

Lecture Outline:

- Goemans-Williamson Algorithm for Max-Cut

In this lecture we will use semi-definite programming (SDP) approach as a black box to solve the Max-Cut problem. The SDP machinery used to solve this problem has developed the whole new class of approach to solve a bunch of problems. Both the algorithm and analysis by Goemans and Williamson are very simple and short.

The approach would be to write a Strict Quadratic Program for Max-Cut and convert it to Vector Program. And then solve the Vector Program using SDP solver. In fact most of the machinery that has been developed using SDP in approximation algorithms uses vector programs.

1 Introduction

In this section we introduce and review some notions related to various mathematical programs.

- Semi-Definite Programs where we have positive semi-definite matrix constraints.
- Strict Quadratic Programs are quadratic programs (i.e. the constraints and the objective function could be quadratic) where sum of power of the variables in each term is either 0 or 2.
- Vector Programs (refer to last lecture)

Relationship of Vector Programs with Strict Quadratic Programs: In Vector Programs, in each constraint and objective function, each term is of the form constant c or $v_i \cdot v_j$ where v_i and v_j are vectors and operation is dot product. We can take a Strict Quadratic Program, replace the scalars (or variables) by vectors to get Vector Program.

Machinery to solve a problem: Take the problem, write it as Strict Quadratic Program and then replace the variables by vectors to get Vectors. And we know the fact that semi-definite programming can solve Vector Programs.

2 Max-Cut Problem

Problem: Given a graph $G = (V, E)$ and $w : E \rightarrow \mathbb{Z}^+$. Determine cut (S, \bar{S}) where $w(S, \bar{S})$ is maximized.

2.1 Algorithms

There can be various approaches to solve Max-Cut Problem:

(i) Local Greedy Algorithm:

We discussed a $1/2$ -approximation algorithm in the last class. In this approach, we start with a cut and greedily try to improve the size of the cut.

(ii) "Naive" Randomized Algorithm:

Place v in S independently with probability $1/2$.

$$\Pr[(u, v) \text{ crosses the cut}] = 1/2$$

By linearity of expectation,

$$E[w(S, \bar{S})] = \frac{\sum_{i>j} w_{ij}}{2}$$

$$\begin{aligned} \text{Approximation Ratio} &= \frac{E[w(S, \bar{S})]}{OPT} \\ &= \frac{\sum_{i>j} w_{ij}}{2} \cdot \frac{1}{OPT} \\ &= \frac{\sum_{i>j} w_{ij}}{2 \cdot OPT} \\ &\geq \frac{2}{\sum_{i>j} w_{ij}} \\ &= 1/2 \end{aligned} \tag{1}$$

So here we have the trivial algorithm, in which we are not even looking at the graph. To get (1), we bounded OPT i.e. OPT is at most $\sum_{i>j} w_{ij}$. Now, if we can provide a better bound for OPT , we will get a better approximation ratio. In this analysis the bound for OPT is naive. One attempt to get a good approximation ratio is to improve this bound. So, there is a possibility that this algorithm might be a better algorithm, but our analysis is not perfect. A question to address: Is the algorithm really an $1/2$ -approximation? One can easily come up with an example where this algorithm (or minor variant) is indeed an $1/2$ -approximation.

(iii) Goemans Williamson Method:

It seems very hard to write an LP for Max-Cut problem. One way to write a LP can be the following:

We have variable $x_{S, \bar{S}}$ for cut (S, \bar{S}) .

$$\begin{aligned} &\max x_{S, \bar{S}} \cdot w_{S, \bar{S}} \\ &\text{such that } \sum x_{S, \bar{S}} = 1 \end{aligned}$$

It is not a very useful LP. This LP is just an enumeration of all the cuts. It doesn't capture any structure of the graph. And also we don't know how to solve it in polynomial time.

We should move away from restricting our notion to LPs because there can be more problems which can be written in more general programs than LPs and can be solved in polynomial time. So we should not constraint ourself to LPs.

One another approach might be to have a variable x_e for each edge e such that $x_e = 1$ if it crosses the cut and 0 otherwise. Objective function would be to maximize $\sum_e w_e \cdot x_e$. What are our constraints? These edges should define a cut.

