Approximation Algorithms and Hardness of Approximation April 16, 2013

Lecture 14

Lecturer: Alantha Newman *Scribes: Marwa El Halabi*

1 Semidefinite Programming and Graph Partitioning

In previous lectures, we saw how linear programs can help us devise approximation algorithms for various $\mathcal{NP}$-hard optimization problems. In this lecture, we will introduce a more general class of relaxations, *vector programs*, which allow for variables to be vectors instead of scalars. In particular, vector programs are a relaxation for problems that can be formulated as *strict quadratic programs*.

Definition 1 (Quadratic program) *A quadratic program (QP) is an optimization problem whose objective function is quadratic in terms of integer valued variables, subject to quadratic constraints. If in addition a quadratic program has all its monomials degree 0 or 2, then it's a* ***strict quadratic program****.*

As we will show later, vector programs are equivalent to semidefinite programs, which can be viewed as a generalization of linear programs. We can solve semidefinite programs by the ellipsoid algorithm up to an arbitrary small additive error ϵ, in time polynomial in the input size and $\log(1/\epsilon)$.

We will illustrate the concept of semidefinite programming by examining some graph partitioning problems. We start with the well known maximum cut problem.

1.1 Maximum Cut

Given an undirected graph $G = (V, E)$, the goal of the maximum cut problem is to find a partition of the vertices, $(S, \bar{S})$, that maximizes the number of edges crossing the cut, i.e. edges with one endpoint in S and the other endpoint in $\bar{S}$. We denote the number of edges crossing the maximum cut by $\text{OPT}_{\text{Max-Cut}}$. The max cut problem is know to be $\mathcal{NP}$-Hard, so our goal is to find a good polynomial time approximation algorithm for it. Note that $|E|$ is an upper bound on $\text{OPT}_{\text{Max-Cut}}$. Thus, we can achieve a $\frac{1}{2}$-approximation for max cut simply by placing each vertex in S or $\bar{S}$ with probability 1/2.

Th max cut problem can be formulated as a quadratic program:

$$\begin{aligned} \max \sum_{(i,j)\in E} x_i(1-x_j) \\ \text{s.t } x_i \in \{0,1\} \quad \forall i \in V, \end{aligned} \tag{1}$$

where each variable x_i is set to one if vertex i is in set S, and to zero if in set $\bar{S}$. Note that an edge with both ends in the same set will not contribute to the objective function.

If we relax the integer constraint in this QP, we have the following formulation:

$$\begin{aligned} \max \sum_{(i,j)\in E} x_i(1-x_j) \\ \text{s.t } x_i \in [0,1] \quad \forall i \in V. \end{aligned} \tag{2}$$

If we can solve this relaxed version optimally, we will still be able to find a max cut. Consider the solution for the relaxed QP, OPT_{rel}. For any vertex $h \in V$, with fractional value assigned to x_h, we can rewrite the objective function (1) as follows. Let $\delta(h)$ denote all edges adjacent to vertex h.

$$\sum_{i,j\in E\setminus\delta(h)} x_i(1-x_j) + x_h \overbrace{\sum_{j\in\delta(h)}(1-x_j)}^{A} + (1-x_h)\overbrace{\sum_{j\in\delta(h)} x_j}^{B}.$$

Then if $A \geq B$, we round x_h to one, otherwise we round it to zero. Let's denote by OPT_{rd} the solution we get after rounding OPT_{rel}. Note that $\text{OPT}_{\text{rel}} \leq \text{OPT}_{\text{rd}}$, so by solving the relaxed QP for max cut and rounding the solution, we obtain an integral solution that is at least as good as OPT_{rel}, which is at least as good as the true optimal value $\text{OPT}_{\text{max-cut}}$. We can deduce from this that solving this particular relaxed version is also $\mathcal{NP}$-hard, since it boils down to solving exactly the $\mathcal{NP}$-hard max cut problem in polynomial time. So relaxing the integrality constraint does not help here. Instead, we'll relax the max cut problem to a semidefinite program.

Definition 2 (Semidefinite program (SDP)) *A semidefinite program is an optimization problem whose objective function is linear in terms of its variables x_{ij}, subject to linear constraints over the variables, with the additional constraint that the symmetric matrix $X = [x_{ij}]$ should be positive semidefinite.*

We recall the definition of positive semidefinite matrix:

Definition 3 *A symmetric matrix $X \in \mathbb{R}^{n \times n}$ is positive semidefinite ($X \succeq 0$) iff for all $y \in \mathbb{R}^n$, $y^T X y \geq 0$.*

Theorem 4 *$X \in \mathbb{R}^{n \times n}$ is a symmetric matrix, then the following are equivalent:*

1. *X is positive semidefinite.*
2. *All eigenvalues of X are non negative.*
3. *$\exists V \in \mathbb{R}^{m \times n}$, $m \leq n$, s.t $X = V^T V$.*

Note that we can compute the eigendecomposition of a symmetric matrix $X = Q \Lambda Q^T$ in polynomial time, thus we can test for positive definiteness in polynomial time. However, the decomposition $V^T V$ is not polynomial time computable, since taking the square root of the diagonal matrix Λ can lead to irrational values. We can get an arbitrarily good approximation of this decomposition, so we can assume we have the exact decomposition in polynomial time, given that this inaccuracy can be included in the approximation factor.

