MIT/LCS/TR-506

ALGORITHMS FOR APPROXIMATE GRAPH COLORING

Avrim Blum

June 1991

This blank page was inserted to preserve pagination.

Algorithms for Approximate Graph Coloring

by

Avrim Louis Blum

S.B., Mathematics with Computer Science
S.B., Physics
Massachusetts Institute of Technology
(1987)
S.M., Electrical Engineering and Computer Science
Massachusetts Institute of Technology
(1989)

Submitted to the
Department of Electrical Engineering and Computer Science
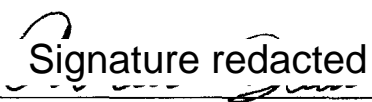in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
May 1991

Signature of Author _____ Signature redacted _____
Department of Electrical Engineering and Computer Science
May 9, 1991

Certified by _____ Signature redacted _____
Ronald L. Rivest
Professor of Computer Science and Engineering
Thesis Supervisor

Accepted by _____
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Algorithms for Approximate Graph Coloring

by

Avrim Louis Blum

Submitted to the Department of Electrical Engineering and Computer Science
on May 9, 1991, in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Abstract

A coloring of a graph is an assignment of colors to the vertices so that no two adjacent vertices are given the same color. 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 thesis explores the approximation problem of coloring k-colorable graphs with as few additional colors as possible in polynomial time, focusing on the case of $k = 3$.

For the worst-case problem, the previous best upper bound on the number of colors needed for coloring 3-colorable n-vertex graphs in polynomial time is $O(\sqrt{n}/\sqrt{\log n})$ colors by Berger and Rompel, improving a bound of $O(\sqrt{n})$ colors by Wigderson. We present an algorithm to color any 3-colorable graph with $O(n^{3/8} \text{polylog}(n))$ colors, breaking an "$O(n^{1/2-o(1)})$ barrier". The algorithm presented 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.

We also examine the problem of coloring *random* k-colorable graphs. We consider a standard model in which vertices are first randomly assigned to one of k color classes and then each edge between two vertices of different color is placed into the graph with probability p. For sufficiently high edge probability, it is known by results of Turner, Dyer and Frieze, and others, that such graphs are easy to k-color. We describe here an algorithm to k-color graphs generated in this way for a much wider range of edge probabilities ($p \geq n^{-1+\epsilon}$ for any constant $\epsilon > 0$) than previously possible.

To study a wider variety of graph distributions, we also present a model of graphs generated by the *semi-random* source of Santha and Vazirani that provides a smooth transition between the worst-case and random models. In this model, the graph is generated by a "noisy adversary" — an adversary whose decisions (whether or not to insert a particular edge) have some small (random) probability of being reversed. We show that even for quite low noise rates, semi-random k-colorable graphs can be colored with high probability using just k colors.

Finally, we use assumptions about the worst-case difficulty of approximate graph coloring to provide lower bounds for other hard problems. Using techniques developed by Berman and Schnitger, we show that if there is no polynomial-time algorithm to color k-colorable graphs with $O(\log n)$ colors, then the largest independent set in a graph (or equivalently the largest clique) cannot be approximated to within a factor of $n^{1-\epsilon}$ for any constant $\epsilon > 0$. This is a much higher lower-bound than achieved by previous results, albeit based on less solid assumptions.

Thesis Supervisor: Ronald L. Rivest

Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

Contents

4

Chapter 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 have a long history in mathematics and computer science. The famous 4-Color Problem of whether every planar graph is 4-colorable, dates back at least to 1852 [33]. Partly through that problem, finally solved by Appel and Haken [3], graph coloring has become a central topic in combinatorics. In computer science, graph coloring problems have long been known to model various scheduling problems such as examination scheduling and register allocation. 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 determining the chromatic number of a graph is NP-Complete. The problem of deciding whether a graph is k-colorable for any *fixed $k \geq 3$* is NP-Complete as well. Thus, coloring an arbitrary k-colorable graph with k colors for $k \geq 3$ cannot be done in polynomial time unless P = NP (for $k = 2$, 2-coloring is easy). Knowing that the coloring problem is NP-hard does not make it disappear, however, and it also does not necessarily mean nothing useful can be done. It does mean that as for many other famous hard problems such as the Traveling Salesman Problem (TSP) and the Bin Packing problem, researchers attempting to find good fast algorithms must consider issues of approximation.

This thesis concerns the algorithmic problem of finding good approximate colorings of graphs for several natural forms of approximation. We focus here on deriving polynomial-time algorithms for coloring graphs of constant chromatic number and on improving upon previously known algorithmic guarantees. In particular, we both improve upon previous guarantees for the number of colors needed in the worst case to properly color k-colorable graphs in polynomial-time, and extend the known classes of graphs for which *optimal* colorings can be found quickly. We will not be so concerned here with precisely optimizing the running time of the algorithms (so long as they are polynomial); instead we focus 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 thesis 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 as well.

1.1 Applications of graph coloring

Graph coloring problems arise in situations where one would like to assign a small number of values to objects under pairwise constraints of the form that object x and object y cannot receive the same value. Such situations occur often in various scheduling problems and we present a few examples here.

Example 1: Examination Scheduling.

Consider the problem of scheduling n final exams into a small number of different time slots. One would like to do so in a way such that no student has a conflicting schedule: that is, no student has two of her examinations at the same time. Suppose we assign one vertex in a graph to each examination and place an edge between two vertices if some student is taking both corresponding exams. Then the problem of scheduling the examinations into k time slots so no student has a conflict is exactly the problem of coloring the corresponding graph with k colors [42][5].

Example 2: Register Allocation.

A more "real computer science" problem, for which graph coloring techniques have actually been used in practice is the problem of register allocation in compilers. Work in this direction has been done by several researchers including Chaitin [15], Chaitin et al. [16], and Briggs et al. [13]. During compilation, a standard compiler [15] transforms the source program into an intermediate language based on a hypothetical machine with an unlimited number of fast syntactic (virtual) registers. Since the real machine has only a bounded number of registers, the compiler in a "register allocation phase" must then map the computed values in the syntactic registers into the true registers of the machine (e.g., 17 registers in work of Chaitin et al. [16] or 32 registers in work of Chaitin [15]). If the compiler cannot do this exactly, it will be forced to "spill" some values into main memory through load and store operations. Because the registers are fast, the hope is to spill as little of the computation as possible.

The relationship of this problem to graph coloring is as follows. For each procedure in a program, Chaitin et al. build a "register interference graph" containing one vertex for each value (e.g. a variable in the program) and an edge between two vertices if

the two values interfere and cannot be placed into the same register. Interference is checked roughly by examining if both values are "live" at the same time, or more precisely if one value is live at a definition point of the other. Thus, if we think of the real registers as colors, an assignment of 17 or 32 colors to the vertices of the interference graph (depending on which machine is being used) corresponds to an assignment of registers to the computed values of the procedure.

Of course, it may be that the interference graph cannot be colored with the required number of colors. In that case, the uncolored vertices are "spilled" into main memory. So, the goal here is to color as many vertices as possible with the given number of colors (where "many" may be defined by some additional measure of cost and not just sheer quantity). As it turns out, once values are spilled, this requires additional vertices, usually of low degree, to be added onto the graph, so the abstraction as a standard coloring problem is not quite exact. Nonetheless, simple graph coloring heuristics appear to work well in practice [15][16][13].

1.2 Forms of approximation, and past work

For the graph coloring problem, the issue of approximation splits naturally into two general directions. One direction is to consider worst-case graphs, but allow the number of colors used to be non-optimal. In particular, one would like answers to the question:

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

A second general direction is to relax the restriction that the graph be worst case and attempt to find optimal colorings for large or nicely characterized subsets of the inputs. That is, one would like answers to the question:

> While coloring k-colorable graphs with k colors in the *worst-case* is hard, can you find a large subset of cases where k-coloring is easy?

1.2.1 Approximate coloring in the worst case

For graphs of constant chromatic number, the first nontrivial result along the first direction presented here was due to Wigderson [43]. 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 [6], Linial, Saks, and Wigderson [24], and Raghavan [32] independently have improved upon this bound to color k-colorable graphs with $O\left((n/\log n)^{1-\frac{1}{k-1}}\right)$ 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. The difficulty in improving these results motivated work of Linial and Vazirani [25] who provide some evidence for an n^ϵ lower bound for the general chromatic number approximation problem.

For general graphs of arbitrary chromatic number, the best algorithmic result known to date is due to Halldórsson [22]. 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 [12] for the Independent Set problem which finds an independent set within an $n/(\log n)^2$ factor of the maximum.

There has also been recent work on coloring graphs presented in an *on-line* manner; that is, coloring graphs presented one vertex at a time in some arbitrary order. Vishwanathan [41] presents an algorithm for such a model that uses a number of colors within a logarithmic factor of the Wigderson bound.

1.2.2   Exact coloring in special cases

Many classical results on graph coloring can be thought of from the point of view of the second direction described here. These results prove nice characterizations that are sufficient conditions for k-colorability, and the characterizations are often testable in polynomial time. For example, the famous 4-Color Problem and Theorem gives an easy way to prove a graph to be 4-colorable — one simply checks that the graph is planar. In fact, the 4-Color Theorem of Appel and Haken is known to yield a polynomial-time coloring algorithm for such graphs. Of course, if the graph turns out not to be planar, then this technique says nothing about the graph's chromatic number. For graphs of chromatic number 3, similar classical results are known. Grötzsch ([5], p.355) proved that any planar graph without triangles must be 3-colorable, and this was extended to hold for graphs with at most 3 triangles by Grünbaum [21].[1] The proofs of both results involve reducing a graph to one with fewer vertices in ways that yield polynomial-time coloring algorithms. For graphs of general chromatic number, Brooks' Theorem [14][26] states that any connected graph of maximum degree d ($d > 2$) is either d-colorable or else is a single $(d+1)$-clique. Note that it is very easy to d-color any graph with maximum degree $d - 1$: for each vertex in an arbitrary order, simply give to that vertex any color in $\{1, \ldots, d\}$ not held at the time by any of its neighbors. Steinberg [37] presents a survey of such classical results, focusing on 3-chromatic graphs.

---

[1]In some sense this is "best possible" since the 4-clique K_4 is planar with four triangles and is not 3-colorable. See [5].

Instead of presenting such explicit characterizations of easy-to-color families of graphs, one can also study random families of graphs. Turner [38], Kucera [23], and Dyer and Frieze [18] give polynomial-time algorithms that color *random* k-colorable graphs with k colors with high probability, for any constant k. So, *most* k-colorable graphs are easy to k-color. In fact, Dyer and Frieze go further and provide an algorithm that when amortized over all n-vertex k-colorable graphs, spends on average polynomial time per graph. Petford and Welsh [31] present experimental work using heuristics for coloring random 3-colorable graphs and claim success for a wide range of edge probabilities.

It is not known how to color general random graphs (where we do not restrict the chromatic number) in polynomial-time with the minimum of colors, but one can get fairly close. For the model $\mathcal{G}(n,p)$ of an n-vertex graph in which each edge is included with probability p, Bollobás [10] has shown that the chromatic number will be $(1 + o(1))\, n/(2 \log_b n)$ with high probability, for $b = \frac{1}{1-p}$. It is not hard to show that the greedy algorithm: in some order give to each vertex the color of least index not yet held by any of its neighbors, finds a coloring of at most $(1 + o(1))n/\log_b n$ colors, a factor of 2 above optimal. Matula [27] provides quasi-polynomial approaches with provably better bounds.

1.3   New results and a plan of the thesis

This thesis presents results in both of the two directions discussed above. For k-colorable graphs for constant k, we both provide better approximation guarantees for the worst-case problem and expand the classes of graphs for which optimal coloring is known to be easy.

The major portion of this thesis concerns the the first direction discussed of finding improved approximation guarantees for the worst-case problem. We present an algorithm that uses a quite different strategy from that used by the algorithms of Wigderson and Berger and Rompel and others, 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 we present 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 (Chapter 4) colors 3-colorable graphs with $O(n^{2/5+o(1)})$ colors, and the second part (Chapter 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 Chapter 10. The algorithms presented for the worst-case problem 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 Chapter 3.

Along the second direction, we extend the class of *randomly* chosen k-colorable graphs for

which a k-coloring can be found in polynomial time. In particular, we consider a standard model of a random k-colorable graph in which vertices are first randomly assigned to one of k color classes and then each edge between two vertices of different color is placed into the graph with probability p. For this model, we are able to find colorings for a wider range of edge probabilities ($p \geq n^{-1+\epsilon}$ for any constant $\epsilon > 0$) than was previously known. These results are described in Chapter 7.

While the known results on random graphs imply that *most* k-colorable graphs are easy to k-color, random k-colorable graphs tend to be of a very special type. For example, with high probability all vertices of a random k-colorable graph have nearly the same degree and vertices of the same color class all have nearly the same number of common neighbors. So, graphs created in only a "somewhat random" manner may not be colored well by algorithms for the random case. To explore a wider variety of graph distributions, we present in Chapter 8 a model of graphs created by the *semi-random* source of Santha and Vazirani [34] that provides a smooth transition between the worst-case and random models. In this model, the graph is generated by a "noisy adversary" — an adversary whose decisions (whether or not to insert a particular edge) have some small probability of being reversed. We show that even for quite low noise rates, these semi-random k-colorable graphs can be colored with high probability using just k colors. The discussion of random and semi-random graph models is based in part on work joint with Joel Spencer.

In addition to the above-mentioned general directions, we describe in Chapter 9 how hardness assumptions for approximately coloring graphs in the worst case can be used to provide lower bounds for other hard problems. In particular, we use a technique developed by Berman and Schnitger [7] to prove the following result. Suppose there were a polynomial-time algorithm to find an independent set in a graph of size at most a factor of $n^{1-\epsilon}$ smaller than the size of the largest independent set, for some constant $\epsilon > 0$. Then one could convert such a procedure into one that colors k-colorable graphs with $O(\log n)$ colors, for any constant k. Also, one could convert such a procedure into one that colors $(\log n)$-colorable graphs with polylog(n)-colors. This contrasts with the best algorithm known to date [22] for coloring $(\log n)$-colorable graphs which uses more than $n/(\log n)^2$ colors. So, these results imply that a seemingly small improvement in approximating independent sets implies one can get a much larger improvement for approximate graph coloring. In contrapositive form, these results present a high lower-bound for Independent Set approximation based on a hardness assumption for graph coloring that is quite far from the best algorithmic guarantees currently known.

Some of the work in this thesis has previously appeared in extended abstract form [8][9].

Chapter 2

Notation, definitions, and previous algorithms

In this chapter we review some standard graph-theoretic definitions and introduce basic notation that will be used throughout this thesis. At the end of the chapter 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 $v \in N(N(v))$.

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 every edge in the graph has at least one endpoint in W; that is, it 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 [29], 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 *t-colors* a graph if it

colors the graph with at most t colors, and it *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, we use the following general standard notation:

- $(m)_i = m(m-1)(m-2)\cdots(m-i+1)$.

- K_t is the clique on t vertices.

- For S a subset of vertices of graph G, the graph $H = G|_S$ is the subgraph of G induced by set S. That is, $V(H) = S$ and $E(H) = \{(i,j) \in E(G) \mid i,j \in S\}$.

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

As is well known, 2-colorable graphs can easily be 2-colored in polynomial time. For example, the following procedure suffices to color any 2-colorable graph with the colors 0 and 1. First, assign a color, say 0, to one vertex in each connected component in the graph. Then assign color 1 to each neighbor of a vertex colored 0. Finally, repeat, assigning color 0 to any uncolored neighbor of a vertex of color 1, and color 1 to any uncolored neighbor of a vertex colored 0, and so on, until the entire graph is colored. The resulting coloring will be legal since 2-colorable graphs have no odd cycles.

Let us now review Wigderson's algorithm [43] 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 $\sqrt{8}$. A more formal description of the algorithm is given below.

Wigderson's Algorithm

Given $G = (V, E)$, a 3-colorable graph on n vertices.

1. *Initialize color c to 0.*

2. *While there exists a vertex $v \in V$ with $d(v) \geq \sqrt{n}$,*

   (a) 2-color $N(v)$ with colors: $c, c + 1$.

   (b) Let $c \leftarrow c + 2$, $V \leftarrow V - N(v)$.

   (note that the loop in this step can be executed at most $\sqrt{n}$ times.)

3. *Color the remaining graph with colors $c, c + 1, \ldots, c + \sqrt{n} - 1$, by arbitrarily assigning to each vertex a color not held by any of its neighbors.*

The improvement to $O(\sqrt{n}/\sqrt{\log n})$ of Berger et al. mentioned previously results from choosing $(\log n)$ starting vertices instead of one. This can be done by selecting an arbitrary subset of vertices of size $(3 \log n)$, and trying each subset of size $(\log n)$; one such subset must be monochromatic under some legal 3-coloring of G and so has a 2-colorable neighborhood. The way that this set is then exploited is described in [6]. We will revisit this algorithm in Chapter 3, where the algorithm and bounds guaranteed follow as an easy corollary of the machinery described there.

In contrast to the above strategies, the new worst-case 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 earlier, is 2-colorable; the structure of the distance-2 neighbors will have to be more carefully brought out.

Chapter 3

Worst-case bounds: preliminaries

3.1 New worst-case approach: the basic idea

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, and not just the immediate neighborhoods of vertices as in previous algorithms. 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 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**

---

[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.

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 [4] and (independently) to Monien and Speckenmeyer [28].[3] 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 the graph. If we apply the algorithm to the graph induced by S, we find a vertex cover W in S of size at most $\frac{1}{2}|S|\left(2 - \frac{\log\log n}{\log n}\right)$, 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. In Chapter 7, we see how in fact to do much better for random graphs and actually *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 $\epsilon > 0$).

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.

Chapters 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 Chapter 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 Chapter 5) is more complicated, but results in an improved bound of $\tilde{O}(n^{3/8})$ colors.

---

[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] Their algorithms differ slightly but the bounds are essentially the same. A version of their algorithm is described in Appendix A for completeness.

3.2   A few additional definitions

We now present a few additional definitions that will be needed in Chapters 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.

3.3   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.

Definition 3.1 *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,*

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

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, throw away the colored vertices, pick two new colors to work with and continue. 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 because each set has a small neighborhood; 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 3.1 *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.*

Progress Type 1 and a weaker variant of Type 2 were used by Wigderson [43]. In fact, if we do not care about constants, we can state Wigderson's algorithm for coloring n-vertex 3-colorable graphs with $O(n^{1/2})$ colors 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 [6] to color any 3-colorable graph with $O(\sqrt{n}/\sqrt{\log n})$ colors using these types of progress (here, $f(n) = \sqrt{n}/\sqrt{\log n}$). 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.

---

[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.

We now prove Lemma 3.1, showing that these types of progress really do "make progress".

Proof of Lemma 3.1: First, if algorithm $\mathcal{A}$ ever makes progress of Type 3 [Same-Color] on a subgraph of G, then since the two vertices u and v found must be the same color under any k-coloring of the subgraph, they also must be the same color under any k-coloring of G. So, we can just merge the vertices u and v into a new vertex with neighborhood $N(u) \cup N(v)$ 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 $C(m) \leq 2 + C(m - \epsilon m/f(m))$. Since f is a nearly-polynomial function, 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

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

Thus, 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:

$$U \;\leftarrow\; U \cup S$$
$$V' \;\leftarrow\; V' - (S \cup N(S)).$$

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 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 3.1, 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 Chapters 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.)

Corollary 3.2 *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.*

---

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

Chapter 4

Worst-case bounds for 3-colorable graphs: first algorithm

In this chapter, we describe an algorithm to color any n-vertex 3-colorable graph with $\tilde{O}(n^{0.4})$ colors. As mentioned in the last chapter, 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 sets. This fact will be useful for us later in Chapter 6 for extending these techniques to graphs of higher chromatic number.

4.1 Forcing expansion

In this section, we show that if our goal is to color a 3-colorable graph G with $O(f(n))$ colors, where f is a nearly-polynomial function as in Definition 3.1, 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 expansion in the graph.

Bounding the number of neighbors that may be shared by two vertices forces expansion in the following way. Suppose we wish to color with n^α colors. If we look at the neighborhood of some vertex v and consider an arbitrary subset of $m + d(v)$ edges leaving $N(v)$, then we may assume those edges enter into at least $m/n^{1-2\alpha}$ other vertices. The reason is that otherwise, some vertex $w \neq v$ must have more than $n^{1-2\alpha}$ neighbors in $N(v)$. This fact will be useful when we show in Section 4.3 how to find such a set of m edges whose endpoints contain an easy-to-find independent set.

Given the three methods for making progress defined in the last chapter, this method for forcing expansion falls out easily. Throughout this section, we assume f is a nearly-polynomial function.

21

Theorem 4.1 *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 4.2 *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, more complicated method for forcing expansion (described in Chapter 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 chapter, the algorithm uses a vertex cover approximation algorithm of Bar-Yehuda and Even [4] and (independently) Monien and Speckenmeyer [28] 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 thesis is given as Algorithm Approx-IS in Appendix A.

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 $O(n^{2/5}(\log n)^{8/5})$-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 4.1.*

3. *[Dist-2 Neighbors] Otherwise, for each vertex v, for each $i, j \in \{0, 1, \ldots, 5\log^2 n\}$:*

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

(Recall the definitions of Section 3.2.)

4. *[VC approx] Run the* BE/MS *Vertex-Cover approximation algorithm 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 are devoted to proving the following theorem.

