New Approximation Algorithms for Graph Coloring

Avrim Blum*
Laboratory for Computer Science
MIT

Abstract

The problem of coloring a graph with the minimum number of colors is well known to be NP-hard, even restricted to k-colorable graphs for constant $k \geq 3$. This paper explores the approximation problem of coloring k-colorable graphs with as few additional colors as possible in polynomial time, with special focus on the case of $k = 3$.

The previous best upper bound on the number of colors needed for coloring 3-colorable n-vertex graphs in polynomial time was $O(\sqrt{n}/\sqrt{\log n})$ colors by Berger and Rompel, improving a bound of $O(\sqrt{n})$ colors by Wigderson. This paper presents an algorithm to color any 3-colorable graph with $O(n^{3/8}$ polylog$(n))$ colors, thus breaking an "$O(n^{1/2-o(1)})$ barrier". The algorithm given here is based on examining second-order neighborhoods of vertices, rather than just immediate neighborhoods of vertices as in previous approaches. We extend our results to improve the worst-case bounds for coloring k-colorable graphs for constant $k > 3$ as well.

1 Introduction

A k-coloring of a graph is an assignment of one of k distinct colors to each vertex in the graph so that no two adjacent vertices are given the same color. The *chromatic number* of a graph is the smallest k such that the graph can be k-colored. Graph coloring problems model a collection of scheduling problems such as examination scheduling and register allocation [Cha82][CAC+81][BCKT89][Ber73]. Graph coloring is also closely related to other combinatorial problems such as finding the maximum independent set in a graph (the largest set of vertices such that no two have an edge between them). Unfortunately from the algorithmic point of view, as is well known, the problem of coloring a graph with the minimum number of colors is NP-hard, even restricted to graphs of constant chromatic number at least 3. Thus, researchers attempting to find good fast algorithms must consider issues of approximation.

In this paper, we explore the approximation problem of coloring worst-case graphs with as few additional colors as possible. That is, we consider the following problem:

> Given an n-vertex k-colorable graph, how many colors do you need in order to color the graph in polynomial time?

In particular, we present here algorithms that improve upon previously known guarantees for coloring graphs of constant chromatic number. We will not be so concerned with precisely optimizing the running time of the algorithms (so long as they are polynomial); instead we focus

*Supported by an NSF Graduate Fellowship, NSF grant CCR-8914428 and the Siemens Corporation. Author's present address: School of Computer Science, CMU, 5000 Forbes Ave., Pittsburgh, PA 15213. Email: avrim@theory.cs.cmu.edu

more on the quality of the approximation. Because 3-chromatic graphs are the simplest and in a sense the most fundamental graphs for which optimal coloring is NP-hard, much of this paper will focus on the special case of coloring graphs of chromatic number 3. We then describe extensions of these results to graphs of higher constant chromatic number.

A second standard approximation issue that we do *not* consider here is to provide algorithms that find optimal colorings for large or nicely characterized subsets of the inputs. Work along this direction has been done by Kucera [Kuc77], Turner [Tur88], Dyer and Frieze [DF89], and Blum [Blu91]; in particular, these results show that large classes of random or "semi-random" k-chromatic graphs can be optimally colored with high probability.

1.1 Past work

For graphs of constant chromatic number, the first nontrivial worst-case approximation result was due to Wigderson [Wig83]. Wigderson gives an algorithm to color any n-vertex 3-colorable graph with $O(\sqrt{n})$ colors, and more generally to color any k-colorable graph with $O(n^{1-\frac{1}{k-1}})$ colors. More recently, several researchers: Berger and Rompel [BR88], Linial, Saks, and Wigderson [LSW], and Raghavan [Rag] independently improved this bound to $O((n/\log n)^{1-\frac{1}{k-1}})$ colors, which for $k = 3$ results in a coloring of 3-colorable graphs with $O(\sqrt{n}/\sqrt{\log n})$ colors.

The result of Berger and Rompel, et al. was important because no progress had been made for some time and it showed that $\sqrt{n}$ was in no sense a lower bound for coloring 3-colorable graphs. However, for the kinds of techniques used it was clear that, say, $O(\sqrt{n}/\log^2 n)$ colors would be completely out of reach. For general graphs of arbitrary chromatic number, the best algorithmic result known to date is due to Halldórsson [Hal90]. Halldórsson's algorithm has a *performance guarantee*—that is, a ratio of the number of colors used to the chromatic number—of $O(n(\log\log n)^2/(\log n)^3)$. This result is based upon an algorithm by Boppana and Halldórsson [BH90] for the Independent Set problem which finds an independent set within an $n/(\log n)^2$ factor of the maximum.

The difficulty in improving the algorithmic results has motivated work on lower bounds for this problem. Very recently, Lund and Yannakakis [LY92], based on work of Arora, Lund, Motwani, Sudan, and Szegedy [ALM+92], have shown that for some $\epsilon > 0$, the chromatic number cannot in general be approximated to a ratio better than n^ϵ unless P=NP.

There has also been recent work on coloring graphs presented in an *on-line* manner: graphs presented one vertex at a time in some arbitrary order, with the requirement that an algorithm color the vertex presented before the next one is shown. Vishwanathan [Vis90] presents an algorithm for such a model that uses a number of colors within a logarithmic factor of the Wigderson bound.

1.2 New results

In this paper, we present an algorithm that uses a quite different strategy from the previous approaches, and colors any 3-colorable graph with $O(n^{3/8}\log^{5/2} n)$ colors. Thus, we improve the previous bound of $O(\sqrt{n}/\sqrt{\log n})$ colors and break a "soft-$O(\sqrt{n})$ barrier" (that is, ignoring polylogarithmic factors). The algorithm also extends to graphs of higher constant chromatic number and improves upon the previous bounds for such graphs. We present the new algorithm in two parts: the first part (Section 4) colors 3-colorable graphs with $O(n^{2/5+o(1)})$ colors, and the second part (Section 5) achieves the better bound claimed above. The strategy used also suggests a plausible path for further significant reductions in the color bounds, and a discussion of this is given in Section 7.

The algorithms given here are based on using information obtained from examining second-order neighborhoods of vertices and not just immediate neighborhoods as in previous approaches. The new algorithms are motivated by techniques that would work if the graph were in fact chosen *randomly*, and this motivation and the general flavor of the algorithms are given in Section 3. Some of the work in this paper has previously appeared in extended abstract form [Blu89][Blu90], and additional results with more detailed discussion appears in [Blu91].

2 Notation, definitions, and previous algorithms

In this section we review some standard graph-theoretic definitions and introduce basic notation that will be used throughout this paper. At the end of the section we will describe some previous worst-case coloring algorithms in order to introduce a few useful techniques.

Given a graph G, let $V(G)$ denote the vertices of G and $E(G)$ denote the edges of G. We will use $N(v)$ to denote the *neighborhood* of a vertex v and $d(v)$ to denote the vertex *degree*. That is, for $G = (V, E)$:

- $N(v) = \{w \in V \mid (v, w) \in E\}$, and
- $d(v) = |N(v)|$.

It will also be convenient to define the degree $D(S)$ of a set of vertices S by:

- $D(S) = \sum_{v \in S} d(v)$,

and the neighborhood $N(S)$ of set S by:

- $N(S) = \bigcup_{v \in S} N(v) = \{w \in V \mid (v, w) \in E \text{ for some } v \in S\}$.

Notice that $D(S)$ may be much larger than $|N(S)|$ if vertices in S share many neighbors in common. We will also use the term "distance-2 neighbors" of a vertex v to mean the set $N(N(v))$. Note that if $N(v) \neq \phi$ then $v \in N(N(v))$. Finally, for S a set of vertices in G, the graph $H = G|_S$ is the subgraph of G induced by set S. That is,

- $G|_S = (S, \{(i, j) \in E \mid i, j \in S\})$.

An *independent set* in a graph is a set of vertices no two of which are adjacent to each other. A *vertex cover* is a set W such that $V - W$ is independent.

As mentioned in the introduction, the *chromatic number* of a graph is the least number of colors needed to color the graph so that no two adjacent vertices are given the same color. As is standard terminology [NW90], we will say that a graph is *k-chromatic* to mean that the chromatic number is exactly k, and that a graph is *k-colorable* to mean that the chromatic number is at most k. For the most part, this distinction will not be important and we will use the terms interchangeably. We say that an algorithm *optimally* colors a graph if it colors with the fewest number of colors possible.

For the special case where G is a 3-colorable graph, we use red, blue, and green to denote the colors of vertices in G under some legal (but unknown) 3-coloring. We also use these terms to denote the sets of vertices belonging to each color class under that legal coloring.

For functions f and g we say $g(n) = \tilde{O}(f(n))$ to denote that $g(n) = O(f(n) \log^c n)$ for some constant c. Similarly, we will use $g(n) = \tilde{\Omega}(f(n))$ to denote that $g(n) = \Omega(f(n)/\log^c n)$ for some constant c. We also use "$g(n) \gg f(n)$" to mean that $f(n) = o(g(n))$. Finally, the term "$\log n$" will be used to denote $\log_2 n$, and $\log^p n$ will be used to denote $(\log n)^p$.

2.1 Previous algorithms

We first just note that 2-colorable graphs can easily be 2-colored in polynomial time.

Let us now review Wigderson's algorithm [Wig83] for the special case of 3-colorable graphs. Wigderson's algorithm looks at the immediate neighborhoods of vertices, and uses the fact that in a 3-colorable graph the neighborhood of any vertex is 2-colorable. The algorithm proceeds as follows. If there exists a vertex of degree at least $\sqrt{n}$ in the graph, then we color its neighborhood with two unused colors and then delete the colored nodes from the graph. If all vertices have degree less than $\sqrt{n}$, we can greedily $\sqrt{n}$-color the remaining graph, since with $\sqrt{n}$ colors, for each vertex we are guaranteed that at least one color is not used on its neighbors. The total number of colors used is at most $3\sqrt{n}$. If we pick a degree cutoff of $\sqrt{2n}$ instead of $\sqrt{n}$, we can optimize the constant for this type of strategy to $\sqrt{8}$.

The improvement to $O(\sqrt{n}/\sqrt{\log n})$ of Berger et al. mentioned previously is more complicated, but essentially results from choosing $O(\log n)$ starting vertices instead of one. The precise algorithm is described in [BR88]. We will revisit this algorithm in Section 3.2, where the algorithm and bounds guaranteed follow as an easy corollary of the machinery described there.

In contrast to the above strategies, the algorithm presented here is a multi-pronged attack. The main idea of the new approach is to take advantage of information from not just the immediate neighbors of vertices, but from distance-2 neighbors as well. One difficulty with looking at distance-2 neighbors is that they have not so obvious a structure as the immediate neighbors. For example, the immediate neighborhood, as noted above, is 2-colorable; the structure of the distance-2 neighbors will have to be more carefully brought out.

3 New algorithms: preliminaries

3.1 The basic idea of the new approach

The previous best algorithms for coloring 3-colorable graphs all used $\tilde{O}(n^{1/2})$ colors in the worst-case. This section describes the basic idea for an algorithm to color any n-vertex 3-colorable graph G with $\tilde{O}(n^{\alpha})$ colors, for some $\alpha < 1/2$. Note that to do so, it is enough, as in Wigderson's algorithm, to find an independent or 2-chromatic set of size $\tilde{\Omega}(n^{1-\alpha})$, since that set can be colored with 1 or 2 colors and the procedure repeated on the graph remaining.

The idea of the new algorithm is to try to make progress from examining distance-2 neighbors. We will describe the motivation for the approach by considering the question: "what if the edges in the graph were distributed *randomly*?" That is, what if after an adversary decided which nodes to place in the sets red, blue, and green (the color classes under a legal 3-coloring unknown to the algorithm) a coin of some bias p was then flipped for each pair of vertices u, v of different colors to determine whether edge (u, v) would be in the graph? In that case, the following strategy finds an independent set of size $\tilde{\Omega}(n^{2/3})$.

First, we may assume there are about the same number of red, blue, and green vertices, since otherwise we could immediately separate at least one of the color classes from the others by just looking at the vertex degrees.[1] Second, we may assume that the vertices have average degree at least $n^{1/3}$, since otherwise we could just greedily gather an independent set of size $\Omega(n^{2/3})$. Finally, for simplicity, we assume that the average degree d is at most $n^{1/2-\epsilon}$ for some $\epsilon > 0$ (so

[1] Once we have separated one of the color classes from the others, we can then easily 2-color the graph remaining. This fact about the sizes of the color classes for random graphs does not generalize to worst-case graphs, and in fact, there is no analog of this step used in the worst-case algorithm. It is inserted here solely to simplify our picture of the graph.

we have $n^{1/3} \leq d \leq n^{1/2-\epsilon}$). This last requirement will simplify the motivational argument, but is not necessary.

Suppose v is a red vertex. Then, the neighborhood of v consists of blue and green vertices, with approximately half of each color if the numbers of blue and green vertices in the graph are roughly equal. Each blue vertex in $N(v)$ similarly has about half green neighbors and half red neighbors, and each green vertex has about half blue neighbors and half red neighbors. So, if we look at the set of the distance-2 neighbors $S = N(N(v))$, red vertices are significantly more predominant than blue or green vertices. In fact, about half of S is red, a quarter blue, and a quarter green, since we have assumed d is small enough (at most $n^{1/2-\epsilon}$) that not many vertices of S are neighbors of several vertices of $N(v)$. Thus, S is a set of size at least $\Omega(n^{2/3})$ that has within it an independent set (the red vertices) of about one half the size of S.[2]

Given a set S of size $\Omega(n^{2/3})$ containing an independent set of size $\frac{1}{2}|S|$, and therefore a vertex cover of size $\frac{1}{2}|S|$, we can algorithmically find an independent set of size $\tilde{\Omega}(n^{2/3})$ by applying a vertex-cover approximation algorithm due to Bar-Yehuda and Even [BYE85] and, independently, to Monien and Speckenmeyer [MS85]. (Their algorithms differ slightly but the bounds are essentially the same; a version of their algorithm is described in the appendix for completeness.) Their algorithm finds a vertex cover of size at most $\left(2 - \frac{\log\log n}{\log n}\right)$ times the size of the minimum vertex cover in an n-node graph. If we apply the algorithm to the graph induced by S, we find a vertex cover W in S of size at most $\frac{1}{2}|S|\left(2 - \frac{\log\log|S|}{\log|S|}\right)$, which is at most $|S| - |S|/(4\log|S|)$. So, the complement, $S - W$, is an independent set inside S of size at least $\Omega(|S|/\log|S|) = \tilde{\Omega}(n^{2/3})$. Thus, in the case where the edges in the graph are chosen by a *random* process, we have found a large independent set. [3]

Worst-case graphs, however, are *not* random. Instead, we will use various techniques to force the graph to have properties of random graphs—or at least weak versions of these properties—that we need. One such property is that of being "well-distributed": we want $N(N(v))$, or at least an easy-to-select subset of $N(N(v))$, to have nearly half red vertices, so that the vertex-cover approximation algorithm can be used. The second such property is an expansion property: we want the selected subset of $N(N(v))$ to be significantly larger than $N(v)$, so that our performance is much better than that achieved by looking only at immediate neighbors.

Sections 4 and 5 describe one general method for proving the existence of a form of good distribution in worst-case graphs and two methods for forcing expansion. The first method for forcing expansion (described in Section 4) is simple and elegant and results in a coloring of any 3-colorable graph with $\tilde{O}(n^{2/5})$ colors; the second (described in Section 5) is more complicated, but results in an improved bound of $\tilde{O}(n^{3/8})$ colors.

3.2 Useful definitions of progress

In order to more easily describe and analyze the coloring algorithms presented, it will be useful to have several formal notions of "making progress" towards an $f(n)$-coloring of an n-vertex graph. These notions simplify the analysis by allowing us to aim for intermediate goals. While we will only need to consider $f(n)$ a function of the form $O(n^\alpha \log^\beta n)$, the notions of progress in fact hold for a more general class of "nearly-polynomial" functions, as defined below.

[2] We can remove the restriction $d < n^{1/2-\epsilon}$ by choosing S to be a subset of $N(N(v))$ generated by conceptually deleting edges from the graph at random until the average degree *is* below $n^{1/2-\epsilon}$, and then letting $S = N(N(v))$ in this new graph.

[3] In fact, random 3-colorable graphs are easy to actually 3-color for a wide range of edge probabilities [DF89, Tur88, Blu91]. In [Blu91], we show how to 3-color random 3-colorable graphs for $p \geq n^{o(1)-1}$ (i.e., where the average degree is at least n^ϵ for some $\epsilon > 0$).

Definition 1 *A function f over $\mathbf{Z}^+$ is* "nearly-polynomial" *if it is non-decreasing and there exist constants $c, c' > 1$ such that for all sufficiently large N,*

$$f(2N) \geq cf(N) \quad \textit{and} \quad f(2N) \leq c'f(N).$$

For example, if $f(n) = n^{1/2}$, then we may choose $c = c' = 2^{1/2}$. If $f(n) = n^\alpha \log^\beta n$ for $\alpha > 0$, then we may choose $c = 2^\alpha(1-\epsilon)$ and $c' = 2^\alpha(1+\epsilon)$ for any constant $\epsilon > 0$.

