Lecture 8 :- Algorithmic applications of convex programs. Rounding.

- There are fast (both in theory & practice) algorithms for linear programming

$$\begin{aligned} & c^T x \\ & Ax = b \\ & x \geq 0. \end{aligned} \qquad x \in \mathbb{R}^n$$

Thm:- (Kachian / Karmarkar). Given $\epsilon > 0$, One can compute in time poly (bitsize (A, b, c), $\log(1/\epsilon)$) a ϵ- approximation to the optimal value.

Combinations of these methods (Ellipsoid / Interior point) + Simplex method work well in practice as well.

Questions:-

(1) What other classes of optimization admit such algorithms?

(2) Can these methods be used for combinatorial optimization?

Q 1:-

→ What about convex fns. over convex sets? (Not always possible in general, but interesting cases do admit such algorithms).

An interesting example:- Semi- definite programming

Positive semi-definite matrix:- A $n \times n$ real

matrix M is called positive semi-definite if

(i) M is symmetric

(ii) $v^T M v \geq 0$ for every $v \in \mathbb{R}^n$.

This is denoted as $M \succcurlyeq 0$. The notation $A \succcurlyeq B$ means that $A - B$ is PSD.

Observation:- $\{M \in \mathbb{R}^{n \times n} \mid M \succcurlyeq 0\}$ is a convex cone.

$\because v^T(\lambda M + \mu N)v = \lambda v^T M v + \mu v^T N v \geq 0$

when $\lambda, \mu \geq 0$ and $M, N \succcurlyeq 0$)

Ex:- If $A, B \succcurlyeq 0$ is $AB \succcurlyeq 0$??

Matrix inner product. $A \bullet B := Tr(AB^T)$

$$:= \sum_{\substack{1 \leq i \leq n \\ 1 \leq j \leq n}} A_{ij} B_{ij}$$

when A, B are $n \times n$ matrices.

Semi-definite program:

$$\min_X \; C \bullet X$$

$$A_i \bullet X = b_i \quad 1 \leq i \leq k$$

$$X \succcurlyeq 0$$

where $X, C \in \mathbb{R}^{n \times n}$, $A_i \in \mathbb{R}^{n \times n}$, $b_i \in \mathbb{R}$.

Thm:- Can be solved to ε-accuracy in time $poly\left(bitsize\left(C, \{b_i\}_{i=1}^k, \{A_i\}_{i=1}^k\right), \log(1/\varepsilon)\right)$.

Q2 :- Can these be used for combinatorial optimization?

MAX-CUT :- Input :- A weighted undirected graph $G = (V, E, W)$, w_e is the weight of edge e.

Output: $S \subseteq V$ s.t.

$$Cut(S) := \sum_{e : e \cap S = 1} w_e$$

is as large as possible.

[NP-hard]

Ex: Given an algorithm (possibly randomized) s.t. it produces and $R \subseteq V$ s.t.

$$Cut(R) \geq \frac{1}{2} \max_{S \subseteq V} Cut(S).$$

Just independly choosing each vertex to be in R with prob. 1/2 gives

$$E[Cut(R)] = \frac{1}{2} \sum_{e \in E} w_e \geq \frac{1}{2} \max_{S \subseteq V} Cut(S).$$

(Can also be easily derandomized)

So the approximation ratio achieved by this algorithm is at least 1/2.

Goemans - Williamson :- Let us try to 'embed' the vertices as vectors (or points) in $\mathbb{R}^d$ for some large d, in such a way that cuts defined by halfplanes (all 'vertices' on side of a hyperplane are in S) are good.

Suppose our embedding puts each vertex as a unit vector, on a line.

$$\max \frac{1}{2} \sum_{e=\{i,j\} \in E} w_{\{i,j\}} (1 - v_i^T v_j)$$

"unit vector" $\|v_i\|^2 = 1$ $1 \le i \le n$ (n is the number of vertices)

$$\text{rank}\left(\text{span}\left(\{v_i\}_{i=1}^n\right)\right) = 1$$

But this is an exact translation, and so this is still NP-hard.

Goemans-Williamson relaxation :- Drop the rank constraint.

$$\max \frac{1}{2} \sum_{e=\{i,j\} \in E} w_{\{i,j\}} (1 - v_i^T v_j)$$

$$\|v_i\|^2 = 1.$$

Then given such $\{v_i\}_{i=1}^n$, pick a 'random hyperplane' through the origin, and define S to be the set of those vectors which lie on one side of the hyperplane.

Q1 :- Where are we going to get the v_i's?

Q2 :- How good will the approximation ratio of this ratio be?

For q 1 :- $A \in \mathbb{R}^{n \times n}$ is positive semi-definite if and only if there exist $v_1, \ldots, v_n \in \mathbb{R}^d$ s.t. $A_{ij} = v_i^T v_j$.

Thus The program will become.

$$\max \frac{1}{2} \sum_{\substack{e \in E \\ e = \{i,j\}}} w_e (1 - A_{ij})$$

$$A_{ii} = 1 \qquad 1 \leq i \leq n.$$

$$A \succeq 0.$$

Fact 2 :- Moving from A to $\{v_1 \ldots v_n\}$ can be done via eg. the Cholesky decomposition.
(in particular, $\{v_i\}_{i=1}^n$ can be chosen to be in $\mathbb{R}^n$.)

Upshot :- We will assume that in time poly(n) we have vectors $\{v_i\}_{i=1}^n \in \mathbb{R}^n$ s.t. $\|v_i\|_2 = 1, 1 \leq i \leq n$ and

$$\text{MAX-CUT}(G) \leq M = \frac{1}{2} \sum_{\substack{e \in E \\ e = \{i,j\}}} w_e (1 - v_i^T v_j)$$

Q:- How to go from $\{v_i\}_{i=1}^n$ to an actual cut without a large loss in the objective value?