Theorem 4.3 (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 3.1 (the usefulness of making progress), we get the following corollary.

Corollary 4.4 *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 4.1).

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 $\tilde{\Omega}(f(n)^4/n) = \tilde{\Omega}(n^{3/5})$ 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 the neighbors of the neighbors of v for some vertex v in the graph.

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

$$D_{\text{red}}(\text{blue} \cup \text{green})\ \geq\ \frac{1}{2}D(\text{blue} \cup \text{green}). \tag{4.1}$$

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

4.3.1   The basic approach, and a counterexample to 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 $S \in \text{blue} \cup \text{green}$ such that nearly half of the edges leaving S enter into **red** vertices. We know that if we look at the *entire* set **blue** $\cup$ **green**, at least half of the edges leaving that set enter into **red** vertices (equation (4.1)). The problem is: we do not know how to find **blue** $\cup$ **green**. We can, however, look at subsets of **blue** $\cup$ **green** by considering vertex neighborhoods, many of which (for **red** starting vertices) will be **blue** and **green**.

Given the property of **blue** $\cup$ **green** described in equation (4.1), one might expect that this property would hold for the neighborhood of some vertex as well: that is, that for some $v \in \text{red}$, we would have $D_{\text{red}}(N(v)) \geq \frac{1}{2}D(N(v))$. Unfortunately, this may not necessarily be the case, and what follows is a counterexample to this seemingly innocent claim.

Consider a graph with m **red** vertices $r_0, \ldots, r_{m-1}$, $m + 1$ **green** vertices $g_0, \ldots, g_m$, and $m + 1$ **blue** vertices $b_0, \ldots, b_m$. Vertices g_m and b_m are two distinguished vertices with large degree and twice as many edges into **blue** or **green** vertices than into **red** vertices. The rest of the vertices have low degree, but together there are enough edges with **red** endpoints so that $D(\text{red})$ is greater than $D(\text{blue})$ or $D(\text{green})$. More specifically, the edges in the graph
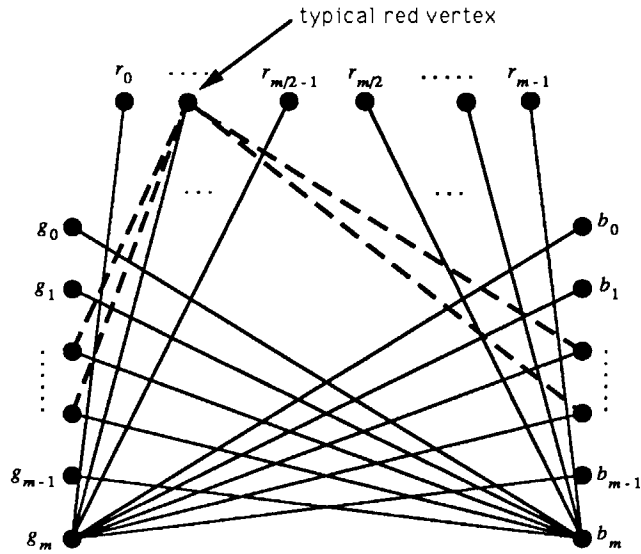
Figure 4.1: A counterexample to the naive strategy. For clarity, only edges incident to the distinguished vertices g_m and b_m, and incident to a typical red vertex are given. The four edges between the red vertex and the non-distinguished blue and green vertices are shown as dashed lines.

are: (see Figure 4.1)

$$\{(g_m, r_0), (g_m, r_1), \ldots, (g_m, r_{m/2-1})\}$$

$$\cup \ \{(g_m, b_0), (g_m, b_1), \ldots, (g_m, b_{m-1})\}$$

$$\cup \ \{(b_m, r_{m/2}), (b_m, r_{m/2+1}), \ldots, (b_m, r_{m-1})\}$$

$$\cup \ \{(b_m, g_0), (b_m, g_1), \ldots, (g_m, b_{m-1})\}$$

$$\cup \ \{(g_i, r_i), (g_i, r_{(i+1)\bmod m})\} \text{ for each } 0 \le i \le m - 1$$

$$\cup \ \{(b_i, r_i), (b_i, r_{(i+1)\bmod m}) \text{ for each } 0 \le i \le m - 1.$$

That is, vertices g_m and b_m are each connected to a different half of of the red vertices and each are connected to all the vertices of index less than m of the remaining color. In addition, each r_i is connected to two green and two blue vertices of index less than m.

So, $D(\text{red}) = 5m$, $D(\text{green}) = (4 + \frac{1}{2})m$, and $D(\text{blue}) = (4 + \frac{1}{2})m$. But, for each red vertex v, we have $D_{\text{red}}(N(v)) = 8 + m/2$ and $D_{V-\text{red}}(N(v)) = 4 + m$, which implies $D(N(v)) = 12 + 3m/2$. So, $D_{\text{red}}(N(v))$ is approximately one third of $D(N(v))$ rather than one half. One can also construct variations of this counterexample in which the ratio between $D_{\text{red}}(N(v))$ and $D(N(v))$ is even worse.

The problem here is that the vertices have wildly varying degrees. While one can also find variations on this counterexample that hold even when all vertices have degrees in the

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 there exists $v \in V$ 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 4.5 below.

Once we have a set $S \subseteq N(v)$ with nearly half the edges leaving it entering into **red** vertices, we again 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 4.6.

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 4.5 *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:*

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

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 4.6 *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:

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

Assuming for now the correctness of Theorems 4.5 and 4.6, 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 4.7 *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 $d_{min} \geq \max\{s(1 + \delta), (3\log_{1+\delta} n)/\delta\}$, 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 4.7: By definition of set red in G, the conditions of Theorem 4.5 are satisfied for $R = $ red and $\lambda = 1/2$ (see equation (4.1)). Let vertex v and bin I_j be such that claims (1) and (2) of Theorem 4.5 are satisfied for $S = N(v) \cap I_j$. By claim (2) of Theorem 4.5, set S satisfies the conditions of Theorem 4.6 with $\lambda' = \frac{1}{2}(1 - 3\delta)$. Let i be the index such that claims (1) and (2) of Theorem 4.6 are satisfied and let $T = N_i(S)$. Then:

$$
\begin{array}{llll}
D_{T \cap R}(S) & \geq & \delta D_R(S)/(\log_{1+\delta} n) & \text{(Theorem 4.6, claim 1)} \\
& \geq & \delta\left[\lambda(1 - 3\delta)D(S)\right]/(\log_{1+\delta} n) & \text{(Theorem 4.5, claim 2)} \\
& \geq & \delta\lambda(1 - 3\delta)\left[d_{min}|S|\right]/(\log_{1+\delta} n) & \text{(for all } v, d(v) \geq d_{min}) \quad (4.2) \\
& \geq & \delta^3\lambda(1 - 3\delta)d_{min}^2/(\log_{1+\delta} n)^2 & \text{(Theorem 4.5, claim 1)} \\
& = & \Omega\left(\delta^5 d_{min}^2/(\log^2 n)\right) & \text{(using } \log_{1+\delta} n = O(\frac{1}{\delta}\log n)) \\
& = & \Omega\left(d_{min}^2/(\log^7 n)\right). & (\delta = \frac{1}{5\log n})
\end{array}
$$

Since no two vertices share more than s neighbors and $S \subseteq N(v)$, we know no vertex $w \neq v$ has more than s neighbors in S. Since we have also assumed that $d_{min} \geq s(1 + \delta)$, we know that the set $N_{i'}(S)$ containing v contains no other vertices besides v by definition of N_i. Also, since $d_{min} \geq (3\log_{1+\delta} n)/\delta$, by equation (4.2) we have $D_{T \cap R}(S) > |S|$ so we know $T \neq \{v\}$ and thus $v \notin T$. So, set T consists only of vertices with at most s neighbors in S and we have:

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

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

$$
\begin{array}{llll}
|T \cap R|/|T| & \geq & \lambda(1 - 2\delta)(1 - 3\delta) & \text{(Theorems 4.5 claim 2, and 4.6 claim 2)} \\
& \geq & \frac{1}{2}(1 - 5\delta) & \text{(by definition of red, we have } \lambda \geq 1/2) \\
& \geq & \frac{1}{2}\left(1 - \frac{1}{\log n}\right).
\end{array}
$$

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

Before proving Theorems 4.5 and 4.6, we state a simple combinatorial lemma:

Lemma 4.8 *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 4.5: 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 4.8 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 4.8 tells us, taking $\epsilon = \delta$, that for some j_0, if we let $I = I_{j_0} \cap (V - R)$, then:

$$D_R(I) \geq \delta D_R(V - R)/(\log_{1+\delta} n) \quad \text{and} \tag{4.3}$$

$$D_R(I) \geq \lambda(1 - \delta)D(I). \tag{4.4}$$

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 (4.3)), that almost a λ fraction of the edges leaving I enter red nodes (equation (4.4)), 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 4.5. 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 4.5. 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 4.5 holds. So, from the definition of R', we have:

$$
\begin{aligned}
D_{R'}(I) &\geq D_R(I) - |R|\delta^2 d/\log_{1+\delta} n \\
&\geq D_R(I) - D_R(V-R)\delta^2/\log_{1+\delta} n \qquad \text{(since } D_R(V-R) = D(R) \geq d|R|) \\
&\geq D_R(I) - \Big(D_R(I)(\log_{1+\delta} n)/\delta\Big)\Big(\delta^2/\log_{1+\delta} n\Big) \quad \text{(by equation (4.3))} \\
&\geq D_R(I)(1-\delta).
\end{aligned}
$$

Finally, applying equation (4.4) we have:

$$
D_{R'}(I) > \lambda(1-2\delta)D(I). \tag{4.5}
$$

We now claim that for some $v \in R'$, the set $N(v) \cap I$ satisfies claim (2) of Theorem 4.5. 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: [1]

For all $v \in R'$, $\quad D_{R'}(N(v) \cap I) < \lambda(1-3\delta)D(N(v) \cap I)$. $\qquad$ (contr 4.6)

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

$$
\sum_{v \in R'} D_{R'}(N(v) \cap I) < \lambda(1-3\delta)\sum_{v \in R'} D(N(v) \cap I). \tag{contr 4.7}
$$

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 $\sum_{v \in R'} \Big[\sum_{w \in N(v) \cap I} d(w)\Big]$ 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 4.7) we have assumed for contradiction, we get:

$$
\begin{aligned}
\sum_{w \in I} d_{R'}(w)^2 &< \lambda(1-3\delta)\sum_{w \in I} d_{R'}(w)d(w) \\
&< \lambda(1-3\delta)\sum_{w \in I} d_{R'}(w)(1+\delta)^{j_0+1} \quad \text{(since } d(w) < (1+\delta)^{j_0+1} \text{ for all } w \in I) \\
&= \lambda(1-3\delta)(1+\delta)^{j_0+1}D_{R'}(I). \quad \text{(by definition of } D_{R'}) \tag{4.8}
\end{aligned}
$$

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

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

---

[1]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.

So, $D_{R'}(I)^2/|I| \leq \sum_{w \in I} d_{R'}(w)^2$. Combining this fact with equation (4.8), we have:

$$\frac{1}{|I|} D_{R'}(I)^2 \; < \; \lambda(1 - 3\delta)(1 + \delta)^{j_0+1} D_{R'}(I). \tag{4.9}$$

Multiplying both sides of equation (4.9) 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 (4.5) and completes the proof of Theorem 4.5.   ∎

Proof of Theorem 4.6: 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 4.8 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 4.8, 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 (4.10)

2. $D_{N_{i_0}(S) \cap R}(S)/D_{N_{i_0}(S)}(S) \; \geq \; (1 - \delta)\lambda'$. (4.11)

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:

$$\begin{aligned} |N_{i_0}(S) \cap R|/|N_{i_0}(S)| \; &\geq \; \left[D_{N_{i_0}(S) \cap R}(S)/D_{N_{i_0}(S)}(S) \right]/(1 + \delta) \\ &\geq \; (1 - \delta)\lambda'/(1 + \delta) \\ &\geq \; (1 - 2\delta)\lambda'. \end{aligned} \tag{4.12}$$

Equations (4.10) and (4.12) 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 [4][28] discussed earlier (and also presented as algorithm **Approx-IS** in Appendix A) 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:

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

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 4.9 *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 (4.3).

Proof of Theorem 4.3: 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 4.7 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 4.9, 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:

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

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. ∎

Chapter 5

Worst-case bounds for 3-colorable graphs: improved algorithm

In this chapter, we present a procedure that improves on the bounds achieved by Algorithm **First-Approx** given in Chapter 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.

Algorithm **First-Approx** performs most poorly when the input graph consists of a collection of high-density regions or "clumps," with a lower density of edges between clumps. In particular, it performs worst when the set $S = N(v) \cap I_j$ has a large fraction of its neighbors hit by about $n^{0.2}$ edges from vertices in S. Here we present an additional tool for making progress from such dense regions and thus improve the coloring bound.

5.1 A useful lemma

We now present a strengthening of Corollary 4.2, described in Lemma 5.1 below, that 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. As with Corollary 4.2, this lemma requires the graph to be 3-colorable.

Let $f(n)$ be some nearly-polynomial function.

Lemma 5.1 *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 $\max\{|N(S) \cap \textbf{green}|, |N(S) \cap \textbf{blue}|\}$. 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 4.5 and 4.6.

Proof of Lemma 5.1:

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 the endpoints from S' (let $S' \leftarrow S' - \{a, b\}$). If we ever have deleted more than $\frac{|S|}{4 \log n}$ edges from S, 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 3.2. 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* $N'(v)$
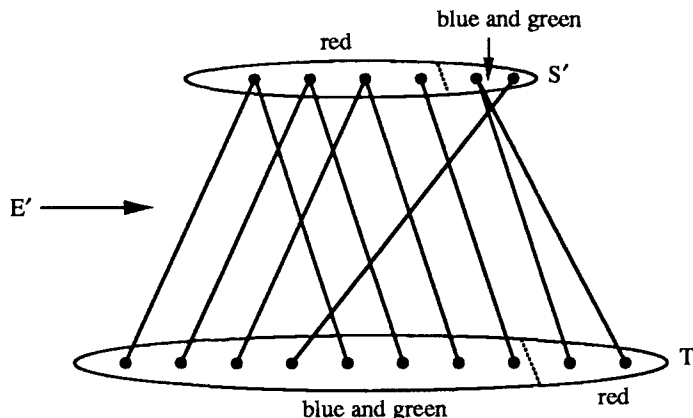
Figure 5.1: Vertices in S' have disjoint marked neighborhoods. If the vertices had nearly identical "marked degree," then a mostly red set S' would imply a mostly blue and green set T.

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 5.1.)

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 $\left(1 - \frac{1}{2\log n}\right)$ 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).

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, the independent set approximation algorithm **Approx-IS** given in Appendix A) 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 4.9, 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:

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

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 5.1 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 3.2).

Theorem 5.2 *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. *$[D_T(S)]^3 = \Omega\left(\left[|S| + \max_{v \in S} d_T(v)\right] \times \left[|S||T|^2(n\log n)/f(n)^2 + |T||S|^2 n^2/f(n)^4\right]\right)$,*

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:

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

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

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

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

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

- 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,

$$D_{T'}(S) \geq D_{\text{total}}/2. \tag{5.1}$$

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 5.1 implies:

$$D_{T'}(S') \geq D_{\text{total}}/4. \tag{5.2}$$

Also, by definition of S' and equation (5.1), if $v \in S'$ then $d_{T'}(v) \geq \frac{1}{4}\frac{D_{\text{total}}}{|S|}$ or equivalently,

$$d_{T'}(v) \geq \frac{1}{4}d_{\text{avg}} \quad \text{for all } v \in S'. \tag{5.3}$$

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 5.1 (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 5.2):

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

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

So, we have:

$$
\begin{aligned}
|X| &\geq \frac{1}{8}D_T(S')/(|T| \log n) \\
&\geq \frac{1}{32}D_{\text{total}}/(|T| \log n) \\
&= \Omega\left(\left(\frac{|S|}{|T|}\right)\left(\frac{d_{\text{avg}}}{\log n}\right)\right).
\end{aligned}
\tag{5.4}
$$

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
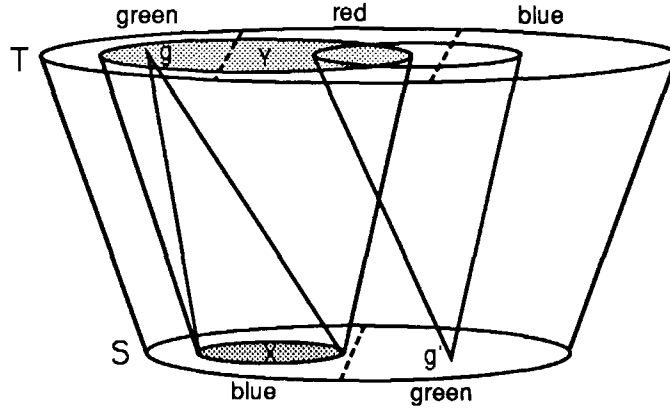
Figure 5.2: Vertex g and the sets X and Y. Also, green vertex $g' \in S$ (defined later) and the intersecting neighborhoods.

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 4.1 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 all of the other vertices in X. This implies, by equation (5.3), 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:

$$|Y| \;=\; \Omega\left(\min\left\{|X|d_{\text{avg}},\; \frac{f(n)^2}{n}(d_{\text{avg}})^2\right\}\right). \tag{5.5}$$

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:

$$
\begin{aligned}
D_S(Y) &\geq \left(\frac{1}{2}\frac{D_{\text{total}}}{|T|}\right)|Y| \\
&= \frac{1}{2}\frac{|S|}{|T|}d_{\text{avg}}|Y| \\
&= \Omega\left(\min\left\{|X|\frac{|S|}{|T|}(d_{\text{avg}})^2,\; \frac{f(n)^2}{n}(d_{\text{avg}})^3\frac{|S|}{|T|}\right\}\right) \quad \text{(by equation 5.5)} \\
&= \Omega\left(\min\left\{\left[\frac{|S|}{|T|}\right]^2(d_{\text{avg}})^3/\log n,\; \frac{f(n)^2}{n}(d_{\text{avg}})^3\frac{|S|}{|T|}\right\}\right). \quad \text{(by equation 5.4) (5.6)}
\end{aligned}
$$

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:

$$\left[\frac{|S|}{|T|}\right]^2(d_{\text{avg}})^3/\log n \;=\; \left[|S| + \max_{v \in S}d_T(v)\right] \cdot \Omega\left(\frac{n}{f(n)^2}\right) \tag{5.7}$$

and

$$\frac{f(n)^2}{n}(d_{\text{avg}})^3 \left[\frac{|S|}{|T|}\right] = \left[|S| + \max_{v \in S} d_T(v)\right] \cdot \Omega\left(\frac{n}{f(n)^2}\right). \tag{5.8}$$

Thus, combining both equations (5.7) and (5.8) with equation (5.6), we get:

$$D_S(Y) = \Omega\left(\frac{n}{f(n)^2}\left[|S| + \max_{v \in S} d_T(v)\right]\right). \tag{5.9}$$

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 (5.9), 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 5.2. Thus, we can make progress on this monochromatic set using Corollary 4.2.

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:

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

which by equation (5.9) implies:

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

Since Z is monochromatic (**blue**) we can now use Corollary 4.2 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 5.2 in some graph G.*
>
> **Output:** *Progress towards an $O(f(n))$-coloring of G.*
>
> > 1. *Run the algorithm of Lemma 5.1 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 4.1 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 need to use the sets S' and T'; they were just convenient for the analysis.)*

   (b) *Run the algorithm of Corollary 4.2 on Z.*

   (c) *For each $w \in Z$, run the algorithm of Corollary 4.2 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 5.2, then make progress using Algorithm Dense-Region-Progress.*

Theorem 5.3 *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 Chapter 4, let $R = $ red be the color class with $D(\text{red}) = \max(D(\text{red}), D(\text{blue}), D(\text{green}))$.

