CS 466/666: Algorithm Design and Analysis **University of Waterloo: Fall 2025**

Lecture 13

October 23, 2025

Instructor: Sepehr Assadi *Scribe: Helia Yazdanyar*

Topics of this Lecture

In the previous lecture, we introduced Semi-definite Programming (SDP) and saw how it can be used to obtain an algorithm for the Max-Cut problem. In this lecture, we use this technique to design an algorithm for graph coloring.

1 The 3-Coloring Problem

We study the 3-coloring problem, namely, the problem of coloring the vertices of a graph using three colors such that no two adjacent vertices have the same color. Formally, a **3-coloring** of a graph $G = (V, E)$ is a function $\phi : V \to \{1, 2, 3\}$ such that $\phi(u) \neq \phi(v)$ for every edge $(u, v) \in E$.

Deciding whether a graph is 3-colorable or not is an NP-hard problem. In this lecture, we want to explore if we can use SDPs to color G with a "good" number of colors. Here by "good", we mean some number that is not trivial. For example we can always color the graph using n colors (where n is the number of vertices in the graph), or by combinatorial bounds we know we can always color the graph using $\Delta + 1$ colors where Δ is the maximum degree in the graph.

1.1 Independent Sets and Coloring

A set $U \subseteq V$ is called an *independent set* if no two vertices in U are adjacent. A k-coloring of G can be viewed as a partition of V into k independent sets.

Suppose G is k-colorable. It is easy to see that in that case G has an independent set of size n/k – simply take the color class in its k-coloring with the largest number of colors. We can also prove a somewhat inverse of this statement: if we can always find an independent set of size $1/k$-fraction of vertices in a given graph, then, we will be able to color it with $O(k \log n)$ colors.

Lemma 1. *Let $G = (V, E)$ be an n-vertex graph and k be a fixed parameter. Suppose we have an algorithm A that given any subgraph G' of G with n' vertices, finds an independent set of size at least n'/k in G'. Then, we can use A to obtain an $O(k \log n)$ coloring of G.*

Proof. Consider the following algorithm B: let $U = \emptyset$ and $G' = G$. While G' is non-empty, run $A(G')$ to obtain an independent set S, color S with a new color and update $G' = G' \setminus S$.

The output of algorithm B is clearly a proper coloring of G since all vertices eventually belong to some set S, and we always color an independent set with the same color, so this is proper coloring. It thus remains to bound the number of colors used in B. This is equal to the number of iterations of the while-loop; since each step reduces the vertices to at most $(1-1/k)$ fraction of what it was, after the first $t = k \ln n$ iterations, we have,

$$\text{size of } G' \leqslant \left(1 - \frac{1}{k}\right)^t \cdot n \leqslant e^{-\frac{t}{k} + \ln n} = e^{-\ln n + \ln n} = 1.$$

In the next iteration then G' becomes empty (as a singleton vertex is always an independent set). Thus, the algorithm uses $O(k \log n)$ colors. □

We now go back to our 3-coloring problem: Our goal is to find a "large" independent set in a 3-colorable graph G with maximum degree Δ. Once we can do that, we can apply Lemma 1 to obtain a proper coloring of G with "small" number of color also.

2 Finding Large Independent Sets in 3-Colorable Graphs

2.1 A Simple Randomized Algorithm

One simple approach for finding a large independent set is the following:

Algorithm 1 (A simple randomized algorithm for independent set in arbitrary graphs)**.**

1. Sample each vertex independently with probability p.
2. Let S be the set of sampled vertices.
3. Remove both endpoints of any edge inside S (keeping only the independent vertices).
4. Return the resulting independent set I.

