Optimization and Approximation ENS Lyon

Lecture 11

Lecturer: Alantha Newman *December 13, 2016*

1 Coloring 3-Colorable Graphs

In previous lectures, we illustrated how to formulate the maximum cut problem as a vector program (equivalently a semidefinite program (SDP)), by relaxing the variables in the exact formulation from scalars to vectors. We noted that this vector program can be solved up to an arbitrarily small additive error using the ellipsoid algorithm, and we showed how to round the vector solution to a feasible solution for the original problem to obtain a polynomial time approximation algorithm for this problem. In this lecture, we will continue exploring applications of vector programs by studying the problem of coloring a 3-colorable graph, which is a special instance of the general graph coloring problem.

In the graph coloring problem, we are given an undirected graph $G = (V, E)$. Our goal is to find a legal coloring (also called a proper coloring, or simply coloring) of the graph with the minimum number of colors. A legal coloring of a graph is a coloring where each vertex is assigned a color such that no two adjacent vertices have the same color. This problem is known to be $\mathcal{NP}$-hard and even finding an approximation with a factor better than $n^{1-\epsilon}$, for any $\epsilon > 0$, is $\mathcal{NP}$-hard. Instead, we focus our attention on special instances of the graph coloring problem in which we are given a k-colorable graph, and our goal is to find a legal coloring of this graph with the fewest number of colors. In this lecture, we will assume G is a 3-colorable graph. Note that it is still $\mathcal{NP}$-hard to decide if a graph can be colored with three colors. It is also $\mathcal{NP}$-hard to find a coloring of a 3-colorable graph with at most five colors. Here, we focus on finding a coloring that uses fewer than n colors. We will often use $\tilde{O}(\cdot)$ notion to hide log factors.

1.1 Coloring with $O(\sqrt{n})$ Colors

We denote the maximum degree of (any vertex in) a graph by Δ.

Lemma 1. *We can efficiently color any graph with $\Delta + 1$ colors, by greedily coloring the graph.*

Given a 3-colorable graph G, we can find a legal coloring using at most $O(\sqrt{n})$ colors.

1. If G has a vertex v with degree $d(v) \geq \sqrt{n}$, we use three new colors to color this vertex and its neighbors $\delta(v)$. Since G is 3-colorable, the neighbors of any vertex v form a bipartite graph (since none of these vertices can have the same color as vertex v). Thus, we can color the set $\delta(v)$ using two colors. We use a third color to color vertex v. We repeat this step until no vertices with degree higher than $\sqrt{n}$ remain.
2. Once G has $\Delta < \sqrt{n}$, we can apply Lemma 1 to color the rest of the graph using $\sqrt{n}$ colors.

Note that Step 1 will be executed at most $\frac{n}{\sqrt{n}}$-times, since at each iteration we are coloring at least $\sqrt{n}$ vertices. At each of these iterations we are using three new colors. So at the end of this phase, we will have used at most $3\sqrt{n}$ colors. In the second step, we use only $\sqrt{n}$ colors. So in total we will color the graph with at most $4\sqrt{n}$ colors.

1.2 $\tilde{O}(n^{0.631})$ Colors via Vector Program

To find a coloring of a k-colorable graph, we need to partition the graph into sets such that all edges are cut. We want to formulate this problem as a vector program, so we represent each vertex by a vector $v_i \in \mathbb{R}^n$. As we have seen in the previous lecture, if we are partitioning the vertices randomly, two vertices are more likely to be separated if their corresponding vectors are far away from each other, which corresponds to a small (negative) inner product, so our goal is to minimize the quantity $v_i \cdot v_j$ for any edge $(i, j) \in E$. We formulate the following vector program for the graph coloring problem of a k-colorable graph:

$$\begin{aligned} &\min \lambda \\ &\text{s.t } v_i \cdot v_j \leq \lambda \quad \forall (i,j) \in E \\ &\quad v_i \cdot v_i = 1 \quad \forall i \in V \\ &\quad v_i \in \mathbb{R}^n \end{aligned} \tag{P$_{color}$}$$

When G is 3-colorable, there exists a feasible solution for the above vector program when $\lambda = -\frac{1}{2}$. To see this, note that any legal 3-coloring of G will partition the vertex set into three *independent sets* (an independent set is a set where no two vertices are connected by an edge) where each set correspond to a color. An optimal exact solution for (P$_{color}$) consists of mapping each independent set to one of the three 'color' vectors, which are each at an angle of $\frac{2\pi}{3}$ from each other. Thus, any legal coloring of a 3-colorable graph yields a feasible solution for the following vector program:

$$\begin{aligned} &v_i \cdot v_j \leq -\frac{1}{2} \quad \forall (i,j) \in E \\ &v_i \cdot v_i = 1 \quad \forall i \in V \\ &v_i \in \mathbb{R}^n \end{aligned} \tag{P$_{3color}$}$$

Let G be a 3-colorable graph. To find a coloring of G, we will solve (P$_{3color}$) and then apply a randomized rounding strategy to color the vertices. The goal in rounding is to avoid using too many colors and at the same time avoid introducing monochromatic edges. We will use the following notion of a *semicoloring*.