We now apply some of the facts proven in Section 4.3.2. Theorem 4.5 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:

$$|S| \geq \delta^2 f(n)/\log_{1+\delta} n, \text{ and} \tag{5.10}$$

$$D_R(S) \geq \tfrac{1}{2}(1 - 3\delta)D(S), \tag{5.11}$$

where $\delta = \frac{1}{5\log n}$. Note that for the given value of f, equation (5.10) and the definition of δ imply that:

$$|S| = \Omega(n^{3/8}/(\log n)^{3/2}). \tag{5.12}$$

Theorem 4.6 (using $\lambda' = \tfrac{1}{2}(1 - 3\delta)$) shows that for some index i, the set $T = N_i(S)$ of step 2(b) has the property that:

$$D_{T \cap R}(S) \geq \delta D_R(S)/\log_{1+\delta} n, \text{ and} \tag{5.13}$$

$$|T \cap R|/|T| \geq \tfrac{1}{2}(1 - 2\delta)(1 - 3\delta). \tag{5.14}$$

Let us now, for the rest of the proof, fix two such sets S and T satisfying equations (5.10) through (5.14). 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 5.14 above, set T contains an independent set $(T \cap R)$ of at least a fraction $\tfrac{1}{2}(1 - \frac{1}{\log n})$ of its vertices (using $\delta = \frac{1}{5\log n}$). So by Lemma 4.9, 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 5.2. 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 (5.11) and (5.13) as follows.

$$D_T(S) \geq D_{T \cap R}(S) = \Omega\left(D(S)/(\log n)^3\right) \tag{5.15}$$

$$= \Omega(d|S|/(\log n)^3). \tag{5.16}$$

So,

$$D_T(S)/|S| \geq \Omega(n^{3/8}/(\log n)^{1/2}) \tag{5.17}$$

$$= \Omega(n^{1/4}/(\log n)^3). \tag{5.18}$$

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

$$[D_T(S)]^3 \;=\; \Omega\left(\Big[|S| + \max_{v \in S} d_t(v)\Big] \cdot \Big[|S|\,|T|^2 \frac{n^{1/4}}{(\log n)^4} + |T|\,|S|^2 \frac{n^{1/2}}{(\log n)^{10}}\Big]\right). \tag{5.19}$$

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

From equation (5.17), we have

$$|S| \;=\; O\Big((\log n)^{1/2} D_T(S)/n^{3/8}\Big). \tag{5.20}$$

Next, our very condition for this case was that:

$$|T| \;=\; O\Big(n^{5/8}/(\log n)^{3/2}\Big). \tag{5.21}$$

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:

$$\begin{aligned}
\max_{v \in S} d_T(v) &= O(D(S)/|S|) \\
&= O(D_T(S)(\log n)^3/|S|) & \text{(using equation 5.15)} \\
&= O\Big(D_T(S)(\log n)^3(\log n)^{3/2}/n^{3/8}\Big) & \text{(using equation 5.12)} \\
&= O\Big(D_T(S)(\log n)^{9/2}/n^{3/8}\Big). & (5.22)
\end{aligned}$$

The three equations (5.20), (5.21), and (5.22) allow us to reduce requirement (5.19) to the condition that:

$$\begin{aligned}
[D_T(S)]^3 &= \Omega\left(\Big[(\log n)^{9/2}\frac{D_T(S)}{n^{3/8}}\Big] \cdot \Big[D_T(S)\frac{n^{9/8}}{(\log n)^{13/2}} + D_T(S)^2 \frac{n^{3/8}}{(\log n)^{21/2}}\Big]\right) \\
&= [D_T(S)]^2 \cdot \Omega\left(\frac{n^{3/4}}{(\log n)^2} + \frac{D_T(S)}{(\log n)^6}\right). & (5.23)
\end{aligned}$$

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 (5.17) and (5.12) 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.   ∎

Chapter 6

Worst-case bounds for 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 [43]. 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 4.1), 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 6.1. 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. Note: the bounds in the last two rows are with high probability over the coin tosses of the algorithm. See Corollaries 6.2 and 6.7 for more precise bounds.

6.1 A simple recursive approach

A standard method [43][6][22] 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:

Where	$k = 3$	4	5	6	7	general
Wigderson [43]	$n^{1/2}$	$n^{2/3}$	$n^{3/4}$	$n^{4/5}$	$n^{5/6}$	$n^{1-\frac{1}{k-1}}$
	$n^{0.5}$	$n^{0.667}$	$n^{0.75}$	$n^{0.8}$	$n^{0.833}$	
base: $k = 3$	$n^{3/8}$	$n^{8/13}$	$n^{13/18}$	$n^{18/23}$	$n^{23/28}$	$n^{1-\frac{1}{k-7/5}}$
	$n^{0.375}$	$n^{0.615}$	$n^{0.722}$	$n^{0.783}$	$n^{0.821}$	
base: $k = 4$	—	$n^{3/5}$	$n^{5/7}$	$n^{7/9}$	$n^{9/11}$	$n^{1-\frac{1}{k-3/2}}$
	—	$n^{0.6}$	$n^{0.714}$	$n^{0.778}$	$n^{0.818}$	
best we have	$n^{3/8}$	$n^{3/5}$	$n^{\frac{91}{131}}$	$n^{\frac{105}{137}}$	$n^{\frac{5301}{6581}}$	
	$n^{0.375}$	$n^{0.6}$	$n^{0.695}$	$n^{0.766}$	$n^{0.806}$	

Table 6.1: Summary of results in "$\tilde{O}$" notation 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.

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 chapters 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:

$$r(\mathcal{A}) = k_0 - \frac{1}{1-\alpha}. \tag{6.1}$$

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 chapter 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:

$$\tilde{O}\left(n^{1-\frac{1}{k-7/5}}\right) \quad \text{colors.} \tag{6.2}$$

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 6.1 *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:*

$$C_k(n) = [c + (k - k_0)]n^{1-1/(k-r)}(\log n)^{\beta\left[\frac{k_0-r}{k-r}\right]} \tag{6.3}$$

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

Using Theorem 6.1 and the bounds achieved by algorithm Improved-Approx, ($k_0 = 3, \alpha = 3/8, \beta = 5/2$),we can restate formula (6.2) more precisely in the following corollary.

Corollary 6.2 *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 (6.3).*

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 6.1: 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 (6.3) in the statement of the theorem holds for the base case of $k = k_0$ since for $k = k_0$, we have:

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

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 6.1 inductively for $k' < k$, we need to show that $C_k(n) \le c_k f(n,k)$.

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

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

So, substituting in the bounds of Theorem 6.1 inductively, we have:

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

6.2 Directly extending the $k = 3$ algorithm

6.2.1 Motivation

In this section, we describe how the methods of Algorithm First-Approx of Chapter 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 4.5 and 4.6 ensures that one such set will be nearly half red under some legal 3-coloring of the graph, and the expansion property of Theorem 4.1 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 4.5 and 4.6. 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:

$$D_{R'}(V' - R') \quad \approx \quad \tfrac{1}{3}D(V' - R'), \tag{6.4}$$

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 (6.4) above, we could then re-apply 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 $\beta(k) \le 5.5$. 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:

$$\frac{1}{1-\alpha(k)} \;=\; 2 - \frac{1}{2^{k-2}} + \frac{1}{1-\alpha(k-2)}\left(1 - \frac{1}{2^{k-2}}\right). \tag{6.5}$$

We will examine this formula in more detail later, but we just note here that α is nondecreasing 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:

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

The sets I_j and $N_i(v)$ used in Chapter 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 $D_R(V - R) \ge \lambda D(V - R)$ for some constant λ and that the graph have sufficiently large minimum degree. Let us, in fact, repeat Corollary 4.7 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 $\lambda \ge 1/2$.)

Corollary 6.3 (Variant of Corollary 4.7) *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 $d_{min} \geq \max(s(1 + \delta), (3 \log n)/\delta^2)$, 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 6.4 *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:*

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

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

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

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 6.4.   ∎

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 $\alpha \in [0,1]$, $\beta \in \mathbf{Z}$ 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 $\delta^2 m n^{\alpha-1} \log^\beta n$: 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 $\delta^2 m n^{\alpha-1} \log^\beta n$, we can greedily find an independent set inside W_i of size at least:

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

   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 6.5 [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 $\delta^2 m n^{\alpha-1} \log^\beta n$ 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:

$$\begin{aligned}
\mathbf{Pr}[A_i] &= D(R_i)/|E_i| \\
&= 2D(R_i)/D(V_i).
\end{aligned} \tag{6.6}$$

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 every time we remove an edge the probability it has an endpoint in R is less than p, by Chernoff bounds [2] 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:

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

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 6.5, 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 (6.5), 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 6.6 *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 (6.5).*

Before proving Theorem 6.6, let us examine the claimed performance more closely. Let $\gamma(k) = \frac{1}{1-\alpha(k)}$. So, equation (6.5) can be written as:

$$\gamma(k) = 2 - \frac{1}{2^{k-2}} + \gamma(k-2)\left(1 - \frac{1}{2^{k-2}}\right). \tag{6.7}$$

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 6.1 at the beginning of this chapter shows the results for both algorithms and for various combinations. In particular, for example, we can substitute the bound of Theorem 6.6 for $k = 4$ into the bound of Theorem 6.1 to get the following corollary.

Corollary 6.7 *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 6.6:

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 6.3, one of the graphs $G' = G_{v,i,j}$ created in Step 4 will both have size at least:

$$\begin{aligned} m_1 &= (d_{min})^2/(s_k(n)\log^7 n) \\ &= n^{2\alpha}\log^{2\beta} n/(s_k(n)\log^7 n), \end{aligned} \tag{6.8}$$

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.[1]

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 6.5, if Step P3 does not halt with success immediately, then one of the graphs H_l produced will have both a minimum degree of $\delta m_i n^{\alpha-1} \log^\beta n$

---

[1]One can verify that the minimum degrees and the values m_i defined satisfy the technical conditions of Corollary 6.3 (min degree > $\max(s(1 + \delta), (3\log n)/\delta^2)$) and Theorem 6.5 ($m_i \gg 1/\delta^2$).

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:

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

Using the minimum degree bound and degree ratios above, Corollary 6.3 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:

$$
\begin{aligned}
m_{i+1} &= \delta^4 m_i^2 n^{2\alpha-2}(\log n)^{2\beta}/(s_k(n)\log^7 n) \\
&= \Omega(m_i^2 n^{2\alpha-2}(\log n)^{2\beta}/(s_k(n)\log^{11} n) \\
&= \Omega(m_i^2 n^{2\alpha-2}/s_k(n)), \quad\quad\quad \text{(for } \beta = 5.5) \quad\quad (6.9)
\end{aligned}
$$

$$
\begin{aligned}
\text{and} \quad \lambda_{i+1} &\geq \frac{\lambda_i - 2\delta}{1 - \lambda_i + 2\delta} - 5\delta \\
&\geq \frac{\lambda_i - 4\delta}{1 - \lambda_i} - 5\delta \\
&\geq \frac{\lambda_i}{1 - \lambda_i} - 13\delta \quad\quad \text{for } \lambda_i \leq 1/2. \quad\quad (6.10)
\end{aligned}
$$

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 (6.10), we have:

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

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

$$\lambda_{k-2} \quad \geq \quad (\frac{1}{2} - \frac{1}{\log n}). \quad\quad\quad\quad (6.11)$$

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 (6.8) and the fact that for $\beta = 5.5$ that $\log^{2\beta} n > \log^7 n$. For $i > 1$, we can check

inductively that (6.9) satisfies the claim as follows:

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

So,

$$m_{k-2} = \Omega(n^{(2^{k-1}-2)\alpha} \cdot n^{2-2^{k-2}} \cdot [s_k(n)]^{1-2^{k-2}}). \qquad (6.12)$$

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 (6.11)) and have size at least m_{k-2}, as given in equation (6.12). By lemma 4.9, Step 6 will find an independent set of size at least $m_{k-2}/\log n$.

Thus, to prove Theorem 6.6 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 (6.12), 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:

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

Rearranging terms, this formula is equivalent to:

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

or:

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

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

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

But, this formula is exactly the definition of $\alpha(k)$ given in equation (6.5). So Algorithm **Multi-Stage-Color** works as claimed. ∎

Chapter 7

Random models for k-colorable graphs

While the problem of coloring *worst-case* k-colorable graphs seems quite difficult, it turns out that coloring *random* k-colorable graphs is much easier. In fact, it is well known by results of Kucera [23], Turner [38], and others that random k-colorable graphs can be k-colored in polynomial-time with high probability. These results show that, in fact, *most* k-colorable graphs are easy to k-color. Dyer and Frieze [18] go further and provide an algorithm that when amortized over all n-vertex k-colorable graphs, spends polynomial time *on average* per graph. Experimental work on various heuristics for coloring random k-colorable graphs has been done by Petford and Welsh [31].

The standard model for a random n-vertex graph is the model $\mathcal{G}(n, p)$ in which each possible edge (u, v) is placed into the graph with probability p. This model has the property that the distribution $\mathcal{G}(n, 1/2)$ is the same as that obtained by selecting a labeled n-vertex graph uniformly at random from the set of all n-vertex graphs.

There are several natural models, however, for what one means by a *random k-colorable* graph. Dyer and Frieze examine several and prove relationships among them [18]. We focus here on one model that happens to be simplest to analyze, which we shall denote $\mathcal{G}(n, p, k)$. A graph is selected in $\mathcal{G}(n, p, k)$ according to the following procedure. First each labeled vertex is independently assigned to one of k color classes with equal probability $1/k$. Then, independently for each pair u, v of vertices in different color classes, the edge (u, v) is placed into the graph with probability p. We use the notation:

- $G \leftarrow \mathcal{G}(n, p, k)$

to mean that G is selected according to the distribution defined by this model.

The $\mathcal{G}(n, p, k)$ model is a natural one for a random n-vertex k-colorable graph, though even for $p = 1/2$ it is not equivalent to selecting a graph uniformly at random from the set of all n-vertex k-colorable graphs. In particular, graphs that can be k-colored in multiple different ways are over-represented in $\mathcal{G}(n, 1/2, k)$ since different assignments to the color classes may still lead to the same graph. (See Dyer and Frieze [18] for more on the relationship between the models.)

In this chapter, we consider the problem of k-coloring graphs in $\mathcal{G}(n,p,k)$ for as low an average edge density as possible. We present an algorithm to k-color such graphs with high probability for any constant k, and for $p \geq n^{o(1)-1}$; that is, the procedure will work for the average degree as low as n^ϵ for any fixed $\epsilon > 0$. Before describing that algorithm, however, we point out first a quite easy method to k-color $G \leftarrow \mathcal{G}(n,p,k)$ for $p \geq n^{-1/2+\epsilon}$ ($\epsilon > 0$).

This idea of the easier procedure is simply this: two vertices from the *same* color class in G will tend to share more neighbors in common than two vertices of different color. Two vertices from the same class have an expected $n - n/k$ vertices they might potentially share as neighbors, and so can be expected to share $n(1-1/k)p^2$ neighbors in common. However, two vertices from different color classes have only an expected $n - 2n/k$ vertices they may share as neighbors, and so they can be expected to share only $n(1 - 2/k)p^2$ neighbors in common. For $p \geq n^{-1/2+\epsilon}$, these values are $n^{2\epsilon}(1-1/k)$ and $n^{2\epsilon}(1-2/k)$ respectively. Since for any given pair of vertices x, y, the indicator random variables X_v for the event that v is a neighbor to both x and y are mutually independent over all v (and each occurring with probability p^2 if v is a different color from both x and y), we may apply Chernoff bounds. In particular, if $X = \sum X_v$, and $\mu = \mathbf{E}[X]$ is the expected number of neighbors in common between x and y, Chernoff bounds state that for any $\delta > 0$,

$$\mathbf{Pr}[X < (1-\delta)\mu \text{ or } X > (1+\delta)\mu] \; < \; 2e^{-\delta^2\mu/3}$$

(See Angluin and Valiant [2]). For $\mu = \Theta(n^{2\epsilon})$, this probability is so small that even when summed over all pairs of starting vertices x, y, the probability *any* pair shares a number of neighbors that differs by more than $\delta\mu$ from the expectation is $o(1)$.

One thus finds that with high probability, *all* pairs of vertices selected in the same color class share $n^{2\epsilon}[1 - 1/k](1 + o(1))$ neighbors and *all* pairs of vertices of different color share only $n^{2\epsilon}[1 - 2/k](1 + o(1))$ neighbors in common. Thus, one can easily algorithmically separate the color classes.

7.1 An improved algorithm

In this section, we describe an algorithm based on an extension of the above idea that k-colors graphs in $\mathcal{G}(n,p,k)$ for much lower values of p. The results presented here are based on work joint with Joel Spencer.

Another way to view the above observation is that vertices of the same color will have more paths of length 2 between them than vertices of different colors. This idea can be extended to paths of a longer constant length l for improved bounds. If l is *even*, it turns out (see Section 7.1.1) that the expected number of paths of length l between two vertices of the same color is higher than the expected number for vertices of different color. If l is

odd, the reverse holds. The difficulty in analyzing the case $l > 2$, however, is that the events corresponding to the paths of length l between two vertices are no longer independent. Different paths of length 2 between two vertices x and y share no edges in common, but two paths of length 3 might share an edge: for example, consider the two paths (x, w, w', y) and (x, w', w, y). So, to prove that the number of paths will be with high probability close to the number expected, one needs a more sophisticated probabilistic analysis. Luckily, such analysis for a general class of this type of problem has already been provided by Spencer [35] in the context of the random graph model $\mathcal{G}(n, p)$. It turns out that the same analysis holds for the $\mathcal{G}(n, p, k)$ model as well.

7.1.1 Calculating expectations

Let $l \geq 2$ be some integer constant and let us fix two vertices x and y. By a "path" we will always mean a simple path; that is, one that never touches any vertex more than once. In this section, we calculate the *expected* number of paths of length l between x and y in $G \leftarrow \mathcal{G}(n, p, k)$ and show this expectation differs by a constant factor depending on whether or not x and y are in the same color class.

In particular let $E_l(p)$ be the expected number of paths of length l between x and y in $G \leftarrow \mathcal{G}(n, p, k)$, and let $E_l^{\text{same}}(p)$ and $E_l^{\text{diff}}(p)$ be the expected number of such paths *given* that x and y are chosen in the same or in different color classes respectively. Also, for $p > 0$ let $\lambda_l(p) = [E_l^{\text{same}}(p) - E_l^{\text{diff}}(p)]/E_l(p)$. When p is clear from context, we will just write $E_l, E_l^{\text{same}}, E_l^{\text{diff}}$, and λ_l for the above quantities. We show now that for constant k and l, the value λ_l is bounded away from 0 by a constant.

We can calculate the expected number of paths between x and y by fixing some arbitrary sequence of distinct vertices (also distinct from x and y) $v_1, \ldots, v_{l-1}$ and calculating the probability of the event B_l that each pair $\langle x, v_1 \rangle, \langle v_1, v_2 \rangle, \ldots, \langle v_{l-2}, v_{l-1} \rangle, \langle v_{l-1}, y \rangle$ consists of vertices chosen in different color classes. Given that the event B_l occurs, the probability the path $\langle x, v_1, \ldots, v_{l-1}, y \rangle$ appears in G is simply p^l. Given that B_l does not occur, the probability is 0. Since there are $(n - 2)_{l-1} = (n - 2)(n - 3) \cdots (n - l) = n^{l-1}[1 - o(1)]$ possible such sequences $v_1, \ldots, v_{l-1}$, the expectation E_l is simply $[1 - o(1)]p^l n^{l-1} \mathbf{Pr}[B_l]$.

For any random variable X, let $\mathbf{Pr}^{\text{same}}[X]$ and $\mathbf{Pr}^{\text{diff}}[X]$ be the probability that event X occurs *given* that x and y are in the same color class, or *given* that x and y are in different color classes, respectively. Thus, we have:

$$E_l = E_l(p) = [1 - o(1)]p^l n^{l-1} \mathbf{Pr}[B_l], \tag{7.1}$$

$$E_l^{\text{same}} = E_l^{\text{same}}(p) = [1 - o(1)]p^l n^{l-1} \mathbf{Pr}^{\text{same}}[B_l], \tag{7.2}$$

$$E_l^{\text{diff}} = E_l^{\text{diff}}(p) = [1 - o(1)]p^l n^{l-1} \mathbf{Pr}^{\text{diff}}[B_l]. \tag{7.3}$$

Also, since the $p^l(n-2)_{l-1}$ terms factor out of the expression for λ_l, we have:

$$\lambda_l = \lambda_l(p) \quad = \quad \frac{\mathbf{Pr}^{\text{same}}[B_l] - \mathbf{Pr}^{\text{diff}}[B_l]}{\mathbf{Pr}[B_l]} \tag{7.4}$$

So, to compute E_l, E_l^{same}, E_l^{diff}, and λ_l, we need only examine the fixed sequence $v_1, \ldots, v_{l-1}$ and the event that all are chosen colors such that the path $\langle x, v_1, \ldots, v_{l-1}, y \rangle$ is a "potential path" in the graph.

The value $\mathbf{Pr}[B_l]$ is quite easy to calculate: each vertex in the path has a $(1 - \frac{1}{k})$ probability of being given a different color than the preceding vertex. Thus,

$$\mathbf{Pr}[B_l] \quad = \quad (1 - \tfrac{1}{k})^l. \tag{7.5}$$

Also, clearly, $\mathbf{Pr}[B_l] = \mathbf{Pr}^{\text{same}}[B_l] \cdot \mathbf{Pr}[x$ and y are chosen the same color$] + \mathbf{Pr}^{\text{diff}}[B_l] \cdot \mathbf{Pr}[x$ and y are chosen of different color$]$. So,

$$\mathbf{Pr}[B_l] \quad = \quad \tfrac{1}{k}\mathbf{Pr}^{\text{same}}[B_l] + (1 - \tfrac{1}{k})\mathbf{Pr}^{\text{diff}}[B_l]. \tag{7.6}$$

So, from equations (7.5) and (7.6), in order to calculate $\mathbf{Pr}^{\text{same}}[B_l]$ and $\mathbf{Pr}^{\text{diff}}[B_l]$ it suffices to prove the following theorem.

Theorem 7.1 $\mathbf{Pr}^{\text{same}}[B_l] - \mathbf{Pr}^{\text{diff}}[B_l] = (-1)^l(\tfrac{1}{k})^{l-1}$.

Proof: Define the following events A_t and B_t for $t \leq l$. Notice that this definition of B_t coincides with the previous definition of B_l for $t = l$.

- For $t \leq l$, let A_t be the event that each pair $\langle x, v_1 \rangle, \langle v_1, v_2 \rangle, \ldots, \langle v_{t-2}, v_{t-1} \rangle$ consists of vertices chosen in different color classes.

- For $t \leq l$, let B_t be the event that A_t occurs *and* in addition vertex v_{t-1} is chosen in a different color class from y.

