Advanced Approximation Algorithms **(CMU 18-854B, Spring 2008)**

Lecture 14: Semidefinite Programming and Max-Cut

Feb 28, 2008

Lecturer: Ryan O'Donnell *Scribe: Ali Kemal Sinop*

1 Maximum Cut

In the Maximum Cut problem, we are given a weighted graph $G = (V, E, W)$. The goal is to find a partitioning $(S, \bar{S}), S \subset V$ of the graph so as to maximize the total weight of edges in the cut. Formally the problem is to find $f : V \to \{0, 1\}$ which maximizes $\sum_{ij \in E : f(i) \neq f(j)} w_{ij}$.

In homework #1, we already saw a greedy $1/2$-approximation algorithm for this problem. Now let's examine a potential LP formulation.

$$\begin{aligned} \text{Variables: } & d_{ij} = \begin{cases} 1, & \text{if edge ij is in cut} \\ 0, & \text{otherwise} \end{cases} \\ & 0 \le d_{ij} \le 1 \\ & \max \textstyle\sum_{ij} w_{ij} d_{ij} \end{aligned}$$

However, the above LP is not enough by itself. What constraints can be added to this formulation? One observation is that on any triangle, the number of edges in the cut can be at most 2. Another observation is that d's form a metric on the graph, so they obey triangle inequality. Hence we have:

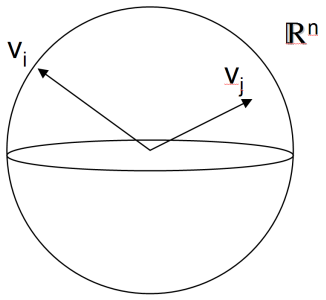
$$\begin{aligned} \text{Constraints: } & d_{ij} + d_{jk} + d_{ki} \le 2, \\ & d_{ij} + d_{jk} \ge d_{ik}. \end{aligned}$$

The integrality gap of this LP is still $1/2$ [7]. In fact $\forall \epsilon > 0$, for random graphs $G_{n, \frac{\log n}{n}}$, $\mathrm{LP}(G) \ge 1 - \epsilon$ but $\mathrm{OPT}(G) \le \frac{1}{2} + \epsilon$. Moreover even if we were to add some local constraints, these results still hold.

Since LP relaxations can't get any better than greedy algorithm, we need to look for some other solution. First let's change the indicator function on variables to $f : V \to \{-1, +1\}$. Then $f(i) \neq f(j) \iff f(i)f(j) = -1$. Using this observation, we can write an Integer-Quadratic Programming (IQP) formulation:

$$\begin{aligned} \text{maximize } & \textstyle\sum_{ij \in E} w_{ij} \left(\frac{1 - v_i v_j}{2} \right) \\ \text{subject to } & v_i^2 = 1, \\ & v_i \in \mathbb{R}. \end{aligned}$$

Notice that this is exactly max-cut problem, so solving this is NP-hard still. In order to obtain a relaxation, we will allow the variables v to be in a higher dimension space. So we change the constraint $v_i \in \mathbb{R}$ with $v_i \in \mathbb{R}^n$. Then $v_i v_j$ becomes $v_i \cdot v_j$ ($\cdot$ denotes the dot product) and $v_i^2 = 1$ becomes $\|v_i\|^2 = v_i \cdot v_i = 1$.



This problem seems very complicated, but it can actually be solved[1] in polynomial time. The reason is that it is actually an LP in disguise. To see this, first let ρ_{ij} denote the dot product $v_i \cdot v_j$. The problem is now:

$$\begin{array}{ll} \text{maximize} & \sum_{ij \in E} w_{ij} \left(\frac{1-\rho_{ij}}{2}\right) \\ \text{subject to} & \rho_{ii} = 1 \end{array}$$

This formulation is currently unbounded. The constraint we are missing now is the geometric constraints saying $\exists v_1, v_2, \ldots, v_n \in \mathbb{R}^n$ such that $\rho_{ij} = v_i \cdot v_j$. (Without loss of generality, we can assume that all vectors lie in dimension n, because only dot products matter.) Although it is not immediately obvious, these constraints turn out to be an infinite collection of linear constraints on the ρ_{ij}'s.

