CSC2411 - Linear Programming and Combinatorial Optimization*
Lecture 13: Semidefinite Programming (SDP) Relaxation.

Notes taken by Kevin Yuen

April 13, 2005

Summary: In this lecture, we give two example applications of approximating the solutions to NP-hard problems by solving the relaxed SDP and rounding. We begin with Max Cut, where we present the Geomans-Williamson Maxcut algorithm, a randomized algorithm that achieves an approximation ratio of 0.878. Next, we look at Graph Coloring, for which we give an algorithm due to Karger-Motwani-Sudan that can color a 3-colourable graph using $O(n^{0.386})$ colors. Finally, we introduce Lovasz Theta-Function, a SDP formulation that defines the vector chromatic number of a graph, and discuss its significance for the class of perfect graphs.

1 Max Cut

In the Max Cut problem, we have an undirected graph $G = (V, E)$. In the unweighted case, we simply want to partition the vertices of the graph into two sets such that the number of edges with vertices in both sets are maximized. In other words, we want to find $S \subset V(G)$ such that $|\{(i,j) \in E(G)| \, |\{i,j\} \cap S| = 1\}|$ is maximized. Refer to Figure 1 for an example of Max Cut.

Motativation Given n activities and m persons. Each activity can be scheduled either in the morning or in the afternoon. Each person is interested in two activities. The task is to schedule the activities to maximize the number of persons that can enjoy both activities. This can be formulated as a Max Cut problem by converting the activities into vertices and persons into edges.

Known complexity results about Max Cut:

- Getting an approximation ratio of 0.942 is NP-hard (Håstad '97).

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

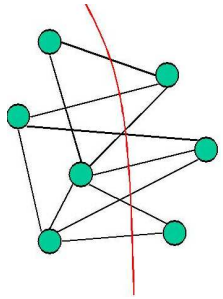

Figure 1: An example of Max Cut.

- Best approximation ratio known, without SDP, is $\frac{1}{2}$ using a greedy algorithm. Here's the greedy algorithm:
  - Start with any arbitrary cut.
  - If some node has more neighbours on its side than on the other side, then move it to the other side. Repeat.

  At this end, at least half of the edges are crossing the cut.
- With SDP, an approximation ratio of 0.878 can be obtained (Goemans-Williamson '95).

Here is a way to describe the problem in Integer Quadratic Programming (IQP), we introduce variables $x_i \in \{-1, 1\}$ for every vertex in the graph. The semantics is that $x_i = 1$ if $i \in S$, and $x_i = -1$ if $i \in \overline{S}$. With this notation, $x_i x_j = 1$ if i and j belong to the same set, and $x_i x_j = -1$ if i and j belong to the opposite sets. Thus,

$$\frac{1 - x_i x_j}{2} = \begin{cases} 1 & |\{i, j\} \cap S| = 1 \\ 0 & \text{otherwise} \end{cases}$$

We now present an IQP formulation of Max Cut.

$$\max \sum_{(i,j)\epsilon E} \frac{1-x_i x_j}{2} \quad \text{s.t.}$$

$$x_i \in \{-1, 1\}$$

Since solving this IQP is NP-hard, we want to relax this formulaton. Due to the product $x_i x_j$ in the IQP, it is no good to relax the integrality constraint to get a Linear Programming (LP) formulation. Instead, we relax the dimensions of variables x_i to get a Vector Programming (VP) formulation, which is equivalent to a SDP formulation. In the VP formulation, we treat each variable x_i as a vector and denote $\langle x_i, x_j \rangle$ as the inner product of x_i and x_j.

$$\max \sum_{(i,j)\epsilon E} \frac{1-\langle x_i, x_j\rangle}{2} \quad \text{s.t.}$$

$$x_i \in \mathbb{R}^n$$

$$\langle x_i, x_i\rangle = 1$$

In this VP formulation, n is the number of vertices in the graph. The objective function above rewards large separation between vertices that are connected by edges. Geometrically, the SDP relaxation of Max Cut embeds the vertices of the graph on an unit sphere such that if two vertices are joined by an edge, then these two vertices are far apart. See Figure 2.