Also, for convenience, let $A_t - B_t$ be the event that A_t occurs and B_t does not. Notice that since $B_t \subseteq A_t$, we have $\mathbf{Pr}[A_t - B_t] = \mathbf{Pr}[A_t] - \mathbf{Pr}[B_t]$. Also note that event A_t does not depend on whether x and y are chosen in the same or different color class.

The probability of event A_t is easy to calculate: we just need v_1 a different color from x, v_2 a different color from v_1, and so on up to v_{t-1}. Thus:

$$\mathbf{Pr}[A_t] \quad = \quad (1 - 1/k)^{t-1}. \tag{7.7}$$

For $t = 2$, event B_2 is the event that v_1 is a different color from both x and y, so $\mathbf{Pr}^{\text{same}}[B_2] = 1 - 1/k$ and $\mathbf{Pr}^{\text{diff}}[B_2] = 1 - 2/k$. For $t > 2$, event B_t occurs if either: (1) B_{t-1} occurs *and*

v_{t-1} is of one of the $k - 2$ colors not used in y or v_{t-2}, or (2) event $A_{t-1} - B_{t-1}$ occurs *and* v_{t-1} is of one of the $k - 1$ colors not used in y or v_{t-2}. Thus,

$$\begin{aligned}
\mathbf{Pr}^{\text{same}}[B_t] &= \mathbf{Pr}^{\text{same}}[B_{t-1}](1 - 2/k) + \Big(\mathbf{Pr}[A_{t-1}] - \mathbf{Pr}^{\text{same}}[B_{t-1}]\Big)(1 - 1/k) \\
&= (1 - 1/k)^{t-1} - \tfrac{1}{k}\mathbf{Pr}^{\text{same}}[B_{t-1}].
\end{aligned}$$ (7.8)

Similarly, $\quad \mathbf{Pr}^{\text{diff}}[B_t] = (1 - 1/k)^{t-1} - \tfrac{1}{k}\mathbf{Pr}^{\text{diff}}[B_{t-1}].$ (7.9)

Thus, we can solve for $\mathbf{Pr}^{\text{same}}[B_l]$ as follows.

$$\begin{aligned}
\mathbf{Pr}^{\text{same}}[B_l] &= (1 - 1/k)^{l-1} - \tfrac{1}{k}\Big[(1 - 1/k)^{l-2} - \tfrac{1}{k}\mathbf{Pr}^{\text{same}}[B_{l-2}]\Big] \\
&= (1 - 1/k)^{l-1} - \tfrac{1}{k}(1 - 1/k)^{l-2} + \tfrac{1}{k^2}\Big[(1 - 1/k)^{l-3} - \tfrac{1}{k}\mathbf{Pr}^{\text{same}}[B_{l-3}]\Big] \\
&\qquad \vdots \\
&= (1 - 1/k)^{l-1} - \tfrac{1}{k}(1 - 1/k)^{l-2} + \tfrac{1}{k^2}(1 - 1/k)^{l-3} - \\
&\qquad \ldots + (-1)^{l-2}\left(\tfrac{1}{k}\right)^{l-2}\mathbf{Pr}^{\text{same}}[B_2].
\end{aligned}$$ (7.10)

Similarly,

$$\mathbf{Pr}^{\text{diff}}[B_l] = (1 - 1/k)^{l-1} - \tfrac{1}{k}(1 - 1/k)^{l-2} + \ldots + (-1)^{l-2}\left(\tfrac{1}{k}\right)^{l-2}\mathbf{Pr}^{\text{diff}}[B_2].$$ (7.11)

From equations (7.10) and (7.11), we have:

$$\begin{aligned}
\mathbf{Pr}^{\text{same}}[B_l] - \mathbf{Pr}^{\text{diff}}[B_l] &= (-1)^{l-2}(\tfrac{1}{k})^{l-2}\Big[\mathbf{Pr}^{\text{same}}[B_2] - \mathbf{Pr}^{\text{diff}}[B_2]\Big] \\
&= (-1)^l(\tfrac{1}{k})^{l-2}[(1 - 1/k) - (1 - 2/k)] \\
&= (-1)^l(\tfrac{1}{k})^{l-1}.
\end{aligned}$$ (7.12)

Thus, we have proven the theorem. ∎

By Theorem 7.1 and equation (7.4), we have $\lambda_l = [(-1)^l/k^{l-1}]/\mathbf{Pr}[B_l]$, so:

$$\lambda_l \geq (-1)^l/k^{l-1}.$$ (7.13)

Thus, for l and k both constant, we have λ_l bounded away from 0 by some constant > 0. Also, note by equations (7.2) and (7.3) that for $p \geq n^{-1+\epsilon}$ for some constant $\epsilon > 0$, for sufficiently large integer l (in fact $l \geq \lceil 2/\epsilon \rceil$), we have $E_l^{\text{same}}, E_l^{\text{diff}} = \Omega(n)$.

7.1.2 Analysis and the l-path algorithm

Note the following property of paths of constant length l between fixed vertices x and y. The number of edges in the path divided by the number of "non-rooted" vertices (that is,

vertices not including x and y) is $l/(l-1)$. For any proper subgraph S of such a path, the quantity: $|E(S)|/|V(S) - \{x,y\}|$ is strictly less. Because this ratio of edges to non-rooted vertices is strictly less for all proper subgraphs, we say that paths between x and y are "strictly-balanced". (A definition of "strictly balanced" for more general "rooted graphs" appears in Definition 8.3 of Chapter 8.)

Spencer [35] proves that for any such strictly balanced graph and any constants $\delta, c > 0$, if the expected number of copies of the graph in $G \leftarrow \mathcal{G}(n,p)$ is at least $K \log n$ for sufficiently large K, then the actual number of copies of the graph in $G \leftarrow \mathcal{G}(n,p)$ will be within $(1+\delta)$ of the expectation with probability $1 - o(n^{-c})$. In Appendix B, we prove a slightly weaker (and simpler to prove) analog of Spencer's theorem for the model $\mathcal{G}(n,p,k)$. A special case of the analog is the following. Let $\text{Num}_l(G)$ be the number of paths of length l between x and y in G.

Corollary 7.2 (Corollary to Theorem B.2) *For any constants $\delta, c > 0$, if l and p are such that $K \log n \leq E_l^{\text{same}}(p), E_l^{\text{diff}}(p) \leq n^{\epsilon^*}$ for sufficiently large K and sufficiently small ϵ^*, then for $G \leftarrow \mathcal{G}(n,p,k)$:*

1. $\mathbf{Pr}^{\text{same}}\Big[(1-\delta)E_l^{\text{same}} < \text{Num}_l(G) < (1+\delta)E_l^{\text{same}}\Big] \geq 1 - o(n^{-c})$,

2. $\mathbf{Pr}^{\text{diff}}\Big[(1-\delta)E_l^{\text{diff}} < \text{Num}_l(G) < (1+\delta)E_l^{\text{diff}}\Big] \geq 1 - o(n^{-c})$.

So, if the expected number of paths is sufficiently large, but not too large, then we can be assured that with probability $1 - o(n^{-2})$, the number of paths between x and y will be close to the expectation.[1]

For convenience, since $\lambda_l \leq 1$, define constant $\epsilon' > 0$ so that:

$$E_l \in [n^{\epsilon'}, 2n^{\epsilon'}] \quad \Longrightarrow \quad E_l^{\text{same}}, E_l^{\text{diff}} \in [K \log n, n^{\epsilon^*}]. \tag{7.14}$$

By equation (7.13), for l constant, the value $|\lambda_l|$ is at least some constant greater than 0. Also, as noted at the end of Section 7.1.1, for $p = n^{-1+\epsilon}$ for any constant $\epsilon > 0$, there exists integer l such that $E_l^{\text{same}}, E_l^{\text{diff}} = \Omega(n)$. We want l such that $E_l \in [n^{\epsilon'}, 2n^{\epsilon'}]$ but such an integer l might not exist. We can handle this difficulty by noting the following fact.

Fact 7.1 *Let $\mathcal{G}_q(n,p,k)$ be the model such that we first select $G \leftarrow \mathcal{G}(n,p,k)$ and then delete each edge with probability q. Then, $\mathcal{G}_q(n,p,k) = \mathcal{G}(n,p(1-q),k)$.*

That is, if we delete each edge in graph $G \leftarrow \mathcal{G}(n,p,k)$ with some probability q, then the distribution obtained is exactly the same as if we had just put each edge into the graph with

---

[1]In fact, the restriction that $E_l^{\text{same}}, E_l^{\text{diff}} \leq n^{\epsilon^*}$ is most certainly not necessary. We leave for future work to show that Spencer's theorem goes through for the expectation greater than n^{ϵ^*} as well.

probability $p(1 - q)$ in the first place. So, given a graph $G \leftarrow \mathcal{G}(n, p, k)$ and a value l such that $E_l > 2n^{\epsilon'}$, if we delete edges at random from G with probability q so that the expected number of paths between x and y is between $n^{\epsilon'}$ and $2n^{\epsilon'}$, then we can apply Corollary 7.2 to the resulting graph. We now present the algorithm l-path.

Algorithm l-path

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

Output: *A k-coloring of G or else failure.*

1. *Let d_{avg} be the average degree in G and let $\hat{p} = \frac{d_{\text{avg}}}{n(1-1/k)}$.*

   (So, if $G \leftarrow \mathcal{G}(n, p, k)$ for $p = n^{-1+\epsilon}$ then with high probability, $\hat{p} = p[1 + o(1)]$.)

   Pick l such that $\hat{p}^l n^{l-1} = \Omega(n)$ and let $\hat{\lambda} = 1/k^{l-1}$.

2. *Randomly delete each edge in G with probability q so that $E_l(\hat{p}(1 - q)) \in [\frac{4}{3}n^{\epsilon'}, \frac{5}{3}n^{\epsilon'}]$ where ϵ' is such that Corollary 7.2 holds for $c = 2$ and $\delta = \hat{\lambda}/4$, and $E_l(\hat{p}(1 - q))$ is calculated using equations (7.1) and (7.5).*

   Let $E_l^{\text{same}} = E_l^{\text{same}}(\hat{p}(1 - q))$, calculated using equations (7.2) and (7.10).

3. *For $i = 1$ to k do:*

   (a) *Pick an arbitrary uncolored vertex x and let S_i be the set containing x and all vertices with a number of paths of length l to x in the range:*

   $$[(1 - \hat{\lambda}/3)E_l^{\text{same}}, (1 + \hat{\lambda}/3)E_l^{\text{same}}].$$

   If the set S_i is not independent or S_i contains previously-colored vertices, then halt with failure.

   (b) *If S_i is independent, then assign color i to all vertices of S_i.*

4. *If in Step 3 we assigned one of k colors to each vertex in the graph, then halt with success. If we did not color each vertex, then halt with failure.*

Theorem 7.3 *Algorithm l-path k-colors graphs $G \leftarrow \mathcal{G}(n, p, k)$ with high probability for $p \geq n^{-1+\epsilon}$ for any constant $\epsilon > 0$.*

Proof: Let $C_1, \ldots, C_k$ be the sets of vertices in each color class in the creation of graph G in model $\mathcal{G}(n, p, k)$. Let us say that Step 3 *succeeds* in iteration i if the set S_i created equals C_j for some $1 \leq j \leq k$.

In step 1, as noted, with high probability $\hat{p} = p[1 + o(1)]$, and let us for convenience assume now that this is the case. So, $E_l(\hat{p}(1 - q)) = [1 + o(1)]E_l(p(1 - q))$ and $E_l^{\text{same}}(\hat{p}(1 - q)) = [1 + o(1)]E_l^{\text{same}}(p(1 - q))$. Let $E_l^{\text{same}} = E_l^{\text{same}}(p(1 - q))$, let $E_l^{\text{diff}} = E_l^{\text{diff}}(p(1 - q))$, and let $E_l = E_l(p(1 - q))$.

In Step 2, if $E_l(\hat{p}(1 - q)) \in [\frac{4}{3}n^{\epsilon'}, \frac{5}{3}n^{\epsilon'}]$, then $E_l \in [n^{\epsilon'}, 2n^{\epsilon'}]$ and Corollary 7.2 applies. Thus, by Corollary 7.2 and since δ is chosen sufficiently small so that $|E_l^{\text{same}} - E_l^{\text{diff}}| > 2\delta$, we have the following. With probability $1 - n^2[o(n^{-2})] = 1 - o(1)$, for every pair of vertices x, y in the same color class C_j, and for no pairs x, y in different color classes, the number of paths of length l between x and y is in the range $[(1 - \delta)E_l^{\text{same}}, (1 + \delta)E_l^{\text{same}}]$. Thus, with high probability, Step 3 succeeds for each iteration i and Algorithm l-path k-colors the graph. ∎

Since l is a constant, counting the number of simple paths of length l between two vertices can be done in polynomial time and so the l-path algorithm runs in polynomial time. The running time of the algorithm could be improved considerably by counting non-simple paths as well as simple paths. It is likely that the bounds claimed by Corollary 7.2 can be made to apply for that case as well.

Chapter 8
Semi-random graphs

The results of Turner [38] and Dyer and Frieze [18] mentioned in the last chapter show that random k-colorable graphs, and thus *most* k-colorable graphs, are easy to k-color. Random k-colorable graphs, however, tend to be of a very special type. For instance, with high probability all vertices have nearly the same degree and all have nearly the same number of edges to each of the other $(k - 1)$ color classes. So, graphs created in only a "somewhat random" manner may not be colored well by algorithms for $\mathcal{G}(n, p, k)$. On the other hand, worst-case assumptions may be overly pessimistic in many situations. To analyze the coloring of graphs in an intermediate range, we consider here two new graph models that lie in between the random and worst-case models. These new models provide a smooth transition between the random and worst-case scenarios and are based on a notion of a "semi-random source" from the cryptographic literature. We will call these models the "semi-random" graph models.

8.1   Basic definitions and statement of results

We define here two graph models both based on the *semi-random* source (also called a "slightly-random" source) of Santha and Vazirani [34] (see also [40] [39] [17]). In the first model, which we denote $\mathcal{G}_S(n, p, k)$, the graph is generated as follows. First, an adversary splits the n vertices into k color classes (for $k = 3$, we denote these classes by red, blue, and green). Then for each pair of vertices u, v where u and v belong to different color classes (running through such pairs in an order of its choosing), the adversary decides whether or not to include edge (u, v) in the graph. Once the adversary has made a choice for a particular edge (u, v), the choice is then reversed with probability p. Note that later choices of the adversary may depend on the outcomes of earlier decisions, as in the Santha-Vazirani source [34]. An alternative way to view this model, and closer to the point of view used by Santha-Vazirani is the following. For each pair of vertices u, v belonging to different color classes, the adversary picks a bias p_{uv} between p and $1 - p$ of a coin which is then flipped to determine whether edge (u, v) is placed in the graph. The adversary may determine the bias p_{uv} based on the outcome of previous coin tosses. The two views of the model are

equivalent: if the adversary in the first description is deterministic, then it can be thought of as selecting $p_{uv} \in \{p, 1 - p\}$; if it is randomized, it can act as if selecting intermediate values. We call p the *noise rate* of the source and this model, the *semi-random* graph model.

The second model we consider is a slightly modified version of the above, differing in that the sizes of the k color classes are required all to be $\Omega(n)$. We call this second model the *balanced* semi-random graph model and denote it by $\mathcal{G}_{SB}(n, p, k)$. Following the notation in Chapter 7, we write:

- $G \leftarrow \mathcal{G}_S(n, p, k)$ or $G \leftarrow \mathcal{G}_{SB}(n, p, k)$

to denote that G is selected according to the corresponding model for *some* unknown adversary. We denote the semi-random and balanced semi-random models for a *fixed* adversary $\mathcal{A}$ by $\mathcal{G}_S^{\mathcal{A}}(n, p, k)$ and $\mathcal{G}_{SB}^{\mathcal{A}}(n, p, k)$ respectively. Formally, we say that an algorithm t-colors $G \leftarrow \mathcal{G}_S(n, p, k)$ with high probability (or t-colors $G \leftarrow \mathcal{G}_{SB}(n, p, k)$ with high probability) if it does so with high probability for any choice of the adversary.

A nearly equivalent way to view the semi-random models is that each edge between vertices of different color classes is actually placed into the graph with probability exactly p, and then an adversary may elect to place additional edges into the graph if it so chooses. This version is perhaps conceptually the most elegant, and an adversary in this version can simulate the adversaries in $\mathcal{G}_S(n, p, k)$ and $\mathcal{G}_{SB}(n, p, k)$, though the converse does not hold. For example, the adversary here could make the graph a complete k-partite graph if it so desired; also, the adversary here may make its decisions after all coin tosses have been performed. While this version could conceivably be more difficult for coloring algorithms than the semi-random graph models as defined above, all the algorithms presented in this chapter work under both conditions.

The semi-random models separate the algorithms for coloring random k-colorable graphs into two categories. Some of the algorithms for the random model [18][23] are highly dependent on facts such as the edge probabilities all being equal and are easily defeated by a semi-random source. Others, such as Turner's No-Choice algorithm [38] adapt well to the semi-random model. In particular, Turner's bound of $p \geq n^{-1/k+\epsilon}$ for k-coloring $G \leftarrow \mathcal{G}(n, p, k)$ holds in the balanced semi-random model as well.

We present first in Section 8.2 an algorithm that achieves the same bound as Turner's algorithm but with significantly simpler analysis (and for 3-colorable graphs holds in the slightly more general $\mathcal{G}_S(n, p, 3)$ model). We then, in Sections 8.3 and 8.4, present an algorithm with better bounds for the balanced model. This algorithm 3-colors graphs in $\mathcal{G}_{SB}(n, p, 3)$ with high probability for $p \geq n^{-0.6+\epsilon}$, and more generally for k-colorable graphs works for $p \geq n^{\lceil \frac{-2k}{(k+1)k-2} \rceil + \epsilon}$. The algorithm of Sections 8.3 and 8.4 requires a more involved

analysis, and the use of the Janson inequality for estimating probabilities of "almost" independent events. In Section 8.5 we present some relationships between the coloring problems in the balanced and unbalanced semi-random models.

For convenience, we make the following definition.

Definition 8.1 *Let* $G \leftarrow \mathcal{G}_S(n, p, k)$ *or* $G \leftarrow \mathcal{G}_{SB}(n, p, k)$. *The pair* (u, v) *is a* **potential edge** *in* G *if* u *and* v *belong to different color classes in the adversary's color scheme.*

For a subgraph H of G or a subset U of $V(G)$, we will use *colors*(H) and *colors*(U) to denote the set of color classes of G that are represented in the subgraph or subset.

8.2 A first algorithm

We now consider the models $\mathcal{G}_S(n, p, 3)$ and $\mathcal{G}_{SB}(n, p, 3)$ of a 3-colorable graph generated by a semi-random source. Although for small constant noise rates p, say $p = 0.01$, it appears at first that the adversary has a good deal of power to defeat a coloring algorithm, it turns out that it does not. As previously mentioned, Turner's algorithm [38] actually 3-colors such a graph with high probability for any $p \geq n^{-1/3+\epsilon}$ for constant $\epsilon > 0$.

We present first a different algorithm that achieves the same bound as Turner's, but works for the unbalanced case $\mathcal{G}_S(n, p, 3)$ as well and has a much simpler analysis. We then present a straightforward improvement and a natural extension of this algorithm for k-colorable graphs (for constant k) for the balanced model.

The idea for the simplest algorithm is the following. If in the adversary's color scheme $u \in$ **blue** and $v \in$ **green**, then the shared neighborhood $S = N(u) \cap N(v)$ contains only **red** vertices. Thus, $N(S) \subseteq$ **blue** $\cup$ **green**. For $p \geq n^{-1/3+\epsilon}$ we show that with high probability, $N(S)$ actually equals the entire set **blue** $\cup$ **green**. So, given u and v, one can split G into a 2-colorable portion $N(S)$ and an independent portion $V - N(S)$ and thus 3-color the graph.

Algorithm Two-Stage

> **Given:** *A graph* $G = (V, E)$.
>
> **Output:** *Either a 3-coloring of* G *or failure.*
>
> > *1. First try to 2-color* G. *If that works, halt with success. Otherwise, do the following:*
> >
> > *2. For each pair of vertices* u, v *(think of* u *as a candidate green node and* v *as a candidate blue node),*
> >
> > > *(a) Let* $S = N(u) \cap N(v)$.

(b) Let $T = N(S)$.

If T is 2-colorable and $V - T$ is an independent set, then color T blue and green, color $V - T$ red and halt. Otherwise go to the top of the loop with a different pair u, v.

3. If Step 2 did not succeed for any pair u, v, then halt with failure.

Theorem 8.1 (weak version) *Algorithm* **Two-Stage** *3-colors $G \leftarrow \mathcal{G}_S(n, p, 3)$ with high probability (over the coin tosses of the semi-random source) for $p \geq n^{-1/3+\epsilon}$ and constant $\epsilon > 0$.*

Proof: For convenience, let red be the color with the most vertices in the adversary's 3-coloring. If there are either no blue or no green vertices, then we will 2-color the graph at the start. Otherwise, let u be a green vertex and v a blue vertex (in the adversary's 3-coloring). Then, the set $S = N(u) \cap N(v)$ contains only red vertices and so set $T = N(S) \subseteq$ blue$\cup$green. We now prove that with high probability, for $p \geq n^{-1/3+\epsilon}$, set T contains *all* the blue and green vertices.

