-y_i^2/2} = \frac{1}{(2\pi)^{n/2}} e^{-\frac{1}{2} \sum_i y_i^2}.$$

Since it depends only on the distance of  $(y_1, \dots, y_n)$  from the origin, the distribution of  $r$  is spherically symmetric. □ 13/19

## Randomized rounding algorithm

### Algorithm:

1. Find an optimal solution  $v_1^*, \dots, v_n^*$  of the vector program for MAX-CUT.
2. Pick a vector  $r$  uniformly distributed over the  $n$ -dimensional unit sphere.
3. Set  $S = \{i : v_i^{*T} r \geq 0\}$ .

Denote by  $W_A$  the random variable representing the weight of edges in the cut obtained from the algorithm.

We want to find  $\alpha$  for which

$$E[W_A] \geq \alpha \cdot OPT_v.$$

14/19

## Randomized rounding algorithm

Note that

$$\begin{aligned} E[W_A] &= \sum_{1 \leq i < j \leq n} w_{ij} \text{Prob}(i \text{ and } j \text{ are separated}) \\ &= \sum_{1 \leq i < j \leq n} w_{ij} \frac{\theta_{ij}}{\pi} \end{aligned}$$

and

$$OPT_v = \sum_{1 \leq i < j \leq n} \frac{w_{ij}}{2} (1 - \cos \theta_{ij}).$$

Hence, if we select  $\alpha$  so that

$$\frac{\theta_{ij}}{\pi} \geq \alpha \frac{1}{2} (1 - \cos \theta_{ij})$$

for all  $i, j \in V$  then  $E[W_A] \geq \alpha \cdot OPT_v$ . We can take

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

15/19

## Randomized rounding algorithm

Thus, we have shown that the following statement is correct:

### Lemma

$$E[W_A] \geq \alpha \cdot OPT_v,$$

where

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

It can be shown that  $\alpha > 0.87856$ .

### Theorem

There is a randomized approximation algorithm for MAX-CUT with an approximation ratio of 0.87856.

16/19

## Randomized rounding algorithm

### Proof.

We will need some background information concerning the randomized algorithms.

In Monte-Carlo algorithms, the answer has to be correct with “high probability” in a predefined run time.

For a problem of size  $n$ , we say that  $p = 1 - \frac{1}{n^\gamma}$ , where  $\gamma > 1$ , is a high probability.

We say that a randomized algorithm takes  $\tilde{O}(f(n))$  of a resource (such as space or time), if there exist some  $C, n_0 > 0$  such that for any  $n \geq n_0$  the amount of resource used is  $\leq C\gamma f(n)$  with probability  $\geq 1 - \frac{1}{n^\gamma}, \gamma > 1$ .

We need to obtain a “high probability” result using the bound  $E[W_A] \geq \alpha \cdot OPT_v$ .

17/19

## Randomized rounding algorithm

Denote by  $W$  the sum of all edge weights in the graph, and let  $a, p$  be such that

$$E[W_A] = aW; \quad p = \text{Prob}(W_A < (1 - \epsilon)aW),$$

where  $\epsilon$  is a constant.  $W_A$  cannot be greater than  $W$ , hence

$$aW \leq p(1 - \epsilon)aW + (1 - p)W \Rightarrow p \leq \frac{1 - a}{1 - a + a\epsilon}.$$

Since  $OPT \geq W/2$  (exercise), we have

$$W \geq E[W_A] = aW \geq \alpha \cdot OPT_v \geq \alpha \cdot OPT \geq \frac{\alpha W}{2},$$

and thus  $\alpha/2 \leq a \leq 1$ . Therefore, we get

$$p \leq \frac{1 - a}{1 - a + a\epsilon} \leq 1 - \frac{\epsilon\alpha/2}{1 + \epsilon\alpha/2 - \alpha/2} = 1 - c,$$

where  $c = \frac{\epsilon\alpha/2}{1 + \epsilon\alpha/2 - \alpha/2}$ .

18/19

## Randomized rounding algorithm

Thus, for  $p = \text{Prob}(W_A < (1 - \epsilon)aW)$  we have  $p \leq 1 - c$ , so after  $k$  iterations we obtain

$$\text{Prob}(W_A \geq (1 - \epsilon)aW) \geq 1 - (1 - c)^k.$$

To achieve high probability, we need to have

$$1 - (1 - c)^k \geq 1 - n^{-\gamma} \Leftrightarrow k \geq -\frac{\gamma}{\ln(1 - c)} \ln n.$$

Since  $aW \geq \alpha \cdot OPT \geq 0.87856OPT$ , we can pick a value of  $\epsilon > 0$  such that  $(1 - \epsilon)aW \geq 0.87856OPT$ . Therefore, denoting by  $C = -\frac{1}{\ln(1 - c)}$ , after at most  $C\gamma \ln n$  iteration we have

$$\text{Prob}(W_A \geq 0.87856OPT) \geq 1 - n^{-\gamma}.$$

Thus, the algorithm approximates the MAX-CUT problem with the approximation ratio of 0.87856 in  $\tilde{O}(\ln n)$  time.  $\square$

19/19