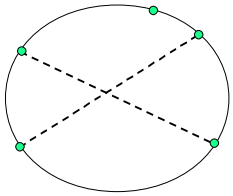

Figure 2: Geometric intuition for an SDP relaxation of Max Cut.

Example 1.1 (Integral vs Relaxed Max Cut). In this example, we consider the graph $G = C_3$. For the IQP, the optimal solution is $OPT_{IQP}(C_3) = 2$. On the other hand, the optimal vectors of the VP relaxation lie in a 2-dimensional subspace and each pair of vectors is 120 degrees apart, this result is proven in assignment 4. See Figure 3 for a graphical interpretation. Thus, $\forall i \neq j, \langle x_i, x_j\rangle = -\frac{1}{2}$ and $OPT_{VP}(C_3) = 3\left(\frac{1+\frac{1}{2}}{2}\right) = 2\frac{1}{4}$.

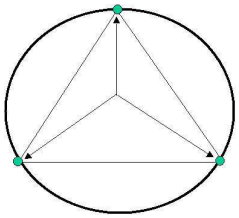

Figure 3: Optimal vectors of VP relaxation with C_3.

1.1 Geomans-Williamson Maxcut algorithm

Geomans-Williamson formulated the above VP and proposed a method of rounding the VP to get a cut. Then they showed that the cut produced will tend to have many edges. We now concentrate on their randomized rounding algorithm.

Algorithm 1.2 (Geomans-Williamson Maxcut algorithm).

1. Solve VP and let $x_1, x_2, \ldots, x_n$ be the solution.
2. Pick a random vector $\vec{v}$ in S^n.
3. Let $S = \{i | \langle x_i, \vec{v} \rangle \geq 0\}$.

   Note: We can compute $\vec{v}$ as follows:

   $$\vec{v} = \frac{(X_1, X_2, \ldots, X_n)}{||(X_1, X_2, \ldots, X_n)||}$$

   where $X_i \sim N(0, 1)$, then $\vec{v}$ is a unit vector whose direction is uniformly distributed over the n-dimensional unit sphere. We can define a hyperplane by treating vector $\vec{v}$ as a normal of that hyperplane.

Some intuition:

- For solution ± 1, we already got the obvious cut.
- The rounding is invariant to rotation, which is reasonable, since the solutions to the VP are invariant to rotation.

Analysis Here, we analyze the approximation ratio of the Geomans-Williamson Maxcut algorithm. For all edges $(i, j) \in E$, if $x_i = -x_j$, then Prob[edge (i,j) is separated] = 1. And this edge's contribution to the objective function is $\frac{1-\langle x_i, x_j \rangle}{2} = 1$. If $x_i = x_j$, then Prob[edge (i,j) is separated] = 0. And this edge's contribution to the objective function is $\frac{1-\langle x_i, x_j \rangle}{2} = 0$.
In general, Prob[edge (i,j) is separated] = $\frac{\alpha}{\pi}$ where α is the angle between x_i and x_j. This fact is obvious if $\vec{v}$ is a random vector in the 2-dimensional circle. Refer to Figure 4 for an illustration. We now claim that this fact can be extended to dimensions greater than 2.

Claim 1.3. *For dimension n greater than 2, Prob[edge (i,j) is separated] = $\frac{\alpha}{\pi}$.*

Proof. We prove this claim by showing that we can project $\vec{v}$ onto the 2-dimensional plane spanned by vector x_i and x_j. From the construction of vector $\vec{v}$, we can project this vector onto any 2-dimensional space by considering only the first two Guassian random variables in $\vec{v}$ while ignoring the rest. □

$$\frac{\text{Excepted \# of separated edges}}{OPT_{SDP}} = \frac{\sum_{(i,j)\epsilon E} \text{Prob[edge (i,j) is separated]}}{\sum_{(i,j)\epsilon E} \frac{1-\langle x_i, x_j \rangle}{2}} \geq$$

$$\min_{(i,j)} \frac{\text{Prob[edge (i,j) is separated]}}{\frac{1-\langle x_i, x_j \rangle}{2}} = \frac{\frac{\alpha}{\pi}}{\frac{1-\cos\alpha}{2}} \geq 0.878$$

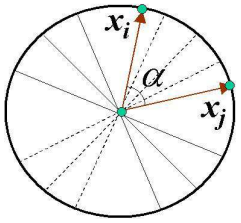

Figure 4: Prob[edge (i,j) is separated] = $\frac{\alpha}{\pi}$.

