

3.2 Reformulation Linearization Technique (RLT)

Consider the most simple quadratic program with binary variables.

$$\begin{cases} \text{minimize} & \mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{q}^T \mathbf{x} \\ \text{subject to} & \mathbf{x} \in \{0, 1\}^n, \end{cases} \quad (2)$$

where $\mathbf{Q} \in \mathbb{R}^{n \times n}$ and $\mathbf{q} \in \mathbb{R}^n$.

Of course, we can perform an obvious linear program relaxation replacing the binary constraint by

$$\begin{aligned} 1 - x_i &\geq 0 & i = 1, 2, \dots, n \\ x_i &\geq 0 & i = 1, 2, \dots, n. \end{aligned}$$

The Reformulation Linearization Technique (RLT) proposed by Sherali and Adams (around 1990's) is based on the following fact. Construct redundant quadratic constraints and perform a linearization. In the above case, we can construct three types of quadratic constraints:

$$\begin{aligned} x_i x_j &\geq 0 & i, j = 1, 2, \dots, n \\ (1 - x_i)(1 - x_j) &\geq 0 & i, j = 1, 2, \dots, n \\ (1 - x_i)x_j &\geq 0 & i, j = 1, 2, \dots, n \end{aligned}$$

Replacing all $x_i x_j$ by w_{ij} , we will have the following linear program relaxation.

$$\begin{cases} \text{minimize} & \sum_{i,j=1}^n Q_{ij} w_{ij} + \sum_{i=1}^n q_i x_i \\ \text{subject to} & w_{ij} \geq 0 & 1 \leq i, j \leq n \\ & 1 - x_i - x_j + w_{ij} \geq 0 & 1 \leq i, j \leq n \\ & x_j - w_{ij} \geq 0 & 1 \leq i, j \leq n. \end{cases} \quad (3)$$

Of course the optimal value of (3) is smaller than or equal to the one of (2).

We can apply the RLT for a more general quadratic constrained quadratic program.

$$\begin{cases} \text{minimize} & \mathbf{x}^T \mathbf{Q}_0 \mathbf{x} + \mathbf{q}_0^T \mathbf{x} \\ \text{subject to} & \mathbf{x}^T \mathbf{Q}_i \mathbf{x} + \mathbf{q}_i^T \mathbf{x} + \gamma_i \leq 0 & 1 \leq i \leq r \\ & \mathbf{a}_j^T \mathbf{x} + b_j \leq 0 & 1 \leq j \leq s. \end{cases} \quad (4)$$

Then we can take the products of the linear constraints and perform a similar linearization.

$$\begin{aligned} 1 &\leq \forall j \leq \forall k \leq s \\ &-(\mathbf{a}_j^T \mathbf{x} + b_j)(\mathbf{a}_k^T \mathbf{x} + b_k) \\ &= -\mathbf{x}^T \mathbf{a}_j \mathbf{a}_k^T \mathbf{x} - b_k \mathbf{a}_j^T \mathbf{x} - b_j \mathbf{a}_k^T \mathbf{x} - b_j b_k \leq 0. \end{aligned}$$

At the end, we can obtain the following relaxation for (4):

$$\begin{cases} \text{minimize} & \sum_{p,\ell=1}^n [\mathbf{Q}_0]_{p\ell} w_{p\ell} + \mathbf{q}_0^T \mathbf{x} \\ \text{subject to} & \sum_{p,\ell=1}^n [\mathbf{Q}_i]_{p\ell} w_{p\ell} + \mathbf{q}_i^T \mathbf{x} + \gamma_i \leq 0 & 1 \leq i \leq r \\ & -\sum_{p,\ell=1}^n [\mathbf{a}_j \mathbf{a}_k^T]_{p\ell} w_{p\ell} - b_k \mathbf{a}_j^T \mathbf{x} - b_j \mathbf{a}_k^T \mathbf{x} - b_j b_k \leq 0 & 1 \leq j, k \leq s \\ & \mathbf{a}_j^T \mathbf{x} + b_j \leq 0 & 1 \leq j \leq s. \end{cases} \quad (5)$$

In some cases, (5) can be unbounded even when (4) has a bounded optimal value. In this case, it is necessary to add some redundant constraints for (4) in (5).

3.3 Exercises

1. Show that the constraints $0 \leq x_i \leq 1$ ($i = 1, 2, \dots, n$) are redundant for (3) and therefore, unnecessary.
2. Give an example of a polyhedron defined by $\{\mathbf{y} \in \mathbb{R}^m \mid \mathbf{A}^T \mathbf{y} \leq \mathbf{c}\}$ where \mathbf{A} and \mathbf{c} have elements equal to $0, \pm 1$, but its vertices do not have integer values (coordinates).