Three important ways of making progress towards an $f(n)$-coloring of an n-vertex k-colorable graph are defined as follows.

Progress Type 1: [Large-IS] Find an independent or 2-colorable[4] set S of size $\Omega(n/f(n))$.

Progress Type 2: [Small-Nbhd] Find an independent or 2-colorable set S such that $|N(S)| = O(f(n)|S|)$.

Progress Type 3: [Same-Color] Find two vertices that must be the same color under any legal k-coloring of the graph.

Progress Type 1 "makes progress" because we can color the set found with at most two colors and then continue on the remaining graph with a new set of colors. The idea for progress Type 2 is that we can use it to find many different 2-colorable sets, each of which is independent of the others; combining the sets found gives us a large 2-colorable set and thereby progress of Type 1. Progress Type 3 always helps us towards any approximate coloring. More formally, besides showing that each type of progress is useful individually, we would like to say that any combination of the three types of progress, in any order, yields an $O(f(n))$-coloring of an n-vertex k-colorable graph.

Lemma 1 *If there exists a polynomial-time algorithm $\mathcal{A}$ that is guaranteed given any k-colorable graph of m vertices, to make progress of either Type 1, 2 or 3 towards an $O(f(m))$-coloring (where f is nearly-polynomial), then there exists a polynomial-time algorithm $\mathcal{B}$ that colors any n-vertex k-colorable graph G with $O(f(n))$ colors.*

Note that if we do not care about constants, we can state Wigderson's algorithm for coloring n-vertex 3-colorable graphs using progress of types 1 [Large-IS] and 2 [Small-Nbhd] as follows. If a vertex v has a neighborhood of size $\Omega(n^{1/2})$ then we make progress of Type 1 using its neighborhood; otherwise, $|N(v)| = O(1 \cdot n^{1/2})$ so we make progress Type 2.

We can also state simply the algorithm of Berger and Rompel [BR88] to color any 3-colorable graph with $O(\sqrt{n}/\sqrt{\log n})$ colors using these types of progress. Select a subset S of $3\log n$ vertices in graph G arbitrarily and examine every independent subset $\tilde{S}$ of S of size $(\log n)$. Note that there are at most $\binom{3\log n}{\log n} < n^3$ such subsets, so this can be done in polynomial time. For each subset $\tilde{S}$, test to see if its neighborhood is 2-colorable; this test will succeed for some $\tilde{S}$ since at least one such subset must consist of vertices all the same color in some legal 3-coloring of G. Now, if $|N(\tilde{S})| \geq \sqrt{n}\sqrt{\log n}$, we have made progress of Type 1. If $|N(\tilde{S})| < \sqrt{n}\sqrt{\log n}$, then we have made progress of Type 2.

We now prove Lemma 1, showing that these types of progress really do "make progress".

[4]Technically, an independent set is 2-colorable. We list both here to emphasize there is no need for the set S to require 2 colors. Also, we label this type of progress by "LARGE-**IS**" since given a 2-chromatic set, one can easily find an independent subset of only a factor of 2 smaller.

Proof of Lemma 1: First, if algorithm $\mathcal{A}$ ever makes progress of Type 3 [Same-Color] on a subgraph of G, then it is clear we can just merge the two vertices found into a new vertex and start again from the beginning: in doing so, we remove one vertex from G and use no colors. Thus, we may assume from now on that $\mathcal{A}$ only makes progress of Types 1 or 2 when applied to any subgraph of G.

Claim: If for some constant $\epsilon > 0$ we can always find a 2-colorable set of size $\epsilon m/f(m)$ in a k-colorable graph of m vertices, then we can achieve an $O(f(n))$-coloring of G as follows. We find such a set in G, color it with two colors, remove those vertices from the graph, and repeat.

Proof of Claim: The proof is just a straightforward calculation given below. The number $C(m)$ of colors used satisfies $C(m) \leq 2 + C\left(m - \epsilon m/f(m)\right)$. For each m' in the range $[m/2, m]$, we have:

$$\begin{aligned} C(m') &\leq 2 + C\left(m' - \epsilon m'/f(m')\right) \\ &\leq 2 + C\left(m' - \epsilon(m/2)/f(m)\right). \qquad \text{(because } f \text{ is non-decreasing)} \end{aligned}$$

Applying this last inequality $f(m)/\epsilon$ times, we get $C(m) \leq 2f(m)/\epsilon + C(m/2)$, which implies

$$\begin{aligned} C(m) &\leq \tfrac{2}{\epsilon}\left[f(m) + f(m/2) + \ldots + f(1)\right] \\ &\leq \tfrac{2}{\epsilon} f(m)\left[1 + \tfrac{1}{c} + \tfrac{1}{c^2} + \tfrac{1}{c^3} + \ldots + O(1)\right] \\ &\quad \text{(since } f(r) \geq c f(r/2) \text{ for } r \text{ large enough)} \\ &= O(f(m)). \qquad \square \text{ (End proof of claim.)} \end{aligned}$$

To prove the lemma, we just need some algorithm $\mathcal{B}'$ that on any k-colorable graph of m vertices finds a 2-colorable set of size $\Omega(m/f(m))$. Algorithm $\mathcal{B}'$ works as follows.

On input (V, E), where $m = |V|$,

1. Initialize set U to the empty set and initialize V' to V.
2. While $|V'| \geq m/2$ do:
   (a) Let (V', E') be the subgraph induced by the vertices in V'. Run algorithm $\mathcal{A}$ on (V', E').
   (b) If $\mathcal{A}$ returns with progress of Type 1 [Large-IS], then since $|V'| \geq m/2$, we have a 2-colorable set of size $\Omega(\frac{m/2}{f(m/2)}) = \Omega(m/f(m))$ (since f is nearly-polynomial), so halt and output that set.
   (c) If $\mathcal{A}$ returns with progress of Type 2 [Small-Nbhd], let S denote the set returned by $\mathcal{A}$. Now, update:

$$\begin{aligned} U &\leftarrow U \cup S \\ V' &\leftarrow V' - (S \cup N(S)). \end{aligned}$$

   Notice that in this step, each time we add vertices to U, we remove all their neighbors from V'. So, we maintain the invariant that U has no neighbors in V'.
3. Halt and output U.

If we reach step 3 in the above algorithm, it must be that at that point, $|V'| < m/2$. Set U is a 2-colorable set since each set S added to U in step 2(c) is 2-colorable and by the invariant mentioned in 2(c), the sets S are all independent of each other (thus, we may use the *same* 2

colors on each set S). Set U is also large because for each set S of size r found in step 2(c), we add r vertices to U and remove at most $r + trf(m)$ vertices from V' for some constant t by the definition of progress Type 2 [Small-Nbhd].[5] Thus, $|V - V'|$ is at least $m/2$ and $|V - V'|$ is at most $|U| + t|U|f(m)$. Combining the two inequalities, we find $|U| + t|U|f(m) \geq m/2$, which implies $|U| = \Omega(m/f(m))$. This large 2-colorable set is exactly what we needed from algorithm $\mathcal{B}'$. ■

By Lemma 1, we now may just aim for progress of one of the three types in our coloring algorithms. This fact will simplify the statements and correctness proofs of algorithms presented in Sections 4, 5, and 6.

Also, as a simple application of these types of progress, note that progress Type 2 [Small-Nbhd] can be used to guarantee that for each vertex v, the set $N(N(v))$ has size $\Omega(f(n)^2)$: we make progress if $|N(v)| \leq f(n)$ since $\{v\}$ is an independent set and make progress if $|N(N(v))| \leq f(n)|N(v)|$ since $N(v)$ is 2-colorable. Thus, we get the following corollary. (We assume here that f is "nearly-polynomial" as in Definition 1.)

Corollary 2 *If G is an n-vertex 3-colorable graph such that $|N(N(v))| = O(f(n)^2)$ for some vertex v, then we can make progress towards an $O(f(n))$-coloring of G.*

3.3 A few additional definitions

We now present a few additional definitions that will be needed in Section 4 and 5. Given a graph $G = (V, E)$ on n vertices:

- For $v \in V$, let $d_T(v) = |N(v) \cap T|$. We call $d_T(v)$ the *degree into T* of v.
- For $S, T \subseteq V$, let $D_T(S) = \sum_{v \in S} d_T(v)$. We call $D_T(S)$ the *degree into T* of S.

  Note that $d_T(v) = D_{\{v\}}(T)$ and $D_T(S) = D_S(T)$.
- Let $\delta = \delta(n) = \frac{1}{5 \log n}$.
- Let $I_j = \{v \in V \mid d(v) \in [(1+\delta)^j, (1+\delta)^{j+1})\}$ for $j = 0, 1, 2, \ldots$. That is, we divide the set of vertices of degree at least 1 into bins I_j so that in each bin, the ratio of the degrees of any two vertices is less than $(1 + \delta)$. The number of bins is at most $\log_{1+\delta} n \leq (1 + o(1))\frac{1}{\delta} \ln n < \frac{1}{\delta} \log n$.
- For $S \subseteq V$, let $N_i(S) = \{v \in N(S) \mid d_S(v) \in [(1+\delta)^i, (1+\delta)^{i+1})\}$ for $i = 0, 1, 2, \ldots$. In other words, $N_i(S)$ $(0 \leq i \leq \log_{1+\delta} n)$ is the subset of vertices in $N(S)$ that are hit by at least $(1 + \delta)^i$ and less that $(1 + \delta)^{i+1}$ edges from S.

4 Coloring 3-colorable graphs: first algorithm

In this section, we describe an algorithm to color any n-vertex 3-colorable graph with $\tilde{O}(n^{0.4})$ colors. As mentioned in the last section, the algorithm consists of two major parts. First, we force the graph without loss of generality to have a useful expansion property. Second, we find and take advantage of a form of good distribution of edges that we show must exist in any 3-colorable graph. Some of the theorems we prove, in particular those in Section 4.3 concerning the distribution property, hold more generally for graphs constrained only to have large independent

[5] Here we use the fact that f is non-decreasing.

sets. This fact will be useful for us later in Section 6 for extending these techniques to graphs of higher chromatic number.

Throughout this section, we assume f is a "nearly-polynomial" function as in Definition 1.

4.1 Forcing expansion

We now show that if our goal is to color a 3-colorable graph G with $O(f(n))$ colors, then we may assume without loss of generality that no two vertices share more than $n/[f(n)]^2$ neighbors. So, for example, if we wish to color with $\tilde{O}(n^\alpha)$ colors, we may assume for all $u, v \in V$, that $|N(u) \cap N(v)| \leq n^{1-2\alpha}$ (for $\alpha = 0.4$, the shared neighborhood may have size at most $n^{0.2}$). This is our first method for forcing a useful form of expansion in the graph. Given the three methods for making progress defined in the last section, this method for forcing expansion falls out easily.

Theorem 3 *If G is an n-vertex 3-colorable graph containing vertices u and v such that*

$$|N(u) \cap N(v)| = \Omega\left(n/[f(n)]^2\right),$$

then we can make progress of Type 1, 2, or 3 towards an $O(f(n))$-coloring of G.

Proof: Suppose u and v are two vertices that share a neighborhood $S = N(u) \cap N(v)$ of size $\Omega(n/[f(n)]^2)$. Clearly, S is 2-colorable since it is a subset of the neighborhood of u. So, if $|N(S)| \leq n/f(n)$, then we have made progress Type 2 [Small-Nbhd]. On the other hand, if $|N(S)| \geq n/f(n)$ *and $N(S)$ is 2-colorable*, then we have made progress of Type 1 [Large-IS]. The last possibility is that $N(S)$ is *not* 2-colorable (and that it is large, but we will not need this fact). But, this last case means that u and v *must* be the same color under any legal 3-coloring of G. The reason is that if u and v could possibly be different colors under some legal 3-coloring (say blue and green) then S would be monochromatic (red), so $N(S)$ would be 2-colorable (blue and green). So, if our attempt to 2-color $N(S)$ fails, then we make progress of Type 3 [Same-Color]. ■

We can use the same argument as above to guarantee without loss of generality that a selected set S of size $\Omega(n/f(n)^2)$ in G is not monochromatic under any legal 3-coloring of G. In particular, suppose S were monochromatic, so $N(S)$ is 2-colorable. Then, if $|N(S)| \geq n/f(n)$ we make progress Type 1 [Large-IS], and if $|N(S)| < n/f(n)$ we make progress Type 2 [Small-Nbhd]. So, we get the following corollary.

Corollary 4 *Given an independent set S of size $\Omega(n/f(n)^2)$ in an n-vertex 3-colorable graph G, we can either make progress towards an $O(f(n))$ coloring of G or else guarantee that the vertices of S are not all the same color under any legal 3-coloring of G.*

While this corollary is not be immediately useful for us here, an improved method for forcing expansion described in Section 5 consists in part of an improvement to this corollary, and leads to better coloring guarantees.

4.2 The algorithm

We now describe the algorithm for coloring n-vertex 3-colorable graphs with $O(n^{2/5}\log^{8/5} n)$ colors. As mentioned in the last section, the algorithm uses a vertex cover approximation algorithm of Bar-Yehuda and Even [BYE85] and (independently) Monien and Speckenmeyer [MS85] that finds a vertex cover of size at most $(2 - \frac{\log\log n}{2\log n})$ times the size of the minimum

vertex cover in a graph. We will call their algorithm the BE/MS algorithm. A simpler version of their procedure for the special case in which it is used in this paper is given as Algorithm Approx-**IS** in the appendix.

Algorithm First-Approx**:**

Given: *$G = (V, E)$, a 3-colorable graph on n vertices. Let $f(n) = n^{2/5}(\log n)^{8/5}$.*

Output: *Progress of Type 1, 2, or 3 towards an $O(f(n))$-coloring of G.*

1. *[Min degree] For each vertex v, if $d(v) < f(n)$, make progress Type 2* [Small-Nbhd].
2. *[Expansion] For each pair of vertices u, v, if $|N(u) \cap N(v)| \geq n/[f(n)]^2$, then make progress using Theorem 3.*
3. *[Dist-2 Neighbors] Otherwise, for each vertex v, for each $i, j \in \{0, 1, \ldots, 5\log^2 n\}$:*

   $$\text{Let } \; T_{v,i,j} = N_i(N(v) \cap I_j).$$

   (Recall the definitions of Section 3.3.)
4. *[VC approx] Run the* BE/MS *Vertex-Cover approximation algorithm (or equivalently, the Independent-Set approximation algorithm* Approx-**IS** *in the appendix) on each $T_{v,i,j}$. If we find an independent set of size $\Omega(n^{3/5}/(\log n)^{8/5})$, we have made progress Type 1* [Large-IS].

The next two sections (4.3 and 4.4) are devoted to proving the following theorem.

Theorem 5 (Main Theorem) *Algorithm* First-Approx *makes progress of Types 1, 2, or 3 towards an $O(n^{2/5}(\log n)^{8/5})$-coloring of any n-vertex 3-colorable graph.*

Using Lemma 1 (the usefulness of making progress), we get the following corollary.

Corollary 6 *There exists a polynomial-time algorithm that will color any 3-colorable n-vertex graph with $O(n^{2/5}(\log n)^{8/5})$ colors.*

Let us calculate the running time of the coloring algorithm. The BE/MS algorithm runs in time $O(NM)$ on any N-vertex graph with M edges. We may assume for simplicity that the graph in Step 4 of algorithm First-Approx has size at most $n^{3/5}$ else we just remove excess vertices at random. So, the running time of algorithm First-Approx, which is dominated by Steps 3 and 4, is at most:

$$[(n \text{ vertices}) \cdot (\log^2 n \; j\text{'s}) \cdot (\log^2 n \; i\text{'s}) \text{ in Step 3}] \times [n^{3/5}(n^{3/5})^2 \text{ for vertex cover in Step 4}] = \tilde{O}(n^{14/5}),$$

which is polynomial in n. Note that this is the time needed to give one color to $\tilde{\Omega}(n^{3/5})$ vertices. One may have to run the algorithm $\tilde{O}(n^{2/5})$ times in order to color the entire graph.

4.3 Forcing good distribution

From the last sections, we know that if we wish to color an n vertex graph with $O(f(n))$ colors, then we may assume that the graph has minimum degree $f(n)$ (or else we make progress Type 2 [Small-Nbhd]) and no two vertices share more than $n/[f(n)]^2$ neighbors (or else we make progress with Theorem 3).

The goal of this section is to show how, given such a graph G, to find a small number of subgraphs such that at least one must be both nearly half red under some legal 3-coloring of G (at least $\frac{1}{2}(1 - \frac{1}{\log n})$ of its vertices red), and large (size $\tilde{\Omega}(f(n)^4/n)$, which equals $\tilde{\Omega}(n^{3/5})$ for $f(n) = \tilde{\Omega}(n^{2/5})$). In particular, we will show this holds true for one of a small number of subsets of $N(N(v))$ for some vertex v in the graph.