We know that each vertex is sampled with probability p inside S, so we have

$$\mathbb{E}\,|S| = n \cdot p.$$

Also we can calculate the probability that an edge $e = (u, v)$ is inside S, i.e., both its endpoints belong to S, as below

$$\Pr(e \text{ is sampled}) = p^2. \tag{1}$$

Thus, since G can have at most $n\Delta/2$ edges given its maximum degree is Δ,

$$\mathbb{E}\,|E(S)| = \sum_e \Pr(e \text{ is sampled}) \leqslant \frac{n\Delta}{2} \cdot p^2.$$

By linearity of expectation, for any p, we can calculate the expected size of the set I returned by Algorithm 1:

$$\mathbb{E}\,|I| \geqslant \mathbb{E}\,|S| - 2 \cdot \mathbb{E}\,|E(S)| \geqslant n \cdot p - 2 \cdot \frac{n \cdot \Delta}{2} \cdot p^2.$$

By choosing p such that the second term is half of the first one, we get that $p = 1/2\Delta$ and have

$$\mathbb{E}\,|I| \geqslant \frac{n}{2\Delta} - \frac{n \cdot \Delta}{4\Delta^2} = \frac{n}{4\Delta}.$$

The bounds obtained in this algorithm are quite weak: we can always find an independent set of size $n/(\Delta+1)$ in any given graph of maximum degree Δ; pick a vertex in the independent set, remove itself and all its neighbors and recurse. This way, we add one vertex to the independent set and remove at most $\Delta + 1$ vertices. Thus, at the end, we will find an independent set of size $n/(\Delta + 1)$. Nevertheless, we will see that the strategy of this randomized algorithm can be used later to get much better bounds.

We should also note that one reason the bounds obtained by this algorithm—or even the greedy one mentioned above—are so weak is because we are not using the fact that the graph is 3-colorable at all. In an arbitrary graph of max-degree Δ, we actually cannot hope for a bound better than $n/(\Delta+1)$ in general. Can you see why?

Remark. A more naive way of creating an independent set via a strategy similar to Algorithm 1 was to pick the set S the same way, but then bound the probability that S is an independent set already; i.e., making sure there are no edges inside S at all. In that case, we needed to have

$$\Pr\left(\text{there is an edge inside } S\right) \leqslant \sum_{e} \Pr\left(e \text{ is sampled}\right) = \frac{n\Delta}{2} \cdot p^2.$$

We thus need this probability to be less than one which implies that

$$p < \sqrt{\frac{2}{n\Delta}}.$$

But then, this would give us

$$\mathbb{E}\,|S| = n \cdot p = \sqrt{\frac{2n}{\Delta}},$$

which is even quadratically worse than the bounds of Algorithm 1.

The approach of Algorithm 1 is often called **alteration method** in randomized algorithms and the probabilistic method. Instead of creating the object in one go probabilistically, we first use a randomized process to "get close to" the object and then *alter* it deterministically to obtain the final object (in Algorithm 1, getting "close" to an independent set meant finding a set that has very few edges inside it).

2.2 A Better Algorithm Tailored to 3-Colorable Graphs

We start by presenting a simple algorithm due to Wigderson [Wig83] that finds an independent set of size $\Omega(\sqrt{n})$ in any 3-colorable graphs, regardless of the maximum degree of the graph.

The algorithm is based on the following observation: given any vertex v, the neighborhood $N(v)$ of v is 2-colorable (because color of v cannot be used on these vertices). But then any 2-colorable graph is bipartite and by finding its bipartition (which can be done easily using DFS/BFS search in polynomial time), we can find an independent set that contains at least half its vertices. This implies that when maximum degree is Δ, we can always find an independent set of size at least $\Delta/2$ in G. But, combined with the previous greedy algorithm (or Algorithm 1 for a slightly weaker result), this implies that we can always find an independent set of size

$$\min\left\{\frac{\Delta}{2}\ ,\ \frac{n}{\Delta+1}\right\} = \Omega(\sqrt{n}),$$

in a 3-colorable graph.

We leave turning this algorithm into a one that finds an $O(\sqrt{n})$-coloring of any 3-colorable graphs as an exercise to the reader. Just note that applying Lemma 1 might not be the most efficient way here and in that lemma, we required k to be fixed where as $\sqrt{n}$ depends on the number of vertices; it will be a lot easier to just directly turn this algorithm into a coloring one.

2.3 Finding Independent Sets via SDPs

We now switch to our main approach which finds a large independent set in a 3-colorable graphs using Semi-Definite Programming (SDP). We will prove the following result, due to Karger, Motwani, and Sudan [KMS98], in this section.

Theorem 2 ([KMS98]). *There exists a polynomial time algorithm that given any 3-colorable graph G with maximum degree Δ, outputs an independent set in G of expected size*

$$\Omega\left(\frac{n}{\Delta^{1/3}\cdot\log\Delta}\right).$$

Combining this algorithm with the approach of Lemma 1, we can also obtain $O(\Delta^{1/3}\cdot\log\Delta\cdot\log n)$ coloring of any 3-colorable graph in polynomial time. We will prove this theorem in the rest of this lecture.

We start with a simple claim: if $G=(V,E)$ admits a 3-coloring, then, we can assign unit-length vectors $b_v\in\mathbb{R}^2$ to vertices of v such that for any edge (u,v), the angle between b_u and b_v is $2\pi/3$. See Figure 1.

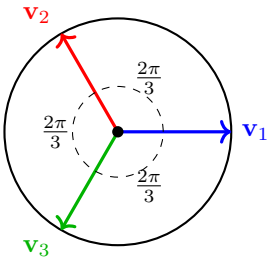

Figure 1: Fix a 3-coloring of G. For any vertex v with color $i\in[3]$, we let $b_v=\mathbf{v}_i$. The vectors $\mathbf{v}_i$ for $i\in[3]$ have the same angle of $2\pi/3$ with each other.

Recall that for any two vectors $x,y\in\mathbb{R}^2$:

$$\langle x\,,\,y\rangle=\|x\|\cdot\|y\|\cdot\cos(\theta_{x,y}),$$

where $\theta_{x,y}$ is the angle between the two vectors. Given $\cos(2\pi/3)=-1/2$, we can write our previous observation as follows.

Observation 3. *In any 3-colorable graph G, there exists vectors $b_v\in\mathbb{R}^2$ for vertices $v\in V$ such that*

$$\begin{aligned}&\|b_v\|=1, && \forall v\in V,\\ &\langle b_u\,,\,b_v\rangle=-1/2, && \forall(u,v)\in E.\end{aligned}\tag{2}$$

It is also easy to see that if we can find such vectors for any graph, we can immediately find a 3-coloring of the graph. The problem of course is that we do not know how to find such vectors efficiently. But, similar to the last lecture, we can *relax* this problem to obtain an SDP.

Firstly, define a symmetric matrix $X\in\mathbb{R}^{n\times n}$, wherein we interpret $X_{uv}=\langle b_u,b_v\rangle$. We can thus write Eq (2) as the following "matrix program": Find a matrix $X\in\mathbb{R}^{n\times n}$ such that:

$$\begin{aligned}&X_{vv}=1, && \forall v\in V,\\ &X_{uv}=-\tfrac{1}{2}, && \forall(u,v)\in E,\\ &X=BB^\top,\end{aligned}$$

for some matrix $B \in \mathbb{R}^{n\times 2}$, where the v-th row of B will be a vector $b_v \in \mathbb{R}^2$. This program is still equivalent to that of Eq (2) and we still do not know how to solve it. But we can see that the constraint $X = BB^\top$ forces X to be a PSD matrix. Thus, by relaxing the constraint to simply require X to be PSD, we obtain our final program:

$$\begin{aligned} &X_{vv} = 1, \quad \forall v \in V, \\ &X_{uv} = -\tfrac{1}{2}, \quad \forall (u,v) \in E, \\ &X \succeq 0. \end{aligned} \tag{3}$$

The problem in (3) is now a proper SDP program (wherein we do not have any objective, the goal is to simply find a feasible point in this program; if you like, you can think of this as an SDP where the objective can be anything, say, maximize $X_{1,1}$).

Observation 3, together with the fact that (3) is a relaxation of (2), implies that as long as G is 3-colorable, the resulting SDP has a feasible solution. We now use this fact to design a *rounding* scheme that given any solution to the SDP (3), finds a large independent set in G.

2.3.1 Rounding SDP (3) via random separators

Having the vectors as described, we want to use an idea similar to the one of last lecture, to separate the vertices and construct a large independent set. In other words, given the SDP solution X we want to pick a random vector r in the sphere of all vectors corresponding to the vertices, and then put all vectors of one side of r in set S as the resulting independent set. We need to pick this vector r such that the expected size of the resulting independent set is large. The algorithm is as follows:

Algorithm 2 (Finding an independent set using the SDP solution)**.**

1. Solve the SDP in (3) to get a PSD matrix X satisfying the constraints (if the SDP is not feasible, return G is not 3-colorable).
2. Recall that any PSD matrix can be written as $X = BB^\top$ for some $B \in \mathbb{R}^{n\times n}$, thus

$$X = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}_{n\times n} \quad \begin{bmatrix} b_1 & b_2 & \cdots & b_n \end{bmatrix}_{n\times n}$$

   where each $b_i \in \mathbb{R}^n$.
3. Pick a $r \in \mathbb{R}^n$ from Gaussian distribution where $r_i \sim N(0,1)$ independently for each $i \in [n]$.
4. For a fixed parameter $c \geqslant 2$ to be determined later, let:

$$S = \{v_i \in V : \langle b_i, r\rangle \geqslant c\}.$$

5. Let

$$F = \{(v_i, v_j) \in E : \langle b_i, r\rangle \geqslant c \quad \wedge \quad \langle b_j, r\rangle \geqslant c, \}.$$

6. Return $I := S \setminus V(F)$

See Figure 2 for an illustration of Algorithm 2. We first note that the set I returned by the algorithm is always an independent set, since F is the set of all edges inside S and we are removing all those edges by removing their vertices. It thus remains to bound the size of I. We do so in the following by using an approach similar to that of Section 2.1 but this time, by relating these probabilities to well-known properties of the Gaussian distribution.

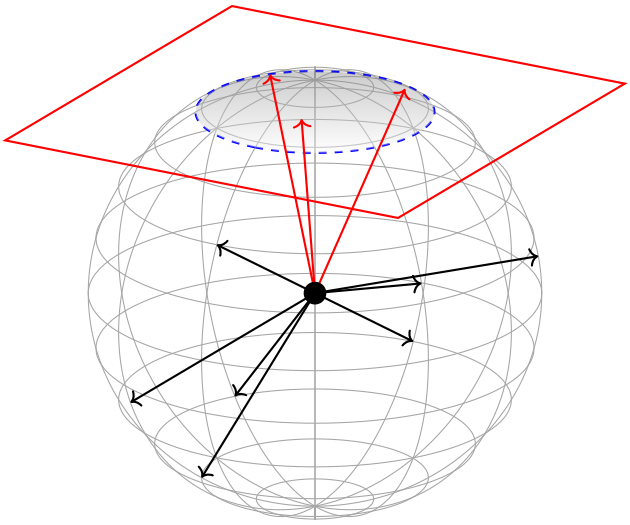

Figure 2: An illustration of the rounding scheme in 3 dimensions (all vectors here are unit-length in 3D and end at the boundary of the sphere). The vertices in the independent set are the ones whose vector are cut by the hyperplane.

Let us first state a standard but highly useful property of Gaussian distribution: sum of two independent Gaussians is also a Gaussian.

Fact 4. *For any two independently sampled Gaussians, we have,*

$$N(\mu_1, \sigma_1^2) + N(\mu_2, \sigma_2^2) = N(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2).$$

We first focus on the probability that a vertex v joins S.

Claim 5. *For any vertex $v \in V$,*

$$\Pr(v \in S) = \Pr(N(0,1) \geqslant c),$$

where the RHS is the probability that a random Gaussian entry from $N(0,1)$ has value at least c.

