CS880: Approximations Algorithms

Scribe: Chi Man Liu **Lecturer:** Shuchi Chawla
Topic: Semi-Definite Programming: Graph Coloring **Date:** 3/29/2007

In the previous lecture , we saw how semi-definite programming (SDP) could be employed to approximate Max-Cut and Max-2-SAT. In this lecture, we look at another problem which can be approximated using SDP: Graph Coloring.

21.1 Graph Coloring: Overview

Let $G = (V, E)$ be a graph. A *k-coloring* for G is a function $f : V \to [k]$ such that $f(u) \neq f(v)$ for all $(u, v) \in E$. In other words, a k-coloring is an assignment of vertices to k colors such that no edge is monochromatic. We say that a graph G is *k-colorable* if there exists a k-coloring for G. The *chromatic number* of G is the least k such that G is k-colorable. Given a k-colorable graph G, finding a k-coloring for G is solvable in polynomial time for $k = 2$, but NP-hard for $k \geq 3$.

In Homework 1, we showed how to find a $O(\sqrt{n})$-coloring for any 3-colorable graph in polynomial time, where n is the number of vertices in the graph. That algorithm is due to Wigderson [1] and can be extended to find $O(n^{2/3})$-colorings for $k = 4$, and $O(kn^{1-1/(k-1)})$-colorings for general k. Blum [2] gave a combinatorial algorithm for coloring 3-colorable graphs using $\tilde{O}(n^{3/8})$ colors[1]. Karger, Motwani and Sudan [3] presented a semi-definite programming based $\tilde{O}(n^{1/4})$-coloring for 3-colorable graphs. Combining the techniques in [2] and [3], Blum and Karger [4] improved the bound to $\tilde{O}(n^{3/14})$. The best known approximation for 3-coloring uses $O(n^{0.2111})$ colors. This algorithm, due to Arora, Chlamtac and Charikar [5], is based on semi-definite programming and the triangle inequality.

On the hardness of Graph Coloring, it is known that coloring a 3-colorable graph using only 4 colors is NP-hard. In fact, the Unique Games Conjecture implies that there is no constant factor approximation for 3-coloring. It is also known that approximating k-coloring better than a factor of $n^{1-\epsilon}$ for arbitrary k is NP-hard.

21.2 SDP Formulation

We need to divide the vertices into k sets such that every edge gets cut. This is different from the Max-Cut and Max-2-SAT problems for which only two sets are sufficient. Our strategy is to map the vertices to unit vectors in $\mathbb{R}^n$, and we want to maximize distances between the edges. Finally, a randomized procedure is used to partition the vertices into k sets according to the optimal solution vectors.

We introduce a vector variable $v_i \in \mathbb{R}^n$ for each $i \in V$. Recall that the dot product between two vectors increases as they get closer to each other. Therefore, to maximize distances between the edges, we want to minimize the quantity $\max_{(i,j)\in E} v_i \cdot v_j$. We formulate Graph Coloring as the

[1]The notation $\tilde{O}$ is sometimes used to hide low-order multiplicative terms, such as $\log n$.

following SDP (equivalently, vector program).

$$\begin{array}{rll} \text{minimize} & t & \\ \text{subject to} & v_i \cdot v_j \leq t & \forall (i,j) \in E \\ & v_i \cdot v_i = 1 & \forall i \in V \\ & v_i \in \mathbb{R}^n & \forall i \in V \end{array}$$

Unfortunately, the relation between an feasible solution to the SDP and a legal coloring for the graph is not obvious — given a feasible solution to the SDP, how to construct a legal k-coloring? Furthermore, how does k depend on t? The following two lemmas answer the latter question.

Lemma 21.2.1 *Let t^* be the optimal value of the SDP. If G is k-colorable, then $t^* \leq \frac{-1}{k-1}$.*

Lemma 21.2.2 *Let t^* be the optimal value of the SDP. If G contains a k-clique, then $t^* \geq \frac{-1}{k-1}$.*

Remark. We define $\theta(G) = 1 - \frac{1}{t^*}$ for any graph G. This function is known as the *Lovász theta function*. Note that if $t^* = \frac{-1}{k-1}$, then $\theta(G) = k$. In this case, the graph is called vector-k-colorable. For a graph G, the *clique number* of G is the size of the largest clique in G. By Lemma 21.2.1, $\theta(G)$ is at most the chromatic number of G. By Lemma 21.2.2, $\theta(G)$ is at least the clique number of G. A graph with its chromatic number equal to its clique number is known as a *perfect graph*. For example, complete graphs are perfect graphs. Note that for perfect graphs, their chromatic numbers and clique numbers can be computed in polynomial time, namely by computing the Lovász theta function.