Now instead of assigning variables for cuts or edges, we will assign a variable to each vertex and write a strict quadratic program as follows:

Assign variables to vertices (say y_i for vertex i).

$$y_i = \begin{cases} +1 & : \text{ if } i \in S \\ -1 & : \text{ if } i \notin S \end{cases}$$

Strict Quadratic Program [Q][MC]:

$$\begin{aligned} \max \sum_{i>j} w_{ij} \frac{1 - y_i \cdot y_j}{2} \\ \text{such that } y_i \in \{-1, 1\} \end{aligned}$$

Note that if $(i, j) \in S$, then $\frac{1 - y_i \cdot y_j}{2} = 1$. Also, we can write $y_i \in \{-1, 1\}$ as $y_i^2 = 1$. So we have written a Strict Quadratic Program (SQP) for our problem.

Goemans and Williamson relaxed the notion of scalars in [Q] by vectors and wrote the following vector program.

Vector Program [P]:

$$\begin{aligned} \max \frac{1}{2} \sum_{i>j} w_{ij} \cdot (1 - v_i \cdot v_j) \\ \text{such that } v_i \cdot v_i = 1 \end{aligned}$$

Since there are n vertices, we have n vectors. W.l.o.g we can assume these vectors to be n -dimensional i.e. $v_i \in R^n$. Note that the constraint $v_i \cdot v_i = 1$ says that all these n -dimensional vectors are on an n -dimensional sphere. Now we know how to solve this Vector Program in polynomial time using SDP.

$$Z_{MC}^* = Z_Q^* = \text{Optimal Max Cut}$$

Z_P^* can be found by SDP to an additive error of ϵ .

So the vector program [P] is solved by SDP solver and we get the solution, say (v_1, v_2, \dots, v_n) . Now we need to round this solution to a cut.

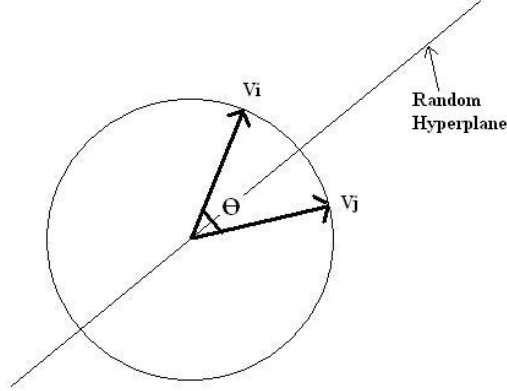
If all these vectors are in 1-dimension, then if the value is $+1$, we will put in S and if -1 then we put it in \bar{S} . So, in this case we get cut of value Z_P^* . Note that $Z_P^* \geq Z_{MC}^*$, since any solution to [MC] is a solution to [P]. Note that this is the place where we are getting a better upper-bound than just taking the total weight of all the edges (which we did in (1)).

ALGORITHM

1. Solve vector program [P] and get optimal solution (v_1, v_2, \dots, v_n)
2. Pick random hyperplane H (or a random vector r)
3. If $v_i \cdot r > 0$, put vertex i in S else put in \bar{S} .

Analysis:

Let θ be the angle between v_i and v_j .



$$\begin{aligned} \Pr[(i, j) \text{ is in cut}] &= \frac{\theta}{\pi} \\ &= \frac{\cos^{-1}(v_i \cdot v_j)}{\pi} \end{aligned} \tag{2}$$

Now, if W be value of the cut

$$\begin{aligned} E[W] &= \frac{1}{\pi} \sum w_{ij} \cos^{-1}(v_i \cdot v_j) \\ &\geq \alpha \cdot Z_P^* \end{aligned} \tag{3}$$

Note that,

$$Z_P^* = \frac{1}{2} \sum w_{ij} (1 - v_i \cdot v_j)$$

The inequality of (3) holds, because, we can prove the inequality term by term, as follows.

$$\frac{\cos^{-1}(v_i \cdot v_j)}{\pi} \geq \alpha \cdot \frac{1}{2} (1 - v_i \cdot v_j)$$

$$\frac{\theta}{\pi} \geq \alpha \cdot \frac{1}{2} (1 - \cos \theta)$$

It can be shown that for $\alpha = 0.878$, the above inequality always holds. More precisely, it can be shown by simple trigonometry that 0.878 is minimum ratio $\frac{2\theta}{\pi(1-\cos\theta)}$

It is shown that it is NP-hard to approximate Max-Cut better than 0.94. There is a belief that 0.878 may be the best we can approximate within polynomial time. This is a randomized algorithm and its derandomization has also been done.