Part 11 Max Cut Problem

(**Definition 1**) (**Max Cut Problem**) Given an undirected graph G=(V, E) with positive edge weights. Find a **partition** (S, V-S) of the vertex set that maximize the total weight of the edges between S and V-S.

Notes: The **general Min Cut Problem**, where there's no specific source vertex s and sink vertex t, has the similar goal with the **Max Cut Problem**. In fact, the general Min Cut Problem can be solved in **polynomial time**, in which we can treat every pair of vertices as s and t to find the corresponding minimum cut. The number of vertex pairs is n^2.

Notes: However, the Max Cut Problem is **NP-Hard**.

(**Algorithm 1**) (**Approximation Algorithm** for **Max Cut Problem**) Randomly place each vertex in S or V-S with the probability 1/2.

(**Theorem 1**) The **approximation ratio** of **Algorithm 1** is 1/2.

Proof of **Theorem 1**. Let $OPT_{\text{MCP}}(I)$ be the optimal value of the maximum cut. Let S be the cut given by **Algorithm 1**. For an arbitrary graph, the value of maximum cut must be less than the sum of all the edge weights. Namely, we have

$$OPT_{\text{MCP}}(I) \le \sum_{(u,v)\in E} w_{uv}\cdot$$

For the **expected weight** of S, we have

$$\mathrm{E}[cost(S)] = \sum_{(u,v)\in E} \frac{1}{2} w_{uv} = \frac{1}{2}\sum_{(u,v)\in E} w_{uv} \ge \frac{1}{2} OPT_{\text{MCP}}(I)\cdot$$

Hence, the **approximation ratio** of **Algorithm 1** is

$$\frac{\mathrm{E}[cost(S)]}{OPT_{\text{MCP}}(I)} \ge \frac{OPT_{\text{MCP}}(I)/2}{OPT_{\text{MCP}}(I)} = \frac{1}{2}\cdot$$

Note: It seems that we can introduce another approximation algorithm with better approximation ratio by using the ILP, LP-Relaxation, and Duality of LP. However, the **integral gap** of the LP w.r.t. Max Cut Problem is 2.

(**Definition 2**) (**Semi-Definite Programming** for Max Cut Problem) Let $V = \{1, 2, \cdots, n\}$, where we associate each vertex with an index. Let w_{ij} denote the weight of edge (i, j). Associate each vertex i with a variable $x_i \in \{-1, 1\}$, in which

$$x_i = \begin{cases} 1, & i \in S \\ -1, & i \in (V - S) \end{cases}.$$

The **Max Cut Problem** can be formulated as the following **Quadratic Program**:

$$\max \sum_{(i,j)\in E} w_{ij} \cdot \frac{1 - x_i x_j}{2}.$$
$$\text{s.t. } x_i \in \{-1, 1\}, \forall i \in V$$

It can be further rewritten into the following form:

$$\max \frac{1}{2} \sum_{(i,j)\in E} w_{ij}(1 - x_i x_j).$$
$$\text{s.t. } x_i^2 = 1, \forall i \in V$$

Notes: When the two end points of an edge (i, j) are partitioned into different set (i.e., S and V-S), we have (1) x_i=1 and x_j=-1 or (2) x_i=-1 and x_j=1, which is equivalent to $x_i x_j$=-1. In contrast, when the two end points are partitioned into the same set, we have (1) x_i=x_j=1 or (2) x_i=x_j=-1, which is equivalent to $x_i x_j$=1.

Notes: The aforementioned objective is an **Integral Quadratic Program**, which is **NP-hard** due to the integral constraints.

Notes: In fact, we can design the **approximation algorithm** for **Max Cut Problem** based on the **Relaxation** of the **Integral Quadratic Program**.

(**Definition 3**) (**Vector Reformulation** of **Quadratic Program**) We associate each vertex $i \in V = \{1, \cdots, n\}$ with a 1-dimensional vector $\mathbf{v}_i$. The **Quadrative Program** of **Max Cut Problem** can be reformulated as follow:

$$\begin{aligned} &\max \frac{1}{2} \sum_{(i,j)\in E} w_{ij}(1-\mathbf{v}_i \cdot \mathbf{v}_j) \\ &\text{s.t. } \mathbf{v}_i \cdot \mathbf{v}_i = 1, i \in \{1, 2, \cdots, n\}, \\ &\quad \mathbf{v}_i \in R^1 \end{aligned}$$

where $\mathbf{v}_i \cdot \mathbf{v}_j$ denotes the **dot product** of vector $\mathbf{v}_i$ and $\mathbf{v}_j$.

