CSC2411 - Linear Programming and Combinatorial Optimization[*] Lecture 12: Semidefinite Programming(SDP) Relaxation

Notes taken by Xinwei Gui

May 1, 2007

Summary: This lecture introduces the semidefinite programming(SDP) relaxation. We begin with the MAX-CUT problem, where we present some simple approximation algorithms first and then Goemans-Williamson Maxcut algorithm. This is a randomized algorithm that makes use of SDP relaxation. And it is proved to have an expected approximation factor of 0.878, which is much better than the other approaches. Next, we discuss the graph coloring problem, where we present Wigderson's Graph Coloring algorithm. Also, we introduce the vector programming approach for 3-colorable graphs there.

1 Introduction

The fact that Linear Programming can be solved in polynomial time and also has a rich geometric theory makes LP a powerful unifying concept in the theory of algorithms. Naturally, when we are looking for algorithms for an NP-hard combinatorial problem, one possible approach is to express the problem as a zero-one integer program, relax it to a linear program, solve the corresponding LP and round the solution back to the IP. In this way, we hope to get a good approximation factor with regard to the original integer problem. However, for some problems including MAX CUT, LP relaxation is not as tight as we expected. And it is under this situation that SDP relaxation arises as an alternative approach.

[*] Lecture Notes for a course given by Avner Magen, Dept. of Computer Sciecne, University of Toronto.

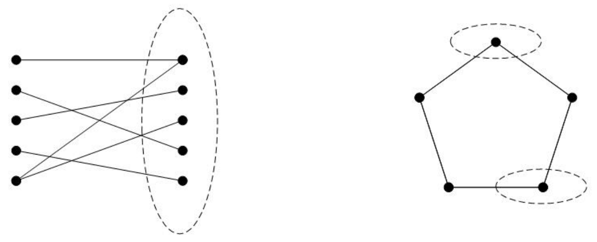
(a) G is a bipartite graph (b) $G = C_5$

Figure 1: instances of MAX CUT

2 MAX CUT

2.1 Definition

In the MAX CUT problem, given an undirected graph G, our goal is to find a subset $S \subset V(G)$ that maximizes the total weight of cross edges between S and its complement S^c. In the unweighted case, MAX CUT is defined formally as follows:

Input: undirected graph G

Output: a subset $S \subset V(G)$ so that $|cut(S)|$ is maximized, where $cut(S) = \{e \in E(G) | \, |e \cap S| = 1\}$.

Note that MAX CUT can be thought of a variant of the 2-coloring problem in which we try to maximize the number of edges consisting of two different colors instead.

Example 2.1. G is a complete graph K_n, where n is even. In this case, a set S of size k has cut-size equal to $k(n-k)$. So the size of the maximum cut is $(\frac{n}{2})^2$.

Example 2.2. G is a bipartite graph. Considering the bipartite graph in Figure 1(a). Obviously, to achieve the maximum number of cross edges, we should group the left vertices into one set and the right ones into the other. So the optimum solution is the total number of edges in graph G.

Example 2.3. G is an odd cycle C_n. Again, this instance of MAX CUT is trivial. An optimum solution takes every other vertex on the cycle, but has to accept two consecutive ones. Therefore, the maximum cut has $n-1$ edges. Take C_5 for example, the optimum solution is shown in Figure 1(b), in which one set contains two vertices and the other contains the rest three.

2.2 Simple Approximation Algorithms for MAX CUT

In [Hås97], Håstad showed that it is NP-hard to get an approximation ratio of 0.942 for the problem. But it is easy to get a $\frac{1}{2}$-approximation. To achieve that, we have the following algorithms:

Algorithm 2.4 (Greedy Algorithm with an Approximation Ratio of $\frac{1}{2}$)**.**

Input: undirected graph G

Output: a subset $S \subset V(G)$ with the maximum $|cut(S)|$

1. start with any $cut(S)$ of graph G
2. if we can improve the number of edges in the cut by moving a vertex from the cut to its complement or the other way around, we do so
3. repeat step 2 until no such a vertex exists

Claim 2.5. *The size of the cut in termination is greater than or equal to $\frac{1}{2}|OPT(G)|$, where $|OPT(G)|$ denotes the maximum number of edges for all possible cuts.*

Proof. The second step of the above algorithm is equivalent to:

we move a vertex to the other side iff there are more neighbors in its set than outside.

Each time we move a vertex, the size of the cut increases by at least 1. Since the size of the cut is upper bounded by $E(G)$, the algorithm must terminate at some time . And when it terminates, any vertex should satisfy that the number of edges outside is not less than that in its set. Thus, $|cut(S)| \geq \frac{1}{2}|E(G)| \geq \frac{1}{2}|OPT(G)|$.