Note: $\langle x_i, x_j \rangle = \cos\alpha$ since x_i and x_j are unit vectors.

Remark 1.4. From (Feige-Schechtman '00), the integrality gap (IG) is arbitary close to 0.878. Specifically, there exists a graph, G, for which $\frac{\text{Max Cut(G)}}{OPT_{SDP}(G)} \approx 0.878$. This means that no rounding can lead to a guarantee which is (always) better than 0.878.

Tightening the IG We can improve the IG by adding triangle inequality constraints to the relaxed VP formulation. $||x_i - x_j||^2 \leq ||x_i - x_k||^2 + ||x_k - x_j||^2$ is an example of a triangle inequality constraint for vertices i, j, and k. Notice that such an inequality holds for $x_i \in \{+1, -1\}$, but it does not necessarily hold for $x \in S^n$, the unit sphere on n dimensions. For example, consider three close points on the unit sphere (see Figure 5), on the microscopic level, we can consider these three points as collienar points. Let x_i, x_j, x_k be three collinear points where x_k is the middle point between x_i and x_j. Let the distance between (x_i, x_j) be 1, then the distance between (x_i, x_k) and (x_k, x_j) will be $\frac{1}{2}$. In this case, the triangle inequality constraint $||x_i - x_j||^2 \leq ||x_i - x_k||^2 + ||x_k - x_j||^2$ does not hold. Thus, adding triangle inequality constraints may not necessarily improve the IG.

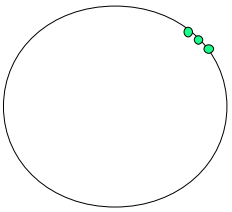

Figure 5: Collinear points on the unit sphere.

2 Graph Coloring

In the Graph Coloring problem, we have an undirected graph $G = (V, E)$. We want to color all vertices in the graph with as few colors as possible such that the coloring is

valid (no edge is monochromatic). Let $\chi(G)$ be the chromatic number of G, which is equivalent to the minimum number of colors needed to color G.

Example 2.1.

1. $\chi(K_n) = n$, where K_n is a clique with n vertices.
2. $\chi(\text{bipartite graph}) \leq 2$.
3. $\chi(\text{planar graph}) \leq 4$.

Known complexity results about Graph Coloring:

- NP-hard to approximate $\chi(G)$ to within $n^{\frac{1}{7}-\epsilon}$.
- Best approximation ratio known is $n^{\frac{(loglog\ n)^2}{(log\ n)^3}}$.

Suppose we restrict ourselves to 3-colorable graphs ($\chi(G) = 3$) only, then how well can we solve the Graph Coloring problem? Khanna-Linial-Safra (KLS '93) showed that Graph Coloring using 4 colors is still NP-hard. (Wigderson '81) proposed an algorithm that can color with $O(\sqrt{n})$ colors if $X(G) = 3$.

Facts Wigderson's algorithm is based on the following facts:

1. For a 3-colorable graph, the neighborhood of every vertex can be colored using 2 colors.
2. For a graph with max degree δ, it can be colored greedily with $\delta + 1$ colors. Consider colors $1, 2, \ldots, \delta, \delta + 1$. List the vertices $v_1, v_2, \ldots, v_n$. The following recursive/iterative algorithm colors G using at most $\delta + 1$ colors:
   (a) Color v_1 with color 1.
   (b) For $i = 2, \ldots, n$, color v_i with smallest color not assigned already to the vertices among $\{v_1, \ldots, v_{i-1}\}$ joined to v_i (i.e. colors among the vertices adjacent to v_i that have already been colored). Since the max degree is δ and we have $\delta + 1$ colors, there will always be an available color.