We will assume without loss of generality that red is the color in G such that $D(\mathsf{red}) = \max(D(\mathsf{red}), D(\mathsf{blue}), D(\mathsf{green}))$. That is, of the three colors, red is the color with the most edges incident. The assumption on red implies that $D(\mathsf{red}) \geq \frac{1}{2}(D(\mathsf{blue}) + D(\mathsf{green}))$, so

$$D_{\mathsf{red}}(\mathsf{blue} \cup \mathsf{green}) \quad \geq \quad \frac{1}{2} D(\mathsf{blue} \cup \mathsf{green}). \tag{1}$$

Note also that if d is the average degree of the vertices in G, then $D(\mathsf{red}) \geq d|\mathsf{red}|$.

4.3.1 The basic approach, and a problem with the naive strategy

In order to find a large subgraph that is nearly half red, the first step will be to find a large subset $S \subseteq \mathsf{blue} \cup \mathsf{green}$ such that nearly half of the edges leaving S enter into red vertices. We know that if we look at the *entire* set blue ∪ green, at least half of the edges leaving that set enter into red vertices by equation (1). The problem is: we do not know how to find blue∪green. We can, however, look at subsets of blue ∪ green by considering vertex neighborhoods, many of which (for red starting vertices) will be blue and green.

Given the property of blue ∪ green described in equation (1), one might expect that this property would hold for the neighborhood of some vertex as well: that is, that for some $v \in \mathsf{red}$, we would have $D_{\mathsf{red}}(N(v)) \geq \frac{1}{2} D(N(v))$. Unfortunately, this may not necessarily be the case. Basically, the problem is that a blue or green vertex w affects the sum of the $D_{\mathsf{red}}(N(v))$ over $v \in$ red in an amount proportional to the square of its degree into red, but w affects $D_{\mathsf{red}}(\mathsf{blue} \cup \mathsf{green})$ in an amount only linear in its degree. For a more detailed counterexample to this naive strategy, see [Blu91].

Essentially, the difficulty occurs when vertices have wildly varying degrees. While one can also find counterexamples that hold even when all vertices have degrees in some range $[n^{\alpha-\epsilon}, n^{\alpha+\epsilon}]$ for any $\epsilon > 0$, if we restrict the vertex degrees *extremely* tightly then the desired property does hold. That is, if the degrees are nearly identical, then it turns out there does exist $v \in V$ such that $N(v)$ has nearly half the edges leaving it entering into red vertices. This is the purpose of the bins I_j and is the intuition for Theorem 7 below.

Once we have a set $S \subseteq N(v)$ with nearly half the edges leaving it entering into red vertices, we use a similar idea to find a large set inside $N(S)$ which is nearly half red. The trick again is to separate vertices according to degree, which is the purpose of the sets $N_i(S)$. This step is handled by Theorem 8.

4.3.2 Theorems and proofs

We now describe the theorems that allow the above basic idea and the algorithm First-Approx to succeed. These theorems are stated in terms of not-necessarily 3-colorable graphs containing a large independent set R. (The symbol "R" is used to be suggestive of the set red.)

Theorem 7 *Given an n-vertex graph $G = (V, E)$ with average vertex degree d, and an independent set R such that (1) $D_R(V - R) \geq \lambda D(V - R)$ for some $0 \leq \lambda \leq 1$ and (2) $D(R) \geq d|R|$, then for some $v \in R$ and some bin I_j:*

1. $|N(v) \cap I_j| \geq \delta^2 d / \log_{1+\delta} n$,
2. $D_R(N(v) \cap I_j) \geq \lambda(1 - 3\delta) D(N(v) \cap I_j)$.

In other words, for some $v \in R$, the set $N(v) \cap I_j$ is a reasonably large fraction of $N(v)$ and has almost a fraction λ of the edges incident to it going into R. We now look at the neighbors of $N(v) \cap I_j$ and show that for some i, the set $N_i(N(v) \cap I_j)$ has the properties we need.

Theorem 8 *Given an n-vertex graph $G = (V, E)$, a set $R \subseteq V$, and $\lambda' \in [0, 1]$:*
For any set S such that $D_R(S) \geq \lambda' D(S)$, there must exist some $i < \log_{1+\delta} n$ such that:

1. $D_{N_i(S) \cap R}(S) \geq \delta D_R(S) / (\log_{1+\delta} n)$,
2. $|N_i(S) \cap R| / |N_i(S)| \geq (1 - 2\delta)\lambda'$.

Assuming for now the correctness of Theorems 7 and 8, we can prove a corollary showing why at least one of the sets created in Step 3 of Algorithm First-Approx will both be large and contain an independent set of nearly half its vertices (and so be of the right form for the vertex-cover algorithm used in Step 4).

Corollary 9 *Given an n-vertex 3-colorable graph $G = (V, E)$ such that (1) no two vertices share more than s neighbors and (2) G has minimum degree $d_{min} \geq 10(\log_{1+\delta} n)/\delta$, then for some $v \in V$ and some $i, j \in [0, 5 \log^2 n]$, the set*

$$T = N_i(N(v) \cap I_j)$$

has at least $\Omega\left((d_{min})^2 / (s \log^7 n)\right)$ vertices of which at least a fraction $\frac{1}{2}(1 - \frac{1}{\log n})$ are colored red *under some legal 3-coloring of G.*

Proof of Corollary 9: By definition of set red in G, the conditions of Theorem 7 are satisfied for $R =$ red and $\lambda = 1/2$ (see equation (1)). Let vertex v and bin I_j be such that claims (1) and (2) of Theorem 7 are satisfied for $S = N(v) \cap I_j$. By claim (2) of Theorem 7, set S satisfies the conditions of Theorem 8 with $\lambda' = \frac{1}{2}(1 - 3\delta)$. Let i be the index such that claims (1) and (2) of Theorem 8 are satisfied and let $T = N_i(S)$. Then:

$$\begin{aligned}
D_{T \cap R}(S) &\geq \delta D_R(S)/(\log_{1+\delta} n) && \text{(Theorem 8, claim 1)} \\
&\geq \delta\Big[\lambda(1 - 3\delta) D(S)\Big]/(\log_{1+\delta} n) && \text{(Theorem 7, claim 2)} \\
&\geq \delta\lambda(1 - 3\delta)\Big[d_{min}|S|\Big]/(\log_{1+\delta} n) && \text{(for all } v,\ d(v) \geq d_{min}) \qquad (2) \\
&\geq \delta^3\lambda(1 - 3\delta) d_{min}^2/(\log_{1+\delta} n)^2 && \text{(Theorem 7, claim 1)} \\
&= \Omega\left(\delta^5 d_{min}^2/(\log^2 n)\right) && \text{(using } \log_{1+\delta} n = O(\tfrac{1}{\delta}\log n)) \\
&= \Omega\left(d_{min}^2/(\log^7 n)\right). && (\delta = \tfrac{1}{5\log n})
\end{aligned}$$

Since no two vertices share more than s neighbors and $S \subseteq N(v)$, we know no vertex $w \neq v$ in T has more than s neighbors in S. Thus, we can lower bound the size of T by $[D_S(T) - d_S(v)]/s$, which is at least $[D_{T \cap R}(S) - |S|]/s$. By equation (2) and our assumption that $d_{min} \geq 10 \log_{1+\delta} n/\delta$, we have $|S| \leq \frac{1}{2} D_{T \cap R}(S)$. So:

$$\begin{aligned}
|T| &\geq \tfrac{1}{2} D_{T \cap R}(S)/s \\
&= \Omega\left(d_{min}^2/(s \log^7 n)\right).
\end{aligned}$$

Also, the fraction of red vertices in T is large:

$$\begin{array}{rcll} |T \cap R|/|T| & \geq & \lambda(1-2\delta)(1-3\delta) & \text{(Theorems 7 claim 2, and 8 claim 2)} \\ & \geq & \frac{1}{2}(1-5\delta) & \text{(by definition of red, we have } \lambda \geq 1/2) \\ & \geq & \frac{1}{2}\left(1-\frac{1}{\log n}\right). & \end{array}$$

Thus, set T satisfies both claims of the corollary. ■

Before proving Theorems 7 and 8, we state a simple combinatorial lemma:

Lemma 10 *Given b balls of which r are red, all placed in k boxes, then for any ϵ $(0 \leq \epsilon < 1)$, there is some box with at least $\epsilon r/k$ red balls such that the ratio of the number red balls to the total number of balls inside that box is more than $(1-\epsilon)r/b$.*

Proof: Throw out all boxes with fewer than $\epsilon r/k$ red balls. The minimum possible ratio of red balls to total balls left is: $(r-\epsilon r)/(b-\epsilon r)$ since at worst we throw out k boxes containing only red balls. This ratio is strictly greater than $(1-\epsilon)r/b$. So, by pigeonholing, there must exist at least one box left with a ratio of red balls to total balls at least this large. ■

Proof of Theorem 7: For convenience, we call vertices in the independent set R "red". First, we show there exists a good bin. We are given that $D_R(V-R) \geq \lambda D(V-R)$. We apply Lemma 10 where there is one "box" for each of the $\log_{1+\delta} n$ bins I_j. For each $v \in V-R$, if $v \in I_j$, we place $d(v)$ "balls" of which $d_R(v)$ are red into box j. So, the number of balls in box j equals $D(I_j \cap (V-R))$ out of which $D_R(I_j \cap (V-R))$ are red, and the number of balls total is $D(V-R)$ of which $D_R(V-R)$ are red. Lemma 10 tells us, taking $\epsilon = \delta$, that for some j_0, if we let $I = I_{j_0} \cap (V-R)$, then:

$$D_R(I) \geq \delta D_R(V-R)/(\log_{1+\delta} n) \text{ and} \quad (3)$$

$$D_R(I) \geq \lambda(1-\delta)D(I). \quad (4)$$

Informally, the set I of non-red vertices has the property that many edges have endpoints in I (since $D_R(I) = \tilde{\Omega}(D(V-R))$ by equation (3)), that almost a λ fraction of the edges leaving I enter red nodes (equation (4)), and that all nodes in I have similar degrees (since $I \subseteq I_{j_0}$). We do not know how to distinguish between edges with endpoints in R and other sorts of edges, so we do not know which I_j contains I, only that such an I_j must exist.

We now show that for some $v \in R$, the set $N(v) \cap I$ satisfies claims (1) and (2) of Theorem 7. Note that this completes the proof because $N(v) \cap [I_{j_0} \cap (V-R)] = N(v) \cap I_{j_0}$ since $v \in R$ and R is an independent set.

Define:

- $R' = \{v \in R : |N(v) \cap I| \geq \delta^2 d/\log_{1+\delta} n\}$.

R' is the set of red vertices such that $N(v) \cap I$ satisfies claim (1) of Theorem 7. We first show that nearly λ of the edges from the set I enter into R' and then use this to show that for some $v \in R'$, claim (2) of Theorem 7 holds. So, from the definition of R', we have:

$$\begin{array}{rcll} D_{R'}(I) & \geq & D_R(I) - |R|\delta^2 d/\log_{1+\delta} n & \\ & \geq & D_R(I) - D_R(V-R)\delta^2/\log_{1+\delta} n & \text{(since } D_R(V-R) = D(R) \geq d|R|) \\ & \geq & D_R(I) - \Big(D_R(I)(\log_{1+\delta} n)/\delta\Big)\Big(\delta^2/\log_{1+\delta} n\Big) & \text{(by equation (3))} \\ & \geq & D_R(I)(1-\delta). & \end{array}$$

Finally, applying equation (4) we have:

$$D_{R'}(I) \;>\; \lambda(1-2\delta)D(I). \tag{5}$$

We now claim that for some $v \in R'$, the set $N(v) \cap I$ satisfies claim (2) of Theorem 7. Essentially, the reason for this is that all vertices in I have similar degrees. The actual proof is by contradiction, using a counting argument.

Suppose for contradiction that: [6]

$$\text{For all } v \in R', \quad D_{R'}(N(v)\cap I) \;<\; \lambda(1-3\delta)D(N(v)\cap I). \tag{contr 6}$$

If this is the case, then it must also be true that:

$$\sum_{v\in R'} D_{R'}(N(v)\cap I) \;<\; \lambda(1-3\delta)\sum_{v\in R'} D(N(v)\cap I). \tag{contr 7}$$

Now, instead of writing each quantity as a sum over $v \in R'$, we would like to write each as a sum over $w \in I$. We can do this as follows.

We may write the sum $[\sum_{v\in R'} D(N(v)\cap I)]$ as $\sum_{v\in R'}\left[\sum_{w\in N(v)\cap I} d(w)\right]$ by the definition of D. Now, each vertex $w \in I$ is counted in the inside sum $d_{R'}(w)$ times since w is in the neighborhood of $d_{R'}(w)$ different vertices of R'. Thus, $\sum_{v\in R'} D(N(v)\cap I) = \sum_{w\in I} d_{R'}(w)d(w)$. Similarly, $\sum_{v\in R'} D_{R'}(N(v)\cap I) = \sum_{w\in I} d_{R'}(w)^2$.

Applying the inequality (contr 7) we have assumed for contradiction, we get:

$$\begin{aligned}
\sum_{w\in I} d_{R'}(w)^2 &< \lambda(1-3\delta)\sum_{w\in I} d_{R'}(w)d(w) \\
&< \lambda(1-3\delta)\sum_{w\in I} d_{R'}(w)(1+\delta)^{j_0+1} \quad (\text{since } d(w) < (1+\delta)^{j_0+1} \text{ for all } w \in I) \\
&= \lambda(1-3\delta)(1+\delta)^{j_0+1} D_{R'}(I). \qquad (\text{by definition of } D_{R'})
\end{aligned} \tag{8}$$

For any collection of values, the average of the squares is at least the square of the average. Thus:

$$\frac{1}{|I|}\sum_{w\in I} d_{R'}(w)^2 \;\geq\; \left[\frac{1}{|I|}\sum_{w\in I} d_{R'}(w)\right]^2 \;=\; \frac{D_{R'}(I)^2}{|I|^2}.$$

So, $D_{R'}(I)^2/|I| \leq \sum_{w\in I} d_{R'}(w)^2$. Combining this fact with equation (8), we have:

$$\frac{1}{|I|}D_{R'}(I)^2 \;<\; \lambda(1-3\delta)(1+\delta)^{j_0+1} D_{R'}(I). \tag{9}$$

Multiplying both sides of equation (9) by $|I|/D_{R'}(I)$, we get:

$$\begin{aligned}
D_{R'}(I) &< \lambda(1-3\delta)(1+\delta)^{j_0+1}|I| \\
&\leq \lambda(1-3\delta)(1+\delta)D(I) \quad (\text{since } d(w) \geq (1+\delta)^{j_0} \text{ for all } w \in I) \\
&< \lambda(1-2\delta)D(I).
\end{aligned}$$

This contradicts equation (5) and completes the proof of Theorem 7. ■

[6] It is always dangerous to display false equations, so we are labeling these inequalities with the symbol "contr" to emphasize that they are just being assumed for contradiction.

Proof of Theorem 8: We are given a set S such that $D_R(S) \geq \lambda' D(S)$; that is, at least a fraction of λ' of the edges leaving the set S (double-counting edges with both endpoints in S) enter into R. We want to show that at least one of the sets $N_i(S)$ both is large and has nearly a fraction λ' of its vertices in R. To do so, we apply Lemma 10 where we have one "box" for each set $N_i(S)$. We place a ball in box i for each endpoint in $N_i(S)$ of an edge from S to $N_i(S)$. A ball is red if the endpoint to which it corresponds is in R. The number of balls in box i is $D_{N_i(S)}(S)$ of which $D_{N_i(S)\cap R}(S)$ are red, and the number of balls total in the $\log_{1+\delta} n$ boxes is $D(S)$ of which $D_R(S)$ are red. By Lemma 10, taking $\epsilon = \delta$, for some i_0 ($0 \leq i_0 < \log_{1+\delta} n$),

1. $D_{N_{i_0}(S)\cap R}(S) \;\geq\; \delta D_R(S)/(\log_{1+\delta} n)$ and (10)
2. $D_{N_{i_0}(S)\cap R}(S)/D_{N_{i_0}(S)}(S) \;\geq\; (1-\delta)\lambda'$. (11)

By definition of $N_{i_0}(S)$, each vertex in $N_{i_0}(S)$ is incident to at least $(1+\delta)^{i_0}$ and less than $(1+\delta)^{i_0+1}$ edges from S. Thus,

$$D_{N_{i_0}(S)\cap R}(S) \;<\; |N_{i_0}(S)\cap R|(1+\delta)^{i_0+1}$$

and

$$D_{N_{i_0}(S)}(S) \;\geq\; |N_{i_0}(S)|(1+\delta)^{i_0}$$

which implies that:

$$\begin{aligned} |N_{i_0}(S)\cap R|/|N_{i_0}(S)| &\geq \left[D_{N_{i_0}(S)\cap R}(S)/D_{N_{i_0}(S)}(S)\right]/(1+\delta) \\ &\geq (1-\delta)\lambda'/(1+\delta) \\ &\geq (1-2\delta)\lambda'. \end{aligned} \tag{12}$$

Equations (10) and (12) show that the index i_0 satisfies both claims of the theorem. ∎

4.4 Applying the vertex-cover approximation

Given a graph H on N vertices, M edges, and with a minimum vertex cover of size N_{VC}, the BE/MS vertex-cover algorithm [BYE85][MS85] finds a vertex cover of size at most $\left(2 - \frac{\log\log N}{2\log N}\right) N_{VC}$ in time $O(NM)$.

If H has an independent set with at least $\frac{1}{2}(1-\frac{1}{\log N})N$ vertices, it must have a vertex cover of at most $\frac{1}{2}(1+\frac{1}{\log N})N$ vertices. So, the algorithm will find a vertex cover $W \subset V(H)$ of size at most:

$$\begin{aligned} \tfrac{1}{2}\left(1+\tfrac{1}{\log N}\right)\left(2-\tfrac{\log\log N}{2\log N}\right)N &= \left[1 - \tfrac{\log\log N}{4\log N} + \tfrac{1}{\log N} - \tfrac{\log\log N}{4(\log N)^2}\right]N \\ &< \left[1 - \Omega(\tfrac{1}{\log N})\right]N. \end{aligned}$$

Since W is a vertex cover, $V(H) - W$ is an independent set of size at least $\Omega(\frac{N}{\log N})$. So, we have the following lemma.

Lemma 11 *Given a graph H on N vertices with an independent set of size at least $\frac{1}{2}(1-\frac{1}{\log N})N$, the BE/MS algorithm can be used to find in polynomial time an independent set of size $\Omega(N/\log N)$.*

We now prove the Main Theorem.

Proof of Theorem 5: Step 1 of algorithm First-Approx ensures that no vertex has degree less than $f(n)$ for $f(n) = n^{2/5}\log^{8/5} n$. Step 2 ensures that no two vertices share more than $n/f(n)^2$ neighbors. Applying these values to Corollary 9 of the previous section yields the result that of the $O(n\log^4 n)$ subsets generated in Step 3 of Algorithm First-Approx, at least one set $T = T_{v,i,j}$ has $\Omega(f(n)^4/(n\log^7 n))$ vertices of which at least a fraction $\frac{1}{2}(1 - \frac{1}{\log n})$ are colored red under some legal 3-coloring of G. By Lemma 11, since $(1 - \frac{1}{\log n}) \geq (1 - \frac{1}{\log |T|})$, Step 4 of algorithm First-Approx will find an independent set in T of size $\Omega(f(n)^4/(n\log^8 n))$. We can thus make progress of Type 1 [Large-IS] on some $T_{v,i,j}$ in Step 4 of Algorithm First-Approx so long as:

$$f(n)^4/(n\log^8 n) \quad = \quad \Omega(n/f(n)).$$

Equivalently, we make progress towards an $O(f(n))$-coloring so long as $f(n)^5 = \Omega(n^2\log^8 n)$, or $f(n) = \Omega(n^{2/5}\log^{8/5} n)$. Thus, we have proved the Main Theorem. ■

5 Coloring 3-colorable graphs: improved algorithm

In this section, we present a procedure that improves on the bounds achieved by Algorithm First-Approx given in Section 4. The essence of the new algorithm is an improved method for forcing expansion (see Section 4.1) and making progress from regions of high density in a 3-colorable graph. This improves performance and results in coloring n-vertex 3-colorable graphs with only $\tilde{O}(n^{3/8})$ colors.

5.1 A useful lemma

We now present a lemma which is a strengthening of Corollary 4, and allows us to force a 3-colorable graph G to behave in a certain "nice" way. In particular, for any vertex v of G, for any subset S we select of $N(v)$ of size at least $(n\log^2 n)/f(n)^2$, the lemma allows us without loss of generality to force S to contain $\tilde{\Omega}(|S|)$ vertices of each of the two available colors (that is, the colors that v does not have), or else make progress towards an $f(n)$-coloring of G. This will be useful for forcing sets to expand "roughly evenly" into vertices of the available colors in the graph. This lemma requires the graph to be 3-colorable.

Let $f(n)$ be some "nearly-polynomial" function as in Definition 1.

Lemma 12 *Given a set $S \subseteq V(G)$ of size $\Omega((n\log^2 n)/f(n)^2)$, we can either make progress towards an $O(f(n))$-coloring of G or else guarantee that under every legal 3-coloring of G, set S contains less than $(1 - \frac{1}{4\log n})|S|$ vertices of any given color class.*

The idea of the proof is that if S consists of vertices nearly all of one color, say red, then its neighborhood should contain mostly blue and green vertices and have few red vertices. If this occurs, then $N(S)$ will have a large independent set of size $\max\{|N(S) \cap \text{green}|, |N(S) \cap \text{blue}|\}$. One can thus make progress on $N(S)$ using the BE/MS Vertex-Cover algorithm. The difficulty with this approach is that the neighborhood $N(S)$ need *not* have few red vertices. It could be, for example, that the red vertices in S tend to have a smaller degree than the others. Or, even if all vertices have the same degree, it could be that edges from the blue and green vertices of S all enter into different vertices in $N(S)$, but edges from red vertices in S tend to hit many vertices multiple times. To handle these difficulties, we will run a procedure separating vertices

and neighborhoods into bins depending on degree, in a similar manner to that done in the proofs of Theorems 7 and 8.

Proof of Lemma 12:

For convenience, let red be the color with the most vertices in S. The first goal is to find a large independent set $S' \subseteq S$. We can do this in a greedy fashion by deleting arbitrary edges from S. That is, begin with $S' = S$, and while S' is not an independent set, pick an arbitrary edge (a, b) between two vertices of S' and delete both endpoints from S' (let $S' \leftarrow S' - \{a, b\}$). If we ever have deleted more than $\frac{|S|}{4 \log n}$ pairs, this means we must have removed over $\frac{|S|}{4 \log n}$ vertices not in red from S (an edge can have at most one endpoint in red). So, we can guarantee that no color comprises more than $(1 - \frac{1}{4 \log n})$ of the vertices of S and halt. Otherwise (we do not delete more than $\frac{|S|}{4 \log n}$ edges from S), we will end with S' an independent set of size at least $(1 - \frac{1}{2 \log n})|S|$, which is $\Omega((n \log^2 n)/f(n)^2)$.

Since S' is independent and has size $\Omega((n \log^2 n)/f(n)^2)$, we can make progress Type 2 [Small-Nbhd] towards an $O(f(n))$-coloring of G if $|N(S')| \leq (n \log^2 n)/f(n)$, in which case we halt with "progress made". Otherwise, let $T = N(S')$, so $|T| \geq (n \log^2 n)/f(n)$.

The basic idea of the procedure now is the following. We first "throw out" edges so that the vertices in S' have disjoint neighborhoods in T. If at this point all vertices in S' had the same degree, we would be done: if set S' consisted almost entirely of red vertices, then set T would consist almost entirely of blue and green vertices. Since the vertices of S' may have differing degrees, we partition S' into bins based on degree in a similar fashion as done with the sets I_j defined in Section 3.3. For each bin, either it contains a good fraction of non-red vertices, or else its neighborhood is mostly blue and green. Thus, if a bin has many neighbors in T, we can either make progress using the BE/MS algorithm on the neighborhood or else have a guaranteed number of non-red vertices in S' (recall, our final goal is to guarantee that S has at least $\frac{1}{4 \log n}|S|$ non-red vertices.) Formally, we perform the following steps.

1. For each vertex w in T, arbitrarily mark one of the edges from w into S'. Let E' be the set of marked edges. Now, for each $v \in S'$, define its *marked neighborhood* $N'(v)$ by:

   $$N'(v) \;=\; \{w \in T \mid (v, w) \in E'\}.$$

   For any set $A \subseteq S'$, define the marked neighborhood of A similarly to be:

   $$N'(A) \;=\; \bigcup_{v \in A} N'(v).$$

   Note that by definition of E', if A and B are disjoint subsets of S', then their marked neighborhoods are disjoint as well, because each $w \in T$ is in the marked neighborhood of only one vertex of S'. (See Figure 1.)

2. Partition S' into subsets such that in each subset, if we consider only the edges in E', the minimum degree is at least half of the maximum degree. In particular, we partition S' into sets $S_0, \ldots, S_m$ for $m \leq \log n$ such that:

   $$S_i \;=\; \{v \in S' : |N'(v)| \in [2^i, 2^{i+1} - 1]\}.$$

   (We may ignore vertices in S' with no marked neighbors.)

   Observation: Notice that if more than a fraction $(1 - \frac{1}{2 \log n})$ of the vertices of some S_i are red, then *at most* $\frac{1}{\log n}$ of the vertices in $N'(S_i)$ can be red, since the non-red vertices in S_i can have at *most* twice as large a marked neighborhood in T as the red vertices do (and, as noted in Step 1, marked neighborhoods of disjoint subsets of S' are disjoint).

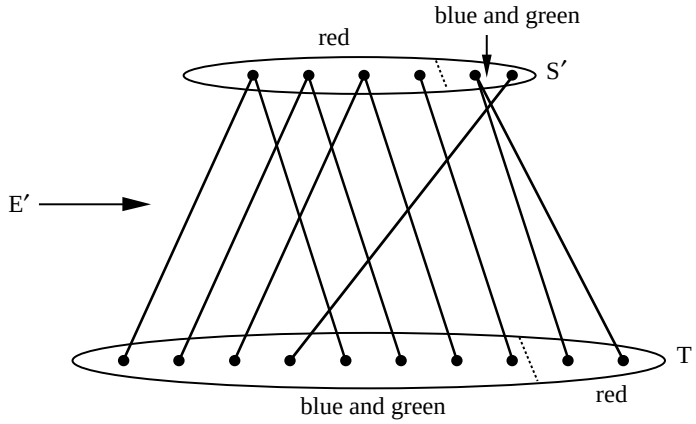

Figure 1: Vertices in S' have disjoint marked neighborhoods. If the vertices had nearly identical "marked degree," then a mostly red set S' would imply a mostly blue and green set T.

3. Now, pick i_0 such that $|N'(S_{i_0})|$ is maximized; so $|N'(S_{i_0})| \geq (\frac{1}{1+\log n})|T|$ since there are at most $(1+\log n)$ sets S_i and their neighborhoods are disjoint. Note that i_0 is not necessarily the largest index, since lower index sets might have enough vertices to compensate for having fewer neighbors per vertex.

4. We now apply the BE/MS vertex-cover algorithm (or equivalently, algorithm Approx-**IS** in the appendix) to the set $N'(S_{i_0})$. If it finds an independent set of size $\Omega(n/f(n))$, then we have made progress Type 1 [Large-IS] and can halt with "progress made".

   The reason we apply the BE/MS vertex cover algorithm is that *if* more than a fraction $(1 - \frac{1}{2\log n})$ of the vertices of S_{i_0} are red, *then* by the observation in Step 2, $N'(S_{i_0})$ has at most a $\frac{1}{\log n}$ fraction of its vertices red, so $N'(S_{i_0})$ has an independent set of at least $\frac{1}{2}(1-\frac{1}{\log n})$ of its vertices, namely either $N'(S_{i_0}) \cap$ blue or $N'(S_{i_0}) \cap$ green, whichever is larger. Thus, by Lemma 11, we find an independent set of size $\Omega(|N'(S_{i_0})|/\log n) = \Omega(n/f(n))$ since we have assumed $|T| \geq (n\log^2 n)/f(n)$ and $|N'(S_{i_0})| \geq \frac{1}{1+\log n}|T|$.

   So, if we do *not* make progress, we know it is *not* true that more than $(1 - \frac{1}{2\log n})$ of the vertices of S_{i_0} are red.

5. If we did not make progress in step 4, we know that at least $\frac{1}{2\log n}$ of the vertices in S_{i_0} are blue or green. Now, let $S' \leftarrow S' - S_{i_0}$ and let $T = N(S')$.

   If S' has not been reduced to less than 1/3 its original size, then go back to Step 1. Notice that in this case, we may still assume that $|T| \geq (n\log^2 n)/f(n)$ since S' still has size $\Omega((n\log^2 n)/f(n)^2)$.

   If S' is less than 1/3 its original size, then go on to Step 6.

6. If we reach this step, it means we have reduced S' to less than a third of its original size, and have done so by removing from S' sets containing at least a $\frac{1}{2\log n}$ fraction of blue and green vertices. Since S' originally had size at least $(1 - \frac{1}{2\log n})|S|$, this implies we must

have removed more than:

$$\frac{2}{3}\frac{1}{2\log n}\left[\left(1-\frac{1}{2\log n}\right)|S|\right] \quad \geq \quad \frac{1}{4\log n}|S|$$

blue and green vertices from S. So, we may halt with the guarantee asked for in the statement of the lemma since set S could not possibly have contained more than $(1 - \frac{1}{4\log n})|S|$ red vertices. ■

5.2 Making progress from dense regions

We will now use Lemma 12 to help take advantage of certain types of dense regions in 3-colorable graphs. In particular, we consider the case of two sets of vertices S and T where S is 2-colored under some legal 3-coloring of G and the number of edges between S and T is large compared with the sizes of the two sets. This occurs when S is a subset of the neighborhood of a vertex (e.g., a set $N(v) \cap I_j$) and T is some set $N_i(S)$ for a large i (see Section 3.3).

Theorem 13 *Given sets of vertices S and T in an n-vertex 3-colorable graph G, such that*

1. *S is 2-colored under some legal 3-coloring of G,*
2. *$D_T(S) = \Omega(|S|(n\log^2 n)/f(n)^2)$, and*
3. *$[D_T(S)]^3 = \Omega\Big(\Big[|S| + \max_{v\in S} d_T(v)\Big] \times \Big[|S||T|^2(n\log n)/f(n)^2 + |T||S|^2n^2/f(n)^4\Big]\Big)$,*

then we can make progress towards an $O(f(n))$-coloring of G.

Before proving this theorem, let us first make sense of the condition on $[D_T(S)]^3$ by considering a few examples. Suppose we wish to color with $f(n) = n^{3/8}$ colors, the set S has size $n^{3/8}$, and each vertex v in S has degree $n^{3/8}$ into T. Then, $\frac{D_T(S)}{|S|} = n^{3/8}$, which is greater than $n^{1/4}\log^2 n$ (condition 2). The main condition (condition 3) reduces to:

$$n^{18/8} \;\geq\; cn^{3/8}\left[|T|^2n^{5/8}\log n + |T|n^{10/8}\right].$$

Ignoring logarithmic factors, the theorem assures us we make progress if $|T| = \tilde{O}(n^{5/8})$. This is the basic idea for the $O(n^{3/8}\log^{5/2} n)$-coloring algorithm described later. For that application of this theorem, if T has $\tilde{\Omega}(n^{5/8})$ vertices, we will be able to find a large independent set inside T, and thus make progress of Type 1.

As another example, if we wished to color with $n^{0.35}$ colors, S had size $n^{0.35}$ and each vertex in S had degree $n^{0.35}$ into T, then the main condition reduces to

$$n^{2.1} \geq cn^{0.35}\left[|T|^2n^{0.65}\log n + |T|n^{1.3}\right].$$

In this case, we only make progress if $|T| = \tilde{O}(n^{0.45})$ (here the $|T|n^{1.3}$ term is dominant). However, we do not know how to make use of forcing $|T| = \tilde{\Omega}(n^{0.45})$.

Proof of Theorem 13: For convenience, let blue and green be the two colors that appear in S, and let us define the following notation.

- Let $D_{\text{total}} = D_T(S) = D_S(T)$.

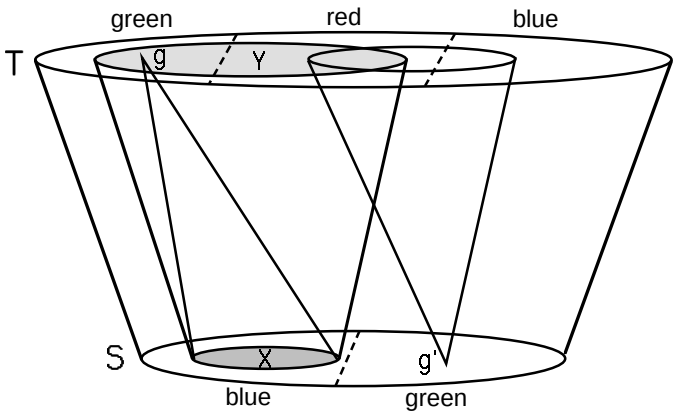

Figure 2: Vertex g and the sets X and Y. Also, green vertex $g' \in S$ (defined later) and the intersecting neighborhoods.

- Let $d_{\text{avg}} = D_{\text{total}}/|S|$ be the average degree into T of vertices in S.

We want to keep track of those vertices of T that have a reasonably large degree into S, so we define a subset T' of T by:

- $T' = \{w \in T \mid d_S(w) \geq \frac{1}{2}\frac{D_{\text{total}}}{|T|}\}$.

Since $D_S(T - T') < |T|\left[\frac{1}{2}\frac{D_{\text{total}}}{|T|}\right]$, we have $D_S(T') \geq \frac{1}{2}D_{\text{total}}$, or equivalently,

$$D_{T'}(S) \quad \geq \quad D_{\text{total}}/2. \tag{13}$$

We also want to look at those vertices in S that have reasonably large degree into T', so define:

- $S' = \{v \in S \mid d_{T'}(v) \geq \frac{1}{2}\frac{D_{T'}(S)}{|S|}\}$.

