

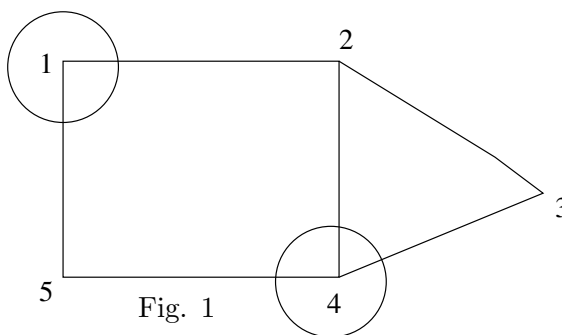
Semidefinite Programming Relaxation

1 Introduction

Today, we talk the use of **Semidefinite Programming** in the design of **approximation algorithms**. This technique was invented by Goemans and Williamson in 1995. It is because of this technique both of them won the Fulkerson Prize in 2000. I'll talk about this technique in the context of the MAX CUT Problem. First let me give the definition of MAX CUT.

MAX CUT: Given an undirected graph G , find a subset $S \subset V$ that maximizes the number of cross edges between S and its complement \bar{S} . Namely, $\text{Max}_{S \subset V} |\text{Cut}(S, \bar{S})|$.

For example,



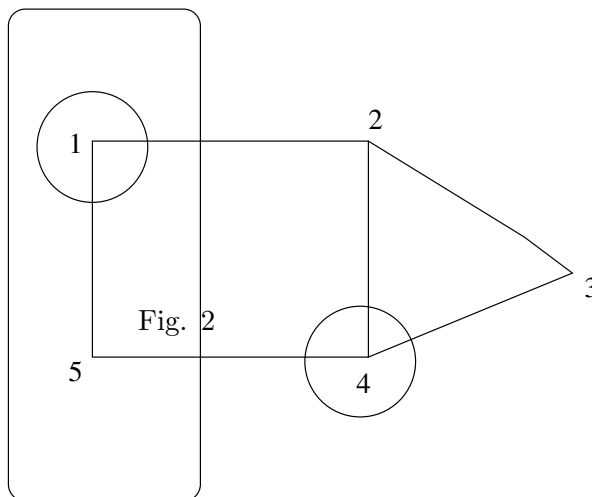
As the structure of this graph is very simple, we can easily get that

$$\text{Max}_{S \subset V} |\text{Cut}(S, \bar{S})| = 5.$$

It seems easy to deal with MAX CUT of this graph, but generally MAX CUT is a NP-hard problem. It is unlikely that there exists an efficient algorithm for it. A typical approach to solving such a problem is to find an **approximation algorithm**.

Naturally, we can give a greedy algorithm for MAX CUT.

- Start with any arbitrary cut.
- If some node has more neighbors on its side than on the other side, then move it to the other side.



Take our previous example.

- We start with $S = \{1, 5\}$.
- Then we change S to $\{1, 4, 5\}$.
- Finally, we get $S = \{1, 4\}$.

It is easy to see that this algorithm is a $\frac{1}{2}$ -approximation algorithm, and for a long time, the best known approximation algorithm for MAX-CUT is $\frac{1}{2}$ -approximation algorithm.

However, by using Semidefinite Programming, Geomans and Williamson[1] give an α -approximation algorithm for MAX CUT, where $\alpha > 0.87856$. This is a great breakthrough in approximating MAX CUT and it also leads people to apply semidefinite programming into the design of approximation algorithms.

It sounds magic? Is every one willing to see how the semidefinite programming works?

