Chapter 12

Semidefinite Programming II: Vertex Coloring

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

12.1 The 3-Coloring Problem

Recall that for any $k \geqslant 1$, a ***k*-coloring** of a graph $G = (V, E)$ is a function $\phi : V \to \{1, \ldots, k\}$ such that $\phi(u) \neq \phi(v)$ for every edge $(u, v) \in E$. We are interested in the 3-coloring problem in this chapter.

Problem 12.1. Given an undirected graph $G = (V, E)$ determine whether or not G admits a 3-coloring.

Deciding whether a graph is 3-colorable or not is an **NP**-hard problem. Thus, we will be interested in approximation algorithms for this problem in this chapter. As a benchmark, notice that we can always color the graph using n colors (where n is the number of vertices in the graph), or even $\Delta + 1$ colors where Δ is the maximum degree in the graph (recall our algorithm from Chapter 1).

Our main approach in this chapter will be to approximate graph coloring by finding "large" independent sets. We start with a quick overview of independent sets and their connection to graph coloring first.

12.1.1 Independent Sets and Coloring

In a graph $G = (V, E)$, 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 G has an independent set of size at least n/k also: simply take the color class in its k-coloring with the largest number of vertices. We can also prove a somewhat inverse to this statement: if we can always find an independent set of size $(1/k)$-fraction of vertices in a graph, then, we will be able to color it with $O(k \log n)$ colors.

Lemma 12.2. *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 $G' = G$ and 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 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. 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. Once we can do that, we can apply Lemma 12.2 to obtain a proper coloring of G with "small" number of colors.

12.2 Finding Large Independent Sets in 3-Colorable Graphs

We start by examining two algorithms for finding "large" independent sets in 3-colorable graphs.

12.2.1 A Simple Randomized Algorithm

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

Algorithm 12.3.

1. Sample each vertex independently with probability p. ▷ We will later fix $p \approx 1/\Delta$
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.

Lemma 12.4. *Given graph $G = (V, E)$ with maximum degree Δ, Algorithm 12.3 with parameter $p = 1/(2\Delta)$ outputs an independent set I with*

$$\mathbb{E}\,|I| \geqslant \frac{n}{4\Delta}.$$

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

$$\mathbb{E}\,|S| = n \cdot p \quad \text{and} \quad \Pr(\text{both endpoints of } e \text{ are sampled}) = p^2, \tag{12.1}$$

for every edge $e \in E$. As G can have at most $n\Delta/2$ edges given its maximum degree is Δ,

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

By linearity of expectation, we can calculate the expected size of the returned set I as:

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

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

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

concluding the proof. □

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. An even more naive way of creating an independent set via a strategy similar to Algorithm 12.3 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

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

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 12.3.

The approach of Algorithm 12.3 is often called the **alteration method** in randomized algorithms and the probabilistic method. Instead of creating the object entirely 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 12.3, getting close to an independent set meant finding a set that has very few edges inside it).

12.2.2 A Better Algorithm Tailored to 3-Colorable Graphs

We start by presenting a simple algorithm due to [Wig83] that finds an independent set of size $\Omega(\sqrt{n})$ in any 3-colorable graph, 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 12.3 for a slightly weaker result), this implies that we can always find an independent set of size

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

in a 3-colorable graph.

We leave turning this algorithm into one that finds an $O(\sqrt{n})$-coloring of any 3-colorable graphs as an exercise to the reader. Just note that applying Lemma 12.2 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.

12.3 Finding Independent Sets via SDPs

We now switch to our main approach in this chapter which finds a large independent set in a 3-colorable graph using semidefinite programming. We will prove the following result, due to [KMS98], in this section.

Theorem 12.5 ([KMS98]). *There is a randomized polynomial time algorithm that given any 3-colorable graph G with maximum degree Δ, outputs an independent set in G of expected size*

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

Combining Theorem 12.5 with Lemma 12.2, we obtain an $O(\Delta^{1/3}\cdot\sqrt{\log\Delta}\cdot\log n)$ coloring of any 3-colorable graph in polynomial time. We will prove this theorem in the rest of this section.