4 Semidefinite Program Relaxation

4.1 Semidefinite Program (SDP)

If we take $\mathcal{K} = \mathbb{S}_+^n$ in (CLP), where \mathbb{S}_+^n stands for the closed convex cone of real $n \times n$ symmetric positive semidefinite matrices, we have a Semidefinite Program (SDP).

$$(\text{SDP}) \begin{cases} \text{minimize} & \langle \mathbf{C}, \mathbf{X} \rangle \\ \text{subject to} & \langle \mathbf{A}_i, \mathbf{X} \rangle = b_i \quad (i = 1, 2, \dots, m) \\ & \mathbf{X} \in \mathbb{S}_+^n, \end{cases}$$

and its associated dual problem:

$$(\text{DSDP}) \begin{cases} \text{maximize} & \langle \mathbf{b}, \mathbf{y} \rangle \\ \text{subject to} & \sum_{i=1}^m \mathbf{A}_i y_i + \mathbf{S} = \mathbf{C}, \\ & \mathbf{S} \in \mathbb{S}_+^n. \end{cases}$$

4.2 Maximum Cut Problem

Given a graph $G = (V, E)$ where $V := \{1, 2, \dots, n\}$ is the set of vertices and $E \subseteq V \times V$ is the set of edges, let us define *positive* weights for each existing edge $(u, v) \in E$ as $w : E \rightarrow \mathbb{R}_{++}$.

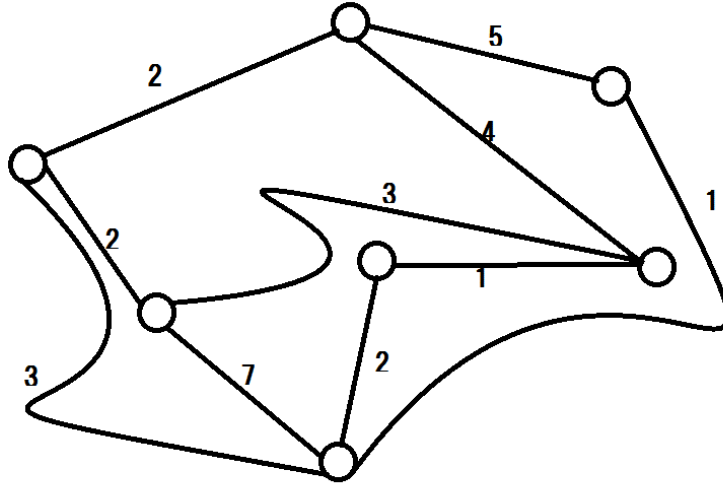


Figure 1: A undirected graph with positive weights.

Given a subset $S \subseteq V$, the subset of edges $\delta(S) := \{(u, v) \in E \mid u \in S, v \in V \setminus S\}$ is called a *cut* of the graph $G = (V, E)$. We can associate a *capacity* for the cut by summing all the edges' weights of the cut, $\delta_w(S) := \sum_{u \in S, v \in V \setminus S} w(u, v)$.

The problem to find a *minimum cut* (capacity) among all possible ones of a undirected graph with positive weights is an “easy” problem. This can be understood in the sense that we can use for instance the famous Ford and Fulkerson method to find the maximum flow of the graph which also gives the minimum cut of the graph from a specific source vertex to a sink vertex. The application of this method for all pair of vertices for instance will give the desired result. Of course, there are other complex algorithms which guarantee polynomial-time number of steps to determine the minimum cut of a undirected graph.

Now, let us turn on to the problem of finding the *maximum cut*. It is known that this problem is *NP-complete*, i.e., it is *Non-deterministic Polynomial time (NP)* and *Non-deterministic Polynomial time hard (NP-hard)*.

Suppose we have a cut $S \subseteq V := \{1, 2, \dots, n\}$, and we assign the value 1 for x_i if $i \in S$ and -1 for x_i if $i \in V \setminus S$. Also, let us assume that $w_{ij} := 0$ for $(i, j) \notin E$. Then, the capacity of a cut S will be

$$\frac{1}{4} \sum_{i,j=1}^n w_{ij}(1 - x_i x_j), \quad x_i = \begin{cases} 1, & i \in S \\ -1, & i \in V \setminus S \end{cases}$$

Therefore, we can formulate the max-cut problem as an optimization problem.

$$\begin{cases} \text{maximize} & \frac{1}{4} \sum_{i,j=1}^n w_{ij}(1 - x_i x_j) \\ \text{subject to} & x_i^2 = 1 \quad i = 1, 2, \dots, n. \end{cases} \quad (6)$$

For a *maximization* problem, we say that one algorithm gives an α -*approximation* if for an $\alpha \in (0, 1)$, we have

$$[\text{approximate value obtained by the algorithm}] \geq \alpha [\text{optimal value of the problem}]$$

for all instance of the problem class.

4.2.1 A Very Simple Randomized Algorithm

Let us consider the most simple randomized algorithm for the max-cut problem.

```

Set maxcut :=  $-\infty$ .
For  $k := 1$  to MAX
  Choose a cut  $S_k \subseteq V$  such that each vertex of  $V$  is chosen with probability  $\frac{1}{2}$ .
  Compute  $\delta_w(S_k)$ .
  If  $\delta_w(S_k) > \text{maxcut}$ ,
    then maxcut :=  $\delta_w(S_k)$ .

```

Since each possible edge in $G = (V, E)$ can be chosen with probability 0.5, we obtain:

$$E[\text{rand}] = \frac{1}{2} \sum_{1 \leq i \leq j \leq n} w_{ij} = \frac{1}{2} \sum_{(i,j) \in E} w_{ij} \geq \frac{1}{2} [\text{opt. max-cut}].$$

where “*rand*” is the optimal value obtained by the above algorithm and “*opt. max-cut*” is the actual optimal value of the maximum capacity of the cut in the given graph.

4.3 Semidefinite Program Relaxation of the Max-Cut Problem

Associating a new variable X_{ij} for $x_i x_j$ ($1 \leq i \leq j \leq n$), we can rewrite (6) as the following equivalent problem: