Advanced Approximation Algorithms **(CMU 18-854B, Spring 2008)**

Lecture 15: Coloring 3-Colorable Graphs using SDP

March 4, 2008

Lecturer: Ryan O'Donnell *Scribe: Dafna Shahaf*

1 3-Coloring

3Col is the problem of deciding whether there is a legal 3-Coloring of a graph (all edges bichromatic). Recall that **3Col** is NP-hard ([5]). **3Col** has two natural corresponding optimization problems:

Max-3Cut: Color with 3 colors, make the coloring as legal as possible. Can be thought of as partitioning vertices into three sets, and maximizing the number of edges between the sets.

Coloring (Min-Coloring): Unlike **Max-3Cut**, the coloring has to be legal. The goal is to use as few colors as possible. In this talk we will focus on this variant.

We start with some elementary observations about coloring.

1.1 Basic observations

1. **2Col** is easy.

   If we know that the graph is 2-colorable, finding a coloring is easy. Pick a vertex, color it red. This forces its neighbours to be colored blue; keep propagating the colors until all the reachable vertices are colored. Repeat for every connected component of the graph.

2. If G has max-degree Δ, it is easy to color G with $\Delta + 1$ colors.

   A greedy-like algorithm works here. When you reach a new vertex, look at its neighbours. There can be at most Δ of them, so there must be at least one color that is free to use; therefore, you can never get stuck.

3. 3-vs.-n coloring is easy

   Choose a different color for every vertex.

2 Known Approximation Results

In this section we present some known approximation algorithms and hardness results. Note: everything in this lecture (pretty much) can be generalized to 4-coloring, 5-coloring, etc., with more work.

Theorem 2.1 (Wigderson [7]). *3-vs.-$O(\sqrt{n})$ coloring is easy*

Proof. If max-degree $\Delta \leq \sqrt{n}$, color as before. Otherwise, look at a vertex with a higher degree. The graph is 3-colorable, so its neighbours must be 2-colorable. 2-color them. Progress with the rest of the graph; never use those colors again (Refer to Figure 1 for the algorithm).

The loop happens at most $\frac{n}{\sqrt{n}} = \sqrt{n}$ times (since we delete at least $\sqrt{n}$ vertices every iteration), so we use no more than $2\sqrt{n}$ colors in the loop, and $\leq 3\sqrt{n}$ colors totally. $\square$

Theorem 2.2 (Blum [1]). *3-vs.-$O(n^{\frac{3}{8}})$ is easy*

Proof. Requires a lot of tricky work. $\square$

```
PROCEDURE Color(G)
{G graph}
1: while $\Delta_G \geq \sqrt{n}$ do
2:     Pick $v \in V$ with $d(v) \geq \sqrt{n}$
3:     2-color its neighbourhood $N(v) = \{u \mid (u,v) \in E\}$
4:     Never use those colors again.
5:     Delete $N(v) \cup \{v\}$ from $G$
6:     Color(G)
7: Color $G$ with $\sqrt{n}$ colors (using observation 2).
```

Figure 1: Wigderson's Algorithm

What about hardness results? Unfortunately, it is less impressive.

Theorem 2.3 ([6])**.** *3-vs.-4 coloring is* NP*-hard.*

The first proof of this fact used the result that cliques are hard to approximate to within any constant factor (which is a consequence of the PCP theorem; also recall question 3 on homework 1). Later, [3] discovered a new proof which does not rely on PCP, but uses only an elementary reduction from Independent-Set.
That was hardness, I'm afraid. Now back to improving upper bounds:

Theorem 2.4 (KMS [4])**.** *3-vs.-$\tilde{O}(\Delta^{\frac{1}{3}})$ is easy*[1].

Corollary 2.5. *3-vs.-$\tilde{O}(n^{\frac{1}{4}})$ is easy.*

Proof. While $\Delta \geq n^{\frac{3}{4}}$, do Wigderson. When $\Delta < n^{\frac{3}{4}}$, color the rest using KMS.
Can do the loop no more than $n^{\frac{1}{4}}$ times, $O(n^{\frac{1}{4}})$ colors. Total $\tilde{O}(n^{\frac{1}{4}})$ colors. □

Theorem 2.6 ([2])**.** *3-vs.-$O(n^{0.2072})$ is easy.*

Proof. Unholy combination of [1] and a better SDP analysis. □

3 SDP Solution

The last theorem in the previous section used SDP. How can we apply this technique to the 3-coloring problem?

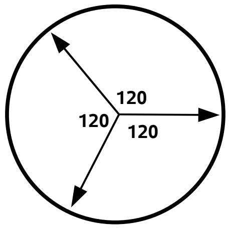

Figure 2: 3 special vectors, represent colors

[1]Notation: $\tilde{O}(f(n)) = O(f \cdot polylog(f))$

3.1 Formalizing the Problem

We would like to have a unit vector $\vec{v}$ for every vertex v (we sometimes refer to $\vec{v}$ as v, when the context is clear). Like the previous lecture, we want vertices to be far apart if there is an edge between them.

Intuition: When we were looking at $\mathbf{Max} - \mathbf{Cut}$, we had 2 special vectors that represented the two sets. Similarly, we want 3 special vectors to represent colors. Refer to Figure 2: We want to use unit vectors, and we'd like them to be as far as possible from each other if they are directly connected. That is, we want their dot-product[2] to be $-\frac{1}{2}$.

We want to map every vertex to a color s.t.

$$\forall_{(u,v)\in E} \vec{u}\cdot\vec{v} = -\frac{1}{2}$$

We can now construct the SDP. First, constraints:

$$\begin{aligned} \forall_{v\in V} \quad \vec{v}\cdot\vec{v} &= 1 \\ \forall_{(u,v)\in E} \quad \vec{u}\cdot\vec{v} &= -\frac{1}{2} \end{aligned}$$

What about the objective function? There is none, it is a feasibility problem.
If G is 3-colorable, this is feasible. Use SDP to find a feasible solution.
Fine print: Cannot really solve this exactly in polytime. Instead, we'll find many unit vectors s.t. $\vec{u}\cdot\vec{v} = -\frac{1}{2} \pm \epsilon$, and we pay runtime overhead of poly($\log(\frac{1}{\epsilon})$). If we take $\epsilon = \frac{1}{n}$, you may check that nothing is lost in the subsequent analysis.

3.2 From the SDP Solution to a Coloring

Suppose we found a feasible solution to the SDP. How do we convert it to a legal coloring? Any rounding scheme will probably miscolor some edges.

Solution: We use rounding that gives us a large independent set. This can help us make good progress towards coloring: color all of its vertices in one color (this is an independent set, so the coloring is legal). Remove this set and repeat, never using this color again.

Theorem 3.1. *Given a feasible vector solution to the SDP, we can find an independent set of fractional*[3] *size* $\tilde{\Omega}(\Delta^{-\frac{1}{3}})$
.

Note that we get the KMS Theorem (2.4) as a corollary:
Color the independent-set with one color, never use that color again, delete set, remaining graph is still 3-colorable. Repeat until less than $\Delta^{-\frac{1}{3}}$ vertices, then just use $\Delta^{-\frac{1}{3}}$ new colors. We repeat the loop at most $\tilde{O}(\Delta^{\frac{1}{3}})$ times.

3.2.1 Finding an Independent Set: General Idea

We have many vertices in a high-dimensional sphere, and we look for an independent set. Like in the previous lecture, we can choose a uniformly random hyperplane through the origin, and use it to cut vectors/vertices into two parts (those that are in the set, and those that are not).

1. To draw a random hyperplane, we can also pick a random normal vector g from the surface of the unit sphere. In order to pick a random unit vector, we can pick n random Gaussian variables.

$$\vec{g} = (g_1, ..., g_n)$$
$$g_i \in \mathbb{R} \quad \textit{independently drawn from a Normal distribution}$$

[2]Remember we promised to generalize? For 4d, we would have 4 vectors in 3d, $-\frac{1}{3}$ dot product. For 5d, we have 5 4d vectors with $-\frac{1}{4}$ dot product, and so on.

[3]"Fractional" means of the original graph size, so we lose a $\log n$ factor.

(An aside: what is the expected length of $\vec{g}$? Its expected squared length is $\sum \mathbf{E}[g_i^2] = n$, so we may guess the expected length is roughly $\sqrt{n}$, and indeed this is true (refer to Figure 3).)

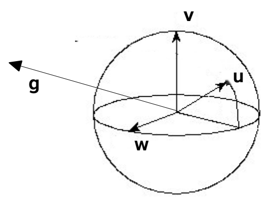

Figure 3: Unit sphere, $\vec{g}$ is a random Gaussian vector

We will use this vector for partitioning. The straightforward way (what we did for $\mathbf{Max} - \mathbf{Cut}$) is not going to result in an independent set, necessarily. We know every two vertices that are directly connected are far apart. Therefore, we can try taking only vectors that are far apart (intuition: pushing the hyperplane up).

2. Consider

$$V_{\vec{g}}(t) = \{v \mid \vec{g} \cdot \vec{v} \geq t\}$$

where t is a distance parameter, TBD.

We hope $V_{\vec{g}}(t)$ has more vertices than internal edges. We retain only a subset of $V_{\vec{g}}(t)$ that has no such edges:

$$I = \{v \in V_{\vec{g}}(t) \mid v \text{ has no neighbours in } V_{\vec{g}}(t)\}$$

I is an independent set (by definition).
(Of course, it might also be empty. The smarter thing to do would be deleting a maximal matching from $V_{\vec{g}}(t)$, but we will stick with our definition of I for simpler analysis.)

3.2.2 Finding an Independent Set: Filling in the Details

Claim 3.2. *Choosing t carefully leads to expected* $E(\frac{|I|}{n}) \geq \tilde{\Omega}(\Delta^{-\frac{1}{3}})$.

We start by bounding size $V_{\vec{g}}(t)$. (this bounds the size of I)

$$E(|V_{\vec{g}}(t)|) = \sum \mathbf{Pr}(\vec{g} \cdot \vec{v_i} \geq t) = n\mathbf{Pr}(\vec{g} \cdot \vec{v} \geq t)$$

where v denotes any unit vector, since $\vec{g}$ is spherically symmetric.
What can we say about $\mathbf{Pr}(\vec{g} \cdot \vec{v} \geq t)$? Note that a weighted sum of Gaussians is a Gaussian. We can rotate the picture; by spherical symmetry, we can rotate $\vec{v}$ to be any unit vector. Say we choose $(1, 0, ..., 0)$. Then

$$\vec{g} \cdot \vec{v} = \vec{g} \cdot (1, 0, ..., 0) = g_1 \sim N(0, 1)$$
and
$$n\mathbf{Pr}(\vec{g} \cdot \vec{v} \geq t) = n\mathbf{Pr}(g_1 \geq t)$$

What can we say about the probability that a random Gaussian variable is $\geq t$? This has been well investigated in the past. First, define

$$\mathcal{N}(t) := \mathbf{Pr}(g_1 \geq t)$$

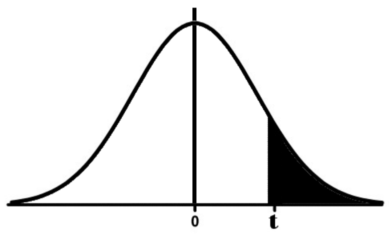

Figure 4: $\mathscr{N}(t)$

(Refer to Figure 4 for more intuition)

Some facts about $\mathscr{N}(t)$:

Fact 3.3. $\mathscr{N}(t) \sim (1/t) \cdot \varphi(t) = (1/t) \cdot (1/\sqrt{2\pi})e^{-t^2/2} = \tilde{\Theta}(e^{-t^2/2})$
where $\varphi(t)$ is the pdf of a Gaussian.

Fact 3.4. *When c is a positive constant, $\mathscr{N}(ct) = \tilde{\Theta}(\mathscr{N}(t)^{c^2})$*

Proof.

$$\tilde{\Theta}(e^{-(ct)^2/2}) = \tilde{\Theta}(e^{-t^2/2})^{c^2} = \tilde{\Theta}(\mathscr{N}(t)^{c^2})$$

□

So the tail decreases very fast.

In order to finish the proof, we still need to choose t. However, there is a tradeoff: the bigger we make t, the fewer vertices we get. We want to make it small, to get enough vertices; we also want to make it big, to get fewer edges.

Claim 3.5. *Let $\dagger := \mathbf{Pr}(v$ has a neighbour in $V_{\vec{g}}(t) \mid v \in V_{\vec{g}}(t))$*
(the probability that v gets removed later)
If $\dagger \leq 1/2$ then $E(|I|/n) \geq \Omega(\mathscr{N}(t))$.