12.3.1 An SDP for 3-Colorable Graphs

We start with a basic claim: whenever $G=(V,E)$ is 3-colorable, we can assign unit-length vectors $b_v \in \mathbb{R}^2$ to vertices such that for any edge (u,v), the angle between b_u and b_v is $2\pi/3$. See Figure 12.1 for this assignment.

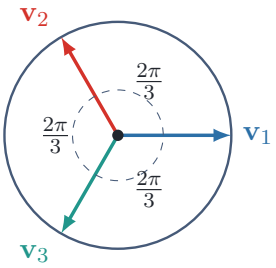

Figure 12.1: Fix a 3-coloring of G. For any vertex v with color $i \in [3]$, we let $b_v = \mathbf{v}_i$. The vectors $\mathbf{v}_i$ for $i \in [3]$ have the same angle of $2\pi/3$ with each other.

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

$$x^\top \cdot y = \|x\| \cdot \|y\| \cdot \cos(\theta_{x,y}),$$

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

$$\begin{aligned} &\|b_v\| = 1, && \text{for all vertices } v \in V,\\ &{b_u}^\top \cdot b_v = -1/2, && \text{for all edges } (u,v) \in E. \end{aligned} \tag{12.2}$$

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 chapter, we can *relax* this problem to obtain an SDP.

Firstly, define a symmetric matrix $X \in \mathbb{R}^{n \times n}$, wherein we interpret $X_{uv} = b_u^\top \cdot b_v$. We can thus write Eq (12.2) as the following "matrix program": Find a matrix $X \in \mathbb{R}^{n \times n}$ such that:

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

for some matrix $B \in \mathbb{R}^{2 \times n}$, where the v-th column of B will be a vector $b_v \in \mathbb{R}^2$. This program is equivalent to that of Eq (12.2) and we still do not know how to solve it. But we can see that the constraint $X = B^\top B$ forces X to be a PSD matrix (Proposition 11.5). Thus, by relaxing the constraint to simply require X to be PSD, we obtain our final program:

$$\begin{aligned} X_{vv} &= 1, && \forall v \in V, \\ X_{uv} &= -\tfrac{1}{2}, && \forall (u,v) \in E, \\ X &\succeq 0, \end{aligned} \tag{12.3}$$

The problem in (12.3) 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 12.6, together with the fact that (12.3) is a relaxation of (12.2), 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 (12.3), finds a large independent set in G.

Note that in general, as long as G is 3-colorable this SDP is always feasible but the inverse is certainly not true – it is possible that G is not 3-colorable and yet we can still find a solution to this SDP. Our rounding scheme shows that given *any* solution to this SDP, regardless of whether or not G is 3-colorable, we can always find a "large" independent set in G (of course, if we run this SDP and cannot find a feasible solution, we can simply return the original graph is not 3-colorable).

12.3.2 Rounding the SDP via Random Separators

Recall that any PSD matrix X can be written as $X = B^\top B$ by Proposition 11.5. Specifically, we can think of the matrix X of Eq (12.3) as

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

where each $b_i \in \mathbb{R}^n$. Thus, solving Eq (12.3) will give us the vectors $b_1, \ldots, b_n \in \mathbb{R}^n$ satisfying

$$\begin{aligned} \|b_i\| &= 1 && \forall v_i \in V, \\ b_i^\top \cdot b_j &= -\frac{1}{2} && \forall (v_i, v_j) \in E. \end{aligned} \tag{12.4}$$

Equipped with these vectors, we want to use an idea similar to the last Chapter to separate the vertices and construct a large independent set. We do this again by picking a random Gaussian vector r and put all vectors on one side of r in set S as the resulting independent set. A key difference with Algorithm 11.13 is that this time we are going to cut the unit ball not from the center, but actually much closer to one of the poles.

Algorithm 12.7.