We can further **relax** the integral constraint in the aforementioned **Quadrative Program** by allowing each 1-dimensional vector $\mathbf{v}_i$ to be a vector with n dimensions (where n is the number of vertices in the graph):

$$\begin{aligned} &\max \frac{1}{2} \sum_{(i,j)\in E} w_{ij}(1-\mathbf{v}_i \cdot \mathbf{v}_j) \\ &\text{s.t. } \mathbf{v}_i \cdot \mathbf{v}_i = 1, i \in \{1, 2, \cdots, n\}. \\ &\quad \mathbf{v}_i \in R^n \end{aligned}$$

Clearly, this is a **relaxation** because **unit vectors** in the 1-dimensional space are special case. Turns out this can in fact be solved in **polynomial time** (to a desired degree of accuracy) using the 'ellipsoid method'.

Note: This is an instance of a **vector program** (which is equivalent to 'semi-definite programming'). A **vector program** is defined over n vector variables in $\Re^n : \{\mathbf{v}_1, \cdots, \mathbf{v}_n\}$. We need to optimize a **linear function** of the **linear product** $\mathbf{v}_i \cdot \mathbf{v}_j$ subject to **linear constraints** on the linear products. Namely, we can think of a **vector program** as being obtained from a **linear program** by replacing each variable with a linear product of a pair of vectors.

For example, given the following vector program

$$\begin{aligned} &\min 3\mathbf{v}_1 \cdot \mathbf{v}_2 + 4\mathbf{v}_2 \cdot \mathbf{v}_3 + 7\mathbf{v}_1 \cdot \mathbf{v}_3 \\ &\text{s.t. } 4\mathbf{v}_1 \cdot \mathbf{v}_3 + 5\mathbf{v}_2 \cdot \mathbf{v}_3 \geq 17 \quad , \\ &\quad \mathbf{v}_i \in \Re^3 \end{aligned}$$

we can treat it as a linear program

$$\begin{aligned} &\min 3x_1 + 4x_2 + 7x_3, \\ &\text{s.t. } 4x_3 + 5x_2 \geq 17 \end{aligned}$$

where we use x_1, x_2, and x_3 to replace $\mathbf{v}_1 \cdot \mathbf{v}_2$, $\mathbf{v}_2 \cdot \mathbf{v}_3$, and $\mathbf{v}_1 \cdot \mathbf{v}_3$.

(**Algorithm 2**) (**Goemans-Williamson Algorithm**)

(1) Solve the **optimal solution** to the **Vector Program** of Max Cut Program.

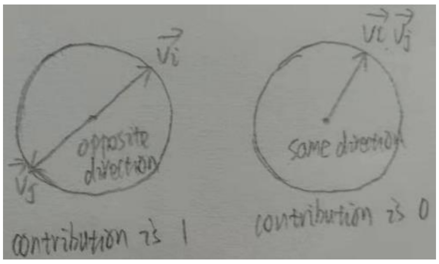
(2) Randomly choose a hyper plane H. Let all vertices with vectors **above** H be on one side of the partition and all vertices with vectors **below** H be on the other side.

Notes: Let $\{\mathbf{v}_1,\cdots,\mathbf{v}_n\}$ denote the **optimal solution** to the **Vector Program**. Let $OPT_{\mathrm{MCP}}(I)$ be the optimal value of the **Max Cut Problem**. We have

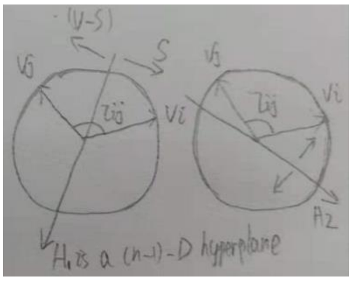
$$OPT_{\mathrm{MCP}}(I)\le\frac{1}{2}\sum_{(i,j)\in E}w_{ij}(1-\mathbf{v}_i\cdot\mathbf{v}_j)\cdot$$

Note: To obtain a **feasible solution** to the **Max Cut Problem**, we need to **round** the vector representation $\{\mathbf{v}_1,\cdots,\mathbf{v}_n\}$ to scales $\{x_1,\cdots,x_n\}$ with $x_i\in\{-1,1\}$. For an edge (i, j), if $(1-\mathbf{v}_i\cdot\mathbf{v}_j)/2$ has large value, then $(1-x_ix_j)/2$ should also have large value.