Since $D_{T'}(S - S') < |S|\left[\frac{1}{2}\frac{D_{T'}(S)}{|S|}\right]$, we have: $D_{T'}(S') \geq \frac{1}{2}D_{T'}(S)$, which by equation 13 implies:

$$D_{T'}(S') \quad \geq \quad D_{\text{total}}/4. \tag{14}$$

Also, by definition of S' and equation (13), if $v \in S'$ then $d_{T'}(v) \geq \frac{1}{4}\frac{D_{\text{total}}}{|S|}$ or equivalently,

$$d_{T'}(v) \quad \geq \quad \tfrac{1}{4}d_{\text{avg}} \qquad \text{for all } v \in S'. \tag{15}$$

Since we are given (condition 2) that $d_{\text{avg}} = \Omega((n\log^2 n)/f(n)^2)$, this implies that all $v \in S'$ have $d_T(v) \geq d_{T'}(v) = \Omega((n\log^2 n)/f(n)^2)$. Thus, by Lemma 12 (applied to the sets $N(v) \cap T$), we can guarantee that each vertex $v \in S'$ has at least a fraction $\frac{1}{4\log n}$ of its edges into T entering into non-red vertices.

So, for some non-red color, say green without loss of generality, at least $D_T(S')/(8\log n)$ edges from S' enter into green vertices of T. This implies that some green vertex $g \in T$ has degree at least $D_T(S')/(8|T|\log n)$ into S'. Now, define (see Figure 2):

- $X = N(g) \cap S'$.

- $Y = N(X) \cap T'$.

So, we have:

$$\begin{aligned} |X| &\geq \tfrac{1}{8} D_T(S')/(|T| \log n) \\ &\geq \tfrac{1}{32} D_{\text{total}}/(|T| \log n) \\ &= \Omega\left(\left(\tfrac{|S|}{|T|}\right)\left(\tfrac{d_{\text{avg}}}{\log n}\right)\right). \end{aligned} \tag{16}$$

Note that set X consists entirely of blue vertices, and since Y is in the neighborhood of a blue set, Y contains only red and green vertices. We want to show that Y is large, because we will later intersect Y with a red and blue set to get a large monochromatic (red) set, which will allow us to make progress. We show that Y must be large as follows.

By Theorem 3 we may assume that no two vertices of X share more than $n/f(n)^2$ neighbors in T'. Now suppose that $|X| < \frac{f(n)^2}{n}(\frac{1}{8} d_{\text{avg}})$. In this case, each vertex $v \in X$ can share at most $|X|(n/f(n)^2) < \frac{1}{8} d_{\text{avg}}$ neighbors with other vertices in X. This implies, by equation (15), that v must have at least $\frac{1}{8} d_{\text{avg}}$ neighbors in T' *not* shared with any other vertices of X. So, set Y must have size at least $\Omega(|X| d_{\text{avg}})$.

If $|X| \geq \frac{f(n)^2}{n}(\frac{1}{8} d_{\text{avg}})$, then if we only consider the first $\frac{f(n)^2}{n}(\frac{1}{8} d_{\text{avg}})$ of the vertices of X, we still get that $|Y| = \Omega(\frac{f(n)^2}{n}(d_{\text{avg}})^2)$. So, whichever case occurs, we have:

$$|Y| = \Omega\left(\min\left\{ |X| d_{\text{avg}},\ \tfrac{f(n)^2}{n}(d_{\text{avg}})^2 \right\}\right). \tag{17}$$

By definition, Y is a subset of T' and vertices of T' all have a high degree into S. So, we can lower bound the degree of Y into S by:

$$\begin{aligned} D_S(Y) &\geq \left(\tfrac{1}{2}\tfrac{D_{\text{total}}}{|T|}\right)|Y| \\ &= \tfrac{1}{2}\tfrac{|S|}{|T|} d_{\text{avg}} |Y| \\ &= \Omega\left(\min\left\{ |X|\tfrac{|S|}{|T|}(d_{\text{avg}})^2,\ \tfrac{f(n)^2}{n}(d_{\text{avg}})^3 \tfrac{|S|}{|T|} \right\}\right) \quad \text{(by equation 17)} \\ &= \Omega\left(\min\left\{ \left[\tfrac{|S|}{|T|}\right]^2 (d_{\text{avg}})^3/\log n,\ \tfrac{f(n)^2}{n}(d_{\text{avg}})^3 \tfrac{|S|}{|T|} \right\}\right). \quad \text{(by equation 16)} \end{aligned} \tag{18}$$

Now we apply condition 3 in the statement of the theorem. The condition (dividing both sides by $|S|^3$) states that $(d_{\text{avg}})^3 = \left[|S| + \max_{v \in S} d_T(v)\right] \cdot \Omega\left(\frac{|T|^2}{|S|^2}\frac{n}{f(n)^2}\log n + \frac{|T|}{|S|}\frac{n^2}{f(n)^4}\right)$. So, this implies both that:

$$\left[\tfrac{|S|}{|T|}\right]^2 (d_{\text{avg}})^3/\log n = \left[|S| + \max_{v \in S} d_T(v)\right] \cdot \Omega\left(\tfrac{n}{f(n)^2}\right) \tag{19}$$

and

$$\tfrac{f(n)^2}{n}(d_{\text{avg}})^3 \left[\tfrac{|S|}{|T|}\right] = \left[|S| + \max_{v \in S} d_T(v)\right] \cdot \Omega\left(\tfrac{n}{f(n)^2}\right). \tag{20}$$

Thus, combining both equations (19) and (20) with equation (18), we get:

$$D_S(Y) = \Omega\left(\tfrac{n}{f(n)^2}\left[|S| + \max_{v \in S} d_T(v)\right]\right). \tag{21}$$

It now must be that one of the following two cases occurs. The first case is that there is some green vertex $g' \in S$ in the neighborhood of more than $\frac{1}{2} D_S(Y)/|S|$ vertices of Y. In this

case, according to equation (21), it must be that $D_{\{g'\}}(Y) = \Omega(n/f(n)^2)$. So, $N(g') \cap Y$ is a set of $\Omega(n/f(n)^2)$ vertices, *all of which are* red since $N(g') \subseteq$ blue $\cup$ red and $Y \subseteq$ red $\cup$ green; see Figure 2. Thus, we can make progress on this monochromatic set using Corollary 4.

The other possibility is that *no* green vertex in S is in the neighborhood of more than $\frac{1}{2}D_S(Y)/|S|$ vertices of Y. In this case, the set of all vertices in S hit by more than $\frac{1}{2}D_S(Y)/|S|$ edges from Y is all blue. Define Z to be that set; that is:

- $Z = \{v \in S \mid d_Y(v) > \frac{1}{2}D_S(Y)/|S|\}$.

Clearly, the number of edges between vertices of Y and vertices in $(S-Z)$ is at *most* $|S|(\frac{1}{2}D_S(Y)/|S|) = \frac{1}{2}D_S(Y)$. So, $D_Z(Y) \geq \frac{1}{2}D_S(Y)$. Thus, we can bound the size of Z by:

$$\begin{aligned} |Z| &\geq \tfrac{1}{2}D_S(Y)/\max_{v\in S} d_Y(v) \\ &\geq \tfrac{1}{2}D_S(Y)/\max_{v\in S} d_T(v) \end{aligned}$$

which by equation (21) implies:

$$|Z| = \Omega(n/f(n)^2).$$

Since Z is monochromatic (blue) we can again use Corollary 4 to make progress. So, whichever of the two cases occurs, we have made progress towards an $O(f(n))$-coloring.

The final algorithm for making progress given our sets S and T is as follows:

Algorithm Dense-Region-Progress**:**

Given: *Sets S and T satisfying the conditions of Theorem 13 in some graph G.*

Output: *Progress towards an $O(f(n))$-coloring of G.*

1. *Run the algorithm of Lemma 12 on $N(v) \cap T$ for all $v \in S$. If any runs make progress towards an $O(f(n))$-coloring, then halt. Otherwise, we know there are many edges from S into red, blue, and green vertices of T under any legal 3-coloring of G.*
2. *If for some pair of vertices $u, v \in S$, we have $|N(u) \cap N(v)| \geq n/f(n)^2$, then use Theorem 3 to make progress.*
3. *Otherwise, for each vertex $v \in T$,*
   (a) *let $Y = N(N(v) \cap S) \cap T$ and let $Z = \{w \in S : d_Y(w) \geq n/f(n)^2\}$.*
   (Note that we do not actually need to use the sets S' and T'; they were just convenient for the analysis.)
   (b) *Run the algorithm of Corollary 4 on Z.*
   (c) *For each $w \in Z$, run the algorithm of Corollary 4 on $Y \cap N(w)$.*

The above proof guarantees that this algorithm makes progress. ■

5.3 The coloring algorithm

We now combine algorithms First-Approx and Dense-Region-Progress to get an improved algorithm guaranteed to $\tilde{O}(n^{3/8})$-color any n-vertex 3-colorable graph.

Algorithm Improved-Approx**:**

Given: *$G = (V, E)$, a 3-colorable graph on n vertices. Let $f(n) = n^{3/8}(\log n)^{5/2}$.*

Output: *Progress towards an $O(f(n))$-coloring of G.*

1. *For each vertex v, if $d(v) < f(n)$, make progress Type 2* [Small-Nbhd].
2. *Otherwise, for each vertex v, for each $i, j \in \{0, 1, \ldots, 5(\log n)^2\}$:*
   (a) *Let $S = N(v) \cap I_j$.*
   (b) *Let $T = N_i(S)$.*
   (c) *If $|T| \geq n^{5/8}/(\log n)^{3/2}$, run the* BE/MS *Vertex-Cover approximation algorithm. If we find an independent set of size at least $n/f(n)$, we have made progress Type 1* [Large-IS].
   (d) *If S and T satisfy the conditions of Theorem 13, then make progress using Algorithm* Dense-Region-Progress.

Theorem 14 *Algorithm* Improved-Approx *will make progress towards an $O(n^{3/8}(\log n)^{5/2})$-coloring of any n-vertex 3-colorable graph.*

Proof: Assume Algorithm Improved-Approx does not make progress in Step 1. So, we know that the minimum degree $d \geq f(n) = n^{3/8}(\log n)^{5/2}$. As in Section 4, let $R =$ red be the color class with $D(\mathsf{red}) = \max(D(\mathsf{red}), D(\mathsf{blue}), D(\mathsf{green}))$.

We now apply some of the facts proven in Section 4.3.2. Theorem 7 guarantees us that for some vertex $v \in R$ and some index j, the set $S = N(v) \cap I_j$ in Step 2(a) has the property that:

$$|S| \geq \delta^2 f(n)/\log_{1+\delta} n, \text{ and} \tag{22}$$

$$D_R(S) \geq \tfrac{1}{2}(1-3\delta)D(S), \tag{23}$$

where $\delta = \frac{1}{5\log n}$. Note that for the given value of f, equation (22) and the definition of δ imply that:

$$|S| = \Omega(n^{3/8}/(\log n)^{3/2}). \tag{24}$$

Theorem 8 (using $\lambda' = \frac{1}{2}(1-3\delta)$) shows that for some index i, the set $T = N_i(S)$ of step 2(b) has the property that:

$$D_{T\cap R}(S) \geq \delta D_R(S)/\log_{1+\delta} n, \text{ and} \tag{25}$$

$$|T \cap R|/|T| \geq \tfrac{1}{2}(1-2\delta)(1-3\delta). \tag{26}$$

Let us now, for the rest of the proof, fix two such sets S and T satisfying equations (22) through (26). We now show that these equations and the definitions of S and T will ensure success of the algorithm.

Suppose first that $|T| \geq n^{5/8}/(\log n)^{3/2}$. By equation 26 above, set T contains an independent set ($T \cap R$) of at least a fraction $\frac{1}{2}(1-\frac{1}{\log n})$ of its vertices (using $\delta = \frac{1}{5\log n}$). So by Lemma 11, the BE/MS vertex-cover algorithm finds an independent set of size $\Omega\left(n^{5/8}/(\log n)^{5/2}\right) = \Omega(n/f(n))$ so we make progress Type 1 [Large-IS] in Step 2(c).

On the other hand, if $|T| < n^{5/8}/(\log n)^{3/2}$, then we just need to show that S and T satisfy the conditions of Theorem 13. Clearly, S is 2-colored under any legal 3-coloring of G since $S \subseteq N(v)$, so Condition 1 is satisfied. For $f(n) = n^{3/8}(\log n)^{5/2}$, Condition 2 reduces to $D_T(S)/|S| = \Omega\left(n^{1/4}/(\log n)^3\right)$, which is found to be easily met using equations (23) and (25) as follows.

$$D_T(S) \geq D_{T\cap R}(S) = \Omega\left(D(S)/(\log n)^3\right) \tag{27}$$

$$= \Omega(d|S|/(\log n)^3). \tag{28}$$

So,

$$\begin{aligned} D_T(S)/|S| &\geq \Omega(n^{3/8}/(\log n)^{1/2}) && (29)\\ &= \Omega(n^{1/4}/(\log n)^3). && (30)\end{aligned}$$

The last task is to show that Condition 3 is satisfied, which for the given value of f, reduces to the requirement that

$$[D_T(S)]^3 \;=\; \Omega\left(\Big[|S| + \max_{v\in S} d_t(v)\Big] \cdot \Big[|S|\,|T|^2 \frac{n^{1/4}}{(\log n)^4} + |T|\,|S|^2 \frac{n^{1/2}}{(\log n)^{10}}\Big]\right). \qquad (31)$$

To show that this requirement holds, we *upper* bound the quantities $|S|$, $|T|$, and $\max_{v\in S} d_T(v)$. From equation (29), we have

$$|S| \;=\; O\Big((\log n)^{1/2} D_T(S)/n^{3/8}\Big). \qquad (32)$$

Next, our very condition for this case was that:

$$|T| \;=\; O\Big(n^{5/8}/(\log n)^{3/2}\Big). \qquad (33)$$

Finally, since $S \subseteq I_j$ so all vertices of S have nearly the same degree (though not necessarily the same degree into T), we can bound $\max_{v\in S} d_T(v)$ as follows:

$$\begin{aligned} \max_{v\in S} d_T(v) &= O(D(S)/|S|) \\ &= O(D_T(S)(\log n)^3/|S|) && \text{(using equation 27)} \\ &= O\Big(D_T(S)(\log n)^3(\log n)^{3/2}/n^{3/8}\Big) && \text{(using equation 24)} \\ &= O\Big(D_T(S)(\log n)^{9/2}/n^{3/8}\Big). && (34)\end{aligned}$$

The three equations (32), (33), and (34) allow us to reduce requirement (31) to the condition that:

$$\begin{aligned} [D_T(S)]^3 &= \Omega\left(\Big[(\log n)^{9/2}\frac{D_T(S)}{n^{3/8}}\Big] \cdot \Big[D_T(S)\frac{n^{9/8}}{(\log n)^{13/2}} + D_T(S)^2\frac{n^{3/8}}{(\log n)^{21/2}}\Big]\right) \\ &= [D_T(S)]^2 \cdot \Omega\left(\frac{n^{3/4}}{(\log n)^2} + \frac{D_T(S)}{(\log n)^6}\right). && (35)\end{aligned}$$

Equivalently, we just have the requirement that $D_T(S) = \Omega(n^{3/4}/(\log n)^2 + D_T(S)/(\log n)^6)$. Clearly, $D_T(S) = \Omega(D_T(S)/(\log n)^6)$ so we simply need $D_T(S) = \Omega\Big(n^{3/4}/(\log n)^2\Big)$. We are now done, because combining equations (29) and (24) yields:

$$\begin{aligned} D_T(S) &= \Omega\Big(|S|\, n^{3/8}/(\log n)^{1/2}\Big) \\ &= \Omega\Big(n^{3/4}/(\log n)^2\Big).\end{aligned}$$

Thus, Step 2(d) of Algorithm Improved-Approx makes progress. ∎

	$k = 3$	4	5	6	7	general
Wigderson [Wig83]	$n^{1/2}$ $n^{0.5}$	$n^{2/3}$ $n^{0.667}$	$n^{3/4}$ $n^{0.75}$	$n^{4/5}$ $n^{0.8}$	$n^{5/6}$ $n^{0.833}$	$n^{1-\frac{1}{k-1}}$
base: $k = 3$	$n^{3/8}$ $n^{0.375}$	$n^{8/13}$ $n^{0.615}$	$n^{13/18}$ $n^{0.722}$	$n^{18/23}$ $n^{0.783}$	$n^{23/28}$ $n^{0.821}$	$n^{1-\frac{1}{k-7/5}}$
base: $k = 4$	— —	$n^{3/5}$ $n^{0.6}$	$n^{5/7}$ $n^{0.714}$	$n^{7/9}$ $n^{0.778}$	$n^{9/11}$ $n^{0.818}$	$n^{1-\frac{1}{k-3/2}}$
best we have	$n^{3/8}$ $n^{0.375}$	$n^{3/5}$ $n^{0.6}$	$n^{\frac{91}{131}}$ $n^{0.695}$	$n^{\frac{105}{137}}$ $n^{0.766}$	$n^{\frac{5301}{6581}}$ $n^{0.806}$	

Table 1: Summary of results in "$\tilde{O}$" notation for the number of colors used to color k-chromatic graphs for various combinations of algorithms. Items "base: $k = 3$" and "base: $k = 4$" correspond to using Algorithm Recursive-Color with Algorithm Multi-Stage-Color as a base case for $k = 3$ or 4 respectively.