Proof.

$$\begin{aligned}\mathbf{Pr}(v \in I) = &\ \mathbf{Pr}(v \in V_{\vec{g}}(t)) \cdot \mathbf{Pr}(v \text{ has no neighbours in } V_{\vec{g}}(t)) \\ &= \mathbf{Pr}(v \in V_{\vec{g}}(t)) \cdot (1 - \dagger) \geq \tfrac{1}{2} \cdot \mathbf{Pr}(v \in V_{\vec{g}}(t)) = \frac{1}{2}\mathscr{N}(t)\end{aligned}$$

□

In order to get this, we take the union bound on neighbours of v:

$$\begin{aligned}\dagger \leq &\ \textstyle\sum_{u \text{ s.t. } (u,v)\in E} \mathbf{Pr}(u \in V_{\vec{g}}(t) \mid v \in V_{\vec{g}}(t)) \leq \\ &\Delta \mathbf{Pr}(u \in V_{\vec{g}}(t) \mid v \in V_{\vec{g}}(t)) = \Delta \mathbf{Pr}(\vec{u} \cdot \vec{g} \geq t \mid \vec{v} \cdot \vec{g} \geq t)\end{aligned}$$

Recall that $(u, v) \in E$, so $\vec{u} \cdot \vec{v} = -\frac{1}{2}$. Because of spherical symmetry, we can imagine the picture is like Figure 5 in 2d:

$$\begin{aligned}\vec{v} &= (1, 0, ..., 0) \\ \vec{u} &= (-\frac{1}{2}, \frac{\sqrt{3}}{2}, 0, ..., 0)\end{aligned}$$

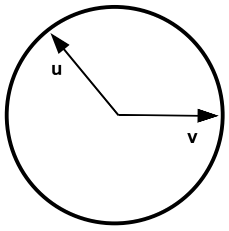

Figure 5: Spherical symmetry allows us to assume this without loss of generality

Then

$$\begin{aligned}\vec{v}\cdot\vec{g} &= g_1\\ \vec{u}\cdot\vec{g} &= -\frac{1}{2}g_1+\frac{\sqrt{3}}{2}g_2\end{aligned}$$

Suppose we know $\vec{v}\cdot\vec{g}\geq t$. Then

$$g_1\geq t\Rightarrow$$
$$\vec{u}\cdot\vec{g}=-\tfrac{1}{2}g_1+\tfrac{\sqrt{3}}{2}g_2\leq-\tfrac{1}{2}t+\tfrac{\sqrt{3}}{2}g_2$$

And if $\vec{u}\cdot\vec{g}\geq t$,

$$t\leq-\frac{1}{2}t+\frac{\sqrt{3}}{2}g_2\Rightarrow$$
$$g_2\geq\sqrt{3}t\Rightarrow$$
$$\dagger\leq\Delta\mathbf{Pr}(g_2\geq\sqrt{3}t)=\Delta\mathscr{N}(\sqrt{3}t)=\Delta\tilde{\Theta}(\mathscr{N}(t)^3)$$

To finish, we need this final bound to be $\leq\frac{1}{2}$, while keeping t as small as possible ($\mathscr{N}(t)$ as big as possible). Choose t s.t. $\tilde{\Theta}(\mathscr{N}(t)^3)=\frac{1}{2\Delta}$, and get that $\mathscr{N}(t)\geq\tilde{\Omega}(\Delta^{-1/3})$, as desired.

References

[1] A. L. Blum. *Algorithms for approximate graph coloring*. PhD thesis, Cambridge, MA, USA, 1992.

[2] E. Chlamtac. Approximation algorithms using hierarchies of semidefinite programming relaxations. In *FOCS '07: Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, pages 691–701, Washington, DC, USA, 2007. IEEE Computer Society.

[3] V. Guruswami and S. Khanna. On the hardness of 4-coloring a 3-colorable graph. In *IEEE Conference on Computational Complexity*, pages 188–197, 2000.

[4] D. R. Karger, R. Motwani, and M. Sudan. Approximate graph coloring by semidefinite programming. In *IEEE Symposium on Foundations of Computer Science*, pages 2–13, 1994.

[5] R. M. Karp. Reducibility among combinatorial problems. *R. E. Miller and J. W. Thatcher (editors): Complexity of Computer Computations*, 1972.

[6] S. Khanna, N. Linial, and S. Safra. On the hardness of approximating the chromatic number. In *Israel Symposium on Theory of Computing Systems*, pages 250–260, 1993.

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