Let $P = (\rho_{ij})_{ij}$ be the $n \times n$ matrix of inner products (Gram matrix). There are three properties of this matrix:

1. P is symmetric, because $\rho_{ij} = v_i \cdot v_j = v_j \cdot v_i = \rho_{ji}$. This implies that all eigenvalues of P are real.

2. All diagonals of P are 1's, because $\rho_{ii} = \|v_i\|^2 = 1$.

3. Let $V = [v_1 \quad v_2 \quad \ldots \quad v_n]$ be the matrix formed by stacking the vectors v. Then we have

$$V^T V = \begin{bmatrix} v_1^T \\ \vdots \\ v_n^T \end{bmatrix} [v_1 \quad v_2 \quad \ldots \quad v_n] = \begin{bmatrix} v_1 \cdot v_1 & v_1 \cdot v_2 & \ldots & v_1 \cdot v_n \\ \vdots & \vdots & \ddots & \vdots \\ v_n \cdot v_1 & v_n \cdot v_2 & \ldots & v_n \cdot v_n \end{bmatrix} = P$$

   Hence there exists a matrix V such that $V^T V = P$.

   This implies a non-trivial fact about matrix P. Consider any $x \in \mathbb{R}^n$. Then

$$x^T P x = x^T V^T V x = (Vx)^T (Vx) = \|Vx\|^2 \geq 0$$

   Hence $\forall x \in \mathbb{R}^n, x^T R x \geq 0$. From now on, such matrices will be called positive semidefinite (PSD) matrices.

[1]essentially

Theorem 1.1. *Let P be a symmetric n-by-n matrix. The following are equivalent:*

1. *There exists an m-by-n matrix V such that $V^TV = P$.*
2. *For all $x \in \mathbb{R}^n$, $x^TPx \geq 0$.*
3. *All eigenvalues of P are non-negative.*

So we can write the missing geometric constraint as:

$$x^TRx \geq 0 \implies \sum_{ij} x_i x_j \rho_{ij} \geq 0, \quad \forall x \in \mathbb{R}^n \tag{1}$$

Note that these are *linear constraints* on the ρ_{ij}'s (albeit infinitely many). However using the Ellipsoid Algorithm, we can still solve this problem in polynomial time as long as we have...

Separation Oracle: Given $P = (\rho_{ij})$, assume it is symmetric (we can easily check this). We know that if P is PSD, then all its eigenvalues are non-negative. Therefore we can compute all eigenvalues of P in polynomial time within a desired accuracy ϵ [5]. If all of them are non-negative, then we are done. Otherwise the eigenvector associated with the negative eigenvalue is the desired vector x violating (1).

After having computer P, we can obtain V by running Cholesky decomposition on P.

1.1 Formal Semidefinite Programming (SDP)

Any LP over n^2 variables arranged to a matrix X with the additional constraint that $X \succeq 0$ (X is PSD) is called a semidefinite program (SDP).

Before going on further, there are some technical issues associated with SDPs. We need the answer, constraints and the solution to be written down with $\text{poly}(n)$ bits, however no such guarantee exists for the solution of SDPs. First, the solution of SDP might be irrational.

Another problem is that the solution might be doubly-exponential. This is not a problem most of the time; for the case of MAX-CUT SDP, since $\|v_i\| \leq 1$, this can't occur.

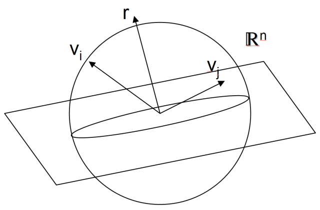
The truth of the matter is that the Ellipsoid Algorithm will give an answer within an additive ϵ of OPT with running time overhead $\text{poly}\log\epsilon^{-1}$. Henceforth we will not worry about these kinds of issues.