Definition 2 (k-Semicoloring)**.** *A k-semicoloring of G is a k-coloring of the vertices such that at most $\frac{n}{4}$ edges have endpoints of the same color.*

Note that any k-semicoloring will directly result in a k-coloring of the rest of the graph after removing all the monochromatic edges, so we obtain a k-coloring of at least $\frac{n}{2}$ vertices.

Lemma 3. *If we can semicolor a graph G with k colors, then we can color G with $k \log n$ colors.*

Proof. Each round we use k colors and remove at least half the vertices. There are $\log n$ rounds, so we use at most $k \log n$ colors. □

Now we are ready to present our approximation algorithm:

1. Solve the vector program (P$_{3color}$) for G.
2. Set $t = 2 + \log_3 \Delta$ and pick t random vectors $\{r_1, r_2, \cdots, r_t\}$ where each entry r_{ij} is drawn from the standard normal distribution $\mathcal{N}(0, 1)$.
3. These t vectors define 2^t different regions based on the sign of the dot products with each of the t vectors. We color each of these regions with a different color.

Lemma 4. *Random hyperplane rounding with t hyperplanes produces a semicoloring using $4\ \Delta^{\log_3 2}$ colors with probability at least $1/2$.*

Proof. The number of colors we are using is $2^t = 4 \cdot 2^{\log_3 \Delta} = 4\Delta^{\log_3 2}$. Using the trivial upper bound of n on Δ, we see that this is at most $4n^{0.631}$ colors.

We want to show that our coloring procedure produces a semicoloring on G, so we need to show that the number of monochromatic edges is at most $\frac{n}{4}$ with probability at least 1/2.

$$\begin{aligned}
\Pr(\text{edge } (i,j) \text{ is monochromatic}) &= (1 - \frac{\arccos(v_i \cdot v_j)}{\pi})^t \\
&= (1 - \frac{\arccos(-0.5)}{\pi})^t \\
&= (1 - \frac{1}{\pi}\frac{2\pi}{3})^t \\
&= (\frac{1}{3})^t \\
&= \frac{1}{9\Delta}.
\end{aligned}$$

Let $m = |E|$. Then $m \leq \frac{n\Delta}{2}$. Let X be a random variable denoting the number of monochromatic edges in our coloring. Then:

$$\mathbb{E}[X] = \sum_{(i,j)\in E} \Pr(\text{edge } (i,j) \text{ is monochromatic}) \leq \frac{n\Delta}{2} \cdot \frac{1}{9\Delta} = \frac{n}{18}.$$

By Markov's inequality, we have:

$$\Pr[X \geq \frac{n}{4}] \leq \frac{\mathbb{E}[X]}{n/4} \leq \frac{2}{9}.$$

Then Lemma 4 holds with probability at least $\frac{7}{9}$. □

Therefore this algorithm results in a $O(n^{0.631} \log n) = \tilde{O}(n^{0.631})$-approximation which is a worse guarantee than that of the combinatorial algorithm described in Section 1.1, but it provides us with a way to combine the above algorithm with the combinatorial algorithm to get a $\tilde{O}(n^{0.387})$-approximation algorithm.

1.3 $\tilde{O}(n^{0.387})$ Colors

Let θ be some parameter that we define later. The following algorithm merges the two approximation algorithms we presented so far.

1. If G has a vertex v with degree $d(v) \geq \theta$, we use three new colors to color the vertex v and its neighbours $\delta(v)$. We repeat this step until no vertices with degree at least θ remain.

2. Once G has $\Delta < \theta$, we use the second algorithm to color the rest of the graph with $4\ \theta^{\log_3 2} \log n$ colors.

Step 1 will use at most $\frac{3n}{\theta}$ colors. Setting $\frac{n}{\theta} = \theta^{\log_3 2}$, we have $n = \theta^{\log_3 6}$. Then we define θ as $\theta = n^{\log_6 3} \approx n^{0.6131}$. In total, this approximation algorithm uses $\tilde{O}(\frac{n}{\theta}) = \tilde{O}(n^{0.387})$ colors.

2 Coloring via Large Independent Sets

We now explore a different method for coloring a 3-colorable graph that results in a coloring with even fewer colors. This method is based on the fact that in a legal coloring, a set of vertices of the same color forms an independent set. We use this fact to construct the following general algorithm:

1. Find an independent set I.
2. Color the vertices in I with a new color.
3. Remove I.
4. Repeat all the steps.

If in each iteration of Step 3 of the above algorithm, at least a γ-fraction of vertices are removed, then after k iterations, at most $(1-\gamma)^k n$ vertices remain. If we remove vertices until there is at least one vertex, then using $\log(1-x) \leq -x$ for $|x| > 1$, we derive k_{max}, the maximum number of iterations, as follows:

$$\begin{aligned} &(1-\gamma)^{k_{max}} n \geq 1 && \Rightarrow \\ &\log n - k_{max}\gamma \geq \log n + k_{max} \log(1-\gamma) \geq 0 && \Rightarrow \\ &k_{max} \leq \frac{1}{\gamma} \log n. \end{aligned}$$

This shows that an algorithm that can find an independent set whose size is at least a γ-fraction of the graph at each iteration uses $O(\frac{1}{\gamma}\log n)$ colors in total.

2.1 $\tilde{O}\left(n^{1/3}\right)$ Colors

As mentioned previously, we would like to map all the vertices onto vectors in a 2-dimensional plane, so that any two vectors corresponding to the two endpoints of an edge are 120 degrees apart. There are three such vectors, each corresponding to a single color, which are uniquely determined up to a rotation. Since such a solution corresponds to an optimal integral solution, it cannot be obtained as the solution to any efficiently solvable mathematical program. Therefore, the vector program (P_{3color}) relaxes the required dimension of the vectors onto which the vertices are mapped. However, in a solution for the relaxation, there may be more than three vectors satisfying the constraints, and it is no longer straightforward to map these vectors to colors. We do know that two vectors v_i and v_j representing an edge are far away from each other, i.e. $v_i \cdot v_j = -1/2$ for every $(i,j) \in E$. This structural guarantee was exploited in the analysis of the algorithm presented in Section 1.2. We now explore another way to use this geometric fact. We randomly sample an n-dimensional vector r and consider vectors that are close to it. Intuitively, vectors that are close to r should also be close to each other and therefore form an independent set. More formally, let

$$r = (r_1, \ldots, r_n) \text{ such that } r_i \in \mathcal{N}(0,1),\ \forall i \in \{1, \ldots, n\},$$

and define

$$S(\epsilon) = \{i \in V : v_i \cdot r \geq \epsilon\},$$

where ϵ will be determined later. Note that $S(\epsilon)$ might not be an independent set (but is "close" to one). We also define

$$S'(\epsilon) = \{i \in S(\epsilon) : i \text{ has no neighbors in } S(\epsilon)\},$$

which *is* an independent set. (Observe that a "smarter" way to define $S'(\epsilon)$ would be to find a maximal matching in $S(\epsilon)$ and delete the vertices defining the matching. However, it would make our analysis more complicated.) Next, our goal is to estimate the size of $S'(\epsilon)$.

First we recall some basic properties about the normal distribution $\mathcal{N}(0,1)$ such as its probability density and cumulative distribution functions:

$$p(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}},$$

$$\Phi(x) = \int_{-\infty}^{x} p(s)\mathrm{d}s \quad \Rightarrow \quad \bar{\Phi}(x) = 1 - \Phi(x) = \int_{x}^{\infty} p(s)\mathrm{d}s.$$

Observe that $v_i \cdot r$, for some vertex $i \in V$, does not depend on the vector v_i since the distribution of r is spherically symmetric. Therefore, to study the distribution of $v_i \cdot r$, we can rotate v_i so that v_i is any vector and without loss of generality, we can assume that $v_i = (1, 0, \ldots, 0)$, and thus $v_i \cdot r = r_1$. Then, $v_i \cdot r$ is distributed as $\mathcal{N}(0,1)$. This implies that:

$$\Pr[i \in S(\epsilon)] = \bar{\Phi}(\epsilon) \quad \Rightarrow \quad \mathbb{E}[\ |S(\epsilon)|\ ] = n\,\bar{\Phi}(\epsilon). \tag{1}$$

Now, we prove the following lemma, which gives an upper bound on the probability that a vertex i "fails", i.e. is not included in $S'(\epsilon)$, the independent set found by our algorithm. Let us now compute the probability that vertex i does not belong to $S'(\epsilon)$ given that it belongs to $S(\epsilon)$.

Lemma 5.
$$\Pr[i \notin S'(\epsilon) \mid i \in S(\epsilon)] \leq \Delta\bar{\Phi}\left(\sqrt{3}\epsilon\right).$$

Proof. We have

$$\Pr[i \notin S'(\epsilon) \mid i \in S(\epsilon)] = \Pr[\exists (i,j) \in E : v_j \cdot r \geq \epsilon \mid v_i \cdot r \geq \epsilon]. \tag{2}$$

From $v_i \cdot v_j = -1/2$ and $v_i \cdot v_i = v_j \cdot v_j = 1$, we conclude that v_j can be written in the following way

$$v_j = -\frac{1}{2}v_i + \frac{\sqrt{3}}{2}u, \text{ such that } v_i \cdot u = 0 \text{ and } u \cdot u = 1.$$

The last equation implies
$$u = \frac{2}{\sqrt{3}}\left(v_j + \frac{1}{2}v_i\right).$$

If $v_i \cdot r \geq \epsilon$ and $v_j \cdot r \geq \epsilon$, then

$$u \cdot r = \frac{2}{\sqrt{3}}\left(v_j \cdot r + \frac{1}{2}v_i \cdot r\right) \geq \frac{2}{\sqrt{3}}\left(\epsilon + \frac{1}{2}\epsilon\right) = \sqrt{3}\epsilon.$$

Thus, we see that *if* $i \in S(\epsilon)$ *and* $j \in S(\epsilon)$ (i.e. if both $v_i \cdot r$ and $v_j \cdot r$ are at least ϵ), then it must be the case that the projection of r onto u is at least $\sqrt{3}\epsilon$. (However, note that if $r \cdot u \geq \sqrt{3}\epsilon$, then this does necessarily imply that both $r \cdot v_i$ and $r \cdot v_j$ are at least ϵ.)

Since $u \cdot v_i = 0$, $r \cdot u$ and $r \cdot v_i$ are independently distributed. Thus, we have:

$$\begin{aligned}\Pr[v_j \cdot r \geq \epsilon \mid v_i \cdot r \geq \epsilon] &\leq \Pr[u \cdot r \geq \sqrt{3}\epsilon \mid v_i \cdot r \geq \epsilon] \\ &= \Pr[u \cdot r \geq \sqrt{3}\epsilon] \\ &= \bar{\Phi}(\sqrt{3}\epsilon).\end{aligned}$$

To conclude the proof, we use a union bound over the neighbors of i. Since there are at most Δ of them, we have:

$$\Pr\left[\exists (i,j) \in E : v_j \cdot r \geq \epsilon \mid v_i \cdot r \geq \epsilon\right] \leq \sum_{j:(i,j)\in E} \Pr[v_j \cdot r \geq \epsilon \mid v_i \cdot r \geq \epsilon] \leq \Delta\bar{\Phi}\left(\sqrt{3}\epsilon\right),$$

which implies the lemma. $\square$

So, how should we choose ϵ? Intuitively, a smaller value of ϵ should result in the smaller number of edges in $S(\epsilon)$ (since two adjacent vertices should be far from each other) and therefore a smalled number of "failed" vertices in $S(\epsilon)$. On the other hand, we would like $S(\epsilon)$ to be large. We will set ϵ so that

$$\Delta\bar{\Phi}\left(\sqrt{3}\epsilon\right) \leq 1/2. \tag{3}$$

This will upper bound the probability that a vertex in $S(\epsilon)$ does not belong to $S'(\epsilon)$. Using (1) and Lemma 5, we have:

$$\mathbb{E}[|\overline{S'(\epsilon)} \cap S(\epsilon)] \quad \leq \quad \frac{\mathbb{E}[|S(\epsilon)|]}{2} \;=\; n\frac{\Phi(\epsilon)}{2},$$

which we can use to lower bound the expected size of our independent set:

$$\mathbb{E}[|S'(\epsilon)|] \quad \geq \quad n\frac{\bar{\Phi}(\epsilon)}{2}. \tag{4}$$

Before we state the main result of this section, we state the following lemma without a proof.

Lemma 6. *For $x > 0$,*

$$\frac{x}{1+x^2}p(x) \leq \bar{\Phi}(x) \leq \frac{1}{x}p(x).$$

Now we can state the following theorem which lower bounds the expected size of $S'(\epsilon)$.

Lemma 7. *If $\epsilon = \sqrt{\frac{2}{3}\ln\Delta}$, then $\Delta\bar{\Phi}\left(\sqrt{3}\epsilon\right) \leq 1/2$.*

Proof. By Lemma 6 we have

$$\begin{aligned}
\bar{\Phi}\left(\sqrt{3}\epsilon\right) &\leq \frac{1}{\sqrt{3}\epsilon}\frac{1}{\sqrt{2\pi}}e^{-\frac{3\epsilon^2}{2}} \\
&= \frac{1}{\sqrt{2\ln\Delta}}\frac{1}{\sqrt{2\pi}\Delta} \\
&= \frac{1}{2\Delta}\frac{1}{\sqrt{\pi\ln\Delta}} \\
&< \frac{1}{2\Delta}.
\end{aligned}$$

So setting ϵ as in the statement of the lemma, we see that (3) holds. □

Theorem 8.

$$\mathbb{E}[|S'(\epsilon)|] \geq \Omega\left(n\,\Delta^{-\frac{1}{3}}(\ln\Delta)^{-\frac{1}{2}}\right).$$

Proof. To estimate the size of $S'(\epsilon)$, we must lower bound the value of $\bar{\Phi}(\epsilon)$. We obtain the following bound by applying Lemma 6 and using inequality $\frac{x}{1+x^2} \geq \frac{1}{2x}$, for $x \geq 1$, as follows

$$\begin{aligned}
\bar{\Phi}(\epsilon) &\geq p(\epsilon)\frac{\epsilon}{1+\epsilon^2} \\
&= \frac{1}{\sqrt{2\pi}}e^{-\frac{\epsilon^2}{2}}\frac{\epsilon}{1+\epsilon^2} \\
&\geq \frac{1}{\sqrt{2\pi}}e^{-\frac{\ln\Delta}{3}}\frac{1}{2\epsilon} \\
&\geq \frac{\sqrt{3}}{4\sqrt{\pi}}\cdot\Delta^{-\frac{1}{3}}\left(\ln\Delta\right)^{-\frac{1}{2}} \\
&= \Omega\Big(\Delta^{-\frac{1}{3}}\left(\ln\Delta\right)^{-\frac{1}{2}}\Big).
\end{aligned}$$

Applying the inequality in Equation (4) concludes the proof. □

Theorem 8 says that in each iteration of the algorithm, we are able to find an independent set of size at least $\tilde{O}\left(n\,\Delta^{-1/3}\right)$, which is a $\frac{1}{\Delta^{1/3}}$-fraction of the graph. This implies that we can color a given 3-colorable graph with $O\left(\Delta^{1/3}\log n\right) = \tilde{O}\left(\Delta^{1/3}\right)$ colors. We summarize the given approach, which we call Independent-Set-Coloring, as follows:

```
Independent-Set-Coloring
  1. Solve $(\mathrm{P}_{3color})$.
  2. Pick a random vector $r$.
     (a) Pick a set $S(\epsilon)$ which is $\epsilon$-close to $r$.
     (b) Let $S'(\epsilon) \subseteq S(\epsilon)$ be vertices with degree zero in $S(\epsilon)$.
     (c) Remove the vertices in the independent set $S'(\epsilon)$ from the graph.
  3. Repeat from Step 2 while the graph is non-empty.
```

Note that there is no need to resolve (P_{3color}) in Step 1 at each iteration, since a solution for the initial graph is valid for any subgraph.

2.2 $\tilde{O}(n^{\frac{1}{4}})$ Colors

Finally, we combine the algorithm from Section 1.1 with the algorithm Independent-Set-Coloring.

1. If G has a vertex v with degree $d(v) \geq n^{3/4}$, we use three new colors to color the vertex v and its neighbours $\delta(v)$. We repeat this step until no vertices with degree at least $n^{3/4}$ remain.
2. Once G has $\Delta < n^{3/4}$, we use the algorithm Independent-Set-Coloring to color the rest of the graph with at most $\tilde{O}((n^{3/4})^{1/3}) = \tilde{O}(n^{1/4})$ colors.

Observe that Step 1 will execute at most $n/n^{3/4} = n^{1/4}$ times, using at most $n^{1/4}$ colors. Thus, the total number of colors used is at most $\tilde{O}(n^{1/4})$.

References

[ACC06] Sanjeev Arora, Eden Chlamtac, and Moses Charikar. New approximation guarantee for chromatic number. In Jon M. Kleinberg, editor, *STOC*, pages 215–224. ACM, 2006.

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

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

[WS11] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms.* Cambridge University Press, 2011.

This lecture is based on Chapters 6 and 13 from [WS11]. The algorithm in Section 1.1 is due to Wigderson [Wig83]. The algorithm in Section 2 is due to Karger, Motwani and Sudan [KMS98, ACC06]. A previous version of these lecture notes scribed by Marwa El Halabi and Slobodan Mitrović was used in a course on approximation algorithms (Lecture 15) at EPFL (`http://theory.epfl.ch/osven/courses/Approx13`).