If we view the semi-random source as choosing biases $p_{uv} \in [p, 1 - p]$, then the sizes of sets S and T are minimized when each p_{uv} equals p. In that case, every vertex in red independently has a probability p^2 of belonging to S. So, using Chernoff bounds, $|S| \geq \frac{1}{2}|\text{red}|p^2 = \Omega(np^2)$ with high probability. Now, each vertex $z \in$ blue $\cup$ green such that $z \notin \{u, v\}$ has a probability $(1 - p)^{|S|}$ of *not* belonging to T. The reason is that for $z \notin \{u, v\}$, for each $w \in$ red, the events $A_{z,w}$ that edge (z, w) appears in the graph occur with probability p and are independent of each other and of the choice of S. So, we have:

$$\mathbf{Pr}[z \notin T] \leq e^{-p|S|} = e^{-\Omega(np^3)} = e^{-\Omega(n^{3\epsilon})} = o(1/n).$$

That is, with high probability *all* vertices $z \in$ blue $\cup$ green belong to T. Thus, with high probability, $T =$ blue$\cup$green and $V - T =$ red and so for some pair u, v considered, algorithm **Two-Stage** succeeds. ∎

Note that if the sizes of the color classes are roughly balanced, we can speed up Algorithm **Two-Stage** considerably by choosing the vertices u and v at *random*. For instance, if the sizes of the color classes are all within constant factors, then we have a constant probability of selecting two "good" vertices each time.

Algorithm **Two-Stage** fails when p falls below $n^{-1/3}$ because then the vertices $u \in$ green and $v \in$ blue may not share enough neighbors for $N(S)$ to equal blue $\cup$ green. However, for p below $n^{-1/3}$, set S might still contain many vertices, and applying additional iterations of the neighbor-taking process can then boost its size if the sizes of the blue, green, and
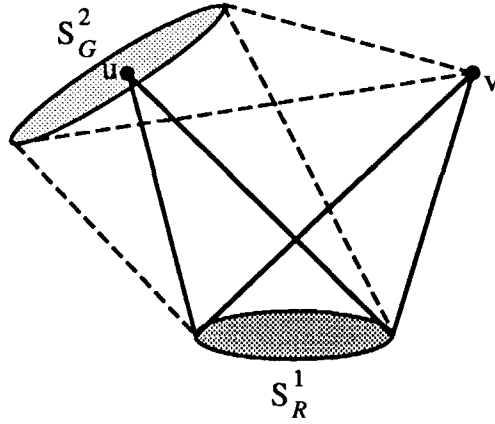
Figure 8.1: Vertices u and v and sets S_R^1 and S_G^2.

red vertex sets are roughly balanced. In fact, we can consider the following straightforward extension of Algorithm Two-Stage that works in the *balanced* semi-random model. (See Figure 8.1.)

Algorithm t-Stage

Given: A graph $G = (V, E)$, and integer t.

Output: Either a 3-coloring of G or failure.

For each edge (u, v):

1. *Let $S_G^1 = \{u\}$, $S_B^1 = \{v\}$, and $S_R^1 = N(S_G^1) \cap N(S_B^1)$.*

2. *Let $S_G^2 = N(S_B^1) \cap N(S_R^1)$, $S_B^2 = N(S_G^1) \cap N(S_R^1)$, and $S_R^2 = N(S_G^2) \cap N(S_B^2)$.*

3. *Let $S_G^3 = N(S_B^2) \cap N(S_R^2)$, $S_B^3 = N(S_G^2) \cap N(S_R^2)$, and $S_R^3 = N(S_G^3) \cap N(S_B^3)$.*

$\vdots$

t. *Let $T = N(S_R^{t-1})$.*

   If T is 2-colorable and $V - T$ is an independent set, then color T blue and green, color $V - T$ red and halt. Otherwise go to the top of the loop with a different edge (u, v).

If we have not succeeded in Step t for any edge (u, v), then halt with failure.

For the balanced model, we have the following stronger version of Theorem 8.1.

Theorem 8.1 (strong version)   *Algorithm t-Stage will 3-color $G \leftarrow \mathcal{G}_{SB}(n, p, 3)$ with high probability for $p \geq n^{-1/2+\epsilon}$, $\epsilon > 0$ constant, and $t > \log_3(1/\epsilon)$.*

Algorithm t-Stage is, in fact, very similar to Turner's No-Choice algorithm, and his algorithm should achieve this stronger bound as well for $\mathcal{G}_{SB}(n, p, 3)$. The algorithm presented here, while more complicated an algorithm, is easier to analyze. However, because we will demonstrate an even better algorithm in the next section, we give here just a proof sketch showing that for $t = 3$, the algorithm will 3-color $G \leftarrow \mathcal{G}_{SB}(n, p, 3)$ for $p \geq n^{-5/11+\epsilon}$, and more briefly describe how this is extended.

Proof sketch: Again, if u is **green** and v is **blue** then for all i, $S_G^i \subseteq$ **green**, $S_B^i \subseteq$ **blue**, and $S_R^i \subseteq$ **red**, and $T \subseteq$ **blue** $\cup$ **green**. For the given value of p, with high probability there will exist an edge between two such vertices u and v. Also, the sizes of the sets S_G^i, S_B^i, S_R^i and T are minimized by the semi-random source that chooses each p_{uv} to equal p. One final fact to note is that for each $i \geq 1$, $S_G^i \subseteq S_G^{i+1}$, $S_B^i \subseteq S_B^{i+1}$, and $S_R^i \subseteq S_R^{i+1}$. The general argument now is just repeated application of bounds for large deviations, being somewhat careful about independence. For this proof sketch, we focus on the case where $t = 3$ and show that algorithm t-Stage will 3-color $G \leftarrow \mathcal{G}_{SB}(n, p, 3)$ for $p = n^{-5/11+\epsilon}$ with high probability. Recall that for $G \leftarrow \mathcal{G}_{SB}(n, p, 3)$ the sizes of the sets **red**, **blue**, and **green** are all $\Theta(n)$.

We can imagine that the coin deciding the presence of an edge is not flipped until we actually examine that edge. So, we first examine all edges (u, w) and (v, w) for $w \in$ **red** and find that almost surely $|S_R^1| = \Theta(|\textbf{red}|p^2) = \Theta(np^2)$. Next, for each $z \in$ **green**, we examine the edges (z, w) for $w \in S_R^1$ and the edge (z, v). For $z \neq u$, these are all previously unexamined edges. So, for $z \in$ **green** $- \{u\}$ we have:

$$
\begin{aligned}
\mathbf{Pr}[z \in S_G^2] &= p(1 - (1-p)^{|S_R^1|}) \\
&= p^2|S_R^1|(1 + o(1)). \qquad \text{(using } p|S_R^1| = o(1))
\end{aligned}
$$

Thus with high probability, $|S_G^2| = \Theta(|\textbf{green}|np^4) = \Theta(n^2p^4)$ and similarly we have $|S_B^2| = \Theta(n^2p^4)$. Now, for each $z \in$ **red** $- S_R^1$ we examine the edges (z, w) and (z, w') for $w \in S_G^2 - S_G^1$ and $w' \in S_B^2 - S_B^1$. Again, these are all previously unexamined edges, so the same argument as above shows that the probability z belongs to S_R^2 is proportional to $p^2|S_G^2 - S_G^1||S_B^2 - S_B^1|$. Thus with high probability, $|S_R^2 - S_R^1| = \Theta(n^5p^{10})$. Finally, we have $T = N(S_R^2)$. Notice that set T contains $S_G^2 \cup S_B^2$ and that for each vertex $z \in$ (**blue** $\cup$ **green**) $- (S_G^2 \cup S_B^2)$, we have not yet examined the edges (z, w) for $w \in S_R^2 - S_R^1$. So, for each such vertex z,

$$
\begin{aligned}
\mathbf{Pr}[z \notin T] &\leq (1-p)^{|S_R^2 - S_R^1|} \\
&= (1-p)^{\Theta(n^5p^{10})} \\
&\leq e^{-\Theta(n^5p^{11})} \\
&= o(1/n), \quad \text{for } p = n^{-5/11+\epsilon}.
\end{aligned}
$$

So, with high probability, $T =$ blue $\cup$ green.

More generally, for $p = n^{-1/2+\epsilon}$, if $|S_R^i| = \Theta(n^{x_i} p^{2x_i})$ we will have with high probability that $|S_R^{i+1}| = \Theta(n^{x_{i+1}} p^{2x_{i+1}})$ for $x_{i+1} = 3x_i + 2$, so long as $n^{x_{i+1}} p^{2x_{i+1}} = o(n)$. Since we begin with $|S_R^1| = \Theta(n^{2\epsilon})$ and at each step the size of S_R^i more than triples, we can continue with the assumption that $n^{x_{i+1}} p^{2x_{i+1}} = o(n)$ for at most $\log_3(1/\epsilon)$ iterations. Suppose i and p are such that $n^{x_{i+1}} p^{2x_{i+1}} \neq o(n)$. Since the probability that Algorithm t-Stage succeeds can only increase with larger p, we may for purposes of analysis decrease p so that $n^{x_{i+1}} p^{2x_{i+1}} = \sqrt{n}$. We will thus have for $z \in ($blue $\cup$ green$) - (S_B^i \cup S_G^i)$ that: $\mathbf{Pr}[z \notin T] \leq (1 - p)^{\Theta(\sqrt{n})} \leq e^{-\Theta(\sqrt{n}p)} = o(1/n)$. So, set T equals blue $\cup$ green with high probability. ∎

It is interesting to note that algorithm **Two-Stage** (or t-**Stage**) extends naturally to graphs in $\mathcal{G}_{SB}(n,p,k)$ for constant $k > 3$. The idea is that instead of selecting two vertices u, v at the start, to select $k - 2$ sets: $U_2, \ldots, U_{k-1}$, each U_i of i vertices, at the start. For some such choice, the vertices of U_{k-1} are all of different colors in $colors(G)$ and the vertices of U_{k-2} are all of different colors in $colors(U_{k-1})$ and so forth. (That is, the vertices of U_{i-1} are all of different colors, and each is of a color used in U_i.) So, for $T_k = V(G)$ and $U_i = \{u_i^1, \ldots, u_i^i\}$, for each $i \in \{2, \ldots, k\}$ simply let

$$T_{i-1} = N_{T_i}(N_{T_i}(u_i^1) \cap \ldots \cap N_{T_i}(u_i^i)),$$

where $N_{T_i}(X) = N(X) \cap T_i$. With high probability, for $p \geq n^{-1/k+\epsilon}$, we will be able to assign one color to each set $T_i - T_{i-1}$ for $i \geq 3$ and two colors to the set T_2, and thus k-color the graph. This yields the same bounds as those achieved by Turner. Again, we will not go into the analysis in detail because in the next section, we show how a quite different idea can be used to get even better bounds.

8.3 A better algorithm: $k = 3$

We now describe a different style of algorithm that improves upon the above bound in the balanced case, to 3-color graphs $\mathcal{G}_{SB}(n,p,3)$ with high probability for $p \geq n^{-0.6+\epsilon}$. The algorithm, while quite simple, requires a more involved probabilistic analysis than the previous one. In particular, we will need to use the Janson inequality [11] to bound probabilities of "nearly" independent events based on pairwise dependencies.[1]

The algorithm is based on the following simple observation. If in a 3-colorable graph G there are two vertices x and y both adjacent to a pair of vertices u and v that are adjacent to each other, then x and y must be the same color in any legal 3-coloring. We call the subgraph induced by $\{x, u, v, y\}$ a *link* between x and y. (See Figure 8.2).

---

[1]The results described in this section and Section 8.4 are based on work joint with Joel Spencer.
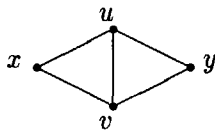
Figure 8.2: A link between x and y.

At first glance, it would seem the above observation does not help, since for fixed vertices x and y, the probability there exists a link between x and y is at most $O(n^2 p^5)$. (There are $O(n^2)$ possible pairs $\langle u, v \rangle$ and for each pair the probability all necessary edges exist is p^5.) Thus, the probability there is a link between x and y is much less than 1 even for $p = o(n^{-0.4})$.

The key fact to note, however, is that we do not need a link between every pair of, say, red vertices x and y. All we need is that for each such pair there is a *sequence* of links between x and some x', between x' and some x'', and so forth, until eventually at some point we reach y. We will call such a sequence a "chain".

Another way to think of this observation is that given a graph G we can create a new graph H as follows. The vertex set $V(H)$ equals $V(G)$, and if x and y are connected by a link in G, we put an edge between x and y into H. So, while edges in G exist only between vertices of different color, edges in H exist only between vertices of the *same* color (in G). The "key observation" is then just that in order to easily select the set of red vertices in G, we do not need red to be a *clique* in H, just a connected set. So, the simple algorithm is as follows.

Algorithm Chain

Given: *A graph $G = (V, E)$.*

Output: *A 3-coloring of G or failure.*

1. *Create graph $H = (V, F)$, where*

$$F = \{(x, y) \mid \exists \text{ a link in } G \text{ between } x \text{ and } y\}.$$

2. *Find all connected components in H. If there are exactly three, halt with success, producing as output the vertices of the three components labeled as red, blue, and green.*

   Otherwise, if there are more than 3 components, then halt with failure.

8.3.1 Motivation

As mentioned above, each connected component in graph H produced by Algorithm Chain consists of vertices that *must* be the same color under any legal 3-coloring of G. The following sections contain a proof that when $p \geq n^{-0.6+\epsilon}$, with high probability there will be only 3 such components in H. Let us first, however, give a motivational argument, supposing that each edge between two vertices of the same color were placed independently with the same probability into H.

Let $p = n^{-0.6+\epsilon}$ and for simplicity, assume that $\epsilon < 1/5$. Given two vertices x and y of the same color (say red) in G, the expected number of links between x and y in G = $\Theta(n^2 p^5) = \Theta(n^{-1+5\epsilon})$. For $\epsilon < 1/5$, the *probability* there exists a link between x and y, and thus the probability that x and y are are neighbors in H is $\Theta(n^{-1+5\epsilon})$ as well. Thus, if we consider the subgraph in H induced by the set red, the average degree of each vertex is $\Theta(n^{5\epsilon})$. It is well known that in the random graph model $\mathcal{G}(n,p)$, once the average degree exceeds $K \log n$ for sufficiently large K, the graph is connected with high probability. So, if the edges in the red subgraph of H were placed *randomly*, the red set would be a single connected component almost surely since $n^{5\epsilon} \gg K \log n$.

8.3.2 Janson's inequality

Janson's inequality [11] is used in the following setting. Consider a universe U of points and a collection of subsets $X_1, \ldots, X_m$ of U. We now create a new subset S of U by placing each $j \in U$ into S independently with probability p. Let A_i be the event that $X_i \subseteq S$. Janson's inequality bounds the probability that no set X_i is contained within S: that is, the probability that no event A_i occurs. [2]

Define:

- $M = \prod_{i=1}^{m} \mathbf{Pr}[\bar{A_i}]$.

If the X_i were all disjoint, then the events A_i would all be independent and so M would be the probability that no A_i occurs. If the X_i are *not* disjoint, then the events A_i are *not* independent. However, Janson's inequality allows us to bound the probability no X_i is contained in S by looking only at *pairwise* dependencies. In particular, Janson's inequality states that:

$$M \leq \mathbf{Pr}[\text{no } X_i \text{ is contained in } S] \leq M e^{\frac{1}{1-x}\frac{\Delta}{2}} \tag{8.1}$$

where λ is an upper bound on $\mathbf{Pr}[A_i]$ and we define:

---

[2] Janson's inequality works even if the probabilities for each point j are different, so long as the points are placed into S independently. We will not need this fact for our purposes.

- $\Delta = \displaystyle\sum_{\substack{\text{ordered pairs } (i \neq j) \\ X_i \cap X_j \neq \phi}} \mathbf{Pr}[A_i \text{ and } A_j].$

Notice that if $\lambda \leq 1/2$ and $\Delta = o(1)$, then by equation (8.1), $\mathbf{Pr}[\text{no } X_i \text{ is contained in } S]$ $= M(1 + o(1))$. That is, under these two conditions, the probability is within $1 + o(1)$ of what the probability would be had the A_i been independent.

In the study of random graphs Janson's inequality is often used to show that some structure exists with high probability. For example, suppose one wishes to prove that the graph $G \leftarrow \mathcal{G}(n, p)$ contains a triangle with high probability for $p \gg 1/n$. For such a setting, we let U be the set of all edges of the n-clique K_n (thought of as possible edges in G) and have one X_i for each set of three edges corresponding to a triangle. Janson's inequality then provides an upper bound on the probability that *no* triangle is contained in G. In the setting of this thesis, we will use Janson's inequality to prove that in the balanced semi-random model, for sufficiently large noise rate p, for any $x, y \in$ red there will be a chain between x and y with probability at least $1 - o(n^{-2})$.

The following definitions are taken (roughly) from Spencer [35]. [3]

Definition 8.2 *Let H be a graph in which some subset R of its vertices are specified to be "roots" and H has no edges between vertices in R. We will call the pair (R, H) a **rooted graph**, or simply say that H is a rooted graph when R is implicit. Define edges(H) to be the total number of edges in H and nonroots(H) to be the number of vertices in H excluding roots. Define the density of H to be dens$(H) = \text{edges}(H)/\text{nonroots}(H)$.*

We will always consider rooted graphs to be graphs on a constant number of vertices, and examine the number of copies of such graphs in larger n-vertex graphs.

Definition 8.3 *A rooted graph (R, H) is **strictly balanced** if for some constant $\epsilon' > 0$, for every proper subgraph (R, H'), we have dens$(H') \leq$ dens$(H) - \epsilon'$. (By a proper subgraph, we mean that $H' \subset H$.)*

Definition 8.4 *Suppose (R, H) is a rooted graph and $G = (V, E)$ is a graph with $V \supseteq R$. An **image** of H over R in G is a subgraph of G isomorphic to H by a map which is the identity on R. When R is clear from context, we will drop the phrase "over R".*

So, for example, if H is a triangle with a root vertex x, then the images of H over $\{x\}$ in G are all triangles in G containing vertex x.

---

[3] The term "image" used here is essentially the same as an "extension" in Spencer's paper, except he counts maps while we count images of maps.

Definition 8.5 *Suppose (R, H) is some strictly balanced rooted graph and V is a set of n vertices containing R. Let $X_1, \ldots, X_m$ denote all distinct images of H in the clique K_n on V. That is, the X_i are all possible images of H fixing root set R in an n-vertex graph. For some model $\mathcal{M}$ (such as $\mathcal{G}(n, p)$ or $\mathcal{G}_{SB}(n, p, k)$), we define:*

$$\Delta(H, \mathcal{M}) = \sum_{\substack{\text{ordered pairs } i \neq j \\ E(X_i) \cap E(X_j) \neq \phi}} \mathbf{Pr}[X_i \subseteq G \text{ and } X_j \subseteq G \mid G \leftarrow \mathcal{M}].$$

Spencer [36][35] proves the following theorem for random graphs $\mathcal{G}(n, p)$.

Theorem 8.2 (Spencer) *Let (R, H) be a strictly balanced rooted graph on a constant number of vertices with $\mathbf{v} = \text{nonroots}(H)$ and $\mathbf{e} = \text{edges}(H)$. Then there exists $\epsilon^* > 0$ so that if $p \leq n^{-\mathbf{v}/\mathbf{e}+\epsilon^*}$, then $\Delta(H, \mathcal{G}(n, p)) = o(1)$.*

Spencer then uses this fact to prove that with high probability, for $p = n^{-\mathbf{v}/\mathbf{e}+\epsilon^*}$, G will contain some image of H.

We can use Spencer's theorem to prove the following.

Theorem 8.3 *Let $\mathcal{G}^{\mathcal{A}}_{SB}(n, p, k)$ be the semi-random model with an adversary that always elects* not *to place edges into the graph. Let (R, H) be a strictly balanced rooted graph on a constant number of vertices with $\mathbf{v} = \text{nonroots}(H)$ and $\mathbf{e} = \text{edges}(H)$. Then there exists $\epsilon^* > 0$ so that if $p = n^{-\mathbf{v}/\mathbf{e}+\epsilon^*}$, then $\Delta(H, \mathcal{G}^{\mathcal{A}}_{SB}(n, p, k)) = o(1)$.*

Proof: Theorem 8.3 follows immediately from Spencer's theorem (Theorem 8.2). Let $X_1, \ldots, X_m$ denote the images of H in the clique K_n and let A_i be the event that $X_i \subseteq G$. Each edge (x, y) is placed into G with probability *at most* p (either probability 0 if x and y are in the same color class or else probability p if they are in different color classes). Thus, for any pair of events A_i, A_j, we have:

$$\mathbf{Pr}[A_i \wedge A_j \mid G \leftarrow \mathcal{G}^{\mathcal{A}}_{SB}(n, p, k)] \leq \mathbf{Pr}[A_i \wedge A_j \mid G \leftarrow \mathcal{G}(n, p)].$$

For sake of completeness, however, we provide a direct proof here as well following the argument of Spencer [35].

We prove the theorem by considering separately for each fixed value of $s \in \{1, \ldots, \mathbf{v}\}$, the pairs X_i, X_j that share s vertices in common in addition to the roots. Note that if $s = 0$, then the graphs X_i and X_j share no edges and so are not counted in the summation in Definition 8.5. The number of pairs X_i and X_j sharing s vertices in common in addition to the roots is $O(n^{2\mathbf{v}-s})$ since there are $2\mathbf{v} - s$ different vertices and only a constant number of permutations.