Currently, SDPs with 1000's of variables are solvable. The state of the art method for solving SDPs is interior point methods with running time $\tilde{O}(n^{3.5}\log\epsilon^{-1})$ [2].

For MAX-CUT SDP, there are combinatorial algorithms utilizing Primal-Dual methods in time $\tilde{O}(\frac{mn}{\epsilon^3})$ for general graphs [6] and in time $\tilde{O}(\frac{m}{\epsilon^{O(1)}})$ for regular graphs [1]. But dependence of these methods on ϵ is very bad, so interior point methods do better in practice.

1.2 Randomized Rounding

Assume we solved the SDP formulation for MAX-CUT and got P and V. So SDP suggests a way of separating nodes i and j such that if $\rho_{ij} = v_i \cdot v_j$ is close to -1, then we should cut it, otherwise there is no need to bother. The Goemans-Williamson randomized rounding technique [4] is to choose a uniformly random hyperplane through the origin and use it to cut vectors/vertices into two parts.



To draw a random hyperplane, we can also pick the normal to the hyperplane to be a random vector r from the surface of unit sphere. Then

$$f(i) = \operatorname{sgn}(r \cdot v_i)$$

In order to pick a random unit vector, we can pick n random Gaussian variables, $r_1, \ldots, r_n$ and let $r = \frac{(r_1, r_2, \ldots, r_n)}{\sqrt{r_1^2 + r_2^2 + \ldots r_n^2}}$.

Fact: A collection of n random Gaussian variables is a spherically symmetric distribution. The probability density function is

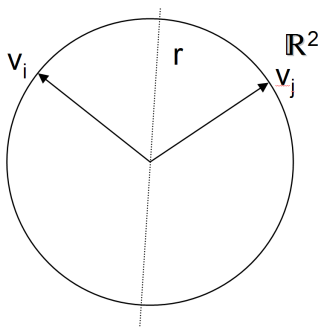
$$\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}} e^{-x_i^2/2} = \frac{1}{(2\pi)^{n/2}} e^{-\|x\|^2/2}.$$

Hence after normalization, r will be a uniformly random vector over the unit sphere.

The expected value of this cut is

$$\mathbf{E}[\text{value of cut}] = \mathbf{E}[\sum_{ij} w_{ij} \mathbf{1}[\text{(i,j) cut}]] = \sum_{ij} w_{ij} \mathbf{Pr}[\text{(i,j) cut}]$$

By spherical symmetry, we can rotate points v_i and v_j so that they both lie in the 2-d plane:

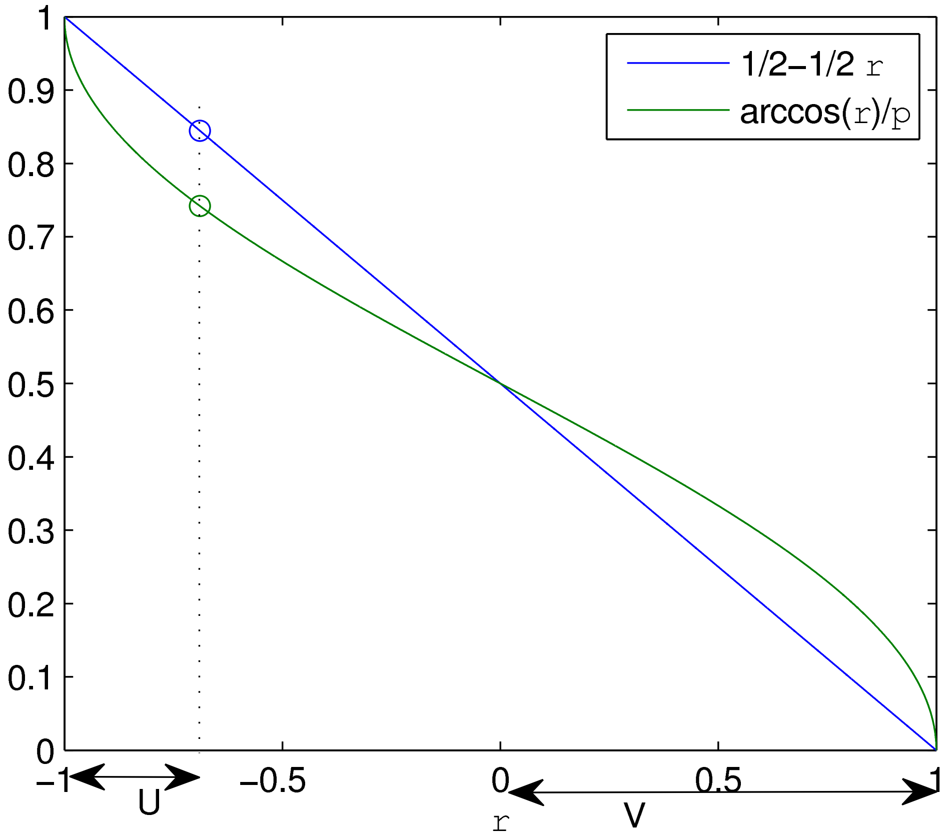


Here the random hyperplane will be equivalent to picking a random diameter of this circle. Then the probability of cutting this edge is equal to the angle between v_i and v_j over π:

$$\mathbf{Pr}[\text{(i,j) cut}] = \frac{\angle(\vec{v_i}, \vec{v_j})}{\pi} = \frac{\arccos(\vec{v_i} \cdot \vec{v_j})}{\pi}$$

Hence the expected value of this randomized rounding procedure is $Alg_{GW}(G) = \sum_{ij} w_{ij} \frac{\arccos(\vec{v_i} \cdot \vec{v_j})}{\pi}$, which we want to compare against $\mathrm{SDP}(G) = \sum_{ij} w_{ij}(\frac{1}{2} - \frac{1}{2}\vec{v_i} \cdot \vec{v_j}) \geq \mathrm{OPT}(G)$.

The plot of functions $(\arccos\rho)/\pi$ and $\frac{1}{2} - \frac{1}{2}\rho$ is given below:



In this plot, V is the place where randomized rounding algorithm is doing better. The dotted lines shown the points on which the gap is maximum.

We know that the gap is smaller than $\alpha_{GW} := \min_{-1 \leq \rho \leq 1} \frac{\arccos\rho/\pi}{1/2 - 1/2\rho} \approx 0.87854$. Hence at every edge, the randomized rounding algorithm will achieve at least a factor of 0.878. Therefore:

Theorem 1.2. $\frac{Alg_{GW}(G)}{\mathrm{SDP}(G)} \geq \alpha_{GW} \approx 0.878$.

Consequently Goemans-Williamson algorithm [4] is a factor $.878$ approximation.

References

[1] S. Arora and S. Kale. A combinatorial, primal-dual approach to semidefinite programs. In *STOC*, pages 227–236, 2007.

[2] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[3] U. Feige, M. Karpinski, and M. Langberg. Improved approximation of MAX-CUT on graphs of bounded degree. *Electronic Colloquium on Computational Complexity (ECCC)*, (021), 2000.

[4] M. X. Goemans and D. P. Williamson. .879-approximation algorithms for max cut and max 2sat. In *STOC*, pages 422–431, 1994.

[5] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 3 edition, 1996.

[6] P. Klein and H.-I. Lu. Efficient approximation algorithms for semidefinite programs arising from max cut and coloring. In *STOC*, pages 338–347, 1996.

[7] S. Poljak and Z. Tuza. On the expected relative error of the polyhedral approximation of the max-cut. *Operational Research Letters*, 16:191–198, 1994.