1. Solve the SDP (12.3) to get a PSD matrix X and the vectors $b_1, \ldots, b_n \in \mathbb{R}^n$ in Eq (12.4) (if the SDP is not feasible, return G is not 3-colorable).
2. Sample a Gaussian vector $r \in \mathbb{R}^n$ where $r_i \sim \mathcal{N}(0,1)$ independently for $i \in [n]$.
3. For a fixed parameter $c \geqslant 2$ to be determined later, let: ▷ We will set $c \approx \sqrt{\log \Delta}$
$$S = \left\{v_i \in V : b_i^\top \cdot r \geqslant c\right\}.$$
4. Let
$$F = \left\{(v_i, v_j) \in E : b_i^\top \cdot r \geqslant c \quad \text{and} \quad b_j^\top \cdot r \geqslant c\right\}.$$
5. Return $I := S \setminus V(F)$

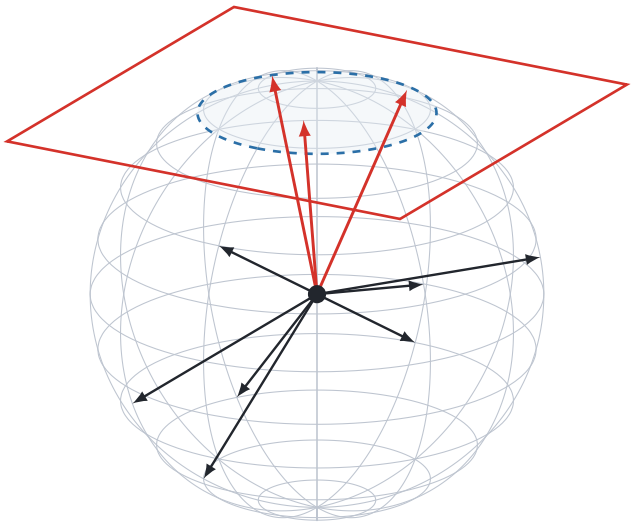

Figure 12.2: An illustration of the rounding scheme in 3 dimensions (all vectors here are unit-length in 3D and end at the boundary of the sphere). The vertices in the independent set are the ones whose vectors are cut by the hyperplane.

See Figure 12.2 for an illustration of Algorithm 12.7. 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 12.2.1 but this time, by relating these probabilities to well-known properties of the Gaussian distribution.

Let us first state, without proof, a standard and highly useful property of Gaussian distribution: sum of two independent Gaussians is also a Gaussian.

Fact 12.8. *For any two independently sampled Gaussians, we have,*

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

(recall that $\mathcal{N}(\mu, \sigma^2)$ denotes the Gaussian distribution with mean μ and variance σ^2).

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

Claim 12.9. *For any vertex $v \in V$,*

$$\Pr(v \in S) = \Pr(\mathcal{N}(0,1) \geqslant c).$$

(The RHS is the probability that a random Gaussian $\mathcal{N}(0,1)$ has value at least c).

Proof. We have,

$$\begin{aligned}
\Pr(v \in S) &= \Pr_{r \in \mathcal{N}(0,1)^n}\left({b_v}^\top \cdot r \geqslant c\right) && \text{(by the definition of the set } S\text{)} \\
&= \Pr\left(\sum_{i=1}^{n} \mathcal{N}(0,1) \cdot b_{v,i} \geqslant c\right) \\
& && \text{(by writing the inner product as the weighted sum of Gaussians)} \\
&= \Pr\left(\sum_{i=1}^{n} \mathcal{N}(0, b_{v,i}^2) \geqslant c\right) \\
& \text{(as } a \cdot \mathcal{N}(0,1) = \mathcal{N}(0,a^2) \text{ (multiplying a variable by } a \text{ changes its variance by } a^2)\text{)} \\
&= \Pr\left(\mathcal{N}(0, \|b_v\|^2) \geqslant c\right) && \text{(by Fact 12.8 and the definition of } \|b_v\|^2\text{)} \\
&= \Pr(\mathcal{N}(0,1) \geqslant c), && \text{(as } \|b_v\| = 1 \text{ in Eq (12.4))}
\end{aligned}$$

as desired. □

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

Claim 12.10. *For any edge $(u,v) \in E$,*

$$\Pr((u,v) \in F) \leqslant \Pr(\mathcal{N}(0,1) \geqslant 2c).$$

Proof. Let b_u and b_v be the vectors of vertices u, v in Eq (12.4). 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,