Let $\epsilon' > 0$ be a value such that every proper subgraph of H containing the roots has density at most $dens(H) - \epsilon'$ (see the definition of "strictly balanced"). If $s = \mathbf{v}$, then since X_i and X_j are distinct, there must be at least $\mathbf{e} + 1$ edges in $X_i \cup X_j$. For any fixed edge, the probability that edge belongs to G is at most p (it could be smaller, e.g. 0, if the two endpoints are in the same color class in the graph). So, the contribution to the summation from such pairs X_i and X_j is at most: $O(n^{\mathbf{v}} p^{\mathbf{e}+1}) = O(n^{\mathbf{v}+(\mathbf{e}+1)(-\mathbf{v}/\mathbf{e}+\epsilon^*)}) = O(n^{-\mathbf{v}/\mathbf{e}+(\mathbf{e}+1)\epsilon^*}) = o(1)$, for ϵ^* sufficiently small.

Now consider $s < \mathbf{v}$ and fix a pair X_i and X_j sharing s vertices in common in addition to roots. Let S be the subgraph $X_i \cap X_j$; that is, $V(S) = V(X_i) \cap V(X_j)$ and $E(S) = E(X_i) \cap E(X_j)$. Since (R, H) is strictly balanced, we know that $|E(S)|/s \le \mathbf{e}/\mathbf{v} - \epsilon'$ for some $\epsilon' > 0$. So, $|E(X_i) \cup E(X_j)| = 2\mathbf{e} - |E(S)| \ge 2\mathbf{e} - s\mathbf{e}/\mathbf{v} + s\epsilon'$ and thus the probability that both X_i and X_j are subgraphs of G is at most $p^{2\mathbf{e}-s\mathbf{e}/\mathbf{v}+s\epsilon'}$.

Finally, summing over all $O(n^{2\mathbf{v}-s})$ pairs X_i and X_j sharing s vertices in common besides the endpoints, the contribution to Δ is at most:

$$
\begin{aligned}
O(n^{2\mathbf{v}-s} p^{2\mathbf{e}-s\mathbf{e}/\mathbf{v}+s\epsilon'}) &= O(n^{2\mathbf{v}-s}(n^{-\mathbf{v}/\mathbf{e}+\epsilon^*})^{2\mathbf{e}-s\mathbf{e}/\mathbf{v}+s\epsilon'}) \\
&= O(n^{2\mathbf{v}-s-2\mathbf{v}+s-s\epsilon'\mathbf{v}/\mathbf{e}+(2\mathbf{e}-s\mathbf{e}/\mathbf{v}+s\epsilon')\epsilon^*}) \\
&= O(n^{-s\epsilon'\mathbf{v}/\mathbf{e}+(2\mathbf{e}-s\mathbf{e}/\mathbf{v}+s\epsilon')\epsilon^*}) \\
&= o(1) \qquad \text{(for } \epsilon^* \text{ sufficiently small)}.
\end{aligned}
$$

Thus, the contribution to Δ from each value of $s \in [1, \mathbf{v}]$ is $o(1)$, and since there are only a constant number of different choices for s, we have $\Delta(H, \mathcal{G}_{SB}^A(n, p, k)) = o(1)$. ∎

We will use this fact in the next section to prove that in balanced semi-random 3-colorable graphs, with high probability there will exist chains between every pair of vertices x and y in the same color class.

8.3.3   The main theorem

We now prove the following theorem.

Theorem 8.4 *Algorithm* Chain *will 3-color* $G \leftarrow \mathcal{G}_{SB}(n, p, 3)$ *with high probability for* $p \ge n^{-3/5+\epsilon}$ *for any constant* $\epsilon > 0$.

The idea of the proof is to consider chains of some length r between two fixed endpoints (roots) x and y and to prove that with probability $1 - o(n^{-2})$ at least one such chain exists in G. This will be done by showing that chains are strictly balanced and then applying Theorem 8.3 and Janson's inequality.

Before proving Theorem 8.4, however, let us first formally define a chain and prove a few preliminary lemmas.
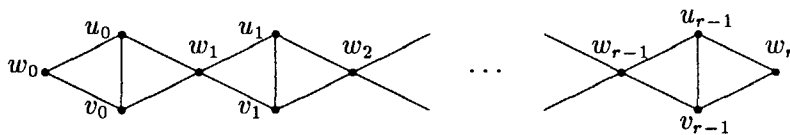
Figure 8.3: A chain of length r between w_0 and w_r.

Definition 8.6 *A* **chain** *C of length r is a rooted graph on $3r+1$ vertices $\{w_0, u_0, v_0, w_1, u_1, v_1,$*
$\ldots, w_{r-1}, u_{r-1}, v_{r-1}, w_r\}$ and $5r$ edges, where:

$$E(C) = \bigcup_{i=0}^{r-1} \{(w_i, u_i), (w_i, v_i), (u_i, v_i), (u_i, w_{i+1}), (v_i, w_{i+1})\}.$$

(See Figure 8.3). Vertices w_0 and w_r are the roots of the chain and will be called the endpoints.

Definition 8.7 *If $G \leftarrow \mathcal{G}_{SB}(n, p, 3)$, we will say that C is a* **potential chain** *between two vertices w_0 and w_r if all w_i are in the same color class, and for each i, vertices u_i, v_i, and w_i are all in different color classes.*

Note that $nonroots(C) = 3r - 1$ and $edges(C) = 5r$, and there are no edges in C between the roots. Also, note that the ratio: $-nonroots(C)/edges(C) = -3/5 + \frac{1}{5r}$, so if $p = n^{-3/5+\epsilon}$ as in the bound for Theorem 8.4, then for C a sufficiently long chain we will have $p = n^{-nonroots(C)/edges(C)+\epsilon^*}$ for some $\epsilon^* > 0$. This is the form of the condition on p in Theorem 8.3 for proving that $\Delta = o(1)$. Our immediate goal is thus to prove that chains are strictly balanced.

Fact 8.8 *If C is a chain of length r, then $dens(C) = \frac{edges(C)}{nonroots(C)} = 5/3 + \frac{5}{3\,nonroots(C)}$.*

The following is a useful fact about subgraphs of chains.

Lemma 8.5 *Let S be a subgraph of a chain C. Then $|E(S)| \le 5/3(|V(S)| - 1)$.*

Proof: Let C have vertex set $\{w_0, u_0, v_0, \ldots, w_{r-1}, u_{r-1}, v_{r-1}, w_r\}$. Let L_i be the ith link in C; that is, $L_i = C|_{\{w_i, u_i, v_i, w_{i+1}\}}$. For convenience, partition the vertices of S into disjoint sets $V_i = V(S) \cap [V(L_i) - \{w_{i+1}\}]$, and partition the edges of S into disjoint sets $E_i = E(S) \cap E(L_i)$, for $0 \le i < r$. So, $S = (\bigcup V_i, \bigcup E_i)$.

For a given index i, if V_i is not empty and $w_{i+1} \in V(S)$, then $|E_i|/|V_i| \le 5/3$. One can easily check that the maximum value of this ratio occurs when E_i and V_i are both "full"

(sizes 5 and 3 respectively). If V_i is non-empty but $w_{i+1} \notin V(S)$, then: if $|V_i| = 3$ we have $|E_i| \leq 3$, if $|V_i| = 2$ we have $|E_i| \leq 1$, and if $|V_i| = 1$ we have $|E_i| = 0$.

Since there must exist *some* i such that V_i is non-empty and $w_{i+1} \notin V(S)$, this implies:

$$
\begin{aligned}
|E(S)| &\leq \max\{\, 5/3\,(|V(S)| - 3) + 3,\; 5/3\,(|V(S)| - 2) + 1,\; 5/3\,(|V(S)| - 1) + 0 \,\} \\
&= 5/3\,(|V(S)| - 1). \quad \blacksquare
\end{aligned}
$$

We can now use Lemma 8.5 to prove that chains are strictly-balanced, and so allow easy application of Janson's inequality.

Lemma 8.6 *Let S be a subgraph of a chain C of some constant length r such that $V(S)$ contains the endpoints w_0 and w_r but $V(S)$ does not contain all the vertices of C. Then, for some constant $\epsilon' = \epsilon'(r) > 0$,*

$$
\frac{edges(S)}{nonroots(S)} \leq \frac{edges(C)}{nonroots(C)} - \epsilon'.
$$

That is, chains are strictly-balanced.

Proof: Since we are giving an upper bound on the number of edges in S, we may as well assume that S is a vertex-induced subgraph.

First, suppose S consists of just one connected component. So, S contains vertices $\{w_0, w_1, \ldots, w_r\}$ and at least one of $\{u_i, v_i\}$ for each $i < r$. Thus, for each vertex in C missing from S, there must be at least 3 edges of C missing from S as well: if $u_i \notin V(S)$ then $(w_i, u_i), (u_i, v_i), (u_i, w_{i+1}) \notin E(S)$. So, if m vertices of C are *missing* from S, then:

$$
\begin{aligned}
\frac{edges(S)}{nonroots(S)} &\leq \frac{edges(C) - 3m}{nonroots(C) - m} \\
&\leq \frac{edges(C)}{nonroots(C)} - \epsilon' \qquad \text{for some } \epsilon' > 0,
\end{aligned}
$$

because $\frac{edges(C)}{nonroots(C)} < 3$ and both $edges(C)$ and $nonroots(C)$ are constant.

If S consists of more than one connected component, then w_0 and w_r cannot be in the same component since we have assumed that S is vertex-induced. We can thus partition S into two disjoint subgraphs: S_{start} and S_{rest} where S_{start} is the component containing w_0 and S_{rest} is everything else (and need not be connected). So, $nonroots(S) = |V(S)| - 2 = |V(S_{start})| + |V(S_{rest})| - 2$. Applying Lemma 8.5, we get:

$$
\begin{aligned}
edges(S) &= |E(S_{start})| + |E(S_{rest})| \\
&\leq 5/3(|V(S_{start})| - 1) + 5/3(|V(S_{rest})| - 1) \\
&= 5/3(|V(S)| - 2) \\
&= (5/3)nonroots(S).
\end{aligned}
$$

So, $\frac{edges(S)}{nonroots(S)} \leq \frac{edges(C)}{nonroots(C)} - \epsilon'$ for $\epsilon' = \frac{5}{3\,nonroots(C)}$, by Fact (8.8). ∎

Proof of Theorem 8.4: First, it is clear from the description of Algorithm Chain that adding additional edges into the graph G cannot decrease the probability of success. Therefore, in order to prove Theorem 8.4, it is enough to prove that Algorithm Chain succeeds when each potential edge is placed into the graph with probability exactly p; that is, the adversary $\mathcal{A}$ always chooses not to place edges into the graph. It is similarly also enough to prove the theorem when p exactly equals $n^{-3/5+\epsilon}$ for some constant $\epsilon > 0$. Let

$$r \in \mathbf{Z}, \quad \hat{\epsilon} > 0 \text{ be constants such that} \quad p = n^{-3/5+\frac{1}{5r}+\hat{\epsilon}}. \tag{8.2}$$

By Lemma 8.6, chains C_r of length r are strictly balanced. So, by Theorem 8.3, there exists $\epsilon^* > 0$ such that if $p \leq n^{-nonroots(C)/edges(C)+\epsilon^*}$ then $\Delta = \Delta(C_r, \mathcal{G}_{SB}^{\mathcal{A}}(n,p,3)) = o(1)$. Because additional edges cannot decrease the probability of success, we may for the purposes of analysis assume that:

$$\hat{\epsilon} \leq \epsilon^*. \tag{8.3}$$

Thus, since $-\frac{nonroots(C)}{edges(C)} = \frac{-3r+1}{5r} = -3/5 + \frac{1}{5r}$, we have by Theorem 8.3 that:

$$\Delta = o(1). \tag{8.4}$$

Fix two points x and y in the same color class in G; without loss of generality, say $x, y \in$ **red**. We now show that with probability $1 - o(n^{-2})$, x and y are connected by a chain of length r, with r as in equation (8.2). This will immediately imply Algorithm Chain succeeds.

In fact, the analysis we provide to show that with high probability there is a chain between x and y, more generally holds in the random model $\mathcal{G}(n,p)$ for any strictly balanced rooted graph H where $p \geq n^{-nonroots(H)/edges(H)+\epsilon^*}$. This general fact is proven by Spencer [36][35]. The proof of Spencer's theorem can be seen to hold in the $\mathcal{G}_{SB}(n,p,k)$ model as well, so long as the number of images of H containing no edges between vertices in the same color class is $\Theta(n^{nonroots(H)})$. This is the case for chains, but is not the case, for example, for non-k-colorable graphs. For completeness, we provide a direct proof for chains along the lines of Spencer's arguments here.

Label each potential chain of length r between x and y as some C_i for $(1 \leq i \leq m)$, where the number of such potential chains is: $m = 2^r(\Theta(n))^{3r-1}(1 - o(1))$ since there are two color choices for each (u,v) pair, there are $\Theta(n)$ vertices in each color class in $\mathcal{G}_{SB}(n,p,3)$, and r is constant. Thus, $m = \Theta(n^{3r-1})$. We bound the probability that x and y are *not* connected by a chain of length r by applying Janson's inequality. The universe U corresponds to the

set of $\Theta(n^2)$ potential edges in G and the sets X_i correspond to the potential chains C_i of length r between x and y, where X_i is the set of all edges in C_i. Every C_i has $5r$ edges, each in G with probability p. So,

$$\begin{aligned}
M &= \prod_{i=1}^{m} \mathbf{Pr}[C_i \not\subseteq G] \\
&= (1 - p^{5r})^{\Theta(n^{3r-1})} \\
&< e^{-p^{5r} \cdot \Theta(n^{3r-1})} \\
&= e^{-\Theta(n^{-3r+1+5r\hat{\epsilon}+3r-1})} \\
&= e^{-\Theta(n^{5r\hat{\epsilon}})} \\
&= o(1/n^2).
\end{aligned} \qquad (8.5)$$

Let us now consider the term $e^{\frac{1}{1-\lambda}\frac{\Delta}{2}}$ in Janson's inequality. For a fixed potential chain C_i, the value $\lambda = \mathbf{Pr}[C_i \subseteq G] = p^{5r} = o(1)$. By our choice of $\hat{\epsilon}$, we have $\Delta = o(1)$ as well (equation 8.4). Thus, $e^{\frac{1}{1-\lambda}\frac{\Delta}{2}} = 1 + o(1)$.

We now apply Janson's inequality using the bound on the above term together with the bound on M in equation (8.5). We thus get that $\mathbf{Pr}[x$ and y are *not* connected by a chain of length r in $G] = M e^{\frac{1}{1-\lambda}\frac{\Delta}{2}} = M(1 + o(1))$, which equals $o(1/n^2)$. So, with high probability, all pairs of vertices from the same color class are connected by some such chain and Algorithm Chain succeeds. ∎

8.4   A better algorithm: general k

We can extend the results of the previous section to graphs of higher chromatic number k. A simple way to do this is just to replace the notion of a "link" by that of a "t-link" defined as follows.

Definition 8.9 *A t-link for some constant t is a $(t + 2)$-vertex graph consisting of two vertices x and y called the* endpoints *both fully connected to a t-clique. (See Figure 8.4).*

Equivalently, a t-link between x and y is a $(t + 2)$-clique with the edge (x, y) removed.

Note that if two vertices in a k-colorable graph are endpoints of a $(k - 1)$-link, then they must be the same color in any legal k-coloring. Using this fact, we can get the following natural generalization of Algorithm Chain to graphs of constant chromatic number $k \geq 3$.

Algorithm t-Chain

> **Given:** $G = (V, E)$, a k-colorable graph.

> **Output:** A k-coloring of G or else failure.
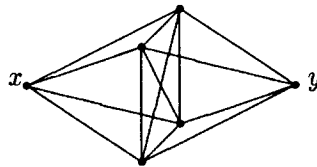
Figure 8.4: A 4-link between x and y.

$k =$	3	4	5	6
p value (fraction)	$n^{-3/5}$	$n^{-4/9}$	$n^{-5/14}$	$n^{-6/20}$
(decimal)	$n^{-0.6}$	$n^{-0.444}$	$n^{-0.357}$	$n^{-0.3}$

Table 8.1: Algorithm t-Chain succeeds with high probability for p at least this value times n^ϵ.

1. *Create graph $H = (V, F)$, where*

$$F = \{(x, y) \mid \exists \text{ a } (k-1)\text{-link in } G \text{ between } x \text{ and } y\}.$$

2. *Find all connected components in H. If there are exactly k components, then halt with success, producing those components as the color classes of G.*

   Otherwise, if there are more than k components, then halt with failure.

Theorem 8.7 *Algorithm t-Chain k-colors $G \leftarrow \mathcal{G}_{SB}(n, p, k)$ for $p \geq n^{\left\lceil \frac{-2k}{(k+1)k-2} \right\rceil + \epsilon}$, $(\epsilon > 0)$ with high probability. (See Table 8.1).*

In order to prove Theorem 8.7, we consider $(k-1)$-chains of some constant length r. We then prove, analogously to the previous section, that $(k - 1)$-chains are strictly-balanced, so Theorem 8.3 applies. We define a t-chain as follows.

Definition 8.10 *A t-chain of length r is a sequence of r t-links connected at their endpoints. For a t-chain C with fixed endpoints x and y, we will treat the chain as a rooted graph, with x and y as the roots.*

Fact 8.11 *If C is a t-chain of length r, then $|V(C)| = r(t+1)+1$, nonroots$(C) = r(t+1)-1$, and $|E(C)| = $ edges$(C) = r\left[\binom{t+2}{2} - 1\right] = \frac{r}{2}[(t+1)(t+2)-2]$. So, $\frac{|E(C)|}{|V(C)|-1} = \frac{1}{2}[t+2-\frac{2}{t+1}] = \frac{t}{2} + \frac{t}{t+1}$.*

Note that if C_r is a $(k-1)$-chain of length r, then the term $\left[\frac{-2k}{(k+1)k-2}\right]$ in the statement of Theorem 8.7 equals $\lim_{r \to \infty} \frac{-nonroots(C_r)}{edges(C_r)}$.

As in the proof of Theorem 8.4, the first step is to prove that t-chains are strictly-balanced. We first prove a fact analogous to Lemma 8.5.

Lemma 8.8 *Let S be a subgraph of a t-chain C of length r. Then:*

$$|E(S)| \leq \left[\frac{t}{2} + \frac{t}{t+1}\right](|V(S)| - 1).$$

Proof:

Let L be a t-link and let $g(t) = \frac{t}{2} + \frac{t}{t+1}$. Define $dens_1(H) = \frac{|E(H)|}{|V(H)|-1}$, so:

$$dens_1(L) = dens_1(C) = g(t). \tag{8.6}$$

Claim 1: If $S \subseteq L$, then $dens_1(S) \leq g(t)$.

Proof of 1:   We may assume S is vertex-induced. Thus, S is either a $(t + 2 - c)$-clique or else S is a $(t - c)$-link for some $c \geq 1$. In the latter case, the claim follows from equation (8.6) since $g(t)$ is an increasing function of t. In the former case, we have $dens_1(S) = \frac{t+2-c}{2} = \frac{t}{2} + 1 - \frac{c}{2} < g(t)$ for $c \geq 1$. $\square$

Now, we prove the lemma that if $S \subseteq C$ then $dens_1(S) \leq g(t)$ by induction on the length of C. The base case is proved in the above claim, so we may assume the lemma holds for any t-chain of length $r - 1$. Let C' be the t-chain of length $r - 1$ consisting of the first $r - 1$ links in C and let S' be S restricted to C'. So, $dens_1(S') \leq g(t)$. Let L be the last link in C and let S_L be S restricted to L. So, $|E(S)| = |E(S')| + |E(S_L)|$ and $|V(S)| \geq |V(S')| + |V(S_L)| - 1$ (note: S' and S_L may share one vertex in common where L joins C'). Thus, $dens_1(S) \leq \frac{|E(S')|+|E(S_L)|}{|V(S')|+|V(S_L)|-2} = \frac{|E(S')|+|E(S_L)|}{[|V(S')|-1]+[|V(S_L)|-1]} \leq g(t)$ by induction and Claim 1.   ∎

Lemma 8.9 *Let S be a subgraph of a t-chain C of some constant length r such that $V(S)$ contains the endpoints x and y but $V(S) \neq V(C)$. Then, for some constant $\epsilon' = \epsilon'(r) > 0$,*

$$\frac{edges(S)}{nonroots(S)} \leq \frac{edges(C)}{nonroots(C)} - \epsilon'.$$

That is, t-chains are strictly-balanced.

Proof: For C a t-chain of length r, we have $edges(C) = \frac{r}{2}[(t+1)(t+2)-2] = \frac{r}{2}[t^2+3t]$. So, we may upper bound the density as follows:

$$
\begin{aligned}
dens(C) &= \frac{\frac{r}{2}[t^2+3t]}{r(t+1)-1} \\
&= \frac{t^2+3t}{2t+2-2/r} \\
&\leq \frac{t+3}{2}.
\end{aligned}
$$

Again, we may assume that S is a vertex-induced subgraph.