6 Coloring k-colorable graphs

We now consider two different methods for using the preceding techniques developed for 3-colorable graphs to improve the bounds for approximately coloring k-colorable graphs for fixed $k > 3$. One method is simply to use the preceding algorithms as an improved base case for a recursive strategy used by Wigderson [Wig83]. A second method is to directly extend the above algorithms for $k > 3$. For the latter approach, one needs both an analog of the shared neighborhood condition (Theorem 3), and a way to cascade together several applications of the distance-2 neighbor-taking process (Step 3 of Algorithm First-Approx) so that we can "pump up" the relative size of the largest independent set. We will see that the second method yields better asymptotic bounds than the first, though with diminishing returns as k increases. However, the running time of the second method grows as $(n \log^2 n)^{2k+O(1)}$ while the running time of the first is dominated just by the time taken by the base-case algorithm. The two methods can be combined, providing a time/performance tradeoff, by choosing some k_0 and using the second method as a base case for the first method for $k \geq k_0$. This will result in an algorithm with running time $O((n \log^2 n)^{2k_0+c})$ for some constant c.

The results of these approaches are summarized (in "$\tilde{O}$" notation) in Table 1. The first row shows the bound for using Wigderson's algorithm with base case at $k = 2$. The second and third rows show how the bounds are improved when we use the new coloring method as base cases for $k = 3$ and $k = 4$ respectively. The last row shows the best bounds we can get using the direct extension. The direct-extension algorithm uses random bits, so the bounds in the last two rows are with high probability over the coin tosses of the algorithm. See Corollaries 16 and 21 for more precise bounds.

6.1 A simple recursive approach

A standard method [Wig83][BR88][Hal90] to approximately color k-colorable graphs is to pick a vertex of high degree and recursively try to color its $(k-1)$-colorable set of neighbors with as few colors as possible. When we get to a 2-colorable set, we can just directly 2-color that set in the standard way. For example, Wigderson's algorithm for coloring k-colorable graphs with $kn^{1-1/(k-1)}$ colors can be described as follows:

Wigderson's Algorithm for k-colorable graphs:

Given: *A k-colorable graph G on n vertices.*

Output: *A coloring with at most $kn^{1-1/(k-1)}$ colors.*

1. *If there exists a vertex v with at least $n^{1-1/(k-1)}$ neighbors, then color the neighborhood recursively with $(k-1)\left(n^{1-1/(k-1)}\right)^{1-\frac{1}{k-2}} = (k-1)\left(n^{\frac{k-2}{k-1}}\right)^{\frac{k-3}{k-2}} = (k-1)n^{\frac{k-3}{k-1}}$ colors. Then remove those nodes from the graph and the colors from the palette. Note that this step can be executed at most $n^{1/(k-1)}$ times, resulting in a total of $(k-1)n^{\frac{k-3}{k-1}+\frac{1}{k-1}} = (k-1)n^{1-1/(k-1)}$ colors used in this step.*
2. *Otherwise, greedily color the graph left with $n^{1-1/(k-1)}$ colors.*
   So, the total number of colors used in both steps together is

$$kn^{1-1/(k-1)}.$$

   (Note that for the base case of $k = 2$, we have $2 = 2n^{1-1/(2-1)}$.)

The algorithms presented in the previous sections allow one to stop at $k = 3$ as a base case instead of $k = 2$ in this type of procedure and thus use fewer colors. More generally, we can describe when a bound achieved for coloring graphs of chromatic number k_0 will improve the performance of this kind of recursive procedure for graphs of higher chromatic number. In particular, suppose we have an algorithm $\mathcal{A}$ to color any n-vertex k_0-colorable graph with $\tilde{O}(n^\alpha)$ colors. Then, the important quantity for this approach, which we call the *recursive performance* $r(\mathcal{A})$ of the algorithm, is:

$$r(\mathcal{A}) \quad = \quad k_0 - \frac{1}{1-\alpha}. \tag{36}$$

If an algorithm has a higher value of r, then the bounds achieved by using that as a base case for $k > k_0$ will be improved. Specifically, the recursive algorithm will color k-colorable graphs for $k \geq k_0$ with $\tilde{O}\left(n^{1-1/(k-r(\mathcal{A}))}\right)$ colors. So, for example, using the fact that we can 2-color 2-colorable graphs ($k_0 = 2, \alpha = 0$), we find $r = 1$ and the bound is $\tilde{O}\left(n^{1-1/(k-1)}\right)$. Using the improved bounds for coloring 3-colorable graphs in Section 5 ($k_0 = 3, \alpha = 3/8$), we get $r = 3 - \frac{1}{5/8} = 7/5$, so the improved bound for $k \geq 3$ is:

$$\tilde{O}\left(n^{1-\frac{1}{k-7/5}}\right) \quad \text{colors.} \tag{37}$$

Later, in Section 6.2, we will see how to color 4-colorable graphs with $\tilde{O}(n^{3/5})$ colors, so we get $r = 4 - \frac{1}{2/5} = 3/2$. Thus, for $k \geq 4$, we can color with $\tilde{O}(n^{1-\frac{1}{k-3/2}})$ colors.

The following theorem more precisely describes the bounds achieved by the recursive approach.

Theorem 15 *Given an algorithm $\mathcal{A}$ to color any m-vertex k_0-colorable graph with $cm^\alpha \log^\beta m$ colors, then algorithm* Recursive-Color($\mathcal{A}$) *below can color any n-vertex k-colorable graph ($k \geq k_0$) with at most:*

$$C_k(n) \quad = \quad [c + (k - k_0)]n^{1-1/(k-r)} \left(\log n\right)^{\beta\left[\frac{k_0-r}{k-r}\right]} \tag{38}$$

colors, where $r = r(\mathcal{A}) = k_0 - \frac{1}{1-\alpha}$.

Using Theorem 15 and the bounds achieved by algorithm Improved-Approx, ($k_0 = 3, \alpha = 3/8, \beta = 5/2$),we can restate formula (37) more precisely in the following corollary.

Corollary 16 *Algorithm* Recursive-Color(Improved-Approx) *colors any n-vertex k-colorable graph ($k \geq 3$) with at most*

$$O\left(n^{1-\frac{1}{k-7/5}}(\log n)^{\frac{4}{k-7/5}}\right)$$

colors.

The recursive algorithm to achieve these bounds is described below.

Algorithm Recursive-Color**:** (Variant on Wigderson's algorithm)

Given: *An n-vertex k-colorable graph G and an algorithm $\mathcal{A}$ to color any m-vertex k_0-colorable graph with at most $C_{k_0}(m) = cm^{\alpha}\log^{\beta} m$ colors ($k_0 \leq k$).*

Output: *A $C_k(n)$-coloring of G, for $C_k(n)$ as defined in equation (38).*

1. *Let $r = k_0 - \frac{1}{1-\alpha}$.*
2. *Let $f(n,k) = n^{1-1/(k-r)}(\log n)^{\beta\frac{k_0-r}{k-r}}$.*
3. *While there exists a vertex with at least $f(n,k)$ neighbors, select $f(n,k)$ of its neighbors and color them with $C_{k-1}(f(n,k))$ colors. Remove those nodes from the graph and the colors from the palette.*

   Note that we can execute this step at most $n/f(n,k)$ times.
4. *Otherwise, greedily color the graph with $f(n,k)$ colors.*

Proof of Theorem 15: Let $\mathcal{A}$ be an algorithm that colors any m-vertex k_0-colorable graph with $cm^{\alpha}\log^{\beta} m$ colors and let $r = r(\mathcal{A})$. We will use $C_k(n)$ to denote the coloring bound achieved on n-vertex k-colorable graphs. First, formula (38) in the statement of the theorem holds for the base case of $k = k_0$ since for $k = k_0$, we have:

$$\begin{aligned} C_{k_0}(n) &= cn^{1-\frac{1}{1/(1-\alpha)}}(\log n)^{\beta\cdot 1} \\ &= cn^{\alpha}\log^{\beta} n. \end{aligned}$$

Let $c_k = c + (k - k_0)$ and let $f(n,k) = n^{\frac{k-r-1}{k-r}}(\log n)^{\beta\frac{k_0-r}{k-r}}$ as in Algorithm Recursive-Color. So, assuming the bounds of Theorem 15 inductively for $k' < k$, we need to show that $C_k(n) \leq c_k f(n,k)$.

Since we can loop in step 3 of Algorithm Recursive-Color at most $n/f(n,k)$ times, this results in the recurrence:

$$C_k(n) \leq C_{k-1}\left(f(n,k)\right)[n/f(n,k)] + f(n,k).$$

So, substituting in the bounds of Theorem 15 inductively, we have:

$$\begin{aligned} C_k(n) &\leq \left[c_{k-1}[f(n,k)]^{1-1/(k-r-1)}[\log f(n,k)]^{\beta\left(\frac{k_0-r}{k-r-1}\right)}\right]\left[\tfrac{n}{f(n,k)}\right] + f(n,k) \\ &< c_{k-1}[f(n,k)]^{1-1/(k-r-1)}[\log n]^{\beta\left(\frac{k_0-r}{k-r-1}\right)}\left[\tfrac{n}{f(n,k)}\right] + f(n,k) \\ &= c_{k-1}n[f(n,k)]^{-1/(k-r-1)}[\log n]^{\beta\left(\frac{k_0-r}{k-r-1}\right)} + f(n,k) \end{aligned}$$

$$
\begin{aligned}
&= c_{k-1} n \left(n^{\frac{k-r-1}{k-r}} \right)^{\frac{-1}{k-r-1}} \left([\log n]^{\beta \frac{k_0-r}{k-r}} \right)^{\frac{-1}{k-r-1}} [\log n]^{\beta\left(\frac{k_0-r}{k-r-1}\right)} + f(n,k) \\
&= c_{k-1} n^{1-\frac{1}{k-r}} [\log n]^{\beta\left(\frac{k_0-r}{k-r}\right)} + f(n,k) \\
&= c_{k-1} f(n,k) + f(n,k) \\
&= c_k f(n,k). \quad \blacksquare
\end{aligned}
$$

6.2 Directly extending the $k = 3$ algorithm

6.2.1 Intuition

In this section, we describe how the methods of Algorithm First-Approx of Section 4 can be applied directly to graphs of higher chromatic number, yielding improved coloring bounds for such graphs. Unfortunately, we do not know a way to extend the approach of Algorithm Improved-Approx in a similar way, though it can still provide a useful "base case".

The main idea of Algorithm First-Approx was to look at large subsets of the distance-2 neighbors of vertices in a 3-colorable graph: in particular, the sets $N_i(N(v) \cap I_j)$ for each vertex v and each pair of indices i, j. The "well-distributed" property proved in Theorems 7 and 8 ensures that one such set will be nearly half red under some legal 3-coloring of the graph, and the expansion property of Theorem 3 ensures the set is large as well.

While the expansion property depended heavily on the graph being 3-colorable, the theorems forcing good distribution require only that the given graph have an independent set of large total degree (see Section 4.3.2). In particular, they simply require that there exist a large independent set R such that $D_R(V-R) \geq \lambda D(V-R)$ for some constant λ and that the graph have sufficiently large minimum degree. So, we could conceivably make progress on graphs of a higher chromatic number than 3 by cascading several applications of the distance-2 neighbor-taking stage in the following way.

Suppose, say, G is a 5-colorable graph and we wish to color G with $f(n)$ colors. Then, we know there exists an independent set R such that $D_R(V-R) \geq \frac{1}{4} D(V-R)$ and we can establish a minimum degree of $f(n)$. If we could guarantee that no two vertices shared too many neighbors, we could look at the sets $T_{v,i,j}$ and be assured that one will be large and have an independent set $R' = R \cap T_{v,i,j}$ such that $|R'| \approx \frac{1}{4}|T_{v,i,j}|$ using Theorems 7 and 8. Let us now focus on the subgraph G' induced by $T_{v,i,j}$, and let $V' = T_{v,i,j}$. Suppose we could in addition somehow ensure that within G', the vertices of R' had about the same average degree as the other vertices of V'. Then we would have $D(R') \approx \frac{1}{4} D(V')$, which would imply that:

$$D_{R'}(V'-R') \approx \tfrac{1}{3} D(V'-R'), \tag{39}$$

since $D_{R'}(V'-R') = D(R')$ and $D(R') \approx \frac{1}{4} D(V') = \frac{1}{4}(D(V'-R') + D(R'))$, where we are now counting degrees only within G'.

Now, if we re-establish a minimum degree without destroying (39) above, we could then reapply the distance-2 neighbor-taking process within G' to get a set V'' containing an independent set R'' such that $|R''| \approx \frac{1}{3}|V''|$. If again we could ensure that $D(R'') \approx \frac{1}{3} D(V'')$ within the new graph G'', we would get:

$$D_{R''}(V''-R'') \approx \tfrac{1}{2} D(V''-R'').$$

Thus, one final application of examining the sets $T_{v,i,j}$ within G'' will yield some set on which the BE/MS vertex-cover algorithm makes progress.

So, the two main ingredients needed to make this procedure go through are (1) how to ensure that no two vertices share too many neighbors in common, and (2) how to get from $|R'| \approx \lambda|V'|$ to $D(R') \approx \lambda D(V')$. These problems are solved in the following sections.

6.2.2 The bootstrapping algorithm

We now describe procedures that allow us to "bootstrap" applications of Algorithm First-Approx to graphs of higher chromatic number. The resulting algorithm Multi-Stage-Color will color any n-vertex k-colorable graph with:

- $f_k(n) = O(n^{\alpha(k)} \log^{\beta(k)} n)$ colors,

  where $\alpha(k)$ will be defined inductively in k, and $\beta(k)$ is a nondecreasing function such that $\beta(k) \leq 5.5$. The exponent β of the logarithm in fact approaches 5.5 as $k \to \infty$. Because α is the critical value and the log factors are low-order terms, for purposes of simpler analysis we will not attempt to get tight bounds and assume β is fixed at 5.5 for all $k > 3$.

For base cases, $\alpha(2) = 0$ and $\alpha(3) = 3/8$ using algorithm Improved-Approx. The recursive formula for $\alpha(k)$ for $k > 3$ is:

$$\frac{1}{1-\alpha(k)} \quad = \quad 2 - \frac{1}{2^{k-2}} + \frac{1}{1-\alpha(k-2)} \left(1 - \frac{1}{2^{k-2}}\right). \tag{40}$$

We will examine this formula in more detail later, but we just note here that α is non-decreasing in k.

We need in this section to redefine the value δ to depend on the chromatic number k of the graph G we wish to color. In particular, we shall use:

- $\delta = \delta(k) \ = \ \frac{1}{4^k \log n}$.

The sets I_j and $N_i(v)$ used in Section 4 now depend on this new quantity.

As mentioned previously, the theorems of Section 4.3.2 forcing good distribution do not require that the graph be 3-colorable, only that there exist a large independent set R such that $D_R(V-R) \geq \lambda D(V-R)$ for some constant λ and that the graph have sufficiently large minimum degree. Let us, in fact, repeat Corollary 9 here, removing all mention of the chromatic number of the graph. (The fact that the graph was 3-colorable was used only in showing that $\lambda \geq 1/2$.)

Corollary 17 (Variant of Corollary 9) *Suppose $G = (V, E)$ is an n-vertex graph such that (1) no two vertices share more than s neighbors, (2) G has minimum degree $d_{min} \geq (10 \log n)/\delta^2]$, and (3) G contains an independent set R such that $D_R(V - R) \geq \lambda D(V - R)$ for some constant $\lambda \in [0, 1]$. Then, for any $\delta = \frac{1}{\Theta(\log n)}$, for some $v \in V$ and some $i, j \in [0, \ldots, \log_{1+\delta} n]$, the set*

$$T_{v,i,j} = N_i(N(v) \cap I_j)$$

has size at least $\Omega\left((d_{min})^2/(s \log^7 n)\right)$ and the property that $|T_{v,i,j} \cap R| \geq \lambda(1 - 5\delta)|T_{v,i,j}|$.

We now present a new method to ensure that no two vertices share too many neighbors.

