COMPSCI 638: Graph Algorithms	November 13, 2019
Lecture 23	
Lecturer: Debmalya Panigrahi	*Scribe: Kevin Sun*

1 Overview

In this lecture, we continue studying applications of semidefinite programming. In particular, we present an approximation algorithm for the graph coloring problem.

2 Graph Coloring

Let $G = (V, E)$ be an undirected graph. We say G is *k-colorable* if there exists a proper coloring using k colors, where a *proper coloring* is a function $f : V \to \mathbb{Z}^+$ satisfying $f(u) \neq f(v)$ for every edge $\{u, v\} \in E$. (We think of each positive integer as a "color.")

It is known that determining whether or not a graph is 3-colorable is NP-hard. Furthermore, finding a 4-coloring of a 3-colorable graph is also NP-hard. In fact, k-coloring a 3-colorable graph in polynomial time, for any constant k, would violate the Unique Games Conjecture. So for the rest of this lecture, we focus on coloring a 3-colorable graph using a minimum number of colors.

Coloring using $O(\sqrt{n})$ colors: Let us make the following observations:

1. A graph is 2-colorable if and only if it is bipartite, and finding a 2-coloring of a bipartite graph can be done using a simple breath-first search procedure.

2. If the maximum vertex degree in G is Δ, then finding a $(\Delta + 1)$-coloring of G is straightforward: a greedy strategy never gets "stuck" because every vertex has at most Δ incident edges.

These two observations yield an algorithm due to Wigderson [Wig83]: if $\Delta \leq \sqrt{n}$, then by the second observation, we can color G in $O(\sqrt{n})$ colors. Otherwise, let v be a vertex with degree greater than $\sqrt{n}$ and notice that its neighborhood $N(v)$ is bipartite (because G is 3-colorable). Thus, we can color $N(v)$ using 2 colors, and we then remove them from the graph. We repeat this iteratively until all degrees are less than $\sqrt{n}$. At this point, by the second observation, we can color the remaining vertices using $\sqrt{n} + 1$ colors.

In each iteration, we remove at least $\sqrt{n}$ vertices, so the total number of iterations is at most $n/\sqrt{n} = \sqrt{n}$. In each iteration we use 2 new colors, and the remaining vertices require at most $\sqrt{n} + 1$ colors, so the total number of colors we use is $O(\sqrt{n})$.

Generalization: The algorithm described above can be seen as a special case of the following approach introduced by Blum [Blu94]: suppose we were given an algorithm that produces a bipartite subgraph containing $\epsilon n/f(n)$ vertices for some constant $\epsilon > 0$ and function $f(n)$. In other words, if $C(n)$ denotes the number of colors we need, then

$$C(n) = 2 + C\left(n - \frac{\epsilon n}{f(n)}\right).$$

(The previous algorithm essentially set $\epsilon = 1$ and $f(n) = \sqrt{n}$.) From this, it can be shown that reducing the number of vertices by half requires $f(n)/\epsilon$ calls to this algorithm, so

$$C(n) \leq \frac{2f(n)}{\epsilon} + C\left(\frac{n}{2}\right).$$

Thus, having such an algorithm yields an $O(f(n)\log n/\epsilon)$-coloring. In his paper, Blum [Blu94] showed that there exists an algorithm with $f(n) = n^{0.4}$, obtaining an $\tilde{O}(n^{0.4})$-coloring algorithm. He also showed that additional preprocessing steps yields an $\tilde{O}(n^{0.375})$-coloring algorithm.

2.1 SDP for Graph Coloring

We now present a semidefinite program (SDP) for the graph coloring problem due to Karger, Motwani, and Sudan [KMS98]. First, notice that graph coloring is equivalent to partitioning the vertex set into k subsets such that no edge has both endpoints in one subset and k is minimized. Notice that this is formulation is similar to the maximum cut problem (see Lecture 22).

So let us associate each vertex i with a unit vector $v_i \in \mathbb{R}^n$. Intuitively, in order for our coloring to be valid, we need to ensure that v_i and v_j are "far apart" for every edge $\{i,j\} \in E$. Notice that $v_i \cdot v_j$ captures this distance in an inverse manner: if $v_i \cdot v_j$ is large (i.e., close to 1), then the angle between the vectors is small, which is undesirable. This leads to the following SDP:

$$\begin{aligned} &\text{(SDP)}: \min t \\ &v_i \cdot v_j \leq t \quad \forall \{i,j\} \in E \\ &v_i \cdot v_i = 1 \quad \forall i \in V. \end{aligned}$$

Let $\chi = \chi(G)$ denote the minimum number of colors needed to color G, and let $\eta = \eta(G)$ denote the size of the largest clique (i.e., complete subgraph) in G.

Lemma 1. *The optimal solution t^* of (SDP) satisfies $t^* \leq -1/(\chi - 1)$.*

Proof. We proceed by inducting on χ. For each value of χ, we will construct a solution with objective value $-1/(\chi - 1)$, and since (SDP) is a minimization, this implies the lemma. Each solution only uses the first $\chi - 1$ coordinates; the remaining $n - (\chi - 1)$ coordinates are all implicitly set to zero. If $\chi = 2$, then the graph is bipartite, so we can set $v_i = 1$ and $v_j = -1$ for every edge $\{i,j\} \in E$. Then $v_i \cdot v_j = -1 = -1/(-1-1)$, so we this is a feasible solution for $t = -1$.

Now assume $\chi = k + 1$ for some $k \geq 1$. The solution for $\chi = k$ gives us vectors $v'_1, \ldots, v'_k$ such that $v'_i \cdot v'_j \leq -1/(k-1)$. We shall design vectors $v_1, \ldots, v_k, v_{k+1}$; recall that we only specify their first k coordinates. The first $k - 1$ coordinates of v_{k+1} are all 0, and the k-th coordinate is 1:

$$v_{k+1} = (0, 0, \ldots, 0, 1).$$

To construct v_i for $i \in \{1, \ldots, k\}$, we set the first $k - 1$ coordinates as $\alpha v'_i$ (for some α to be determined), and the k-th coordinate is $-1/k$:

$$v'_i = \left(\alpha v_i, -\frac{1}{k}\right).$$

Recall that v_i needs to be a unit vector; this is achieved by setting $\alpha = \sqrt{1 - 1/k^2}$. Now we must show that $v_i \cdot v_j \leq -1/k$. If i or j is equal to $k + 1$, then this is trivial. Otherwise, observe that

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

Lemma 2. *The optimal solution t^* of (SDP) satisfies $t^* \geq -1/(\eta - 1)$.*

Proof. Without loss of generality, suppose the vertices $\{1, \ldots, \eta\}$ form a clique of size η. Now observe that the following inequality holds:

$$\sum_{i=1}^{\eta} \sum_{\substack{j=1 \\ j \neq i}}^{\eta} v_i \cdot v_j + \sum_{i=1}^{\eta} v_i \cdot v_i = \left(\sum_{i=1}^{\eta} v_i \right) \cdot \left(\sum_{i=1}^{\eta} v_i \right) \geq 0.$$

Since $v_i \cdot v_i = 1$, this implies that the left-most summation term above is at least $-\eta$. This term contains $\eta(\eta - 1)$ terms of the form $v_i \cdot v_j$, so the average value (among these terms) is at least $-1/(\eta - 1)$, which implies that the maximum is also at least this value. Since t^* corresponds to some feasible solution, this implies the lemma. □

We can formalize the connection between (SDP) and graph coloring using the Lovász Theta Function: let $\theta = 1 - 1/t^*$ where t^* is the optimal solution to (SDP). Then by Lemmas 1 and 2, we have $\eta \leq \theta \leq \chi$. (Notice that $\eta \leq \chi$ follows directly from their definitions.) In this next lecture, we will see how this inequality this us an algorithm for the graph coloring problem.

3 Summary

In this lecture, we studied algorithms for coloring a 3-colorable graph using a minimum number of colors. In particular, we gave an algorithm that uses $O(\sqrt{n})$ colors due to Wigderson [Wig83], and began looking at an SDP-based algorithm due to Karger, Motwani, and Sudan [KMS98].

References

[Blu94] Avrim Blum. New approximation algorithms for graph coloring. *Journal of the ACM (JACM)*, 41(3):470–516, 1994.

[KMS98] David Karger, Rajeev Motwani, and Madhu Sudan. Approximate graph coloring by semidefinite programming. *Journal of the ACM (JACM)*, 45(2):246–265, 1998.

[Wig83] Avi Wigderson. Improving the performance guarantee for approximate graph coloring. *Journal of the ACM (JACM)*, 30(4):729–735, 1983.