Proof. Each vertex $v \in V$ has a vector $b_v \in \mathbb{R}^n$ in the solution $X = BB^\top$ where $\|b_v\| = 1$. Since vector $r \in \mathbb{R}^n$ is sampled from the Gaussian distribution, and since sum of independent Gaussians is itself a Gaussian (Fact 4), we have,

$$\begin{aligned}
\Pr(v \in S) &= \Pr_{r \in N(0,1)^n}\left(\langle b_v, r\rangle \geqslant c\right) && \text{(by the definition of the set } S\text{)} \\
&= \Pr\left(\sum_{i=1}^{n} N(0,1) \cdot b_{v,i} \geqslant c\right) && \text{(by writing the inner product as the weighted sum of Gaussians)} \\
&= \Pr\left(\sum_{i=1}^{n} N(0, b_{v,i}^2) \geqslant c\right) \\
&\quad \text{(as } a \cdot N(0,1) \text{ is distributed as } N(0,a^2)\text{; recall multiplying a variable by } a \text{ changes its variance by } a^2\text{)} \\
&= \Pr\left(N(0, \|b_v\|^2) \geqslant c\right) && \text{(by the aforementioned property of Gaussians and definition of } \|b_v\|^2\text{)} \\
&= \Pr(N(0,1) \geqslant c). && \text{(as } \|b_v\| = 1\text{)}
\end{aligned}$$

□

Next, we bound the probability that an edge (u, v) belongs to F, i.e., has both endpoints in S.

Claim 6. *For any edge $(u,v) \in E$,*

$$\Pr\left((u,v) \in F\right) \leqslant \Pr\left(N(0,1) \geqslant 2c\right).$$

Proof. Let b_u, b_v be the vectors of u, v from $X = BB^\top$. We know $\|b_u\| = \|b_v\| = 1$ and $\langle b_u\,,\, b_v\rangle = -1/2$ which implies that

$$\|b_u + b_v\|^2 = \|b_u\|^2 + \|b_v\|^2 + 2\langle b_u\,,\, b_v\rangle = 1 + 1 + 2\cdot(-1/2) = 1.$$

Using this, we have,

$$\begin{aligned}
\Pr\left(v \in S\right) &= \Pr_{r\in N(0,1)^n}\Big(\langle b_u\,,\, r\rangle \geqslant c \wedge \langle b_v\,,\, r\rangle \geqslant c\Big) && \text{(by the definition of the set } F\text{)}\\
&\leqslant \Pr_{r\in N(0,1)^n}\Big(\langle b_u + b_v\,,\, r\rangle \geqslant 2c\Big) \\
&&& \text{(as each term being larger than } c \text{ implies the sum is larger than } 2c\text{)}\\
&= \Pr\left(N(0, \|b_u + b_v\|^2 \geqslant 2c\right) && \text{(exactly as argued in Claim 5)}\\
&= \Pr\left(N(0,1) \geqslant 2c\right). && \text{(as } \|b_u + b_v\| = 1\text{)}
\end{aligned}$$

□

We are now almost done. Both the RHS of Claim 5 and Claim 6 has closed form analytical solutions from existing literature on Gaussian distribution; thus, we simply need to "plug in" these standard terms and then follow the same exact strategy as in Section 2.1. This is what the rest of this proof is going to do. For that, we need the following fact about the Gaussian distribution.

Fact 7. *For any $t \geqslant 1$,*

$$\left(\frac{1}{t} - \frac{1}{t^3}\right)\cdot\frac{1}{2\sqrt{\pi}}\cdot e^{-t^2/2} \leqslant \Pr\left(N(0,1) \geqslant t\right) \leqslant \frac{1}{t}\cdot\frac{1}{\sqrt{2\pi}}\cdot e^{-t^2/2}.$$

Proof Sketch. Using the the definition of the PDF $p(\cdot)$ of the Gaussian distribution, we have,

$$\Pr\left(N(0,1) \geqslant t\right) = \int_t^\infty p(x)\,dx = \int_t^\infty \frac{1}{\sqrt{2\pi}}e^{-x^2/2}\,dx = \frac{1}{\sqrt{2\pi}}\cdot e^{-t^2/2}\cdot\left(\frac{1}{t} - \frac{1}{t^3} + O(\frac{1}{t^5})\right),$$

where the final part comes from the tailor expansion, and in turn gives us the bound we wanted. □