By replacing the variable x_{ij} in a SDP by the inner product of the two vectors v_i and v_j in the decomposition $X = V^T V$ corresponding to entry x_{ij}, we obtain an equivalent vector program:

$$\begin{aligned} \max & \sum_{i,j} c_{ij} x_{ij} \\ \text{s.t} & \sum_{i,j} a_{ijk} x_{ij} = b_k \\ & x_{ij} = x_{ji} \\ & X \succeq 0 \end{aligned} \quad \Longleftrightarrow \quad \begin{aligned} \max & \sum_{i,j} c_{ij} (v_i \cdot v_j) \\ \text{s.t} & \sum_{i,j} a_{ijk} (v_i \cdot v_j) = b_k \\ & v_i \in \mathbb{R}^n \end{aligned}$$

The max cut problem admits a strict quadratic program formulation, which can be relaxed to a vector program:

$$\begin{aligned} \max & \sum_{(i,j) \in E} \frac{1 - v_i \cdot v_j}{2} \\ \text{s.t} & \ v_i \in \{-1, 1\} \end{aligned} \quad \Longrightarrow \quad \begin{aligned} \max & \sum_{(i,j) \in E} \frac{1 - v_i \cdot v_j}{2} \\ \text{s.t} & \ v_i \cdot v_i = 1 \\ & v_i \in \mathbb{R}^n \end{aligned}$$

1.2 Random Hyperplan Rounding

Given a solution $\{v_u | \ \forall u \in V\}$ to the vector program of max cut with value OPT_v, we round our solution as follows:

- Pick a random vector $r = (r_1, r_2, \cdots, r_n)$, s.t $r_i \in \mathcal{N}(0,1)$.
- For all $u \in V$: $\begin{cases} r \cdot v_u \geq 0 & \Rightarrow u \to S \\ r \cdot v_u < 0 & \Rightarrow u \to \bar{S} \end{cases}$.

This rounding procedure is called **random hyperplane rounding**.

Theorem 5 *There exist a polynomial time algorithm that achieves a 0.878-approximation of the maximum cut with high probability.*

To prove Theorem 5, we will start by stating the following facts:

Fact 1: The normalized vector $\frac{r}{\|r\|}$ is uniformly distributed on $\mathcal{S}_n$, the n-dimensional unit sphere. This is because the distribution of r only depends on $\|r\|$.

Fact 2: The projections of r on to unit vectors e_1 and e_2 are independent iff e_1 and e_2 are orthogonal. Proof: Let's denote r_1 and r_2 the projections of r onto e_1 and e_2, respectively. The projections of a Gaussian random vector are also Gaussian random vectors, so it's sufficient to have $E[r_1 r_2] = 0$ for r_1 and r_2 to be independent. Since $E[r_1 r_2] = E[(e_1^T r)(r^T e_2)] = e_1^T E[rr^T] e_2 = e_1^T e_2$, Fact 2 follows directly.

Corollary 6 *Let r' be the projection of r onto a 2-dimensional plane, then $\frac{r'}{\|r'\|}$ is uniformly distributed on a unit circle in the plane.*

Lemma 7 *The probability that edge (i,j) is cut is $\frac{\arccos(v_i \cdot v_j)}{\pi} = \frac{\theta_{ij}}{\pi}$, where θ_{ij} is the angle between vectors v_i and v_j.*

Proof Project vector r onto the plane containing v_i and v_j. It is easy to see that edge (i,j) is cut iff r falls within the area formed by the vectors perpendicular to v_i and v_j, which has area equal to $2\theta_{ij}/(2\pi)$. ■

Lemma 8 *For $x \in [-1,1]$, one can show that: $\frac{\arccos(x)}{\pi} \geq 0.878\left(\frac{1-x}{2}\right)$.*

Let W be a random variable denoting the weight of the cut we obtain from solving the vector program for max cut and then applying the random hyperplane rounding. Then:

$$\begin{aligned} E[W] &= E[\sum_{(i,j)\in E} \Pr(\text{edge } (i,j) \text{ is cut})] \\ &= \sum_{(i,j)\in E} \frac{\theta_{ij}}{\pi} && \text{(by lemma 7)} \\ &= \sum_{(i,j)\in E} \frac{\arccos(v_i \cdot v_j)}{\pi} \\ &\geq 0.878 \ \frac{1 - (v_i \cdot v_j)}{2} && \text{(by lemma 8)} \\ &= 0.878 \ \mathrm{OPT}_v \\ &\geq 0.878 \ \mathrm{OPT}_{\text{max-cut}} \end{aligned}$$

Given this expected value, one can show the existence of an algorithm that achieves a 0.878-approximation of the maximum cut in polynomial time, with high probability. This concludes the proof of Theorem 5.

Finally, we give an example to show that this approximation factor is almost tight. Consider a 5-cycle graph. $\mathrm{OPT}_{\text{max-cut}} = 4$, while the optimal solution for the SDP is placing the 5 vector in a 2-dimensional plane with an angle $\frac{2\pi}{5}$ between each two vectors. The approximation factor achieved in this case is 0.884.