2 Reformulation of MAX CUT

First let us reformulate the MAX-CUT as an **Integer Quadratic Programming**:

$$\begin{aligned} \text{Max} \quad & \sum_{(i,j) \in E} \frac{(1-y_i y_j)}{2} \\ \text{subject to} \quad & y_i \in \{-1, 1\} \end{aligned}$$

Note that this is equivalent to the MAX CUT problem. Let $S = \{i | y_i = 1\}$, then

$$\begin{aligned} (i, j) \in C(S, \bar{S}) & \iff \frac{(1-y_i y_j)}{2} = 1, \\ (i, j) \notin C(S, \bar{S}) & \iff \frac{(1-y_i y_j)}{2} = 0. \end{aligned}$$

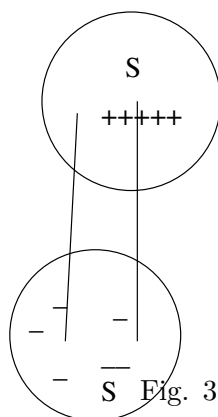


Fig. 3

In other words, only the cross edges between S and its complement \bar{S} contribute to the sum.

It is known that integer programming is NP-hard. So we should avoid solving it. Then what shall we do? Let me remind you of what we have done with the Minimum Vertex Cover problem before.

Minimum Vertex Cover

$$\begin{aligned}
 &\min \quad \sum_{i=1}^n x_i \\
 &\text{Subject to} \quad x_i + x_j \geq 1, (i, j) \in E \\
 &\quad \quad \quad x_i \in \{0, 1\} \\
 &\text{Relax} \quad \Downarrow \text{the above problem to} \\
 &\min \quad \sum_{i=1}^n x_i \\
 &\text{Subject to} \quad x_i + x_j \geq 1, (i, j) \in E \\
 &\quad \quad \quad 0 \leq x_i \leq 1
 \end{aligned}$$

We relax an Integer Programming problem(IP) to a Linear Programming(LP), which enable us to solve an easier problem. Then we "round" the solution to the LP back to the IP. To explain this clearly, suppose x^* is the optimal solution to the above LP. Then we let $S = \{i | x_i^* \geq \frac{1}{2}\}$. Convince yourself that S is a vertex cover of the graph and thus we can use it to approximate the minimum vertex cover of the graph.

Now, we are facing an Integer Quadratic problem. What kinds of relaxation will you apply to

it?

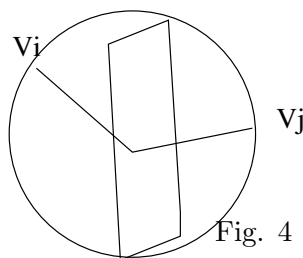
Relax the Integer Quadratic problem to a Semidefinite Programming

$$\begin{aligned} \text{Max} \quad & \sum_{(i,j) \in E} \frac{1 - \langle v_i, v_j \rangle}{2} \\ \text{subject to} \quad & v_i \in R^n \\ & \langle v_i, v_i \rangle = 1 \quad (\langle, \rangle \text{ denote the inner product}). \end{aligned}$$

A semidefinite programming can be solved in polynomial time. However, the solution to the semidefinite programming **probably** is **not** a solution to the original Integer Quadratic problem. This is also the **difficulty** of the technique of **relaxation** lies in. We have to "round" the solution to the relaxed problem back to the solution to the original problem.

3 How to Round the Solution back?

Now let me show you how Goemans and Williamson round the solution to this semidefinite programming back. Any solution to the semidefinite programming gives us an embedding of the vertices of G into S_n . Note that, our **objective function** is $\max \sum_{(i,j) \in E} \frac{1 - \langle v_i, v_j \rangle}{2}$. To make it large, we'd better make the angle between two connected adjacent vertices large. Intuitively, if $(i, j) \in E$, then we want $\frac{1 - \langle v_i, v_j \rangle}{2}$ to contribute as much as possible to the sum. Thus we want to make $\langle v_i, v_j \rangle$ as small as possible. That's to say, we want the **angle** between v_i and v_j as large as possible.



Now we cut the unit sphere by a hyperplane H passing the origine randomly. The two vertices connected will be separated by this hyperplane with high probability. As long as the hyperplane inside this angle, v_i, v_j are separated. Thus if we take all the vertices on one side of the hyperplane as S , we find a cut $C(S, \bar{S})$ of G , which enclose the edge between v_i, v_j with high probability.

