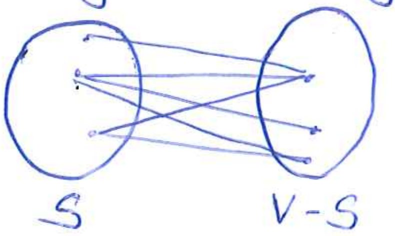
Lecture 14 : Max-Cut

I/p : an undirected graph $G = (V, E)$

O/p : a partition $(S, V-S)$ of the vertex set such that the number of edges crossing this cut is maximized.

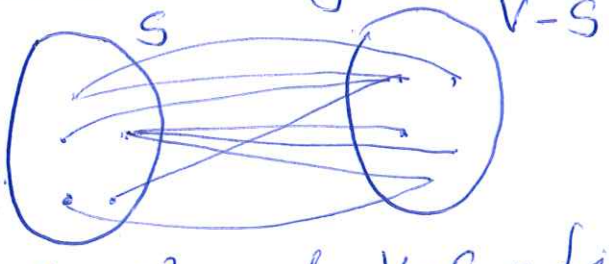


$E(S, V-S) = \{e \in E$: one endpoint of e is in S and the other is in $V-S\}$.

We know that Max-Cut is an NP-hard problem.
We also know there is a simple randomized algorithm that computes a cut $(X, V-X)$ such that the expected number of edges crossing this cut $\geq \frac{OPT}{2}$, where OPT = optimum value.

The above randomized algorithm can be derandomized using the method of conditional expectations.

No c-approximation algorithm was known for this problem for any constant $c > \frac{1}{2}$ till 1994 when Goemans and Williamson showed a 0.878-approximation algorithm.



Let us associate ± 1 values to vertices.

Suppose $V = \{1, 2, \ldots, n\}$.

Let $S = \{i : z_i = 1\}$ and $V-S = \{i : z_i = -1\}$.

The edge (i, j) crosses the cut $\Leftrightarrow z_i z_j = -1$.

So the following quadratic program solves the max-cut problem.

$$\max \sum_{(i,j)\in E} \frac{1 - z_i z_j}{2}$$

$$\text{s.t.} \quad \left.\begin{array}{ll} z_i^2 = 1 & \forall i \\ z_i \in \mathbb{Z}. & \end{array}\right\} \text{That is, } z_i \in \{\pm 1\} \ \forall i.$$

So z_i for each i $\in S^0$, which is the unit sphere in $\mathbb{R}$.

<u>A relaxation</u> : let us go from $\mathbb{R}$ to $\mathbb{R}^n$.

That is, each z_i is relaxed to a vector $u_i \in S^{n-1}$, which is the unit sphere in $\mathbb{R}^n$.

The product $z_i z_j$ is replaced with the dot product (u_i, u_j). So the above quadratic program becomes the following <u>vector program</u>:

$$\max \sum_{(i,j)\in E} \frac{1 - (u_i, u_j)}{2}$$

$$\text{s.t.} \quad \left.\begin{array}{ll} (u_i, u_i) = 1 & \forall i \\ u_i \in \mathbb{R}^n & \end{array}\right\} \begin{array}{l}\text{That is,} \\ u_i \in S^{n-1}.\end{array}$$

This is called a vector program since the unknowns are vectors.

Observe that this vector program is indeed a relaxation of the quadratic program. That is, for every solution of the quadratic program, there is a corresponding solution of the vector program with the same value.

Hence the optimum value of the vector program $\geq$ OPT. (The vector program looks more complicated than the QP.)

Question: How do we solve the vector program?

Semidefinite programming will allow us to solve the vector program efficiently, to any desired accuracy. Recall that an SDP can be solved within an additive error of ε, for any $\varepsilon > 0$, in time $\text{poly}(n, \log\frac{1}{\varepsilon})$ using the ellipsoid algorithm.

Let us perform the following variable substitution:

$x_{ij} = (u_i, u_j)$ for all i, j.