Proof: [Proof of Lemma 21.2.1] Suppose that we have a partition of V into k subsets. We will define explicitly k unit vectors $v_1, \ldots, v_k$ in $\mathbb{R}^k$ such that $v_i \cdot v_j \leq \frac{-1}{k-1}$ for any $i \neq j$. Assigning vertices in different subsets to different vectors, we have found a feasible solution to the SDP with value at most $\frac{-1}{k-1}$, thus proving the lemma. We demonstrate how to find such vectors by induction on k.

The base case ($k = 2$) is trivial. Assume that for some $k \geq 2$ we have unit vectors $v'_1, \ldots, v'_k \in \mathbb{R}^k$ such that $v'_i \cdot v'_j \leq \frac{-1}{k-1}$ for $i \neq j$. We construct $k+1$ unit vectors $v_1, \ldots, v_{k+1} \in \mathbb{R}^{k+1}$ as follows.

- $v_{k+1} = (0, \ldots, 0, 1)$;
- for $1 \leq i \leq k$, $v_i = (\alpha v'_i, -1/k)$, i.e. the $k+1$-vector obtained by adding one coordinate $(-1/k)$ to the k-vector $\alpha v'_i$, where $\alpha = \sqrt{1 - 1/k^2}$.

There are a few things to show. First, the v_i's are unit vectors:

$$v_i \cdot v_i = \alpha^2 (v'_i \cdot v'_i) + \frac{1}{k^2} = 1.$$

Second, $v_i \cdot v_j \leq \frac{-1}{k}$ for $1 \leq i, j \leq k$:

$$\begin{aligned} v_i \cdot v_j &= \alpha^2 (v'_i \cdot v'_j) + \frac{1}{k^2} \\ &\leq \left(1 - \frac{1}{k^2}\right)\left(\frac{-1}{k-1}\right) + \frac{1}{k^2} \\ &= -\frac{k+1}{k^2} + \frac{1}{k^2} \\ &= \frac{-1}{k}. \end{aligned}$$

Third, $v_i \cdot v_{k+1} = \frac{-1}{k}$ for $1 \leq i \leq k$, which is obvious. ■

Proof: [Proof of Lemma 21.2.2] Let $v_1, \ldots, v_k \in \mathbb{R}^k$ be k unit vectors. Consider the dot product of $\sum_{i=1}^{k} v_i$ with itself:

$$\begin{aligned} \left\langle \sum_{i=1}^{k} v_i \, , \sum_{i=1}^{k} v_i \right\rangle &= \sum_{i=1}^{k} (v_i \cdot v_i) + \sum_{\substack{i \neq j \\ 1 \leq i,j \leq k}} (v_i \cdot v_j) \\ &= k + k(k-1)\overline{r}, \end{aligned}$$

where $\overline{r}$ is the average dot product. Noticing that the above dot product is non-negative, we get

$$\begin{aligned} k + k(k-1)\overline{r} &\geq 0 \\ \overline{r} &\geq \frac{-1}{k-1} \\ \max_{\substack{i \neq j \\ 1 \leq i,j \leq k}} (v_i \cdot v_j) &\geq \frac{-1}{k-1} \end{aligned}$$

Suppose that G contains a k-clique. Consider the k unit vectors in the optimal solution corresponding to the vertices in this clique. These vectors satisfy the above inequality. Since there is an edge between every pair of these vertices, t^* must be at least the maximum dot product among these vectors, which is at least $\frac{-1}{k-1}$. ■

In view of Lemma 21.2.1 and Lemma 21.2.2, we interpret our SDP as the following program.

$$\begin{array}{lll} \text{minimize} & k & \\ \text{subject to} & v_i \cdot v_j \leq \frac{-1}{k-1} & \forall (i,j) \in E \\ & v_i \cdot v_i = 1 & \forall i \in V \\ & v_i \in \mathbb{R}^n & \forall i \in V \end{array}$$

21.3 Converting from SDP Solution to Coloring

Next we give a randomized algorithm for transforming a solution to the above SDP into a feasible coloring. For simplicity, we restrict our attention to 3-colorings. The following 3-coloring algorithm can be generalized to handle more colors with some more effort.

