Advanced Algorithms - Lecture December 22, 2005

Lecture 8

Lecturer: Yair Bartal *Scribe: Or Sattath*

1 Semi Definite Programming

The main idea in semi definite programming is the same as LP - we want to maximize/minimize a function, but here, we let the constraints be non-linear.

Example 1 *MAX-CUT let* $G = (V, E, W), W : E \to \mathbf{R}^+$*, a weighted graph. We would like to find a partition of* V *into 2 groups -* $S, \overline{S}(\overline{S} = V \backslash S)$ *s.t.* $C(S)$ *(defined bellow) will be maximized.*

Definition 1 (CUT) $\delta(S) = \{(u, v) \in e \mid u \in S, v \in \overline{S}\}$
$C(S) = \sum_{e \in \delta(S)} W(e)$

In the first recitation we saw a 1/2 approximation.
This lesson we'll see a better approximation by Goemans and Williamnson.

1.1 Quadratic Program

Remark We know that Quadratic Programing is NP hard. For each vertex u_i we will define a variable $y_i \in \{-1, 1\}$ in the following way:
$y_i = 1 \Rightarrow u_i \in S; y_i = -1 \Rightarrow u_i \in \overline{S}$

We got that:
$\delta(S) = \{(u_i, u_j) \mid y_i * y_j = -1\}$.
Or alternatively:
$(u_i, u_j) \in \delta(S) \iff y_i * y_j = -1$
Therefore: $\frac{1-y_i*y_j}{2} = 1$ if $(u_i, u_j) \in \delta(S)$, otherwise 0.
The Quadratic Program will be:
$Minimize_y : \ C(S) = sum_{1 \le i < j \le n} w_{i,j} * \frac{1-y_i*y_j}{2}$ Subject to:

- $y_i^2 = 1$
- $y_i \in \mathbf{R}$

Where $w_{i,j} = 1\ if\ (u_i, u_j) \in E$, 0 otherwise.
The quadratic constraint on y causes that $y_i \in \{-1, 1\}$, which is what we want.

1.2 Vector Program

We'll change this program to another type of a program: Vector Program (VP). For every vertex, u_i we'll define a vector $v_i = (a, b)$. The vector program will be:
$maximize_v : \sum_{1 \le i < j \le n} w_{i,j} * \frac{1 - <v_i, v_j>}{2}$ Subject to:

- $\forall i \ \ < v_i, v_i >= 1.$
- $\forall i \quad v_i \in \mathbf{R}^2.$

Because we can always make $v_i = (y_i, 0)$ we get $OPT_{VP} \ge OPT_{QP} = OPT(I)$. Notice that the Vector Program doesn't induces the groups $S, \overline{S}$, so we don't have a solution for the original problem.

Generally, Vector Programming is defined by: $min/max_{v_1,\ldots,v_n} \sum C_{i,j} * < v_i, v_j >$ Subject to:

- $\forall k \sum_{i,j} a_{i,j,k} < v_i, v_j >= b_k$
- $\forall i \quad v_i \in \mathbf{R}^m$

(notice that n can be different than m).

1.3 Semi Definite Programming

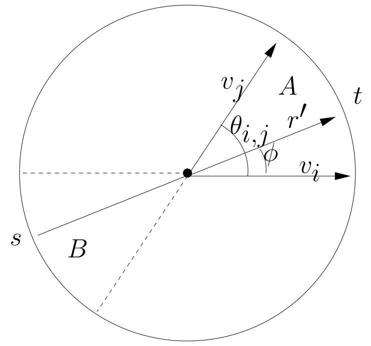
Another type of programming is Semi Definite Programming(SDP). A matrix $X^{n \times n}$ is Positive Semi Definite(PSD) if there exist a matrix $Y^{n \times n}$ s.t. $X = Y^t Y$. If we'll write $Y = ((v_1)(v_2)...(v_n))$, then $X_{i,j} =< v_i, v_j >$ We are trying to solve:
$maximize_{v_1,\ldots,v_n} : \sum_{i<j} W_{i,j} \frac{1 - <v_i, v_j>}{2}$ Subject to

- $\forall i < v_i, v_i >= 1.$

Notice that $< v_i, v_j >= cos(\theta_{i,j})$. Therefore the target function becomes:
$maximize_{v_1,\ldots,v_n} \sum_{i<j} W_{i,j} \frac{1 - cos(\theta_{i,j})}{2}$

It's worth mentioning that unlike VP, SDP has a polyomial solution. But the problem is that if we have the optimal solution for the SDP, then how can we go back to a solution which specifically says whether the vertex is in S or $\overline{S}$. We'll want that if the $\theta_{i,j}$ is big, then the chance that they will be in different sides of the cut will be big. The next section is about a solution that is going in that direction and called randomized rounding.