1.3 Correlation Clustering

Given an undirected graph $G = (V, E)$, we assign for each edge $(i, j) \in E$ the weights W_{ij}^+ and W_{ij}^- to denote how similar or different the endpoints of this edge are, respectively. (In our analysis we'll assume each edge have only one of these two kind of weights, but the approximation algorithm will still work in general.) The goal of the correlation clustering problem is to find a partition of the graph into clusters of similar vertices. In other words, we aim to maximize the following objective function:

$$\max \sum_{i,j \text{ are in the same cluster}} W_{ij}^+ + \sum_{i,j \text{ are in different clusters}} W_{ij}^-.$$

We denote the optimal value by OPT_{cc}.

The correlation clustering problem also admits a simple $\frac{1}{2}$-approximation algorithm by picking the best of the following two procedures:

1. Form one cluster: $S = V$.
2. Set each vertex to be in its own cluster.

Note that if the total sum of W_{ij}^+ in the graph is greater than the total sum of W_{ij}^-, choice 1 will guarantee at least half OPT_{cc}, otherwise choice 2 will.

The correlation clustering problem admits an exact formulation that can be relaxed to a vector program:

$$\begin{aligned} &\max \sum_{(i,j)\in E} \left(W_{ij}^+(v_i \cdot v_j) + W_{ij}^-(1 - v_i \cdot v_j)\right) \\ &\text{s.t } v_i \in \{e_1, e_2, \cdots, e_n\} \ \ \forall i \in V \end{aligned} \quad \Longrightarrow \quad \begin{aligned} &\max \sum_{(i,j)\in E} \left(W_{ij}^+(v_i \cdot v_j) + W_{ij}^-(1 - v_i \cdot v_j)\right) \\ &\text{s.t } v_i \cdot v_i = 1 \\ &\quad v_i \cdot v_j \geq 0 \\ &\quad v_i \in \mathbb{R}^n \end{aligned}$$

where e_k denotes the unit vector with the kth entry set to one. In the exact formulation, v_i is set to e_k if vertex i belongs to cluster k.

1.4 Rounding Algorithm for Correlation Clustering

Given a solution $\{v_u | \ \forall u \in V\}$ to the vector program of correlation clustering with value OPT_v, we round our solution as follows:
- Pick two random vectors r_1 and r_2 s.t each entry is drawn from the standard distribution $\mathcal{N}(0, 1)$.
- Form 4 clusters as follows:

$$\begin{aligned} R_1 &= \{i \in V : \ r_1 \cdot v_i \geq 0 \text{ and } r_2 \cdot v_i \geq 0\} \\ R_2 &= \{i \in V : \ r_1 \cdot v_i \geq 0 \text{ and } r_2 \cdot v_i \leq 0\} \\ R_3 &= \{i \in V : \ r_1 \cdot v_i \leq 0 \text{ and } r_2 \cdot v_i \geq 0\} \\ R_4 &= \{i \in V : \ r_1 \cdot v_i \leq 0 \text{ and } r_2 \cdot v_i \leq 0\}. \end{aligned}$$

Theorem 9 *There exists a polynomial time algorithm that achieves a $\frac{3}{4}$-approximation of the correlation clustering problem with high probability.*

Proof We start with a useful lemma:

Lemma 10 *For $x \in [0, 1]$, one can show that $\frac{(1-\frac{a\cos(x)}{\pi})^2}{x} \geq 0.75$ and $\frac{1-(1-\frac{a\cos(x)}{\pi})^2}{1-x} \geq 0.75$.*

Let x_{ij} be a random variable that takes the value one if i and j are in the same cluster. Note that the probability that v_i and v_j are not separated by either r_1 or r_2, is $(1 - \frac{\arccos(v_i \cdot v_j)}{\pi})^2$. Thus,

$$E[x_{ij}] = (1 - \frac{\arccos(v_i \cdot v_j)}{\pi})^2$$

Let W be a random variable denoting the weight of the clustering we obtain from solving the vector program of correlation clustering and then applying the random 2-hyperplane rounding.

$$\begin{aligned}
E[W] &= E[\sum_{(i,j)\in E} W_{ij}^+ x_{ij} + W_{ij}^-(1 - x_{ij})] \\
&= \sum_{(i,j)\in E} W_{ij}^+(1 - \frac{\arccos(v_i \cdot v_j)}{\pi})^2 + W_{ij}^-(1 - (1 - \frac{\arccos(v_i \cdot v_j)}{\pi})^2) \\
&\geq 0.75 \sum_{(i,j)\in E} W_{ij}^+(v_i \cdot v_j) + W_{ij}^-(1 - v_i \cdot v_j) \qquad \text{(by lemma 10)} \\
&= 0.75 \text{ OPT}_v \\
&\geq 0.75 \text{ OPT}_{cc}.
\end{aligned}$$

Given this expected value, one can show the existence of an algorithm that achieves a 0.75-approximation for the correlation clustering problem in polynomial time, with high probability. ■