CMPUT 675: Approximation Algorithms **Fall 2014**

Lecture 22 (Oct 27): Colouring 3-Colourable Graphs

Lecturer: Zachary Friggstad *Scribe: Bradley Hauer*

In this lecture, we consider the following problem. Suppose we have a graph $G = (V, E)$ that we know is 3-colourable but we are not given the colouring. The goal is to colour G with the fewest possible colours.

It may come as a surprise that the state of the art in this topic seems quite bad. The best algorithm will colour G with $O(n^\delta)$ colours for some constant (see the discussion below). We will see how to colour G using at most $O(n^{0.387})$ colours.

22.1 Using $O(\sqrt{n})$ Colours

We begin with some definitions.

Definition 1 *For every vertex v of a graph $G = (V, E)$, let $N_G(v) = \{u \in V : (u, v) \in E\}$ be the* neighbourhood *of v.*

Definition 2 *For $V' \subseteq V$, $G[V']$ is the graph with nodes V' and edges $\{(i, j) \in E : i, j \in V'\}$.*

The following lemma describes one of the key structures of 3-colourable graphs we will exploit.

Lemma 1 *Let $G = (V, E)$ be a 3-colourable graph. For every $v \in V$, the graph $G[N(v)]$ is 2-colourable.*

Proof. Consider a colouring $\chi : V \to \{1, 2, 3\}$ of a graph G. For every $u \in N_G(v)$ we have $\chi(u) \neq \chi(v)$, so $G[N(v)]$ can be coloured with the two colours $\{1, 2, 3\} - \{\chi(v)\}$. ■

This will be very helpful because, as you will recall, there is a simple linear-time algorithm that either 2-colours a graph or else determines the graph is not 2-colourable.

The colouring algorithms we cover in this lecture come in two phases. The first uses some colours to reduce the degree of uncoloured nodes to some quantity Δ and the second will colour the degree-Δ bounded graph.

Theorem 1 *Given a 3-colourable graph and any integer $\Delta \geq 1$, we can efficiently colour a set of nodes S using $\leq 3 \cdot \frac{n}{\Delta}$ colours such that the graph of uncoloured nodes $G[V - S]$ has degree at most Δ.*

Proof. Algorithm 1 described below accomplishes this. Each iteration uses 3 new colours. Since each iteration removes at least $\Delta + 2$ vertices from G (namely, i and $N_{G[V-S]}(i)$) then there are at most $\frac{n}{\Delta+2} \leq \frac{n}{\Delta}$ iterations. Overall, the algorithm uses at most $3\frac{n}{\Delta}$ colours.

When the while loop terminates, the graph of uncoloured nodes $G[V - S]$ has degree at most Δ. ■

Next we describe a simple algorithm to colour a degree-bounded graph.

Theorem 2 *Any graph with degree $\leq \Delta$ can be efficiently coloured with $\leq \Delta + 1$ colours.*

Algorithm 1

```
  $S \leftarrow \emptyset$
  while $G[V-S]$ has a vertex $i$ with $|N_{G[V-S]}(i)| \geq \Delta + 1$ do
    Use 3 new colours to colour $i$ and $N_{G[V-S]}(i)$. (c.f. Lemma 1)
    $S \leftarrow N_{G[V-S]}(i) \cup \{i\}$
  end while
```

Proof. Process $i \in V$ in any order. Colour i with any of the $\Delta + 1$ colours not yet used by a node in $N_G(i)$. ■

These two algorithms are combined to colour G with $\sqrt{n}$ colours.

Algorithm 2 "Final" Algorithm

```
  $\Delta \leftarrow \sqrt{n}$
  Colour some nodes with $\leq 3 \cdot \frac{n}{\Delta} = 3 \cdot \sqrt{n}$ colours s.t. the degree of the uncoloured nodes is $\leq \sqrt{n}$
  Colour the rest with $\Delta + 1 = \sqrt{n} + 1$ colours
```

Note that this algorithm is guaranteed to succeed if G is 3-colourable. It would also succeed if $G[N(i)]$ was 2-colourable for every $i \in V$ (even if G itself was not 3-colourable).

If we are given a graph G without even the 3-colourability guarantee then we can still either determine that G is not 3-colourable or else colour G with $O(\sqrt{n})$ colours. That is, first test to see if $G[N(i)]$ was 2-colourable for every $i \in V$ before running Algorithm 2. Note that this does not decide the 3-colouring problem because it may still colour a non-3-colourable graph with $O(\sqrt{n})$ colours.