First, suppose S consists of just one connected component. So, S contains at least one non-endpoint for each t-link L in C, and contains all link endpoints. Let us focus on some fixed t-link L in C. If S is missing m vertices from that link, then it must be missing *at least* $(t + 1) + t + (t - 1) + \ldots + (t - m + 2)$ edges from that link as well. So, the ratio of (edges missing) to (vertices missing) is at least $\frac{(t+1)+(t-m+2)}{2} \geq \frac{t+4}{2}$, for $m \leq t - 1$, the largest value of m possible. Thus, if there are in total $\hat{m}$ vertices in C missing from S, then

$$
\begin{aligned}
\frac{edges(S)}{nonroots(S)} &\leq \frac{edges(C) - \hat{m}(t + 4)/2}{nonroots(C) - \hat{m}} \\
&\leq \frac{edges(C)}{nonroots(C)} - \epsilon' \qquad \text{for some } \epsilon' > 0,
\end{aligned}
$$

because $\frac{t+4}{2} \geq \frac{1}{2} + dens(C)$.

If S consists of more than one connected component, then the endpoints of C cannot be in the same component since we have assumed that S is vertex-induced. We can thus partition S into two disjoint subgraphs: S_{start} and S_{rest} where S_{start} contains root x and S_{rest} is everything else. So, $nonroots(S) = |V(S_{start})| + |V(S_{rest})| - 2$. Let $g(t) = \frac{t}{2} + \frac{t}{t+1}$. Applying Lemma 8.8, we get:

$$
\begin{aligned}
edges(S) &= |E(S_{start})| + |E(S_{rest})| \\
&\leq g(t)(|V(S_{start})| - 1) + g(t)(|V(S_{rest})| - 1) \\
&= g(t) nonroots(S).
\end{aligned}
$$

So, $\frac{edges(S)}{nonroots(S)} \leq \frac{|E(C)|}{|V(C)|-1} = \frac{edges(C)}{nonroots(C)+1} \leq \frac{edges(C)}{nonroots(C)} - \epsilon'$ for $\epsilon' > 0$. ∎

Proof of Theorem 8.7: As in the proof of Theorem 8.4, we may assume that the adversary $\mathcal{A}$ always elects *not* to place edges into the graph. Let $C = C(r)$ be a $(k-1)$-chain of length r between two fixed vertices x and y. So:

$$
\begin{aligned}
-nonroots(C)/edges(C) &= -(kr - 1)/ \left(\frac{r}{2}[k(k + 1) - 2] \right) \qquad \text{(see Fact 8.11)} \\
&= \frac{-2k}{k(k + 1) - 2} + \frac{2}{r[k(k + 1) - 2]}.
\end{aligned}
$$

Thus, for sufficiently large r, for some $\hat{\epsilon} > 0$, we have $p \geq n^{-nonroots(C)/edges(C)+\hat{\epsilon}}$. By Lemma 8.9 we know C is strictly balanced, so let $\epsilon^* > 0$ be the constant of Theorem 8.3 such that for $p \leq n^{-nonroots(C)/edges(C)+\epsilon^*}$, we have $\Delta = \Delta(C, \mathcal{G}_{SB}^{\mathcal{A}}(n, p, k)) = o(1)$. Because additional edges cannot decrease the probability of success, we may for the purposes of analysis assume that $\hat{\epsilon} \leq \epsilon^*$. That is,

$$
p = n^{-nonroots(C)/edges(C)+\hat{\epsilon}} \quad \text{for some } r \in \mathbf{Z},\ 0 < \hat{\epsilon} < \epsilon^*. \tag{8.7}
$$

We now examine all potential $(k-1)$-chains C_i of length r between at x and y. That is, all images over $\{x, y\}$ in K_n of C, such that the image contains no edges between two vertices in the same color class in the adversary's k-coloring. Since in the balanced semi-random model there are $\Theta(n)$ vertices in each color class and since r is constant, the number of potential $(k-1)$-chains is $m = \Theta(n^{nonroots(C)})$. Because each edge in some such C_i is placed into G with probability p, for any given C_i we have

$$\mathbf{Pr}[C_i \subseteq G] = p^{edges(C)} = n^{-nonroots(C)+edges(C)\hat{\epsilon}}.$$

So:

$$
\begin{aligned}
M &= \prod_{i=1}^{m} \mathbf{Pr}[C_i \not\subseteq G] \\
&= \left(1 - n^{-nonroots(C)+edges(C)\hat{\epsilon}}\right)^{\Theta(n^{nonroots(C)})} \\
&\leq e^{-n^{(-nonroots(C)+edges(C)\hat{\epsilon})}\Theta(n^{nonroots(C)})} \\
&= e^{-\Theta(n^{edges(C)\hat{\epsilon}})} \\
&= o(n^{-2}).
\end{aligned}
$$

Since $\lambda = \mathbf{Pr}[C_i \subseteq G] = p^{edges(C)} = o(1)$ and $\Delta = o(1)$, we have by Janson's inequality that: $\mathbf{Pr}[x$ and y are not connected by a $(k-1)$-chain of length r in $G] = Me^{\frac{1}{1-\lambda}\frac{\Delta}{2}} = M(1 + o(1)) = o(1/n^2)$. So, with high probability, all pairs of vertices from the same color class are connected by some such chain and Algorithm t-Chain succeeds. ∎

8.5 Relating the balanced and unbalanced models

For graphs of chromatic number 3, we had fairly good performance bounds even in the unbalanced model. However, for graphs of higher chromatic number, the algorithms required the number of vertices in each color class to be roughly balanced. The reason that the unbalanced case is harder is that if a color class is very small, then the noise rate p *as a function of the number of vertices* is dramatically lower. So, if $(k-1)$ color classes each are small, the algorithm is essentially required to solve a problem for a much lower noise rate on the $(k-1)$-chromatic graph defined by those colors. In particular, one gets the following theorem.

Theorem 8.10 *If BPP $\not\supseteq$ NP, then for $k \geq 4$ there is no polynomial-time algorithm for k-coloring graphs in $\mathcal{G}_S(n, p, k)$ with high probability, for $p = n^{-\epsilon}$ for any constant $\epsilon > 0$.*

Proof: Suppose otherwise; that is, there exists an algorithm $\mathcal{B}$ for k-coloring graphs in $\mathcal{G}_S(n, p, k)$ for $p = n^{-\epsilon}$ for some constant $\epsilon > 0$ where $k \geq 4$. We show how to use $\mathcal{B}$ to

optimally color an *arbitrary* $(k-1)$-colorable graph in probabilistic polynomial time. Note that for $k \geq 4$, the problem of optimally coloring $(k-1)$-colorable graphs is NP-hard.

Let $G = (V, E)$ be a $(k-1)$-colorable graph on n-vertices. We create a new N-vertex graph $H = (V \cup V', F)$ where V' is a set of vertices of size $n^{3/\epsilon}$ disjoint from V, and $N = n + n^{3/\epsilon}$ as follows. For each pair $u, v \in V$, if $(u, v) \in E$ then let (u, v) be an edge in F as well. Also, independently for each pair $v \in V$ and $v' \in V'$, let (v, v') be an edge of F with probability $1 - p$ for $p = N^{-\epsilon}$. Note that there are no edges in F between vertices in V'. Now, feed graph H to algorithm $\mathcal{B}$. If $\mathcal{B}$ k-colors H, then with high probability it must assign at most $k-1$ colors to V and therefore $(k-1)$-color G. The reason is that otherwise there are k vertices in V all given different colors by $\mathcal{B}$, and with probability $(1-p)^k > 1 - kp = 1 - o(1)$, any given vertex in V' is connected to all k of them (and in fact with *extremely* high probability, there will be *some* such vertex in V'). This forces $(k+1)$ colors to be used in H.

The main point of the proof is that an adversary with noise rate $p = N^{-\epsilon}$ can create a k-colorable graph on N vertices in a distribution indistinguishable from that used to create H. In particular, as above, the adversary separates the N vertices into one set V of n vertices and $k-1$ colors, and one set V' of $N-n$ vertices and one color. It then attempts to places edges between vertices in V exactly where they appear in the graph G and to put in all edges between V and V'. Since n is so small (less that $N^{\epsilon/3}$), there are at most $N^{2\epsilon/3}$ potential edges in the set V. So for $p = N^{-\epsilon}$, with probability at least $1 - N^{-\epsilon/3}$ the adversary will be able to place exactly the edges it wishes between vertices in V without any noise at all. Thus, since we assumed that $\mathcal{B}$ can k-color graphs created by such an adversary with high probability, then $\mathcal{B}$ must k-color graph H with high probability as well.

∎

In the balanced model, our best bound for 3-coloring $G \leftarrow \mathcal{G}_{SB}(n, p, 3)$ with high probability is $p \geq n^{-0.6+\epsilon}$. For the random model, we were able to 3-color for p as low as $n^{-1+\epsilon}$. We leave as an open problem whether one can achieve such a low bound on p for the semi-random model as well.

Chapter 9

Lower bounds for independent set approximation based on approximate graph coloring

In this section, we describe a lower bound for independent set approximation (or equivalently clique approximation) based on assumptions about the hardness of approximate graph coloring. Thought of in contrapositive form, we show how to get very good bounds for approximate graph coloring if we are given seemingly weak algorithms for approximating the maximum independent set in a graph. These results are corollaries to a basic technique of Berman and Schnitger [7] which they use to provide weaker lower bounds for independent set approximation based on other hard problems.

Let $is(G)$ denote the size of the largest independent set in graph G. For the Independent Set problem, we define the *performance guarantee* of an algorithm to be the worst-case ratio over all graphs G on n vertices, of $is(G)$ to the size of the independent set found (with high probability if the algorithm is randomized) [12]. So, the lower the performance guarantee, the better the algorithm. The best performance guarantee known for a polynomial-time algorithm for Independent Set is $O(n/(\log n)^2)$ by Boppana and Halldorsson [12].

What we show in this chapter is that if there exists an polynomial-time algorithm with performance guarantee $O(n^{1-\epsilon})$ for Independent Set, then there is a polynomial-time algorithm to color k-colorable graphs with $O(\log n)$ colors and to color $(\log n)$-colorable graphs with polylog(n)-colors. The best algorithm known to date [22] for coloring $(\log n)$-colorable graphs uses more than $n/(\log n)^2$ colors. The best algorithm known for 3-colorable graphs (see Chapters 4 and 5 of this thesis) uses $\tilde{O}(n^{3/8})$ colors. So, this result shows that a performance that seems only somewhat better in approximating independent sets implies being able to do quite significantly better for approximate graph coloring.

9.1 Additional definitions and previous results

Given a maximization (minimization) problem, we say an algorithm is a *polynomial-time approximation scheme* (PTAS) if for any constant $\epsilon > 0$, it runs in probabilistic polynomial time and finds a solution of value within a $(1 + \epsilon)$ factor of the maximum (minimum). For example, consider the problem MAX 2-SAT of finding a solution that maximizes the number

Where	Lower Bound	Assumptions
FGLSS	constant	NP $\not\subseteq n^{O(\log\log n)}$-TIME
FGLSS	$2^{(\log n)^{1-\epsilon}}$	NP $\not\subseteq n^{(\log n)^k}$-TIME
BS	n^ϵ	$\not\exists$ PTAS for MAX-SNP
Here	$n^{1-\epsilon}$	No $(\log n)$-coloring algorithm for k-colorable graphs or no polylog(n)-coloring for $(\log n)$-colorable graphs.
Here	$n/2^{(c\log n)^{1/2}}$	No $O(n^\epsilon)$-coloring for k colorable graphs in *quasi-polynomial* $(n^{(\log n)^k})$ time.

Table 9.1: **Lower bounds for Independent Set approximation based on various assumptions.**

of satisfied clauses in a 2-CNF expression. A PTAS for this problem would be an algorithm that for any $\epsilon > 0$, given a sufficiently large 2-CNF expression, finds an assignment that satisfies $1/(1 + \epsilon)$ of the maximum number of clauses possible.

MAX SNP is a syntactically defined class of problems described by Papadimitriou and Yannakakis [30]. It has the property that if one MAX SNP-hard or MAX SNP-complete problem has a polynomial-time approximation scheme then all problems in MAX SNP do as well. Some MAX SNP-complete problems include MAX k-SAT for $k \geq 2$ (finding the maximum number of clauses satisfiable in a k-CNF formula), the problem Independent Set-B of finding the largest independent set in a graph of constant degree bound $B \geq 4$, the TSP with edge weights 1 and 2, and others [30]. It is believed for these problems that no polynomial-time approximation schemes exist.

Berman and Schnitger prove that if there do not exist polynomial-time approximation schemes for MAX SNP-hard problems, then for some constant $\epsilon > 0$, no polynomial-time algorithm approximates Independent Set with performance guarantee n^ϵ. In a recent result of a very different style, Feige, Goldwasser, Lovász, Safra, and Szegedy [19] prove a lower bound for approximating independent sets based on NP not containing "quasi-polynomial time". In particular, they show that there is no polynomial-time algorithm for independent set with performance guarantee $O(2^{(\log n)^{1-\epsilon'}})$ for any $\epsilon' > 0$, if NP $\not\subseteq \bigcup_k \left[n^{(\log n)^k}\text{-TIME} \right]$. In addition, they show there is no algorithm with constant performance guarantee for Independent Set if NP $\not\subseteq n^{O(\log\log n)}$-TIME. Thus, they get a weaker conclusion than Berman and Schnitger (since $2^{(\log n)^{1-\epsilon'}} < n^\epsilon$ for all $\epsilon, \epsilon' > 0$) but based on likely a much more solid assumption. The new results presented in this chapter go in the other direction, proving a much stronger conclusion, but based on what may be much less solid assumptions. The results are summarized in Table 9.1.

9.1.1 The basic idea of the new results

Berman and Schnitger prove their result on approximating independent sets by amplifying "approximation gaps" in a constraint satisfaction problem $MAX_{k,n}$. They then show this problem can be reduced in an approximation-preserving sense to Independent Set, and reduced from, in such a sense, MAX SNP-Complete problem MAX 2SAT (Lemma 4.6 of [7]). Following the chain of reductions yields their n^{ϵ} bound. More simply, however, we can apply their basic technique in a straightforward way directly to the independent set problem. Doing so allows us to relate the approximability of Independent Set not only to that of finding PTAS's for MAX SNP-hard problems (in this case, the problem Independent Set-B), but also to the problem of finding good approximations for graph coloring. In fact, this version of their procedure (in some ways more general, in some more specific than the procedure in [7]) can be thought of as a randomized version of a commonly used graph product, and we describe the procedure from this point of view.

9.2 Randomized graph products

We now describe the randomized graph product technique that will be the key to the results presented in this chapter. The technique is formalized in the procedure Rand-Select below.

Algorithm Rand-Select takes as input an n-vertex graph G and values r, p, and t, and produces as output a new N-vertex graph H. The purpose of this procedure is to amplify gaps in independent set approximation. In particular, the procedure will reduce a problem of finding an independent set of size n/t^p in an n-vertex graph containing an (unknown) independent set of size n/t, to a problem of finding an independent set of size $N/(n^r)^p$ in an N-vertex graph containing an independent set of size N/n^r, where $N = n^{rp+2}$. Thus, for example, if the original graph was 3-colorable and so contained an independent set of size $n/3$, then the problem of *finding* an independent set of size $n/9$ in the original graph (a factor of 3 smaller) is mapped to a problem of finding an independent set a factor of $1/n^r$ smaller than the largest independent set in the new n^{2r+2}-vertex graph. We now describe the procedure.

Algorithm Rand-Select (Variant of procedure in proof of Lemma 4.3 in [7])

> **Given:** *An n-vertex graph $G = (V, E)$ and values r, p, and t.*
>
> **Output:** *An n^{rp+2}-vertex graph H, and a mapping φ from subsets of G to vertices of H.*
>
> 1. *Select $N = n^{rp+2}$ subsets of vertices, each of size $r \log_t n$, at **random** from the vertices of G. Label the subsets $s_1, s_2, \ldots, s_N$.*
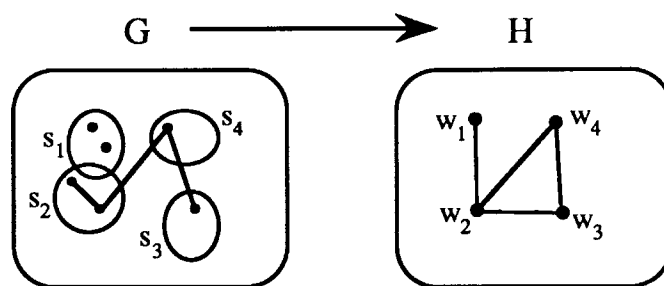
Figure 9.1: **A sample mapping from sets** s_i **in** G **to vertices** w_i **in** H.

2. For each subset s_i, associate a vertex w_i in H. The edge set $E(H) = \{(w_i, w_j) \mid$ $s_i \cup s_j$ is **not** independent in $G\}$. That is: (w_i, w_j) is not an edge in H only if both s_i and s_j are independent sets and in addition there are no edges between any vertex in s_i and any vertex in s_j.

Define a mapping $\varphi(s_i) = w_i$ and $\varphi^{-1}(w_i) = s_i$. (See Figure 9.1.)

(Note that for this to be a polynomial-time procedure, we need the product rp bounded above by a constant. The value t need not be a constant: in fact, we will later plug in $t = \log n$ to apply this technique to $(\log n)$-colorable graphs.)

Given a graph G and new graph H created using **Rand-Select** above, it will be convenient to extend the mapping φ as follows. For $S \subseteq V(G)$, let $\varphi(S) = \{\varphi(s_i) \mid s_i \subseteq S\}$. Also, for T a subset of $V(H)$, define $\varphi^{-1}(T) = \{v \mid v \in s_i \text{ for some } \varphi(s_i) \in T\} = \bigcup_{w \in T} \varphi^{-1}(w)$. Notice that $S \supseteq \varphi^{-1}(\varphi(S))$ and $T \subseteq \varphi(\varphi^{-1}(T))$; we do not necessarily get equality in the first case since S may have elements not inside any $s_i \subseteq S$, and in the second case, for $w_i, w_j \in T$, the set $s_i \cup s_j$ may contain some s_k for $w_k \notin T$. From this extended definition of φ and the definition of $E(H)$ in step 2 of **Rand-Select**, we immediately get the following fact.

Fact 9.1 *If S is an independent set in G, then $\varphi(S)$ is an independent set in H. If T is an independent set in H of size at least 2, then $\varphi^{-1}(T)$ is independent in G.*

Proof: If $w_i, w_j \in \varphi(S)$, then $s_i, s_j \subseteq S$. So if the edge (w_i, w_j) is in H, then $s_i \cup s_j$ is a non-independent subset of S. If T is independent in H of size greater than 1, then for each $w_i \in T$, the set s_i must be independent in G. So, $\varphi^{-1}(T)$ is a union of independent sets that are pairwise independent of each other, and thus is independent itself. ■

The purpose of procedure **Rand-Select** is as follows. Let $H = $ **Rand-Select**(G, r, p, t). If we have an independent set S of size n/t in graph G, then since each s_i has probability

about $(\frac{1}{t})^{r \log_t n}$ of being chosen inside S, the expected size of $\varphi(S)$ in H is $\Theta(n^{rp+2}(\frac{1}{t})^{r \log_t n})$ $= \Theta(n^{r(p-1)+2})$. In fact, with high probability, $\varphi(S)$ will be about that large. However, the expected size of $\varphi(S')$ for S' an independent set of size n/t^p is only $\Theta(n^{rp+2}(\frac{1}{t^p})^{r \log_t n}) = \Theta(n^2)$. In fact, it turns out that with high probability, $\varphi(S')$ will be small for *all* such S' (this is the purpose of the "+2" in **Rand-Select**) as described in the following theorem. [1]

Theorem 9.1 *Let G be an n-vertex graph with an independent set S of size n/t and let H be the output of* **Rand-Select**(G, r, p, t). *Then, with high probability, if $(r \log_t n)^2 = o(n/t^p)$ where $p \geq 1$, both of the following are true:*

(1) $|\varphi(S)| \geq \frac{1}{2} n^{r(p-1)+2}$, and

(2) for every independent set S' in G of size at most n/t^p, we have $|\varphi(S')| \leq 4n^2$.

Note that (1) implies $|\varphi(S)| = \Omega(N/n^r)$ and (2) implies $|\varphi(S')| = O(N/n^{rp})$, for $N = |V(H)|$.

The proof of this theorem uses the following standard (Chernoff variant) probabilistic inequality (e.g., see [1]).

Fact 9.2 *Suppose $X_1, \ldots, X_m$ are mutually independent $\{0, 1\}$-valued random variables. Let $X = X_1 + X_2 + \ldots + X_m$ and let $\mu = \mathbf{E}[X]$. Then:*

$$\mathbf{Pr}[X > 2\mu] < \left(\frac{e}{4}\right)^{\mu}$$

$$\mathbf{Pr}[X < \mu/2] < e^{-\mu/8}.$$

Proof of Theorem 9.1: First, claim (1) (the easy half). Given $S \subseteq V(G)$ of size n/t, consider a run of algorithm **Rand-Select**. Let X_i be a random variable such that $X_i = 1$ if $s_i \subseteq S$, and otherwise $X_i = 0$. So, $X = \sum X_i$ equals $|\varphi(S)|$. Since S has size n/t, we have:

$$\begin{aligned}
\mathbf{Pr}[X_i = 1] &= \binom{n/t}{r \log_t n} / \binom{n}{r \log_t n} \\
&= \left(\tfrac{1}{t}\right)^{r \log_t n} (1 + o(1)) \qquad (\text{ since } (r \log_t n)^2 = o(n/t)) \\
&= n^{-r}(1 + o(1)).
\end{aligned}$$

Thus, $E[X] = n^{r(p-1)+2}(1 + o(1))$ and with high probability, we have $X = |\varphi(S)| \geq \frac{1}{2} n^{r(p-1)+2}$.