Algorithm 2.6 (Randomized Algorithm with an Expected Approximation Ratio of $\frac{1}{2}$)**.**

Input: undirected graph G

Output: a subset $S \subset V(G)$ with the maximum $|cut(S)|$

1. start with $S = \phi$
2. for each vertex in the graph G, put it in S with a probability of $\frac{1}{2}$

Claim 2.7. *Regarding this randomized algorithm, the expected size of the cut in termination is greater than or equal to $\frac{1}{2}|OPT(G)|$, where $|OPT(G)|$ denotes the maximum number of edges for all possible cuts.*

Proof. Based on the algorithm, we know that the probability of a vertex being in S is $\frac{1}{2}$. Therefore, given an edge e, if we denote p_e as its probability of being a cross edge, $p_e = 2 \times (\frac{1}{2})^2 = \frac{1}{2}$. And consequently, $\mathbb{E}(|cut(S)|) = \frac{1}{2}|E(G)| \geq \frac{1}{2}|OPT(G)|$.

2.3 Using SDPs to Design Approximation Algorithms

Recall that the general form of a semidefinte program is: Find $n \times n$ PSD matrix X that satisfies $A_i \bullet X = b_i$, for $i = 1, 2, ..., m$ and minimizes $C \bullet X$. In other words, SDP consists of finding a set of n vectors $v_1, v_2, ..., v_n \in \mathbb{R}^n$ such that the inner products $\langle v_i, v_j \rangle$ satisfy some given linear constraints and we minimize some linear function of the inner products. In the following, we first formulate MAX CUT as an integer quadratic program (IQP), and then relax it to a vector program which is equivalent to Semidefinite Programming.

In IQP, we introduce the variables $x_i \in \{-1, 1\}$ for each vertex in graph G. $x_i = 1$ denotes that vertex $i \in S$ and $x_i = -1$ means vertex $i \in S^c$. Thus, we formulate MAX CUT as the following IQP:

$$\begin{gathered} max \sum_{(i,j)\in E(G)} \frac{1 - x_i x_j}{2} \\ x_i \in \{-1, 1\} \end{gathered}$$

Because it is NP-hard to solve the above IQP, we relax it to the following vector program:

$$\begin{gathered} max \sum_{(i,j)\in E(G)} \frac{1 - \langle v_i, v_j \rangle}{2} \\ \|v_i\|_n = 1 \\ v_i \in \mathbb{R}^n \end{gathered}$$

Note that the above vector program is indeed a relaxation of the IQP because its objective function reduces to $\sum_{i,j\in E(G)} \frac{1-x_i x_j}{2}$ in the case of vectors lying in a one-dimensional space.

Besides, the above vector program has a good geometric meaning. The vertices of graph G can be viewed as points in a unit sphere. And locally, if two vertices are connected by an edge, we are trying to make them far apart in the unit sphere. Regarding the objective function in the relaxed program, the inner product $\langle v_i, v_j \rangle$ is equal to -1 when two vertices i and j are of opposite polarities and equal to 1 when two vertices lies on the same point in the sphere. In other words, if two vertices are of opposite polarities, they contribute most to the objective function.

Obviously, the VP(or SDP) relaxation above is not exact. Consider the following example.

Example 2.8. *G is a cycle graph C_3. In this case, the optimum solution for the IQP is 2. However, the optimum solution for the relaxed SDP is $3 \times \frac{3}{4} = \frac{9}{4}$, as described in Figure 2, where the angle between each pair of vectors is $\frac{2}{3}\pi$. Note that this is indeed the optimum solution because $(v_1 + v_2 + v_3)^2 \geq 0$, from which we get $\langle v_1, v_2 \rangle + \langle v_1, v_2 \rangle + \langle v_2, v_3 \rangle \geq -\frac{3}{2}$. And this means the objective function*

$$\sum_{(i,j)\in E(G)} \frac{1 - \langle v_i, v_j \rangle}{2} = \frac{3}{2} - \frac{1}{2}(\langle v_1, v_2 \rangle + \langle v_1, v_2 \rangle + \langle v_2, v_3 \rangle) \leq \frac{9}{4}$$

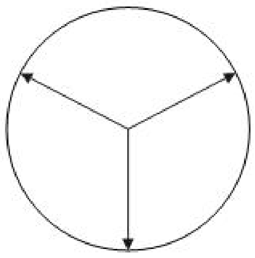

Figure 2: Optimum solution for the SDP relaxation given $G = C_3$

In order to get an algorithm finding the maximum cut, we need to round the solution back. In [GW95], Geomans and Williamson proposed a rounding procedure which guaranteed an approximation factor of 0.878 and we'll discuss it in the following.

2.4 Goemans-Williamson MAX CUT Algorithm

Algorithm 2.9 (Rounding Procedure)**.**

1. Solve the relaxed vector program, obtaining an optimum set of vectors v_i
2. Let r be a vector uniformly distributed in the unit sphere S^{n-1}. In other words, $r \in \mathbb{R}^n$, $\|r\|_n = 1$
3. Set $S = \{i | \langle v_i, r \rangle \geq 0\}$

Note that the vector r actually defines a hyperplane which separates the vertices. See Figure 3 for better understanding. As to the question of picking such a vector r, we can choose r such that:

$$r = \frac{(X_1, X_2, ..., X_n)}{\|(X_1, X_2, ..., X_n)\|_n}, X_i \sim N(0,1)^1, \text{for } i = 1, ..., n$$

Obviously, this rounding procedure is invariant to rotation. This is reasonable because the relaxed vector program is invariant to rotation. Now, we'll show that this rounding procedure actually guarantees a large approximation factor, which is stated in the following.

Lemma 2.10. *Goemans-Williamson MAX CUT algorithm gives an expected approximation factor of* 0.878.

To justify this lemma, we have the following claim.

Claim 2.11. $P_r[(i,j)\ is\ seperated]^2 \geq \alpha(\frac{1-\langle v_i, v_j \rangle}{2})$, *where* $\alpha = 0.878$.

[1]Gaussian distribution

[2]Namely, $P_r[|\{i,j\} \cap S| = 1]$

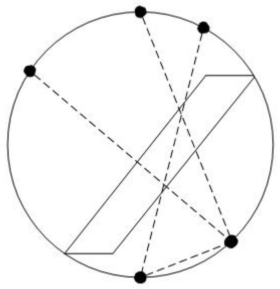

Figure 3: Goemans-Williamson MAX CUT algorithm

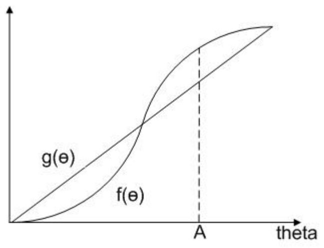

Figure 4: Lower bound for the approximation factor

Proof. Suppose θ is the angle between v_i and $v_j, \theta \in [0, \pi]$. Then, the probability of the edge (i, j) being separated is equal to $\frac{\theta}{\pi}$. This is obvious in the 2-dimensional space. And for an n-dimensional sphere where $n > 2$, it also holds because when we project the separating vector r onto the 2-dimensional space defined by v_i and v_j, its direction is still uniform. Therefore,

$$\frac{P_r[(i,j)\ is\ seperated)]}{\frac{1-\langle v_i, v_j \rangle}{2}} \geq \frac{\frac{\theta}{\pi}}{\frac{1-\cos\theta}{2}} = \frac{2}{\pi} \times \frac{\theta}{1-\cos\theta} \geq 0.878.$$

This lower bound is reached at point A in Figure 4, where we denote $g(\theta) = \theta$ and $f(\theta) = 1 - \cos\theta$.

In the following, we prove Lemma 2.10 based on Claim 2.11.

Proof. We denote Goemans-Williamson randomized algorithm as $\mathcal{A}(G)$, the optimum solution of the original quadratic program as $OPT(G)$ and the optimum solution of the corresponding vector program as $OPT^*(G)$. Then, we have:

$$\mathbb{E}[\mathcal{A}(G)] = \sum_{(i,j)\in E(G)} \mathbb{E}[\mathtt{1}_{|(i,j)\cap S|=1}] = \sum_{(i,j)\in E(G)} P_r[(i,j)\ is\ seperated)]$$

which is greater or equal to

$$\sum_{(i,j)\in E(G)} \alpha\cdot\frac{1-\langle v_i, v_j\rangle}{2} = \alpha\cdot \sum_{(i,j)\in E(G)} \frac{1-\langle v_i, v_j\rangle}{2} = \alpha\cdot OPT^*(G) \geq \alpha\cdot OPT(G)$$

And the lemma follows.

By Lemma 2.10, we know that Goemans-Williamson randomized algorithm guarantees an expected rounding factor and thus an approximation factor of at least 0.878. But is this tight? In other words, can the rounding and approximation factors be equal to 0.878 for some instance? In [FS00], however, Feige and Schechtman have shown that the integrality gap of the SDP relaxation above is 0.878, which indicates that there exist some instances for which the rounding factor is actually 0.878.