22.2 A Better Algorithm

Theorem 2 works for any graph, but we can use even fewer colours in a degree-bounded, 3-colourable graph G.

Theorem 3 *Given a 3-colourable graph G with degree $\leq \Delta$, there is a randomized polynomial time algorithm that colours G with at most $4 \cdot \Delta^{\log_3 2} \cdot \log_2 n$ colours.*

By "randomized polynomial time", we mean it always uses as many colours as stated and the running time is polynomial with high probability.

Before proving this, let us see how it leads to a better colouring algorithm for arbitrary 3-colourable graphs.

Theorem 4 *There is a randomized polynomial-time algorithm that colours a graph G with $O(n^{\log_6 2} \cdot \log n) \approx O(n^{0.387})$ colours.*

Proof. Replace the third line in Algorithm 2, above, with the algorithm described in Theorem 3. Next, select a value Δ that minimizes the number of colours used between the two parts. For simplicity, we select Δ so that $n/\Delta = \Delta^{\log_3 2}$ or $n = \Delta^{\log_3 6}$. That is, we use $\Delta = n^{\log_6 3}$.

The number of colours used in the first step is at most $3\frac{n}{\Delta} = 3n^{1-\log_6 3} = 3 \cdot n^{\log_6 2}$. The number of colours used in the second step is bounded by

$$4 \cdot \Delta^{\log_3 2} \cdot \log_2 n = 4 \cdot n^{\log_6 3 \cdot \log_3 2} \cdot \log_2 n = 4 \cdot n^{\log_6 2} \log_2 n.$$

■

The proof of Theorem 3 follows from iteration the following process $\log_2 n$ times. Since the number of uncoloured nodes decreases by a factor of 2, this will colour all nodes using at most $4 \cdot \Delta^{\log_3 2} \cdot \log_2 n$ colours.

Theorem 5 *There is a randomized polynomial time algorithm that colours at least $n/2$ nodes with at most $4 \cdot \Delta^{\log_3 2}$ colours.*

Proof. We use semidefinite programming. For each $i \in V$, we will have a vector $\mathbf{v}_i \in \mathbb{R}^n$ where $n = |V|$.

The feasible solutions are those where, for each $(i,j) \in E$, $\mathbf{v}_i \circ \mathbf{v}_j = -1/2$. There is nothing in particular we want to minimize here, we just want a feasible solution to use in our algorithm. If you want to implement this using an SDP solver, you can simply have it minimize the constant 0 function.

Lemma 2 *The SDP has a feasible solution if G is 3-colourable.*

Proof. Say the colours are 0, 1, and 2, and $\alpha(i) \in \{0,1,2\}$ is the colour of $i \in V$. Let $\overline{\mathbf{v}_i} = (\cos(\alpha(i) \cdot \frac{2\pi}{3}), \sin(\alpha(i) \cdot \frac{2\pi}{3}), 0, \ldots, 0)$ for every $i \in V$. So $\overline{\mathbf{v}_i} \circ \overline{\mathbf{v}_j} = \cos(2\pi/3) = -\frac{1}{2}$ for any edge $(i,j) \in E$ because $\alpha(i) \neq \alpha(j)$. ■

Set $t = 2 + \log_3 \Delta$ and uniformly and independently sample t random unit vectors $\mathbf{r}_1, \ldots, \mathbf{r}_t \in \mathbb{R}^n$. For $1 \leq j \leq t$ let

$$S_j = \{i \in V : \mathbf{r}_j \circ \overline{\mathbf{v}_i} \geq 0\}.$$

For each t-dimensional boolean vector $(b_1, \ldots, b_t) = \underline{b}$, let $S_{\underline{b}}$ be the nodes i such that, for each $1 \leq j \leq t$, $i \in S_j$ iff $b_j = 1$. Finally we prune these sets a bit: while some $(i, i') \in E$ has $i, i' \in S_{\underline{b}}$ for some $\underline{b}$, discard both i and i' from $S_{\underline{b}}$.

Note that assigning colour $\underline{b}$ to each $i \in S_{\underline{b}}$ is a valid colouring of the nodes that were not discarded simply because no edge has both endpoints in the same $S_{\underline{b}}$ set.