Figure 1:



1.4 Randomized Rounding

1. Solve the SDP. The solution is $v_1, ..., v_n$.
2. Choose randomly a vector r from the uniform distribution of the unity sphere in $\mathbf{R}^n$.
3. The solution for the max cut will be:

$$S = \{u_i \in V \mid < v_i, r > \geq 0\}$$

Remark Remember that v_i is the vector that represents the vertex u_i.

Lemma 2 $Pr[(u_i, u_j) \in \delta(S)] = \frac{\theta_{i,j}}{\pi}$

Proof: From the defintion of randomized rounding we get:
$Pr[(u_i, u_j) \in \delta(S)] = Pr[v_i, v_j$ in the opposite sides of the surface of $r] =$
$= Pr[(< v_i, r >> 0 \wedge < v_j, r >< 0) \vee (< v_i, r >< 0 \wedge < v_j, r >> 0)]$
Now, let r' be the normalized projection of r on the plane that is created by the span v_i, v_j. The angle ϕ is the angle between r' and v_i. ϕ is distributed uniformly on $[0, 2\pi]$ (because r was selected uniformly). See figure 1.
$Pr[v_i, v_j$ in the opposite sides of the surface of $r] = Pr[\phi \in A \cup B] = 2 * \frac{\theta_{i,j}}{2\pi} = \frac{\theta_{i,j}}{\pi}$. ■

Therefore: $E[c(S)] = \sum_{i<j} w_{i,j} \frac{\theta}{\pi}$

We would like to find α such that:

$$E[c(S)] \geq \alpha OPT_{SDP}(I)$$

The reason that we want that is because we know that $OPT_{SDP}(I) \geq OPT(I)$, and therefore we'll have $E[c(S)] \geq \alpha OPT(I)$ - that is an α approximation.

Further development shows that: $\alpha = \min_\theta \frac{2\pi}{\pi(1-cos(\theta))} > 0.878...$
Recently it was shown that this approximation is tight, assuming some computational assumptions (for details on the assupmtions - see http://weblog.fortnow.com/2005/06/unique-games-conjecture.html).

Figure 2:

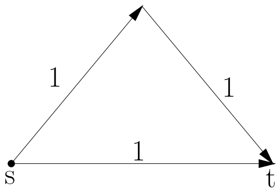
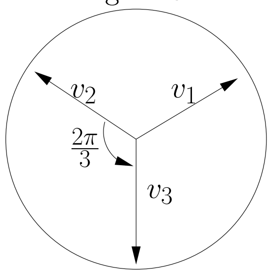


Figure 3:



1.5 Choosing The Random Vector

We'll now show how one can implement part 2 of the algorithm: choose randomly a vector r from the uniform distribution of the unity sphere in $\mathbf{R}^n$.

Define $\forall i \in 1,..,n$, $X_i \sim N(0,1)$ I.I.D. (normal distribution with expectation of 0, and variance of 1). That can be done from a uniform distribution using the Box-Muller transformation, see http://en.wikipedia.org/wiki/Box-Muller_transform for details. Let $X = (X_1, ..., X_n)$, and $r = \frac{X}{\|X\|_2}$.

$$f_r(y_1, ..., y_n) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}} \exp(\frac{-y_i^2}{2}) = \frac{1}{(2\pi)^{n/2}} \exp(-\sum_{i=1}^{n} \frac{y_i^2}{2}) = constant$$

Example 2 *Let's look at the flow problem of the triangle graph, with weights of 1 on each edge - see figure 2. Obviously, $OPT(I)$ is 2. The Vector Program in two dimensions (notice that the SDP will be in 3 dimensions), will be as figure 3. So we get:* $OPT_{VP}(I) = \sum_{i=1}^{3} w(e_i)(\frac{1-<v_i,v_j>}{2} = 3 * \frac{1-cos(2\pi/3)}{2} = \frac{9}{4}$
$\frac{OPT(I)}{OPT_{VP}(I)} = \frac{8}{9}$