$$\begin{aligned}
\Pr((u,v) \in F) &= \Pr_{r \in \mathcal{N}(0,1)^n}\left({b_u}^\top \cdot r \geqslant c \ \wedge\ {b_v}^\top \cdot r \geqslant c\right) && \text{(by the definition of the set } F\text{)} \\
&\leqslant \Pr\left((b_u + b_v)^\top \cdot r \geqslant 2c\right) \\
& && \text{(as each term being larger than } c \text{ implies the sum is larger than } 2c\text{)} \\
&= \Pr\left(\mathcal{N}(0, \|b_u + b_v\|^2) \geqslant 2c\right) && \text{(exactly as argued in the proof of Claim 12.9)} \\
&= \Pr(\mathcal{N}(0,1) \geqslant 2c), && \text{(as } \|b_u + b_v\| = 1 \text{ in Eq (12.4))}
\end{aligned}$$

as desired. □

We are now almost done. Both the RHS of Claim 12.9 and Claim 12.10 have closed form analytical solutions from the vast literature on Gaussian distribution; thus, we simply need to "plug in" these standard terms and then follow the same exact strategy as in Section 12.2.1. This is what the rest of this proof is going to do. For that, we need the following fact about the Gaussian distribution.

Fact 12.11. *For any $t \geqslant 1$,*

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

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

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

where the final part comes from the Taylor expansion, proving the statement. □

We are now ready to prove Theorem 12.5.

Proof of Theorem 12.5. As in Section 12.2.1, we can calculate

$$\begin{aligned}
\mathbb{E}\,|I| &\geqslant \mathbb{E}\,|S| - 2 \cdot \mathbb{E}\,|F| \\
&= \sum_{v \in V} \Pr\left(v \in I\right) - 2 \cdot \sum_{(u,v) \in E} \Pr\left((u,v) \in F\right) && \text{(by the linearity of expectation)} \\
&\geqslant n \cdot \Pr\left(\mathcal{N}(0,1) \geqslant c\right) - 2 \cdot \frac{n\Delta}{2} \cdot \Pr\left(\mathcal{N}(0,1) \geqslant 2c\right) && \text{(by Claim 12.9 and Claim 12.10)} \\
&\geqslant n \cdot \frac{1}{2c} \cdot \frac{1}{\sqrt{2\pi}} \cdot e^{-c^2/2} - n\Delta \cdot \frac{1}{2c} \cdot \frac{1}{\sqrt{2\pi}} \cdot e^{-(2c)^2/2}. && \text{(by Fact 12.11 and since } c \geqslant 2)
\end{aligned}$$

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

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

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

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

as desired.

This concludes the proof of Theorem 12.5. □

Remark. Let us put this result in the context of the simple algorithm of Section 12.2.1. Define

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

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

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

we can "almost" say that

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

Thus, going through the calculations of Section 12.2.1, 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.

In Section 12.2.1, we saw that a naive sampling is enough to ensure each vertex joins S with probability p and each edge joins S with probability p^2. 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 *negative correlation* between the choice of endpoints of an edge. This was the key source of improvement in Algorithm 12.7 compared to that of Section 12.2.1.

Remark. The SDP-based algorithm of this chapter colors any 3-colorable graph with $\widetilde{O}(n^{1/4})$ colors [KMS98], and lowering this exponent (for 3-colorable graphs) has driven a long line of work. Even before [KMS98], Blum improved the $O(\sqrt{n})$-coloring algorithm of [Wig83] by giving a purely combinatorial $\widetilde{O}(n^{3/8})$-coloring [Blu94]. Blum and Karger then combined this with the SDP approach to reach $\widetilde{O}(n^{3/14})$-coloring [BK97], and stronger SDP hierarchies lowered the exponent to $O(n^{0.2111})$ [ACC06] and then $O(n^{0.2072})$ [Chl07]. On the combinatorial side, Kawarabayashi and Thorup gave an $\widetilde{O}(n^{0.19996})$-coloring [KT], later improved to $\widetilde{O}(n^{0.19747})$ [KTY]. The best bound known to date, $O(n^{0.19539})$ [BHL26], comes from revisiting the SDP approach of this chapter. Whether one can color a 3-colorable graph with $\operatorname{poly}\log(n)$ colors in polynomial time remains a major open problem in approximation algorithms.