Thus we get the following SDP:

$$\max \sum_{(i,j)\in E} \frac{1 - x_{ij}}{2}$$

$$\text{s.t.} \quad x_{ii} = 1 \text{ for } i = 1, \ldots, n.$$

$$\underbrace{X \succeq 0}_{\downarrow}$$

X is the $n \times n$ matrix whose (i,j)-th entry is x_{ij}. This is a symmetric matrix and the above constraint $X \succeq 0$ says that it is positive semidefinite.

Claim. The above SDP is equivalent to the vector program.

Suppose $u_1, \ldots, u_n$ constitute a feasible solution to the vector program. So they are unit vectors. Define $x_{ij} = (u_i, u_j)$ for all i, j. Then $X = \mathcal{U}^T\mathcal{U}$ where $\mathcal{U} = [u_1\ u_2 \ldots u_n]$ (the matrix with u_i as its i-th column).

Such a matrix X is positive semidefinite and $x_{ii} = 1$ follows from $(u_i, u_i) = 1$. So X is a feasible solution to the SDP with the same value.

Let us now show that every feasible solution X to the SDP yields a solution to the vector program with the same value. Since X is positive semidefinite, $X = \mathcal{U}^T \mathcal{U}$ for some $n \times n$ real matrix $\mathcal{U}$. So if X is a feasible solution of the SDP, then the columns of $\mathcal{U}$ provide a feasible solution of the vector program. Moreover, due to the constraint $x_{ii} = 1$, they are actually unit vectors.

Thus the SDP has the same optimum value as the vector program. Hence we can find in time $\text{poly}(n, \log\frac{1}{\varepsilon})$ a matrix $X^* \succcurlyeq 0$ with $x^*_{ii} = 1$ for all i and with $\sum\limits_{(i,j)\in E} \frac{1 - x^*_{ij}}{2} \geqslant \text{OPT} - \varepsilon$.

We can also compute in polynomial time a matrix $\mathcal{U}^*$ such that $X^* = (\mathcal{U}^*)^T \mathcal{U}^*$ up to a tiny error. This is a Cholesky factorization of X^*. We will assume that the factorization is exact.

Then the columns $u_1^*, u_2^*, \ldots, u_n^*$ of $\mathcal{U}$ are unit vectors that form an almost-optimal solution of the vector program: $\sum\limits_{(i,j)\in E} \frac{1 - (u_i^*, u_j^*)}{2} \geqslant \text{OPT} - \varepsilon$.

Randomized rounding algorithm

At this point we have an almost optimal solution $u_1^*, \ldots, u_n^*$ to the vector program. Let opt* denote its objective function value.

These vectors lie on S^{n-1}. We need to obtain a cut $(S, V-S)$ whose size is a large fraction of opt*.

Let θ_{ij} denote the angle between the vectors u_i^* and u_j^*. The contribution of this pair of vectors to opt* is $\frac{1-\cos\theta_{ij}}{2}$. The closer θ_{ij} is to π, the larger this contribution will be. That is, we would like vertices i and j to be separated if θ_{ij} is large. The following method accomplishes this:

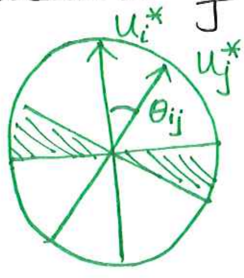
- Pick r to be a uniformly distributed point on the unit sphere S^{n-1}.

  Let $S = \{ i : (u_i^*, r) \geq 0 \}$.

Claim. For any pair of vertices i and j,

$$\Pr[i \text{ and } j \text{ are separated}] = \frac{\theta_{ij}}{\pi}.$$

Proof. Let p be the projection of r on the 2-dimensional plane spanned by u_i^* and u_j^*. Observe that i and j will be separated if and only if p lies in one of the two arcs of angle θ_{ij} shown ~~below~~ here: $\longrightarrow$



Since r is uniformly distributed in S^{n-1}, the direction of its projection p on this plane is uniformly distributed in $[0, 2\pi]$. Thus the probability p falls into this "double wedge" is

$$\frac{2\theta_{ij}}{2\pi} = \frac{\theta_{ij}}{\pi}. \quad ▨$$

Generating Spherically Symmetric Random Vectors using i.i.d. variables

- Choose each coordinate independently from a $N(0,1)$ distribution.
- The density function of each coordinate is $\frac{1}{\sqrt{2\pi}} e^{-x^2/2}$. So the joint density is
$$\boxed{\frac{1}{(\sqrt{2\pi})^n} \cdot e^{-\left(\frac{x_1^2 + \cdots + x_n^2}{2}\right)}}$$
- The joint density is the same for all tuples $(x_1, \dots, x_n)$ with $x_1^2 + \cdots + x_n^2 = r^2$.
- So the joint density function is spherically symmetric.