1. Pick t (whose value to be decided) "directions" (unit vectors) $\alpha_1, \ldots, \alpha_t$ uniformly at random.
2. Divide the graph into 2^t parts according to $sgn(\alpha_j \cdot v_i)$ for every j. Color each part with a unique color.
3. Recurse on monochromatic edges.

If G is 3-colorable, then by Lemma 21.2.1 we get an optimal solution to the SDP with $k \leq 3$ and $v_i \cdot v_j \leq -1/2$ for all $(i,j) \in E$. This implies that for any $(i,j) \in E$, the angle between v_i and v_j is at least $2\pi/3$.

Let Δ be the maximum degree of the graph. We pick $t = \log_3 \Delta + 1$.

Claim 21.3.1 *Suppose that we use the above algorithm to color a 3-colorable graph $G = (V, E)$. Then, the expected number of monochromatic edges after any iteration is at most $n/4$, where $n = |V|$.*

Proof: Consider any $(x, y) \in E$. Fix a j $(1 \leq j \leq t)$. Then, since the angle between v_x and v_y is at least $2\pi/3$ as discussed above, we have

$$\mathbf{Pr}[sgn(\alpha_j, v_x) = sgn(\alpha_j, v_y)] \leq \frac{1}{3}.$$

Hence,

$$\begin{aligned} \mathbf{Pr}[(x,y) \text{ is monochromatic}] &= \mathbf{Pr}[\forall j, sgn(\alpha_j, v_x) = sgn(\alpha_j, v_y)] \\ &\leq \frac{1}{3^t} \\ &= \frac{1}{3\Delta}. \end{aligned}$$

Thus, the expected number of monochromatic edges is at most $\frac{n\Delta}{2} \cdot \frac{1}{3\Delta} \leq n/4$. ■

We know from Claim 21.3.1 that the expected number of vertices need to be considered after any iteration is at most $n/2$ (since there are $n/4$ monochromatic edges). Thus, we have the following corollary.

Corollary 21.3.2 *With high probability, the above algorithm terminates after $O(\log n)$ iterations.*

We now analyze the number of colors used in the coloring. In each iteration, at most 2^t new colors are used. Supposing there are $\log n$ iterations, the total number of colors used is

$$\begin{aligned} 2^t \log n &\approx 2^{\log_3 \Delta} \cdot \log n \\ &= \Delta^{\log_3 2} \cdot \log n \\ &= \Delta^{0.63} \log n \\ &= O(n^{0.63} \log n). \end{aligned}$$

This is worse than the $O(\sqrt{n})$-approximation we got in Homework 1. We can improve the approximation factor by combining the above algorithm with the idea in Homework 1. Note that in the above analysis we simply bound the value of Δ by n. As in Homework 1, we can set a threshold for the value of Δ, say Δ^*. The combined algorithm is as follows.

1. Pick a vertex v with degree greater than Δ^*. 3-color the neighbors of v (including v), and remove them from the graph.

2. Repeat step 1 until all vertices in the graph have degree at most Δ^*.

3. Run the above 3-coloring algorithm on the pruned graph.

The number of colors used in the pruning phase (steps 1 and 2) is at most $\frac{3n}{\Delta^*}$. The (expected) number of colors used in step 3 is $(\Delta^*)^{0.63} \log n$. We can minimize their sum by setting $\Delta^* = n^{1/1.63}$, which leads to a $\tilde{O}(n^{0.39})$-coloring. A smarter preprocessing gives a $\tilde{O}(n^{0.25})$-coloring algorithm for 3-colorable graphs, but we will not discuss it in class.

References

[1] A. Wigderson. Improving the performance guarantee for approximate graph coloring. *Journal of the ACM*, 30(4):729–735, October 1983.

[2] A. Blum. New approximation algorithms for graph coloring. *Journal of the ACM*, 31(3):470–516, 1994.

[3] D. Karger, R. Motwani and M. Sudan. Approximate graph coloring by semidefinite programming. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, November 1994.

[4] A. Blum and D. Karger. An $\tilde{O}(n^{3/14})$-coloring algorithm for 3-colorable graphs. *Information Processing Letters*, 61(1):49–53, 1997.

[5] S. Arora, E. Chlamtac and M. Charikar. New approximation guarantee for chromatic number. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, 2006.