Algorithm 2.2 (Wigderson's algorithm).

1. As long as there exists a vertex v with degree $\geq \delta$, we color the neighbors of v with 2 colors. Afterwards, we discard the neighbors of v and the two colors used.
2. For the remaining graph, color with δ colors.

Analysis Notice that the algorithm iterates step 1 for at most $\frac{n}{\delta}$ times. Hence the number of colors used is at most $2\frac{n}{\delta} + \delta = O(\sqrt{n})$ for $\delta = \sqrt{n}$.

We now present the relaxed VP formulation for Graph Coloring. In this VP formulation, we introduce vector variables v_i for every vertex in the graph. For every edge $(i, j) \in E$, we have the constraint $\langle v_i, v_j \rangle \leq \lambda$. Geometrically, the SDP relaxation of Graph Coloring embeds the vertices of the graph on the unit sphere such that if two vertices are joined by an edge, then the angle between them must be no less than $\cos^{-1} \lambda$. And the objective function of the formulation minimizes λ, which is equivalent to maximizing the smallest angle separation between any two vertices that are connected by an edge.

$$\min \lambda \qquad \text{s.t.}$$

$$\langle v_i, v_j \rangle \leq \lambda \quad \forall (i, j) \in E$$

$$v_i \in \mathbb{R}^n$$

$$\langle v_i, v_i \rangle = 1$$

The following lemma motivates the above VP formulation.

Lemma 2.3. *If $\chi(G) = 3$, then $\lambda \leq -\frac{1}{2}$.*

Proof. Since G is 3-colorable, $v = v_1 \cup v_2 \cup v_3$ where v_1, v_2, v_3 are disjoint and there are no edges between vertices in the same v_i. We embed v_1, v_2, v_3 to vectors G_3 shown in Figure 6. □

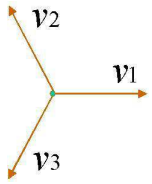

Figure 6: Solution of the VP formulation with $\chi(G) = 3$.

Algorithm 2.4 (KMS '95 Part 1).

1. Solve VP and let $v_1, v_2, \ldots, v_n$ be the soluton.
2. Choose $T = \lceil log_3 \triangle \rceil$ random hyperplanes $\vec{v}_1, \ldots, \vec{v}_t$, where $\triangle$ is the maximum degree of graph G.
3. Assign each vertex v_i with a color that corresponds to its $sign(v_i, \vec{v})$ vector. Where $sign(v_i, \vec{v})$ is a vector with t entries. The j^{th} entry, denoted $sign(v_i, \vec{v})_{(j)}$, is 1 if $\langle v_i, \vec{v}_j \rangle \geq 0$, or -1 otherwise. Since there are a total of 2^t distinct sign vectors, 2^t colors will be used in this step. At this point, we want to analyze the

expected number of monochromatic edges generated by this algorithm so far. For all edges $(i, j) \in E$, v_i and v_j are separated by at least 120 degrees. Thus, Prob[(i,j) is monochromatic] $\leq (\frac{1}{3})^t \leq \frac{1}{9\triangle}$.

E[# of monochromatic edges] $\leq m \cdot \frac{1}{9\triangle} \leq \frac{n\triangle}{2} \cdot \frac{1}{9\triangle} \leq \frac{n}{4}$, where m is at most $\frac{n\triangle}{2}$.

4. After step 3, there will be at most $\frac{n}{4}$ monochromatic edges. We remove colors from vertices with monochromatic edges and continue with the rest of uncolored vertices. Notice that there are at most $\frac{n}{2}$ such vertices. In order to color all the vertices, we need to repeat the steps above for $\log n$ iterations and each iteration requires 2^t colors. See Figure 7. In this part of the algorithm, the total number of colors we used is $2^t \cdot \log n \approx \triangle^{\log_3 2} \approx \triangle^{0.631} \leq n^{0.631}$. So far, there is no improvement in the number of colors used when compared to the Wigderson's algorithm.