Theorem 18 *Given an n-vertex k-colorable graph G containing two vertices that share at least $n^{\frac{1-\alpha(k)}{1-\alpha(k-2)}}$ neighbors and an algorithm $\mathcal{A}$ to color any m-vertex $(k-2)$-colorable graph with $f_{k-2}(m)$ colors, Algorithm* Sharing-Progress *below will make progress towards an $f_k(n)$-coloring of G.*

Algorithm Sharing-Progress**:**

Given: *(1) An n-vertex k-colorable graph G containing two vertices that share at least $n^{\frac{1-\alpha(k)}{1-\alpha(k-2)}}$ neighbors, and (2) an algorithm $\mathcal{A}$ to color any m-vertex $(k-2)$-colorable graph with $f_{k-2}(m)$ colors.*

Output: *Progress towards an $f_k(n)$ coloring of G.*

1. *Let $S = N(x) \cap N(y)$ where x and y share at least $n^{\frac{1-\alpha(k)}{1-\alpha(k-2)}}$ neighbors, and let G_S be the subgraph induced by set S.*
2. *Run algorithm $\mathcal{A}$ on G_S. Note that if G_S is $(k-2)$-colorable, then $\mathcal{A}$ will color G_S with at most:*

$$\begin{aligned} f_{k-2}(|S|) &= O(|S|^{\alpha(k-2)}(\log|S|)^{\beta(k-2)}) \\ &\leq O(|S|^{\alpha(k-2)}(\log n)^{\beta(k)}) \quad \textit{colors,} \end{aligned}$$

(using $|S| \leq n$ and β non-decreasing). Thus, Algorithm $\mathcal{A}$ will find an independent set of size at least:

$$\begin{aligned} \Omega\Big(\frac{|S|^{1-\alpha(k-2)}}{(\log n)^{\beta(k)}}\Big) &= \Omega\Big(\frac{n^{1-\alpha(k)}}{(\log n)^{\beta(k)}}\Big) \qquad \textit{(for the given choice of } |S|\textit{)} \\ &= \Omega(n/f_k(n)). \end{aligned}$$

Thus, if G_S is $(k-2)$-colorable, then we have made progress of Type 1 [Large-IS].

3. *If we did not make progress in Step 2, it must be that G_S was not $(k-2)$-colorable. The only way this could be is if x and y* must *be the same color under any legal k-coloring of G. So, we can merge vertices x and y and make progress of Type 3* [Same-Color].

The argument given in Algorithm Sharing-Progress proves Theorem 18. ■

We now use Algorithm Sharing-Progress in a procedure that allows us to "bootstrap" applications of Step 3 of Algorithm First-Approx.

Algorithm Bootstrap**:**

Given: *(1) Values $\alpha \in [0,1]$, $\beta > 0$ and $\delta = \frac{1}{\Theta(\log n)}$, and (2) An m-vertex subgraph H ($m \gg 1/\delta^2$) of an n-vertex graph G such that H contains an independent set R with $|R| \geq \lambda|V(H)|$ for some constant $\lambda > 0$.*

Output: *Either: (1) progress towards an $O(n^\alpha \log^\beta n)$-coloring of G, or else (2) at most $m/2$ subgraphs $G_0, G_1, \ldots, G_{m/2-1}$ of H such that with high probability at least one G_i has both a minimum degree of $(\delta^2 \frac{m}{n})n^\alpha \log^\beta n$ and considering only edges within G_i, $D(R \cap V(G_i)) \geq (\lambda - 2\delta)D(V(G_i))$.*

1. *Let $G_0 = (V_0, E_0) = H$. Inductively create graph $G_i = (V_i, E_i)$ from graph G_{i-1} for $i = 1, 2, \ldots, m/2 - 1$ by selecting an edge at random in E_{i-1} and deleting both endpoints. So, $|V_i| = |V_{i-1} - 2|$.*

2. *For each G_i with at least δm vertices, while G_i contains a vertex with degree less than $\delta^2 mn^{\alpha-1}\log^\beta n$: delete from G_i the vertex of minimum degree and all incident edges.*

   Suppose we have removed more than $\delta^2 m$ vertices from any G_i. Since within the set W_i of vertices deleted from G_i, the degree of each vertex can be at most $\delta^2 mn^{\alpha-1}\log^\beta n$, we can greedily find an independent set inside W_i of size at least:

$$\frac{\delta^2 m}{\delta^2 mn^{\alpha-1}\log^\beta n} \quad = \quad n/(n^\alpha \log^\beta n).$$

   So, we make progress Type 1 [Large-IS] *towards an $O(n^\alpha \log^\beta n)$-coloring of G.*

3. *If we did not make progress in Step 2, then output the graphs G_i for $i = 0, 1, \ldots, m/2-1$.*

Theorem 19 [Algorithm Bootstrap works as guaranteed] *Given an m-vertex subgraph H ($m \gg 1/\delta^2$) of an n-vertex graph G such that H contains an independent set R with $|R| \geq \lambda|V(H)|$ for some constant λ. Then, either (1) Algorithm* Bootstrap *makes progress towards an $O(n^\alpha \log^\beta n)$-coloring of G in Step 2, or else (2) with high probability, one of the subgraphs $G_i = (V_i, E_i)$ has both a minimum degree of $\delta^2 mn^{\alpha-1}\log^\beta n$ and within the subgraph, $D(R \cap V_i) \geq (\lambda - 2\delta)D(V_i)$.*

Proof: Let us consider the graphs G_i created after Step 1 of Algorithm Bootstrap, but *before deleting vertices in Step 2.* Let $R_i = V_i \cap R$ and let $N = m(1-\delta)/2$; note that set V_N contains δm vertices. We show now that with high probability, for some index $i \leq N$, we have $D(R_i) \geq (\lambda - \delta)D(V_i)$. The idea of the argument is that since we are removing vertices with a probability proportional to their degree, if $D(R_i) < (\lambda - \delta)D(V_i)$ for all such i, then we would remove many fewer vertices from R than from $V - R$. In fact, with high probability we would remove so many fewer that once we reach graph G_N, the set R_N would be larger than than V_N, a clear contradiction.

For each $i \leq N$, let A_i be the event that in creating G_{i+1} from G_i, we delete an edge with an endpoint in R_i. Since the number of edges in E_i with an endpoint in R_i is exactly $D(R_i)$ (because R_i is an independent set), we have:

$$\begin{aligned} \mathbf{Pr}[A_i] &= D(R_i)/|E_i| \\ &= 2D(R_i)/D(V_i). \end{aligned} \tag{41}$$

Suppose for some index $i \leq N$ we have $D(R_i) \leq (\lambda - \delta)D(V_i)$. Then, the probability event A_i occurs is at most $2(\lambda - \delta)$.

Let $p = 2(\lambda - \delta)$ and assume for contradiction that $D(R_i) < (\lambda - \delta)D(V_i)$ for *every* $i \leq N$. So, for each $i \leq N$, the probability that the ith edge removed from G has an endpoint in R is less than p. Since we remove N edges to create G_N and each time we remove an edge the probability it has an endpoint in R is less than p, by Chernoff bounds [AV79] the probability we remove more than $pN(1+\delta)$ vertices from R is at most $e^{-\delta^2\Omega(pN)}$. Since $pN = \Omega(m)$ and we assume $m \gg 1/\delta^2$ in the statement of the theorem, the probability we remove more than $pN(1+\delta)$ vertices from R is $o(1)$. Thus, with high probability:

$$\begin{aligned} |R_N| &\geq \lambda m - pN(1+\delta) \\ &= \lambda m - 2(\lambda-\delta)[m(1-\delta)/2](1+\delta) \\ &= m[\lambda - (\lambda-\delta)(1-\delta^2)] \\ &= \delta m + m\delta^2(\lambda - \delta) \\ &> \delta m. \qquad\qquad (\text{since } \lambda > \delta) \end{aligned}$$

So, with high probability, $|R_N| > |V_N|$, a contradiction. Thus, with high probability our assumption that $D(R_i) < (\lambda - \delta)D(V_i)$ for every $i \leq N$ is incorrect; that is, for some V_i of size at least δm, we have $D(R_i) \geq (\lambda - \delta)D(V_i)$.

Now, let i be such that $|V_i| > \delta m$ and $D(R_i) \geq (\lambda - \delta)D(V_i)$ before Step 2 of Algorithm Bootstrap. In Step 2, if at most $\delta^2 m$ vertices are removed, then we remove at most a fraction δ of the vertices of V_i in order to establish the desired minimum degree. Since we are always removing the vertex of least degree, we remove at most $\delta D(V_i)$ from the total degree sum of the subgraph. Even if, at worst, all the vertices removed were from the set R_i, we still have in the graph remaining that:

$$D(R_i) \geq (\lambda - 2\delta)D(V_i),$$

as claimed. ∎

Given Theorem 19, we have an improved approximation algorithm for coloring graphs of chromatic number $k > 3$ as follows. We first apply algorithm Sharing-Progress; we then run the distance-2 neighbor-taking stage of Algorithm First-Approx $k-2$ times, using Algorithm Bootstrap to "clean up" the graph in between applications; and finally, we use the BE/MS vertex-cover algorithm. The formal algorithm to color any k-colorable graph with $O(n^{\alpha(k)} \log^{\beta(k)} n)$ colors is given below. For simplicity, we have separated out the distance-2-neighbor/bootstrap step into a separate procedure.

Algorithm Multi-Stage-Color**:**

Given: *An n-vertex k-colorable graph G.*

Output: *Progress towards an $O(n^\alpha \log^\beta n)$-coloring of G for $\alpha = \alpha(k)$ as defined by the recursion in equation (40), and β at most 5.5.*

Let $f(n) = n^\alpha \log^\beta n$.

1. *[Base case] If $k = 2$ then just color G with 2 colors. If $k = 3$, then run Algorithm* Improved-Approx *on G.*
2. *[Minimum degree] For each vertex v, if $d(v) < f(n)$, make progress Type 2.*
3. *[Minimum sharing of neighbors] For each pair of vertices u, v, if $|N(u) \cap N(v)| \geq n^{\frac{1-\alpha(k)}{1-\alpha(k-2)}}$, then make progress using Algorithm* Sharing-Progress*. Note that Algorithm* Sharing-Progress *will use Algorithm* Multi-Stage-Color *recursively on $(k-2)$-colorable graphs.*
4. *[Initial distance-2 neighbors] For each vertex v and each pair $i, j \in [0, \ldots, \log_{1+\delta} n]$, let $G_{v,i,j}$ be the subgraph induced by the set $N_i(N(v) \cap I_j)$.*
5. *[Additional neighbor-taking stages] For each graph $G_{v,i,j}$, run Procedure* Iterate-neighbors *below on input $(n, k, G_{v,i,j}, k-3)$.*
   If the algorithm makes progress on any of the inputs given, then halt with success. Otherwise, let $G_1, \ldots, G_q$ be all the graphs returned by Iterate-neighbors*, for $q = O([(\log_{1+\delta} n)^{2k-4} n^{2k-5})$.*
6. *[Vertex-Cover approximation] Run the* BE/MS *vertex-cover algorithm on the graphs $G_1, \ldots, G_q$.*

Procedure Iterate-neighbors**:** $(n, k, G', iter)$

Given: *Values n and k. An m-vertex subgraph G' of some n-vertex graph G, and a number of iterations iter.*

Output: *$O([m^2(\log_{1+\delta} m)^2]^{iter})$ subgraphs of G' or else progress towards an $O(n^{\alpha(k)} \log^{\beta(k)} n)$-coloring of G.*

P1. *If $iter = 0$, then return G'.*

P2. *If $iter \geq 1$, then run Algorithm* Bootstrap *on G' and values $\alpha = \alpha(k), \beta = \beta(k)$, and $\delta = \delta(k)$.*

P3. *If Algorithm* Bootstrap *returns progress towards an $O(n^{\alpha(k)} \log^{\beta(k)} n)$-coloring of G, then halt with success. Otherwise, let $H_0, \ldots, H_{\frac{m}{2}-1}$ be the subgraphs returned.*

P4. *Now, for each H_l, $(0 \leq l \leq \frac{m}{2} - 1)$ for each vertex v in H_l and each index $i, j \in [0, \ldots, \log_{1+\delta} m]$:*
(note: there are at most $m^2(\log_{1+\delta} m)^2$ different 4-tuples (l, v, i, j))

(a) *Let $G_{l,v,i,j}$ be the subgraph of H_l induced by $N_i(N(v) \cap I_j)$, where neighborhoods are taken within H_l.*

(b) *Run:* Iterate-neighbors$(n, k, G_{l,v,i,j}, iter - 1)$.

Theorem 20 *Algorithm* Multi-Stage-Color*, given any n-vertex k-colorable graph, makes progress towards a coloring with $O(n^{\alpha(k)}(\log n)^{5.5})$ colors, for $\alpha(k)$ as defined in equation (40).*

Before proving Theorem 20, let us examine the claimed performance more closely. Let $\gamma(k) = \frac{1}{1-\alpha(k)}$. So, equation (40) can be written as:

$$\gamma(k) \quad = \quad 2 - \frac{1}{2^{k-2}} + \gamma(k-2)\left(1 - \frac{1}{2^{k-2}}\right). \tag{42}$$

One can see from this equation immediately that $\gamma(k) < 2 + \gamma(k-2)$; that is, if we increase k by 2, then γ increases by less than 2. We can compare this with the simpler approach from Section 6.1. Algorithm Recursive-Color given there colors k-colorable graphs with $\tilde{O}(n^{\alpha'(k)})$ colors for $\alpha'(k) = 1 - \frac{1}{k-r}$ for some constant r. Thus, the quantity $\gamma'(k) = \frac{1}{1-\alpha'(k)}$ equals $k - r$ and $\gamma'(k) = 2 + \gamma'(k-2)$. Since the function $g(x) = \frac{1}{1-x}$ is an increasing function with x, for algorithm Multi-Stage-Color the exponent α does not rise as rapidly as in algorithm Recursive-Color. Thus, the new approach yields better bounds. Because Algorithm Multi-Stage-Color is slower than algorithm Recursive-Color, one can achieve time/performance tradeoffs by running the faster algorithm with the slower algorithm as a base case for some $k = k_0$. Table 1 at the beginning of this section shows the results for both algorithms and for various combinations. In particular, for example, we can substitute the bound of Theorem 20 for $k = 4$ into the bound of Theorem 15 to get the following corollary.

Corollary 21 *Algorithm* Recursive-Color *using algorithm* Multi-Stage-Color *as a base case for $k = 4$, colors any n-vertex k-colorable graph $(k \geq 4)$ with at most:*

$$O\left(n^{1-\frac{1}{k-3/2}} (\log n)^{\frac{55}{4}(\frac{1}{k-3/2})}\right)$$

colors.

Proof of Theorem 20:

We may assume $k > 3$ since otherwise, we just run Algorithm Improved-Approx in Step 1 of Multi-Stage-Color. Define $s_k(n) = n^{\frac{1-\alpha(k)}{1-\alpha(k-2)}}$, and let $\alpha = \alpha(k)$ and $\beta = \beta(k)$. Steps 2 and 3

of Algorithm Multi-Stage-Color establish that the graph has a minimum degree of $n^\alpha \log^\beta n$ and that no two vertices share more than $s_k(n)$ neighbors.

Since G is k-colorable, it must contain an independent set R with $D_R(V-R) \geq \frac{1}{k-1} D(V-R)$. So, by Corollary 17, one of the graphs $G' = G_{v,i,j}$ created in Step 4 will both have size at least:

$$\begin{aligned} m_1 &= (d_{min})^2/(s_k(n)\log^7 n) \\ &= n^{2\alpha}\log^{2\beta} n/(s_k(n)\log^7 n), \end{aligned} \tag{43}$$

and contain an independent set of at least a $\lambda_1 = \frac{1}{k-1}(1-5\delta) \geq (\frac{1}{k-1} - 5\delta)$ fraction of its vertices.[7]

We now examine the call to procedure Iterate-neighbors. Suppose Iterate-neighbors is called with a graph G' of at least m_i vertices containing an independent set of at least a λ_i fraction of its nodes. By Theorem 19, if Step P3 does not halt with success immediately, then one of the graphs H_l produced will have both a minimum degree of $\delta m_i n^{\alpha-1}\log^\beta n$ and contain an independent set R' with $D(R') \geq (\lambda_i - 2\delta)D(V(H_l))$. Rewriting the latter inequality, we have $D(R') \geq (\lambda_i - 2\delta)[D(V(H_l) - R') + D(R')]$, so:

$$D_{R'}(V(H_l) - R') = D(R') \quad \geq \quad \tfrac{\lambda_i - 2\delta}{1-\lambda_i+2\delta} D(V(H_l) - R').$$

Using the minimum degree bound and degree ratios above, Corollary 17 implies that one of the sets $G_{l,v,i,j}$ produced in Step P4(a) will both have size at least m_{i+1} and an independent set of at least a fraction λ_{i+1} of its vertices, where:

$$\begin{aligned} m_{i+1} &= \delta^4 m_i^2 n^{2\alpha-2}(\log n)^{2\beta}/(s_k(n)\log^7 n) \\ &= \Omega(m_i^2 n^{2\alpha-2}(\log n)^{2\beta}/(s_k(n)\log^{11} n)) \\ &= \Omega(m_i^2 n^{2\alpha-2}/s_k(n)), \qquad\qquad (\text{for } \beta = 5.5) \end{aligned} \tag{44}$$

$$\begin{aligned} \text{and} \qquad \lambda_{i+1} &\geq \tfrac{\lambda_i - 2\delta}{1-\lambda_i+2\delta} - 5\delta \\ &> \tfrac{\lambda_i - 4\delta}{1-\lambda_i} - 5\delta \\ &\geq \tfrac{\lambda_i}{1-\lambda_i} - 13\delta \qquad \text{for } \lambda_i \leq 1/2. \end{aligned} \tag{45}$$

Thus, one of the graphs G_l returned to Step 5 of Algorithm Multi-Stage-Color will have at least m_{k-2} vertices and contain an independent set of size at least $\lambda_{k-2}|V(G_l)|$, where we must now solve for m_{k-2} and λ_{k-2}.