Claim 1

$$\Pr\left[\sum_{\underline{b}} |S_{\underline{b}}| \geq \frac{n}{2}\right] \geq \frac{1}{2}$$

We prove this in a few steps. First, we show the probability that an edge (i, i') has both i, i' in the same set $S_{\underline{b}}$ is small. Let $\theta = 2\pi/3$ denote the angle between $\mathbf{v}_i$ and $\mathbf{v}_{i'}$. Recall from previous lectures that for a random unit vector $\mathbf{r}$ we have that exactly one of $\mathbf{r} \circ \mathbf{v}_i$ and $\mathbf{r} \circ \mathbf{v}_{i'}$ is negative is $\theta/\pi = 2/3$. That is, $\Pr[S_j \cap \{i, i'\} = 1] = 2/3$ from which we get

$$\begin{aligned} & \Pr[i, i' \text{ in same } S_{\underline{b}}] \\ = \; & \prod_{j=1}^{t} \Pr[|S_j \cap \{i, i'\}| \neq 1] \quad \text{(because the vectors } \mathbf{r}_j \text{ are sampled independently)} \\ = \; & (1/3)^t \end{aligned}$$

We use this to bound the probability that i is discarded.

$$\begin{aligned} \Pr[i, i' \text{ in same } S_{\underline{b}} \text{ for some neighbour } i' \text{ of } i] &\leq \sum_{i' \in N_G(i)} \Pr[(i, i') \text{ in same } S_b] \\ &\leq \Delta \cdot \left(\tfrac{1}{3}\right)^t \\ &= \Delta/9 \cdot 3^{-\log_3 \Delta} \\ &= 1/9 \end{aligned}$$

The first inequality is by the union bound and the second is by the fact that every vertex has at most Δ neighbours. In other words, we have

$$\Pr[i \text{ not coloured}] \le \Pr[i, i' \text{ in same } S_{\underline{b}} \text{ for some neighbour } i' \text{ of } i] \le 1/9.$$

We conclude by noting that $\mathrm{E}[\#i \in V \text{ not coloured}] \le n/9$. So:

$$\Pr[\#i \in V \text{ not coloured } \le n/2] \le \frac{\mathrm{E}[\# \text{ not coloured}]}{n/2} \le \frac{2}{9}.$$

The first inequality is justified by Markov's inequality. So, we will likely get the desired colouring from running this algorithm a few times; better still, we can easily check when we have succeeded in colouring at least $n/2$ nodes so we know exactly when to stop.

To be precise, the probability that we have to iterate this process more than n times is at most $(2/9)^n$, so with extremely high probability the algorithm runs in polynomial time. ■

22.3 Discussion

The algorithm that uses $O(\sqrt{n})$ colours is an old algorithm by Wigderson [W83]. The improved algorithm that uses $O(n^{0.387})$ was given by Karger, Motwani, and Sudan [KMS94]. They actually obtain an even better algorithm that uses $\sim O(n^{1/4})$, which is also described in the Williamson and Shmoys text. The best algorithm so far is by Kawarabayashi and Thorup and uses $O(n^{0.19996})$ colours [KT14]. Obtaining an algorithm that uses $O(\log^c n)$ colours for some constant c or even $O(n^\epsilon)$ colours for any constant $\epsilon > 0$ is an interesting and important open problem.

From the hardness perspective, unless P = NP any efficient algorithm that colours a 3-colourable graph must use at least 6 colours [GK04,KLS00]. Under some still-unresolved complexity-theoretic conjecture (that is somewhat related to the Unique Games Conjecture), it is hard to colour a 3-colourable graph using any constant number of colours [DMR09].

References

DMR09 I. Dinur, E. Mossel, O. Regev, Conditional hardness for approximate coloring, *SIAM Journal on Computing*, 39(3):843–873, 2009.

GK04 V. Guruswami and S. Khanna, On the hardness of 4-coloring a 3-colorable graph,

KMS94 D. R. Karger, R. Motwani, and M. Sudan, Approximate graph coloring by semidefinite programming, *Journal of the ACM*, 45(2):246–265, 1998.

GW05 K. Kawarabayashi and M. Thorup, Coloring 3-colorable graphs with $o(n^{1/5})$ colors, In Proceedings of Symposium on Theoretical Aspects of Computer Science, 2014.

KLS00 S. Khanna, N. Linial, and S. Safra, On the hardness of approximating the chromatic number, *Combinatorica*, 20(3):393–415, 2000.

W83 A. Wigderson, Improving the performance guarantee for graph coloring, *Journal of the ACM*, 30(4):729–735, 1983.