22C:253 Lecture

Scribe:Sriram Penumatcha

6th November 2002

MAX CUT
Input:A graph $G = (V, E)$ with edge weights $W : E- > Q^+$
Output:A partition $(S, \bar{S})$ such that the sum total of the weights of edges crossing $(S, \bar{S})$ is maximized

Applications: MAX-CUT has a lot of applications in VLSI design,also in graph partitioning for solving partial differential equations(variant of MAX-CUT)

Let us analyze what happens if we throw each vertex into S or $\bar{S}$ independently with probability $\frac{1}{2}$ into the solution.We get a factor $\frac{1}{2}$ approximation algorithm(in the expected sense)

$$Prob[edge\{u, v\} crosses (S, \bar{S})] = \frac{1}{2}$$

Therefore the expected contribution of edge $\{u, v\}$ is $\frac{W_v}{2}$. Expected cost of the solution $= \frac{1}{2} \sum_{\{u,v\} \in E} W_{uv} \geq \frac{1}{2} OPT$. This can be easily derandomized using the method of conditional expectation

Algorithm using SDF:

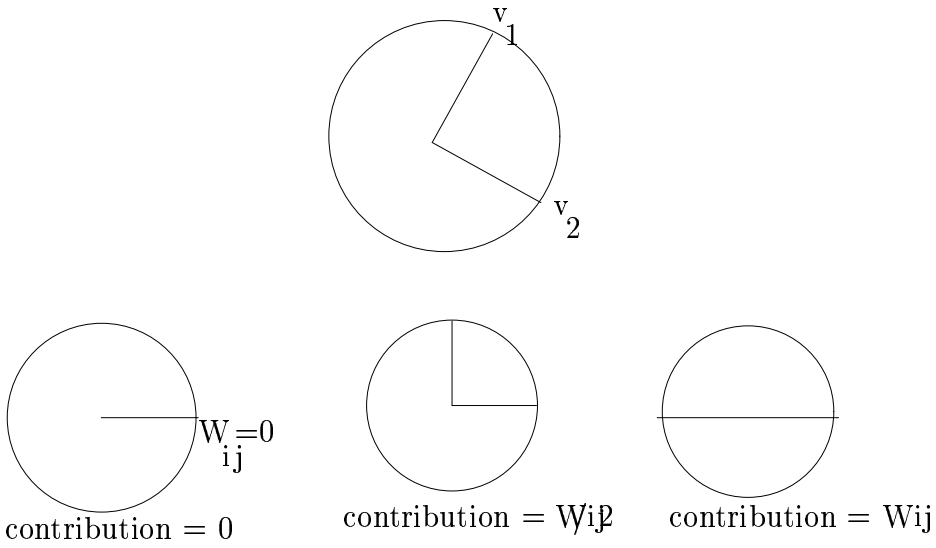
We use a technique called semi-definite programming to solve this problem.This technique was invented by Goemans and Williamson in 1995
The following program solves MAX-CUT

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

such that ${y_i}^2 = 1$
We can observe that this is a quadratic program because sum of the terms in the objective function and constriants are quadratic.Solving quadratic programs is NP-Complete

Consider the following vector relaxation of the quadratic program. Replace y_i by a vector $V_i \in R^n$
We get,

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



such that $V_iV_j = 1$
This is a vector program which is a relaxation of the quadratic program above because for any feasible solution $(y_1, y_2, y_3, ..., y_n)$ of the quadratic program, the solution $V_i = (y_i, 0, 0, ..., 0) \forall i = 1, 2, 3, ..., n$ is a feasible solution of the vector program. Hence if OPT_V is the cost of an optimal solution of the vector program then $OPT_V \geq OPT$

An important observation by Goemans and Williamson is that this vector program is equivalent to Semi-Definite programming,and SDP can be solved in polynomial time. For this lecture we take this as given,the proof will be given in the next lecture

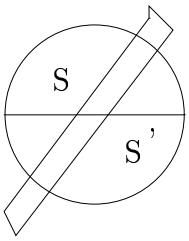
Note any solution to the vector program(VP) is collection of n vectors each on the surface of the n-dimensional unit sphere S_{n-1}
Also note that