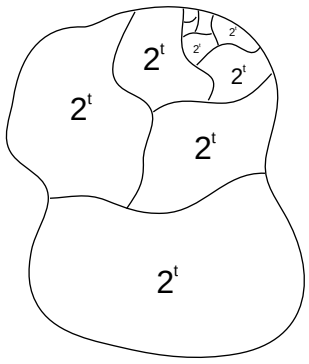

Figure 7: Graph Coloring with (KMS '95 Part 1).

To get the improvement, (KMS '95) suggested the following hybrid algorithm, which combines the algorithm above with Wigderson's algorithm.

Algorithm 2.5 (KMS '95 Part 2).

1. As long as there exists a vertex v with degree $\geq \delta$, we color the neighbors of v with 2 colors. Afterwards, we discard the neighbors of v and the two colors used.
2. For the remaining graph, color with $O(\delta^{0.631})$ colors using part 1 of (KMS '95).

Analysis The total number of colors used in this algorithm is less than $2\frac{n}{\delta} + \delta^{0.631}$. The number of colors used is minimized when $\frac{n}{\delta} = \delta^{0.631}$ or equivalently $\delta = n^{\frac{1}{1.631}}$. Thus, number of colors used $= O(n^{\frac{0.631}{1.631}}) = O(n^{0.386})$.

3 Lovász Theta Function '79

KMS essentially used a relaxed notion of coloring, which is called vector k-coloring: we embed vertices in the unit sphere and require small inner product for two vectors with corresponding vertices that are joined by an edge. A vector k-coloring of a graph $G = (V, E)$ is a sequence of unit vectors $v_1, v_2, \ldots, v_n$ such that if $(i, j) \in E$ then $\langle v_i, v_j \rangle = -\frac{1}{k-1}$. Vector chromatic number is the minimum k for which G is vector k-colorable. We can formulate the vector k-coloring problem as follows:

$$\min k \qquad \text{s.t.}$$

$$\langle v_i, v_j \rangle \leq -\tfrac{1}{k-1} \quad \forall (i,j) \in E$$

$$v_i \in \mathbb{R}^n$$

$$\langle v_i, v_i \rangle = 1$$

Lovász defined the vector chromatic number of a graph G and it is named Lovász Theta Function $\theta(G)$, where

$$\omega(G) \leq \theta(G) \leq \chi(G)$$

and $\omega(G)$ is the size of maximal clique of graph G.

Lemma 3.1. *Vector chromatic number of G is smaller or equal to its chromatic number, $\theta(G) \leq \chi(G)$.*

Proof. Recall Lemma 2.3. If $\chi(G) = 3$, then $\lambda \leq -\frac{1}{3-1} = -\frac{1}{2}$. □

Lemma 3.2. *Size of maximal clique of G is smaller or equal to its vector chromatic number, $\omega(G) \leq \theta(G)$.*

Proof. It is enough to show that $\theta(K_n) \geq n$.

$$0 \leq \langle \sum v_i, \sum v_i \rangle = \sum_i \langle v_i, v_i \rangle + \sum_{i \neq j} \langle v_i, v_j \rangle$$

$$\Rightarrow \sum_{i \neq j} \langle v_i, v_j \rangle \geq -n$$

$$\Rightarrow \exists (i,j) \text{ for which } \langle v_i, v_j \rangle \geq \frac{-n}{n(n-1)} = \frac{-1}{n-1}$$

$$\text{So } \theta(K_n) \geq n \text{ and } \theta(G) \geq \omega(G).$$

□

Definition 3.3. Perfect graphs are graph G for which $\forall$ induced subgraph H of G,

$$\omega(H) = \theta(H) = \chi(H)$$

Claim 3.4. *We can find the maximal clique in perfect graphs using $\theta(G)$.*

Algorithm 3.5 (Finding maximal clique in perfect graphs).
MaxClique(G):

If $\theta(G\backslash x) \leq \theta(G)$ then

- Include x in the maximal clique.
- **MaxClique**$(N(x))$, where $N(x)$ is the neighborhood of x.

Else

- Exclude x in the maximal clique.
- **MaxClique**$(G\backslash x)$.