The Goemans-Williamson algorithm

1. Solve the vector program. Let $u_1^*, \dots, u_n^*$ be the solution returned.
2. Sample r uniformly at random from S^{n-1}.
3. Let $S = \{i : (u_i^*, r) \geq 0\}$.
4. Return $(S, V-S)$.

Lemma. Let C be the random variable denoting the size of the cut returned by the GW algorithm. $E[C] \geq (0.878)\, OPT$.

Proof. $E[C] = \sum_{(i,j)\in E} \Pr[i \text{ and } j \text{ are separated}]$

$$= \sum_{(i,j)\in E} \frac{\theta_{ij}}{\pi}.$$

Claim. For all $\theta \in [0,\pi]$, we have $\boxed{\frac{2}{\pi}\frac{\theta}{(1-\cos\theta)} \geq 0.878}$

Proof. This follows from calculus.

We want to find $\min_{0\leq\theta\leq\pi} \frac{\theta}{(1-\cos\theta)}$. For this, we need to solve $\theta = \tan\frac{\theta}{2}$. Approximately, $\theta \approx \frac{11\pi}{15}$.

Substituting this value, we get $\frac{2}{\pi}\cdot\frac{11\pi/15}{\left(1-\cos\frac{11\pi}{15}\right)} \geq 0.878.$

So $\sum_{(i,j)\in E} \frac{\theta_{ij}}{\pi} \geq 0.878 \sum_{(i,j)\in E} \frac{1-\cos\theta_{ij}}{2}$

$$\geq \underbrace{0.878}_{\text{in fact, this is } 0.8785672\ldots}(opt^*).$$

We have $opt^* \geq OPT - \mathcal{E}$ and $E[C] \geq (0.87856\ldots)\, opt^*$.

By choosing $\mathcal{E}$ to be a small enough constant, we can claim $E[C] \geq (0.878)\cdot OPT$. ▨

By running the GW algorithm k times and outputting the $\overset{\text{largest}}{\text{heaviest}}$ cut found in these runs, we can ensure that with prob. $\geq 3/4$, the cut returned by this algorithm has size $\geq (0.878)$ OPT.

Let q be the probability that the algorithm returns a cut of size $< (1-\varepsilon) \cdot E[C]$.

$$E[C] \leq q \cdot (1-\varepsilon) \cdot E[C] + (1-q) m$$

So $q(m - (1-\varepsilon) \cdot E[C]) \leq m - E[C]$.

Hence $$q \leq \frac{m - E[C]}{m - (1-\varepsilon) \cdot E[C]} = 1 - \frac{\varepsilon \cdot E[C]}{m - (1-\varepsilon) \cdot E[C]}$$

Note that $E[C] \geq (0.87856) \cdot \text{OPT}$ $\rightarrow$ (The lemma statement can be made tighter.)

Also $\text{OPT} \geq \frac{m}{2}$ (why?).

So $$q \leq 1 - \underbrace{\boxed{\frac{\alpha \cdot \varepsilon}{2 + \varepsilon \cdot \alpha - \alpha}}}_{\text{call this } c}$$ where $\alpha = 0.87856$.

Let ε be such that $(1-\varepsilon) \cdot (0.87856) = 0.878$.
We will run the GW algorithm $2/c$ times and return the largest cut found in these runs.

$$\Pr[\text{size of this cut} \geq 0.878 \text{ OPT}] \geq 1 - (1-c)^{2/c}$$
$$\geq 1 - \frac{1}{e^2} \geq \frac{3}{4}.$$

<u>References</u>

1. Approximation Algorithms and Semidefinite Programming. (B. Gärtner and J. Matoušek)

2. Approximation Algorithms. (V. V. Vazirani)