As in Section 2.1, we can now calculate

$$\begin{aligned}
\mathbb{E}\,|I| &\geqslant \mathbb{E}\,|S| - 2\cdot\mathbb{E}\,|F| \\
&= \sum_{v\in V}\Pr\left(v \in I\right) - 2\cdot\sum_{(u,v)\in E}\Pr\left((u,v)\in F\right) && \text{(by linearity of expectation)}\\
&\geqslant n\cdot\Pr\left(N(0,1)\geqslant c\right) - 2\cdot\frac{n\Delta}{2}\cdot\Pr\left(N(0,1)\geqslant 2c\right) && \text{(by Claim 5 and Claim 6)}\\
&\geqslant n\cdot\frac{1}{2c}\cdot\frac{1}{2\sqrt{\pi}}\cdot e^{-c^2/2} - n\Delta\cdot\frac{1}{2c}\cdot\frac{1}{\sqrt{2\pi}}\cdot e^{-(2c)^2/2}. && \text{(by Fact 7 and since } c\geqslant 2\text{)}
\end{aligned}$$

We again would like the second term to be half of the first term. Thus, we would like to pick a choice of $c \geqslant 2$ to satisfy

$$n\cdot\frac{1}{2c}\cdot\frac{1}{2\sqrt{\pi}}\cdot e^{-c^2/2} = 2n\Delta\cdot\frac{1}{2c}\cdot\frac{1}{\sqrt{2\pi}}\cdot e^{-(2c)^2/2},$$

which is equivalent to

$$e^{-c^2/2} = 2\Delta\cdot e^{-4c^2/2}$$

which, by taking the $\ln(\cdot)$ on both sides, gives us

$$c = \sqrt{\frac{2}{3} \cdot \ln(2\Delta)};$$

(this also satisfies the requirement $c \geqslant 2$ as we know $\Delta = \omega(1)$; otherwise, we can find an independent set of size $\Omega(n)$ in the graph using the greedy algorithm).

To conclude, we now have

$$\begin{aligned} \mathbb{E}\,|I| &\geqslant n \cdot \frac{1}{2\sqrt{\frac{2}{3}\ln(2\Delta)}} \cdot \frac{1}{2\sqrt{\pi}} \cdot e^{-\frac{1}{3}\cdot\ln(2\Delta)} && \text{(by our choice of the parameter } c\text{)} \\ &= \Omega\left(\frac{n}{\Delta^{1/3} \cdot \log \Delta}\right), \end{aligned}$$

as desired. This concludes the proof of Theorem 2.

Remark. Let us put this result in the context of the simple algorithm of Section 2.1. Define

$$p := \Pr\left(N(0,1) \geqslant c\right),$$

for $c \geqslant 2$ to be determined later. Given that "effectively" (although not accurately), for any $x \geqslant 1$,

$$\Pr\left(N(0,1) \geqslant x\right) \approx e^{-x^2/2},$$

we can "almost" say that

$$\Pr\left(N(0,1) \geqslant 2c\right) \approx e^{-(2c)^2/2} = \left(e^{-c^2/2}\right)^4 \approx p^4.$$

Thus, going through the calculations of Section 2.1, we get

$$\mathbb{E}\,|I| \gtrsim n \cdot p - n\Delta \cdot p^4,$$

which suggests we should set $p \approx \Delta^{-1/3}$ and get $\mathbb{E}\,|I| \gtrsim n/\Delta^{1/3}$. This is exactly what our actual argument did modulo fixing these informal approximation-terms with actual bounds.

But going back to Section 2.1, there, we saw that naive sampling is enough to ensure each vertex joins S with probability p and each edge joins S with probability p^2 (for some p). The new rounding scheme now ensures that while each vertex still joins S with probability p, each edge joins S with probability $\approx p^4$, so a much lower probability. In other words, looking at the probabilities defined by the SDP, there is a very *negative correlation* between the choice of vertices on endpoints of an edge. This was the key source of improvement in this algorithm compared to that of Section 2.1.

References

[KMS98] David R. Karger, Rajeev Motwani, and Madhu Sudan. Approximate graph coloring by semidefinite programming. *J. ACM*, 45(2):246–265, 1998. 4

[Wig83] Avi Wigderson. Improving the performance guarantee for approximate graph coloring. *J. ACM*, 30(4):729–735, 1983. 3