Another question should be whether we can add some additional constraints to improve the rounding and approximation factors. For example, we may add the triangle inequality: $(x_i - x_j) + (x_j - x_k) \geq (x_i - x_k)$, which corresponds to $\|v_i - v_j\|_n^2 + \|v_j - v_k\|_n^2 \geq \|v_i - v_k\|_n^2$ in the relaxed SDP. Interested readers may refer to recent work of Subhash A. Khot and Nisheeth K. Vishnoi[3].

3 Graph Coloring

In Graph Coloring problem, our goal is to color all vertices of an undirected graph G with as few colors as possible so that no edge in G is homochromatic. In the following, we denote $\chi(G)$ as the minimum number of colors needed to color the graph.

Example 3.1. We look at three simple examples first.

- $\chi(K_n) = n$, where K_n is a complete graph with n vertices
- $\chi(G) \leq 2$ if G is a bipartite graph
- $\chi(G) \leq 4$ if G is a planar graph

It is already known that approximating $\chi(G)$ within $n^{\frac{1}{7}-\varepsilon}$ is NP-hard. Here, we only focus on the cases when $\chi(G) = 3$. And the question is how well a polynomial algorithm can color the graph G. In [Wig83], Wigderson provided a polynomial algorithm coloring a 3-colorable graph using $O(\sqrt{n})$ colors. Their algorithm mainly makes use of two observations.

Observation 3.2. *In a 3-colorable graph G, the neighborhoods of every vertex are 2-colorable.*

Proof. Suppose by contradiction the neighbors of a vertex v can only be colored using at least 3 colors. Then, we have to use one more different color for coloring v, which shows that G is not 3-colorable.

Observation 3.3. *If a graph G has degrees $\leq \Delta$, it can be colored using no more than $\Delta + 1$ colors.*

[3]One good resource is http://www-static.cc.gatech.edu/ khot/publications.html

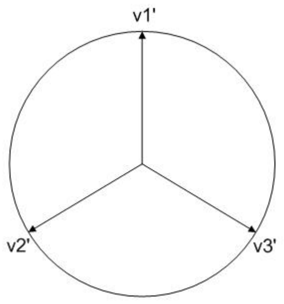

Figure 5: A solution of the vetor programmming with $\chi(G) = 3$

Proof. This is obvious because the degree of any vertex $v_i \leq \Delta$ and we always have at least one different color while coloring v_i.

Now, we show Wigderson's coloring algorithm.

Algorithm 3.4 (Wigderson's Graph Coloring Algorithm)**.**

Input: undirected 3-colorable graph G

Output: a coloring method using $O(\sqrt{n})$ colors

1. as long as there is a vertex v of degree $\geq \sqrt{n}$, color its neighbors in two colors and never use those colors again
2. color with $\Delta \leq \sqrt{n}$ colors the rest of the graph

By simple analysis, we know that Wigderson's coloring algorithm uses at most $2\frac{n}{\sqrt{n}} + \sqrt{n} = 3\sqrt{n} = O(\sqrt{n})$ colors.

However, there are better algorithms. Let's consider the following vector programming for 3-colorable graphs. We just present the VP and prove a related claim here. We'll continue this problem in the next lecture.

$$\begin{aligned} \langle v_i, v_j \rangle \ &\leq -\tfrac{1}{2}, \text{for any edge } (i,j) \in E(G) \\ \|v_i\|_n &= 1 \\ v_i &\in \mathbb{R}^n \end{aligned}$$

Claim 3.5. *The above vector program is feasible with regard to any 3-colorable graph G.*

Proof. Since $\chi(G) \leq 3$, we can divide the vertices of G into 3 sets V_1, V_2, V_3, where V_1, V_2, V_3 are disjoint and no edges exist between vertices in the same set. Therefore, as decribed in Figure 5 where the angles between the vectors are $\frac{2}{3}\pi$, we can embed the vertices in these three sets as the vectors v_1', v_2', v_3' respectively. Thus, we get a solution with $\langle v_i, v_j \rangle$ equal to $-\frac{1}{2}$ for all edges $(i,j) \in E(G)$.

References

[FS00] U. Feige and G. Schechtman. On the optimality of the random hyperplane rounding technique for MAX CUT. Technical report, Rehovot, Israel, 2000.

[GW95] M. X. Goemans and D. P. Williamson. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *J. Assoc. Comput. Mach.*, 42:1115–1145, 1995.

[Hås97] J. Håstad. Some optimal inapproximability results. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, 1997.

[Wig83] A. Wigderson. Improving the performance for approximate graph coloring. *Journal of the ACM*, 30:729–735, 1983.