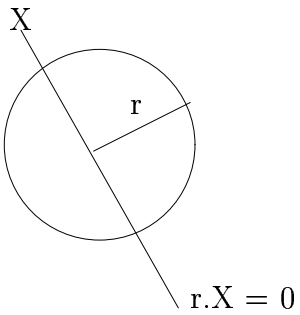
$$V_iV_j = |V_i||V_j|cos(\theta_{ij})$$

Since $|V_i| = |V_j|$, we have $V_iV_j = cos(\theta_{ij})$

$$OPT_V = \frac{1}{2} \sum_{\{i,j\} \in E} W_{ij}(1 - cos(\theta_{ij}))$$

We can observe that the larger the angle, the more the relaxation says that vertces should lie on the opposite sides. So, we apply a key idea here here of picking a random hyperplane passing through the origin
This is done by picking a unit vector r uniformly at random , uniformly from among all unit vectors in R^n.Define the random hyperplane





$$H = \{X | r.X = 0\}$$

$$S = \{i | r.V_i \geq 0\}$$

$$\bar{S} = \{i | r.V_i < 0\}$$

Procedure:
Pick $x_1, x_2, x_3, ..., x_n$ from the unit mormal distribution

$$f(x) = \frac{1}{\sqrt{(2\pi)}} e^{-\frac{x^2}{2}}$$

define

$$d = \frac{(\sum_{1=1}^{n} {x_i}^2)}{2}$$

define $r = (\frac{x_1}{d}, \frac{x_2}{d}, ..., \frac{x_n}{d})$
$r = (y_1, y_2, y_3, ..., y_n)$
probability distribution function for r is given by

$$\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$$

$$= (\frac{1}{\sqrt{2\pi}})^n e^{-\frac{1}{2}\sum_{i=1}^{n} {y_i}^2}$$

$$= \frac{1}{\sqrt{(2\pi)}} e^{-\frac{1}{2}}$$

which doesnt depend on y_is and is therefore uniform

The probability that for any edge $\{u, v\}$ that $\{u, v\}$ crosses $barS$ can be given as $\frac{\theta_{ij}}{\pi}$. Expected contribution of edge $\{i, j\}$ is $W_{ij}\theta_{ij}$. Total expected cost of the solution is

$$\frac{1}{\pi} \sum_{\{i,j\} \in E} W_{ij}\theta_{ij}$$

We are comparing this with

$$OPT_V = \frac{1}{2} \sum_{\{i,j\} \in E} W_{ij}(1 - cos(\theta_{ij}))$$

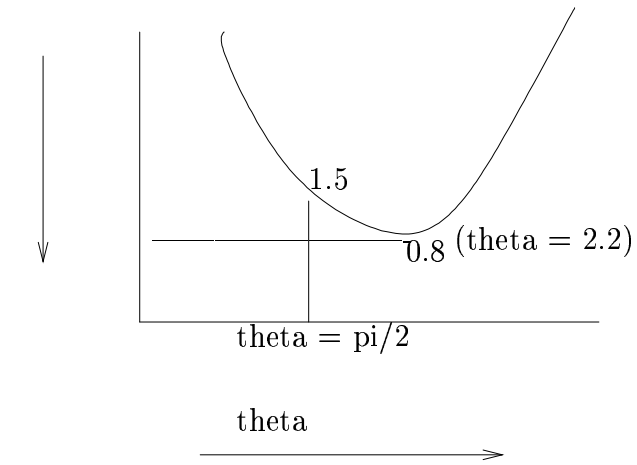
We would like to show that , $\frac{\theta_{ij}}{\pi}$ is not too small as compared to $\frac{(1-cos\theta_{ij})}{2}$. So we look at

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

=

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

where α is our approximation factor ,$\alpha = 0.8785$
which occurs for $\theta = 2.2$.


theta/1-cos(theta)
1.5
0.8
(theta = 2.2)
theta = pi/2
theta