Now we come to Goeman and Williamson's **Randomized Max-Cut Algorithm**:

- Solve the Semidefinite Programming and let v_1, \dots, v_n be the optimal solution.
- Pick a random vector \vec{v} in S^n .

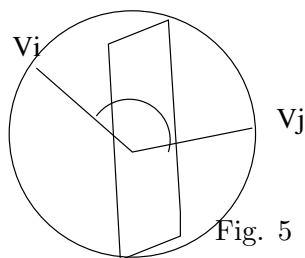
- Let $S = \{i \mid \langle v_i, v \rangle \geq 0\}$.

4 Analysis of the Randomized Algorithm

Till now, the thing left is to analyze the performance of the randomized algorithm.

Lemma 4.1 $Pr[v_i \text{ and } v_j \text{ are separated by hyperplane } H] = \frac{1}{\pi} \arccos(\langle v_i, v_j \rangle)$.

Proof. Let me show you the proof by Fig.5. Suppose this is our sphere, the angle between v_i and v_j is β . So v_i and v_j will be separated if and only if the hyperplane is inside this angle. Thus, $Pr[\text{sgn}(v_i \cdot r) \neq \text{sgn}(v_j \cdot r)] = \frac{2\beta}{2\pi} = \frac{1}{\pi} \arccos(\langle v_i, v_j \rangle)$.



■

Theorem 4.1 The expectation $E(|C(S, \bar{S})|)$ of the number of edges separated by hyperplane H is $\frac{1}{\pi} \sum_{(i,j) \in E} \arccos(\langle v_i, v_j \rangle)$.

Proof. For $(i, j) \in E$, Let

$$U_{ij} = \begin{cases} 1 & \text{if } v_i \text{ and } v_j \text{ are separated by } H \\ 0 & \text{otherwise.} \end{cases}$$

Thus

$$\begin{aligned} E(|C(S, \bar{S})|) &= \sum_{(i,j) \in E} E(U_{ij}) \\ &= \sum_{(i,j) \in E} P(U_{ij} = 1) \\ &= \frac{1}{\pi} \sum_{(i,j) \in E} \arccos(\langle v_i, v_j \rangle). \end{aligned}$$

■

Let $\alpha = \min_{0 \leq \theta \leq \pi} \frac{2}{\pi} \frac{\theta}{1 - \cos(\theta)}$. Using simple calculus, we have $\alpha > 0.87856$. Now we claim that

Theorem 4.2 *The randomized algorithm for MAX CUT is α -approximation algorithm.*

Proof. By the definition of α , we have

$$\begin{aligned} \alpha(1 - \cos \theta) &\leq \frac{2}{\pi}\theta, \quad \text{where } 0 \leq \theta \leq \pi \\ &\Downarrow \\ \alpha(1 - y) &\leq \frac{2}{\pi} \arccos y, \quad \text{where } -1 \leq y \leq 1 \end{aligned}$$

Thus,

$$\begin{aligned} E(|C(S, \bar{S})|) &= \frac{1}{\pi} \sum_{(i,j) \in E} \arccos(\langle v_i, v_j \rangle) \\ &\geq \sum_{(i,j) \in E} \frac{\alpha(1 - \langle v_i, v_j \rangle)}{2} \\ &= \alpha \sum_{(i,j) \in E} \frac{1 - \langle v_i, v_j \rangle}{2} \\ &\geq \alpha \max_{T \subset V} |C(T, \bar{T})| \end{aligned}$$

■

Thus our randomized approximation algorithm for MAX CUT is a α -algorithm, where $\alpha > 0.87856$.

References

- [1] Machel X. Goemans, David P. Williamson, Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming, *Journal of the ACM*, 42:1115-1145, 1995.
- [2] Uriel Feige, Randomized Rounding for Semidefinite Programs Variations on the MAX CUT Example, *Randomization, Approximation, and Combinatorial Optimization, Proceedings of Random-Approx'99*, Lecture notes in Computer Science 1671, Springer 189-196.
- [3] Avner Magen, *Linear Programming and Combinatorial Optimization (Lecture Notes)*, Computer Science, University of Toronto