Claim 1: $\lambda_i \geq \frac{1}{k-i} - 4^{i+2}\delta$ for $0 \leq i \leq k-2$.

Proof: For $i = 1$ the claim holds. For $i > 1$, by induction and using equation (45), we have:

$$\begin{aligned} \lambda_i &\geq (\tfrac{1}{k-i+1} - 4^{i+1}\delta)/(\tfrac{k-i}{k-i+1} + 4^{i+1}\delta) - 13\delta \\ &> (\tfrac{1}{k-i+1} - 2\cdot 4^{i+1}\delta)/(\tfrac{k-i}{k-i+1}) - 13\delta \\ &\geq \tfrac{1}{k-i} - 2\cdot 4^{i+1}\delta(\tfrac{k-i+1}{k-i}) - 13\delta \\ &\geq \tfrac{1}{k-i} - 3\cdot 4^{i+1}\delta - 13\delta \qquad (\text{for } i \leq k-2) \\ &\geq \tfrac{1}{k-i} - 4^{i+2}\delta. \qquad (\text{for } i+1 \geq 2) \qquad \square \end{aligned}$$

So, for $\delta = \delta(k) = \frac{1}{4^k \log n}$, we have:

$$\lambda_{k-2} \quad \geq \quad (\tfrac{1}{2} - \tfrac{1}{\log n}). \tag{46}$$

[7] One can verify that the minimum degrees and the values m_i defined satisfy the technical conditions of Corollary 17 (min degree $> \max(s(1+\delta), (3\log n)/\delta^2)$) and Theorem 19 ($m_i \gg 1/\delta^2$).

Claim 2: $m_i = \Omega(n^{(2^{i+1}-2)\alpha} \cdot n^{2-2^i} \cdot [s_k(n)]^{1-2^i})$.
Proof: One can easily check that the claim holds for the base case of $i = 1$, using equation (43) and the fact that for $\beta = 5.5$ that $\log^{2\beta} n > \log^7 n$. For $i > 1$, we can check inductively that (44) satisfies the claim as follows:

$$\begin{aligned} m_{i+1} &= \Omega(m_i^2 n^{2\alpha-2}/[s_k(n)]) \\ &= \Omega\Big(n^{(2^{i+1}-2)2\alpha} \cdot n^{2(2-2^i)} \cdot [s_k(n)]^{2(1-2^i)} \cdot n^{2\alpha-2}/[s_k(n)]\Big) \\ &= \Omega\Big(n^{(2^{i+2}-4+2)\alpha} \cdot n^{4-2^{i+1}-2} \cdot [s_k(n)]^{2-2^{i+1}-1}\Big) \\ &= \Omega\Big(n^{(2^{i+2}-2)\alpha} \cdot n^{2-2^{i+1}} \cdot [s_k(n)]^{1-2^{i+1}}\Big) \qquad \square \end{aligned}$$

So,

$$m_{k-2} = \Omega(n^{(2^{k-1}-2)\alpha} \cdot n^{2-2^{k-2}} \cdot [s_k(n)]^{1-2^{k-2}}). \tag{47}$$

Thus, one of the graphs of Step 5 of Algorithm Multi-Stage-Color will have an independent set of at least $(\frac{1}{2} - \frac{1}{\log n})$ of its vertices (from equation (46)) and have size at least m_{k-2}, as given in equation (47). By lemma 11, Step 6 will find an independent set of size at least $m_{k-2}/\log n$.

Thus, to prove Theorem 20 we must just show that $m_{k-2}/\log n = \Omega(n/(n^{\alpha(k)} \log^{\beta(k)} n))$. Since $\beta(k)$ is set to 5.5 it is enough to have $m_{k-2} = \Omega(n^{1-\alpha(k)})$. Equivalently, using equation (47), taking $\log_n$ of both sides, and substituting in $s_k(n) = n^{\frac{1-\alpha(k)}{1-\alpha(k-2)}}$, we just need to show that:

$$1 - \alpha(k) \leq \alpha(k)\Big[2^{k-1} - 2\Big] + \Big[2 - 2^{k-2}\Big] + \Big[\frac{1-\alpha(k)}{1-\alpha(k-2)}\Big]\Big(1 - 2^{k-2}\Big).$$

Rearranging terms, this formula is equivalent to:

$$[1 - \alpha(k)](2^{k-1} - 1) - 2^{k-2} \leq \Big[\frac{1-\alpha(k)}{1-\alpha(k-2)}\Big]\Big(1 - 2^{k-2}\Big),$$

or:

$$2^{k-1} - 1 - \frac{2^{k-2}}{1-\alpha(k)} \leq \Big[\frac{1}{1-\alpha(k-2)}\Big]\Big(1 - 2^{k-2}\Big).$$

Dividing both sides by -2^{k-2} and rearranging one final time, we find that we just need:

$$\frac{1}{1-\alpha(k)} \geq 2 - \frac{1}{2^{k-2}} - \frac{1}{1-\alpha(k-2)}\Big(\frac{1}{2^{k-2}} - 1\Big).$$

But, this formula is exactly the definition of $\alpha(k)$ given in equation (40). So Algorithm Multi-Stage-Color works as claimed. ■

7 Possibilities for improvement

Algorithm First-Approx performs most poorly when (1) many vertices share about $n^{0.2}$ neighbors in common, and (2) the average vertex degree is about $n^{0.4}$. If the edges in the graph were distributed randomly, this combination of events would likely not occur; instead, the graph must contain high density regions. For example, a graph could have properties (1) and (2) above if it consists of a collection of "clusters" of size $\Theta(n^{0.6})$ such that each vertex inside a given cluster

has $\Theta(n^{0.4})$ neighbors within the cluster and $\Theta(n^{0.4})$ neighbors distributed throughout the other clusters. Thus, if the edges within a cluster were distributed randomly, then 2 vertices inside the same cluster share on average $\Theta((n^{0.4})^2/n^{0.6}) = \Theta(n^{0.2})$ neighbors in common, even though the degrees are low. (The purpose of giving to each vertex $\Theta(n^{0.4})$ neighbors in the other clusters is so that the distance-2 neighbor set $N(N(v))$ for each vertex v may have size $\Omega(n^{0.8})$ to avoid immediately making progress through Corollary 2.)

Algorithm Improved-Approx achieves better performance by taking advantage of such high density regions when they are found. However, one other possible approach is the following. Suppose by removing 9/10 of the edges in the graph, one could somehow get rid of such high-density regions and prove a stronger analog of Theorem 3 (bounding the number of shared neighbors of two vertices). Then, Theorems 7 and 8 would still apply to show that some set $T = N_i(N(v) \cap I_j)$ in the new graph is both large and has a large fraction of its vertices red. The point here is that even though an independent set in the new graph might not be an independent set in the original graph, there still must be *some* color class in a 3-coloring of the original graph that satisfies the $\lambda = 1/2$ condition (see Theorem 7) in the new graph. Also, the average degree has only changed by a constant factor, so the set T produced will still be large. (The minimum degree can be raised easily to a constant fraction of the average in order to apply Corollary 9.)

A different way one might be able to do significantly better is to consider distance-3 neighborhoods of vertices (or perhaps even distance-t neighborhoods for larger t). However, all the techniques given here for forcing expansion — that is, for forcing the set found to be large — seem to break down completely in this case.

8 Open problems and conclusion

We have described here an algorithm guaranteed to color any 3-chromatic graph with $\tilde{O}(n^{3/8})$ colors in the worst case, and shown how these techniques can be used to improve previous bounds for coloring k-chromatic graphs for $k > 3$ as well. Clearly, however, there remains a long way to go. There is no reason to believe an $\tilde{O}(n^{3/8})$ bound is intrinsic to the coloring problem. In fact, for coloring 3-colorable graphs, to date there is no lower bound known greater than 3. That is, it remains unknown whether there is any intrinsic reason why one could not 4-color any given 3-colorable graph in polynomial time. It would be a very significant contribution to this area if one could make headway in this direction. Some such headway has been recently made by Lund and Yannakakis [LY92], who provide a collection of exciting new lower bounds, though the question about 4-coloring 3-colorable graphs remains open.

Acknowledgments: I would like to thank Bonnie Berger, Ron Rivest, John Rompel, David Shmoys, and Cliff Stein for many helpful discussions.

References

[ALM+92] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 14–23, Pitsburgh, October 1992.

[AV79] D. Angluin and L. G. Valiant. Fast probabilistic algorithms for Hamiltonian circuits and matchings. *Journal of Computer and System Sciences*, 18(2):155–193, April 1979.

[BCKT89] P. Briggs, K. D. Cooper, K. Kennedy, and L. Torczon. Coloring heuristics for register allocation. In *Proceedings of the SIGPLAN '89 Conference on Programming Language Design and Implementation*, pages 275–284, Portland, June 1989.

[Ber73] C. Berge. *Graphs and Hypergraphs.* North-Holland, 1973.

[BH90] R. B. Boppana and M. M. Halldórsson. Approximating maximum independent sets by excluding subgraphs. In *Proc. of 2nd Scand. Workshop on Algorithm Theory. Springer-Verlag Lecture Notes in Computer Science #447*, pages 13–25, July 1990.

[Blu89] A. Blum. An $\tilde{O}(n^{0.4})$-approximation algorithm for 3-coloring (and improved approximation algorithms for k-coloring). In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, pages 535–542, Seattle, May 1989.

[Blu90] A. Blum. Some tools for approximate 3-coloring. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, pages 554–562, St. Louis, October 1990.

[Blu91] A. Blum. *Algorithms for Approximate Graph Coloring.* PhD thesis, Massachusetts Institute of Technology, May 1991. (MIT Laboratory for Computer Science Technical Report MIT/LCS/TR-506, June 1991).

[BR88] B. Berger and J. Rompel. A better performance guarantee for approximate graph coloring. *Algorithmica*, 1988.

[BYE85] R. Bar-Yehuda and S. Even. A local-ratio theorem for approximating the weighted vertex cover problem. *Annals of Discrete Mathematics*, 25:27–46, 1985.

[CAC+81] G. J. Chaitin, M. A. Auslander, A. K. Chandra, J. Cocke, M. E. Hopkins, and P. W. Markstein. Register allocation via coloring. *Computer Languages*, 6:47–57, 1981.

[Cha82] G. J. Chaitin. Register allocation and spilling via graph coloring. In *Proceedings of the SIGPLAN '82 Symposium on Compiler Construction*, pages 98–101, Boston, June 1982.

[DF89] M. E. Dyer and A. M. Frieze. The solution of some random NP-Hard problems in polynomial expected time. *Journal of Algorithms*, 10:451–489, 1989.

[Hal90] M. M. Halldórsson. A still better performance guarantee for approximate graph coloring. Technical Report 90-44, DIMACS, June 1990. Also to appear in IPL.

[Kuc77] L. Kucera. Expected behavior of graph colouring algorithms. In *Lecture Notes in Computer Science No. 56*, pages 447–451. Springer-Verlag, 1977.

[LSW] N. Linial, M. Saks, and A. Wigderson. personal communication.

[LY92] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. Unpublished manuscript, 1992.

[MS85] B. Monien and E. Speckenmeyer. Ramsey numbers and an approximation algorithm for the vertex cover problem. *Acta Informatica*, 22:115–123, 1985.

[NW90] R. Nelson and R. J. Wilson, editors. *Graph Colourings.* Longman Scientific and Technical, 1990.

[Rag] P. Raghavan. personal communication.

[Tur88] J. S. Turner. Almost all k-colorable graphs are easy to color. *Journal of Algorithms*, 9:63–82, 1988.

[Vis90] S. Vishwanathan. Randomized online graph coloring (preliminary version). In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, volume II, pages 464–469, St. Louis, October 1990.

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

A Appendix: the Vertex-Cover / Independent-Set approximation algorithm

For completeness, we include here a simplified version of the Vertex-Cover approximation algorithm of Bar-Yehuda and Even [BYE85] and Monien and Speckenmeyer [MS85], specialized to its use in this paper. This version is taken from a treatment given by Boppana and Halldórsson [BH90]. We will describe the algorithm as an Independent Set approximation algorithm for the special case where the input n-vertex graph contains an independent set of at least $\frac{1}{2}(1 - \frac{1}{\log n})$ of its vertices. The output of the procedure is an independent set of size $\Omega(n/\log n)$.

Algorithm Approx-**IS** *[Simplified version of the* BE/MS *algorithm]*

Given: *An n-vertex graph G which has an independent set of size at least $\frac{1}{2}(1 - \frac{1}{\log n})n$.*

Output: *An independent set of size at least $\Omega(n/\log n)$.*

1. *Remove all odd cycles of length $\leq 2l + 1$ for $l = \frac{\log n}{6} - \frac{1}{2}$. See Note 1 below. (Assume for simplicity that $\frac{\log n}{6} - \frac{1}{2}$ is an integer.)*
2. *Initialize I, the independent set found, to ϕ.*
3. *Choose $v \in V$.*
4. *For $i \in \{0, \ldots, l\}$, let V_i = the set of vertices of distance i from v.*
5. *For $i \in \{0, \ldots, l\}$, let $S_i = V_i \cup V_{i-2} \cup V_{i-4} \cup \ldots$.*
   Note that S_i is an independent set since there are no odd cycles of length $\leq 2l + 1$.
   Also, note that $N(S_i) = S_{i+1}$.
6. *Let $i_0 \leq l$ be an index such that $|N(S_{i_0})| \leq n^{1/(l+1)}|S_{i_0}|$.*
   This property must hold for some $i_0 \in \{0, \ldots, l\}$ because otherwise:
   $|N(S_l)| > n^{1/(l+1)}|S_l| > n^{2/(l+1)}|S_{l-1}| > n^{3/(l+1)}|S_{l-2}| > \ldots > n^{(l+1)/(l+1)}|S_0| = n$,
   a contradiction.
7. *Let $I \leftarrow I \cup S_{i_0}$ and let $V \leftarrow V - S_{i_0} - N(S_{i_0})$.*
   If V is non-empty, then go back to Step 3. Otherwise output set I.
   See note 2 below.

Note 1: Step 1 removes all odd cycles of length $\leq 2l + 1$. An odd cycle of length $2i + 1$ may have at most i vertices in any independent set in G. So, if m vertices remain after this step (so $n - m$ are removed), we have removed at most $\frac{l}{2l+1}(n - m)$ vertices from any independent set

in G. Thus, the maximum independent set in G may have size at most $m + \frac{l}{2l+1}(n-m)$. This implies that the number of vertices m remaining is at least $n/\log n$ since otherwise,

$$\begin{aligned} m + (n-m)\tfrac{l}{2l+1} &\leq m + (n-m)(\tfrac{\log n}{6} - \tfrac{1}{2})/(\tfrac{\log n}{3}) \\ &= m + (n-m)\tfrac{\log n - 3}{2\log n} \\ &\leq \tfrac{n}{\log n} + (n - \tfrac{n}{\log n})(\tfrac{1}{2} - \tfrac{3}{2\log n}) \\ &= \tfrac{n}{2} - \tfrac{n}{\log n} + \tfrac{3n}{2\log^2 n} \\ &< \tfrac{1}{2}(1 - \tfrac{1}{\log n})n. \qquad \text{(for } n \text{ sufficiently large)} \end{aligned}$$

This contradicts our assumption on the largest independent set in G.

Note 2: By Note 1, after Step 1 we know graph G has at least $n/\log n$ vertices. Each application of Step 6 removes from V at most $O(n^{1/(l+1)})$ times as many vertices as added to I. So, the final set I reported in Step 7 must be large enough so that $|I|n^{1/(l+1)} = \Omega(n/\log n)$. That is, it must be the case that:

$$|I| \;=\; \Omega(\tfrac{n}{\log n} n^{-1/(l+1)}) \;=\; \Omega(\tfrac{1}{\log n} n^{l/(l+1)}).$$

For $l = \frac{\log n}{6} - \frac{1}{2}$, we have:

$$\tfrac{l}{l+1} \;=\; (\tfrac{\log n}{6} - \tfrac{1}{2})/(\tfrac{\log n}{6} + \tfrac{1}{2}) \;=\; \tfrac{\log n - 3}{\log n + 3} \;\geq\; \tfrac{\log n - 6}{\log n} \;=\; 1 - \tfrac{6}{\log n}.$$

So, finally, this implies that:

$$\begin{aligned} |I| &= \Omega(\tfrac{1}{\log n} n^{1 - \frac{6}{\log n}}) \\ &= \Omega(\tfrac{n}{\log n} \cdot 2^{-6}) \\ &= \Omega(n/\log n). \qquad \blacksquare \end{aligned}$$