Notes: Consider the following two extreme examples w.r.t. an edge (i, j). When $\mathbf{v}_i$ and $\mathbf{v}_j$ are in the **opposite direction**, the contribution of $(1-\mathbf{v}_i\cdot\mathbf{v}_j)/2$ should be 1. In contrast, when $\mathbf{v}_i$ and $\mathbf{v}_j$ are in the **same direction**, the contribution of $(1-\mathbf{v}_i\cdot\mathbf{v}_j)/2$ should be 0.



Notes: The strategy of **rounding** in **Algorithm 2** is that <u>if the **angle** between $\mathbf{v}_i$ and $\mathbf{v}_j$ is large, then we want x_i and x_j to have **opposite signs**</u>, i.e., vertex i and j are assigned to different sides of the partition.



(**Fact 1**) (**Fact** in **Trigonometric**) For $0\le\tau\le\pi$, we have

$$\frac{\tau/\pi}{(1-\cos\tau)/2}\ge 0.878\cdot$$

Notes: Consider a **weak analysis** of **Fact 1**.

$$\begin{aligned}\frac{2}{\pi}\cdot\frac{\tau}{(1-\cos\tau)} &= \frac{2}{\pi}\cdot\frac{\tau}{2\sin^2(\tau/2)} \\ &= \frac{1}{\pi}\cdot\frac{\tau}{\sin(\tau/2)}\cdot\frac{1}{\sin(\tau/2)}, \\ &\geq \frac{1}{\pi}\cdot\frac{\tau}{\tau/2}\cdot\frac{1}{1} \\ &= \frac{2}{\pi}\approx 0.6369\end{aligned}$$

where we use the well-known **fact** that $\sin\tau \leq \tau$ for $\tau \geq 0$ and $\sin\tau \leq 1$.

(**Theorem 2**) The **approximation ratio** of **Algorithm 2** is 1,138.

Proof of **Theorem 2**. Let $OPT_{\mathrm{MCP}}(I)$ be the **optimal cost** of the **Max Cut Problem**. Let $\{\mathbf{v}_1,\cdots,\mathbf{v}_n\}$ denote the **optimal solution** to the **vector program**. Then, we have

$$OPT_{\mathrm{MCP}}(I) \leq \frac{1}{2}\sum_{(i,j)\in E} w_{ij}(1-\mathbf{v}_i\cdot\mathbf{v}_j)\cdot$$

For each edge (i, j), suppose the angle between vector v_i and v_j is τ_{ij}. Then, the **probability** that vertex i and j are partitioned into different set is τ_{ij}/π. Let S be the cut given by the **Algorithm 2**, the **expected cost** of **Algorithm 2** is

$$\mathrm{E}[cost(S)] = \sum_{(i,j)\in E} w_{ij}\frac{\tau_{ij}}{\pi}\cdot$$

Since each vector $\mathbf{v}_i$ is unit vector, with $|\mathbf{v}_i|=1$, we have $\mathbf{v}_i\cdot\mathbf{v}_j = |\mathbf{v}_i||\mathbf{v}_j|\cos\tau_{ij} = \cos\tau_{ij}$. Then, we further have

$$OPT_{\mathrm{MCP}}(I) \leq \frac{1}{2}\sum_{(i,j)\in E} w_{ij}(1-\mathbf{v}_i\cdot\mathbf{v}_j) = \frac{1}{2}\sum_{(i,j)\in E} w_{ij}(1-\cos\tau_{ij})\cdot$$

Hence, the **approximation ratio** of **Algorithm 2** is

$$\begin{aligned}\frac{\mathrm{E}[cost(S)]}{OPT_{\mathrm{MCP}}(I)} &\geq \frac{\sum_{(i,j)\in E} w_{ij}\tau_{ij}/\pi}{[\sum_{(i,j)\in E} w_{ij}(1-\cos\tau_{ij})]/2} \\ &\geq \frac{0.878[\sum_{(i,j)\in E} w_{ij}(1-\cos\tau_{ij})]/2}{[\sum_{(i,j)\in E} w_{ij}(1-\cos\tau_{ij})]/2}, \\ &= 0.878\end{aligned}$$

where we use **Fact 1** to derive the lower bound.