Now, claim (2): we show that for *every* small set S' (and thus every small independent set S'), $\varphi(S')$ has size at most $4n^2$. We do this by showing that each *individual* set S' has an *extremely* low probability of having an image under φ larger than this value.

---

[1] We can actually replace the "+2" with "+1" to get a slightly better bound in Theorem 9.2 with only a little extra effort. However, the precise value of the constant is not crucial for us unless it could somehow be made significantly less than 1.

Fix a given set S' in G of size n/t^p. Let $X'_i = 1$ if $s_i \subseteq S'$, and $X'_i = 0$ if $s_i \not\subseteq S'$, and let $X' = \sum X'_i$. Again, since $(r \log_t n)^2 = o(n/t^p)$, we have:

$$
\begin{aligned}
\Pr[X'_i = 1] &= (t^p)^{-r \log_t n}(1 + o(1)) \\
&= n^{-rp}(1 + o(1)).
\end{aligned}
$$

Thus, $0.5n^2 < E[X'] < 2n^2$. Applying Fact 9.2, we get that $\Pr[X' > 4n^2] < (e/4)^{0.5n^2}$, which equals 2^{-cn^2} for some constant c. Note that if S' has size less than n/t^p, then $\Pr[X' > 4n^2]$ can only be lower. Now the crucial point: the probability that $X' > 4n^2$ is so small that even if we now sum over all at most 2^n such sets S', we get that $\Pr[|\varphi(S')| > 4n^2]$ for *any* S' of size at most n/t^p, is no more than $2^n 2^{-cn^2} = o(1)$. Thus, with high probability, both conclusions of Theorem 9.1 hold. ∎

So, algorithm **Rand-Select** maps a problem of finding an independent set of size $1/t^{p-1}$ times the largest in the original n-vertex graph to a problem of finding one of size $1/n^{rp-r} = 1/N^{\left(1 - \frac{r+2}{pr+2}\right)}$ times the largest in the new N-vertex graph. In particular, one gets the following theorem.

Theorem 9.2 *Suppose there exists a (randomized) algorithm $\mathcal{A}$ for Independent Set on N-vertex graphs that runs in time $f(N)$ and has performance guarantee $\leq \frac{1}{8}N^{\left(1 - \frac{r+2}{pr+2}\right)}$ for constants r, p. Then, there is a randomized algorithm $\mathcal{B}$ that on n-vertex graphs containing an independent set of size n/t, finds an independent set of size n/t^p with high probability in time $f(n^{pr+2}) + O(n^{rp + O(1)})$, so long as $(r \log_t n)^2 = o(n/t^p)$.*

Proof: Given an n-vertex graph G with independent set S of size n/t, run algorithm **Rand-Select**(G, r, p, t) to create graph H on n^{rp+2} vertices. This step takes $O(n^{rp + O(1)})$ time. By Fact 9.1, we know that $\varphi(S)$ is independent in H, so by Theorem 9.1 claim (1), we have that with high probability H contains an independent set of size $\frac{1}{2}n^{rp-r+2}$. So, algorithm $\mathcal{A}$ in time $f(n^{rp+2})$ finds an independent set T in H of size at least:

$$
\frac{\frac{1}{2}n^{rp-r+2}}{\frac{1}{8}\left[n^{rp+2}\right]^{\left(1 - \frac{r+2}{rp+2}\right)}} = \frac{4n^{rp-r+2}}{n^{rp-r}} = 4n^2.
$$

Now, look at $S' = \varphi^{-1}(T)$ which is independent in G by Fact 9.1. We know, by definition of φ, that $\varphi(S') \supseteq T$ and so $|\varphi(S')| \geq 4n^2$. Thus by Theorem 9.1 claim (2), we have: with high probability S' must have size at least n/t^p. ∎

If we plug $p = 1 + \epsilon$ into Theorem 9.2 and let $r = 2(1 - \delta)/\delta$ so $\frac{r}{r+2} = 1 - \delta$, then:

$$
n^{\left(1 - \frac{r+2}{pr+2}\right)} = n^{\frac{\epsilon r}{(1+\epsilon)r+2}} \geq n^{\left[\frac{\epsilon}{1+\epsilon}\right]\left(\frac{r}{r+2}\right)} = n^{\left[\frac{\epsilon}{1+\epsilon}\right](1-\delta)}.
$$

So, if we view Theorem 9.2 in the contrapositive form, we get the following corollary.

Corollary 9.3 *If for some t and some constant $\epsilon > 0$ there is no randomized polynomial-time algorithm which finds an independent set of size $n/t^{(1+\epsilon)}$ on n-vertex graphs containing an independent set of size n/t, then: for any constant $\delta > 0$ there is no (randomized) polynomial-time algorithm with performance guarantee $o(n^{\frac{\epsilon}{1+\epsilon}(1-\delta)})$ for general Independent Set.*

This corollary immediately implies the Berman-Schnitger result on the approximability of Independent Set [7] by using the MAX SNP-hard problem Independent Set-B. Any n-vertex graph with degree at most B must have an independent set of size $n/(B+1)$. So, we get the following.

Corollary 9.4 (Berman and Schnitger) *If there do not exist randomized PTAS's for MAX SNP-hard problems, then there exists $c > 0$ such that Independent Set does not have a (randomized) polynomial-time approximation algorithm with performance guarantee n^c.*

In particular, if for some ϵ and B, Independent Set-B does not have a randomized polynomial-time approximation algorithm with performance guarantee $(1 + \epsilon)$, then Independent Set does not have a polynomial time approximation algorithm with performance guarantee $o(n^{\frac{\epsilon'}{1+\epsilon'}(1-\delta)})$ for $\epsilon' = \log_{B+1}(1 + \epsilon)$, for any constant $\delta > 0$.

We can also use Theorem 9.2 to provide stronger bounds on independent-set approximation based on assumptions of the hardness of approximate graph coloring. In particular, we can prove the following.

Theorem 9.5 *Suppose there exists a (randomized) polynomial-time algorithm $\mathcal{A}$ for Independent Set with performance guarantee $n^{1-\epsilon}$ for some $\epsilon > 0$. Then, there is a randomized polynomial-time algorithm $\mathcal{B}$ that will color any n-vertex k-colorable graph with $O(\log n)$ colors, and color any n-vertex $O(\log n)$-colorable graph with $O(\log^c n)$-colors (where $c \leq 1 + 3/\epsilon$).*

Theorem 9.6 *Suppose there exists a (randomized) quasipolynomial-time algorithm $\mathcal{A}$ for Independent Set with performance guarantee $N/2^{\sqrt{c \log N}}$ on N-vertex graphs, then there is a randomized quasipolynomial-time algorithm $\mathcal{B}$ to color any n-vertex k-colorable graph with $O(n^\epsilon)$ colors, where $\epsilon = (20 \log k)/c$.*

Proof of Theorems 9.5 and 9.6: Given an n-vertex t-colorable graph G, we know there exists an independent set of size at least n/t. Suppose we had an algorithm $\mathcal{B}'$ that on any n-vertex graph with an independent set of size n/t were guaranteed to find an independent set of size n/t^p for some constant p. We could then find a coloring of G with at most $(t^p \ln n)$ colors by applying $\mathcal{B}'$, coloring the independent set found with one color, and then repeating

on the remaining graph G' of size at most $n(1 - 1/t^p)$. Note that since G is t-colorable, so is graph G', and thus G' has an independent set of at least $1/t$ of its vertices as well and we may reapply $\mathcal{B}'$. The number of colors used by this algorithm $\mathcal{B}$ is at most a value C such that $n(1 - 1/t^p)^C = 1$, so $C \leq -(\ln n)/\ln(1 - 1/t^p) \leq t^p \ln n$. Thus if t is some constant k, the number of colors used is $O(\log n)$ and if $t = \log n$, the number of colors used is $O((\log n)^{p+1})$. (The fact that $\log n$ decreases as the graph gets smaller only helps).

If there exists a polynomial time algorithm $\mathcal{A}$ for Independent Set with performance guarantee $n^{1-\epsilon}$ for some constant $\epsilon > 0$, then for $p > \frac{3-2\epsilon}{\epsilon}$, algorithm A has performance guarantee $o(n^{1-\frac{3}{p+2}})$. So, we can apply Theorem 9.2 with $r = 1$ to get a randomized polynomial-time algorithm $\mathcal{B}'$ with the guarantee we need. This proves Theorem 9.5.

For Theorem 9.6, we must be a bit careful.[2] The quasi-polynomial time algorithm $\mathcal{B}$ is as follows. Given n-vertex k-colorable graph G, let $p = \epsilon \log_k n$ so $n^\epsilon = k^p$, and let $N = n^{p+2}$. Plugging in these values, we get:

$$
\begin{aligned}
2^{\sqrt{c \log N}} &= N^{\sqrt{c}/\sqrt{(p+2)\log n}} \\
&\geq N^{\sqrt{c/(2p \log n)}} \\
&= N^{\sqrt{\epsilon c/(2p^2 \log k)}} \qquad \text{(using } \log n = (p \log k)/\epsilon) \\
&= N^{\frac{\sqrt{10}}{p}} \qquad\qquad \text{(using } \epsilon = \frac{20 \log k}{c}) \\
&\gg N^{\frac{3}{p+2}}.
\end{aligned}
$$

Thus, the performance guarantee $N/2^{\sqrt{c \log N}}$ of algorithm $\mathcal{A}$ for Independent Set on N-vertex graphs is $o(N/N^{\frac{3}{p+2}}) = o(N^{1-\frac{3}{p+2}})$. So, we can again apply Theorem 9.2 with $r = 1$ to get an algorithm $\mathcal{B}$ guaranteed on any k-colorable graph to find an independent set of size $n/k^p = n^{1-\epsilon}$. Thus, $\mathcal{B}$ makes progress (in fact, progress type 1 of Section 3.3) towards an $O(n^\epsilon)$-coloring of G.

Since algorithm $\mathcal{A}$ is quasipolynomial, algorithm $\mathcal{B}$ runs in time quasipolynomial in (n^{p+2}), which is quasipolynomial in n since $n^{p+2} = n^{O(\log n)}$.   ∎

---

[2]Note: it is easy to fall into a trap in Theorem 9.2 in falsely thinking that if p is a function of n (eg. $p = \epsilon \log n$) for algorithm $\mathcal{B}$, then we can plug in the same function of N (eg. $\epsilon \log N$) for algorithm $\mathcal{A}$.

Chapter 10

Possibilities for improvement, open problems, and conclusion

10.1 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 since for such a low average degree, any two given vertices would be expected to share less than one neighbor in common. 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 a 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 3.2.)

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 4.1 (bounding the number of shared neighbors of two vertices). Then, Theorems 4.5 and 4.6 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 main 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 4.5) in the new graph. Also, the average degree has only changed by a constant factor, so the set T produced will still be large. One small difficulty is that Corollary 4.7 relies on a large *minimum* degree which might no longer exist in the new graph. This

problem can be overcome by simply deleting all vertices with degree less than, say, 1/10 of the average in the new graph.

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). From preliminary calculations, I believe that some of the results for distance-2 neighborhoods may go through — for example, that one could find a set T with an independent set of 3/8 of its vertices inside the distance-3 neighborhood. (Note that if the edges were distributed randomly, one would expect a ratio of $\frac{3}{8}:\frac{1}{4}:\frac{3}{8}$ of blues to reds to greens inside the distance-3 neighbors of v for $v \in$ red.) However, all the techniques given here for forcing expansion — that is, for forcing the set found to be large — seem to break down completely.

10.2 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. For the general problem of coloring graphs of arbitrary chromatic number, the best lower bound remains a factor of $2 - \epsilon$ from 1976 by Garey and Johnson [20].

The random and "semi-random" case appears much easier. We have described here an algorithm to color a random k-colorable graph in the model $\mathcal{G}(n, p, k)$ for p as low as $n^{-1+\epsilon}$ (see Section 7). For even smaller values of p, perhaps some other strategy might work well. An intriguing open question is whether there might be a polynomial-time algorithm to color graphs in $\mathcal{G}(n, p, k)$ for *every* p, or whether there is some intrinsic reason such an algorithm should not exist. Experimental work of Petford and Welsh [31] suggests that at least for the heuristics used there, low values of p for which the average degree in the graph is about 5 or 6 may be the hardest.

For the semi-random model we described in Chapter 8 an algorithm to color graphs in $\mathcal{G}_{SB}(n, p, 3)$ for p as low as $n^{-0.6+\epsilon}$, and for higher values of p for $k > 3$ (see Table 8.1). One obvious open question is whether one can optimally color such graphs for lower noise rates p. A second open direction to explore is coloring graphs based on even "harder" semi-random sources that have been proposed and studied in the cryptographic literature. In these models, the "noise" is not independent over each bit; rather, we are simply guaranteed

that no sequence of bits of some length occurs too often. In a graph setting, this might correspond to a model in which we are simply guaranteed that for any given collection of "potential edges," no fixed configuration occurs with more than some specified probability. The reader is referred to papers of Chor and Goldreich [17] and Zuckerman [44] for more details on these "weak random" models.

Appendix A

The Vertex-Cover / Independent-Set approximation algorithm

We now describe a simplified version of the Vertex-Cover approximation algorithm of Bar-Yehuda and Even [4] and Monien and Speckenmeyer [28], specialized to its use in this thesis. The version here is taken from a treatment given by Boppana and Halldórsson [12]. 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 $\leq 2l+1$. For example, if there were an edge between a vertex in V_2 and a vertex in V_4 then there is a cycle of length 7.*
>
>   *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,

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

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:

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

Appendix B

An analog of Spencer's result on counting extensions

In this section, we prove an analog of a theorem of Spencer for counting the number of images of a rooted graph.

If (R, H) is a rooted graph (see Definitions 8.2, 8.3, and 8.4), define $\text{Im}(H, G)$ to be the set of images of H in G and let $\text{Num}(H, G) = |\text{Im}(H, G)|$. Also, for $\mathcal{M}$ some model (such as $\mathcal{G}(n, p)$ or $\mathcal{G}(n, p, k)$) define $\mu(H, \mathcal{M})$ to be the expected number of images of H in $G \leftarrow \mathcal{M}$.

Spencer [35] proves the following result for the random graph model $\mathcal{G}(n, p)$.

Theorem B.1 (Spencer) *Let (R, H) be strictly balanced on some constant number of vertices and let $\delta, c > 0$. Then, $\exists K > 0$ so that if p is such that $\mu(H, \mathcal{G}(n, p)) \geq K \log n$, then for $G \leftarrow \mathcal{G}(n, p)$:*

$$\mathbf{Pr}[(1 - \delta)\mu \leq Num(H, G) \leq (1 + \delta)\mu] = 1 - o(n^{-c}).$$

In order to prove that the l-**path** algorithm of Chapter 7 works as claimed, we need an analog of Spencer's result — at least for the case of H a path of some constant length l — for the model $\mathcal{G}(n, p, k)$. (As noted in Chapter 7, paths of length l between two roots x and y are strictly balanced.) In fact, Spencer's proof goes through in the $\mathcal{G}(n, p, k)$ model with only minor modifications. We describe here what those modifications are and how they affect Spencer's proof.

Spencer's result is easiest to prove for the special (but main) case where there exists some sufficiently small ϵ so that the expected number μ of images of H in G is at most n^ϵ; that is, when $K \log n \leq \mu \leq n^\epsilon$. To simplify our discussion, we will only consider that case here. We will also consider only rooted graphs (R, H) that have no automorphisms fixing the roots. Spencer counts "extensions" which are essentially all the different *maps* of H into G, whereas we count the images of H; for rooted graphs without such automorphisms, these are the same quantity. Note that paths of length l fit into this category.

For H a path of length l between roots x and y, we would like to prove that the number of images of H in $G \leftarrow \mathcal{G}(n, p, k)$ *given* that x and y are chosen the same color, or *given*

that x and y are chosen of different color, are both within $(1 + o(1))$ of the expectation. In order to not prove essentially the same theorem twice — once for each case — let us define the notion of a random k-colorable graph given a particular root coloring. For a root set R, there are $k^{|R|}$ different possible ways to assign k colors to the $|R|$ vertices. So:

- Let $\mathcal{G}_j^R(n, p, k)$ be the model $\mathcal{G}(n, p, k)$ given that the subset R of V has the jth of $k^{|R|}$ possible colorings.

B.1   Modifying Spencer's result

Suppose (R, H) is a rooted graph on a constant number of vertices and X is some image of H in K_n. Let $\mathbf{v} = nonroots(H)$ and $\mathbf{e} = edges(H)$. Then $\mathbf{Pr}[X \subseteq G \mid G \leftarrow \mathcal{G}(n, p)] = p^{\mathbf{e}}$. If H has no automorphisms, then $\mu(H, \mathcal{G}(n, p)) = n^{\mathbf{v}} p^{\mathbf{e}}(1 - o(1))$. The key fact that allows Spencer's argument to go through for $\mathcal{G}(n, p, k)$ is that if H is also k-colorable, then $\mathbf{Pr}[X \subseteq G \mid G \leftarrow \mathcal{G}(n, p, k)] = \Theta(p^{\mathbf{e}})$. The reason is that since H has only a constant number of vertices, there is a constant probability at least $(1/k)^{|V(H)|}$ that in the creation of G, the vertices of X are placed into color classes that legally color the graph. So, $\mathbf{Pr}[X \subseteq G] \geq (1/k)^{|V(H)|} p^{\mathbf{e}} = \Theta(p^{\mathbf{e}})$.

We now describe how to modify Spencer's proof to prove the following result.

Theorem B.2 *Let (R, H) be strictly balanced on some constant number of vertices with no automorphisms fixing the roots, and let $\delta, c > 0$. Then, there exists $K, \epsilon > 0$ so that if*

$\mu = \mu(H, \mathcal{G}_j^R(n, p, k)) \in [K \log n, n^{\epsilon}]$, *then for $G \leftarrow \mathcal{G}_j^R(n, p, k)$:*

$$\mathbf{Pr}[(1 - \delta)\mu \leq Num(H, G) \leq (1 + \delta)\mu] = 1 - o(n^{-c}).$$

Proof: For convenience, let $\mathcal{M} = \mathcal{G}_j^R(n, p, k)$, $\mathbf{v} = nonroots(H)$, $\mathbf{e} = edges(H)$, and let $G \leftarrow \mathcal{M}$. Also, let $X_1, \ldots, X_m$ be the images of H in K_n and let A_i be the event that $X_i \subseteq G$. We may assume that H is k-colorable *given* that the root set has the jth possible assignment of colors, else μ would equal 0.

From our above observations, for any given image X_i, $\mathbf{Pr}[X_i \subseteq G] = \Theta(p^{\mathbf{e}})$ and so $\mu = \Theta(n^{\mathbf{v}} p^{\mathbf{e}})$. For convenience, let us define $\hat{p}$ so that $\mathbf{Pr}[X_i \subseteq G] = (\hat{p})^{\mathbf{e}}$ and thus $\mu = (1 - o(1))n^{\mathbf{v}} \hat{p}^{\mathbf{e}}$.

Spencer's proof for $\mathcal{G}(n, p)$ where $\mu(H, \mathcal{G}(n, p)) \in [K' \log n, n^{\epsilon'}]$ for sufficiently large K' and sufficiently small ϵ', proceeds in three stages. First, he proves a theorem stated here as Theorem 8.2 of Chapter 8. We have already proven the analog in Theorem 8.3.[1]   Second, he proves that for $G' \leftarrow \mathcal{G}(n, p)$, with probability $1 - o(n^{-c})$, the size of every maximal

---

[1]Technically, Theorem 8.3 was proven for the semi-random model $\mathcal{G}_{SB}(n, p, k)$. However, since in $\mathcal{G}_i^R(n, p, k)$ there are w.h.p. $\Omega(n)$ vertices of each color class, the bound holds for this model as well.

family $\mathcal{F}$ of *disjoint* X_i in G' is within $(1 + \delta)$ of μ. Finally he shows that for any fixed maximal family $\mathcal{F}$, with probability $1 - o(n^{-c})$ there are only $O(1)$ images $X_j \notin \mathcal{F}$ in G' that intersect some $X_i \in \mathcal{F}$. Since every image of H must either belong to $\mathcal{F}$ or else intersect some $X_i \in \mathcal{F}$ (as $\mathcal{F}$ is a *maximal* family of disjoint images), the last 2 parts of Spencer's argument imply that $\text{Num}(H, G')$ is within $(1 + \delta)$ of μ with probability $1 - o(n^{-c})$.

Note that for X any subgraph of K_n at all,

$$\mathbf{Pr}[X \subseteq G \mid G \leftarrow \mathcal{M}] \leq \mathbf{Pr}[X \subseteq G' \mid G' \leftarrow \mathcal{G}(n,p)].$$

The reason is simply that each edge is placed into $G \leftarrow \mathcal{M}$ with probability *at most p* (either probability p or probability 0 depending on the colors of the endpoints), while in $\mathcal{G}(n,p)$, each edge is placed into the graph with probability exactly p. So, if we pick ϵ sufficiently small such that $\mu(H, \mathcal{G}(n,p)) < n^{\epsilon'}$ and thus Spencer's argument holds for $G' \leftarrow \mathcal{G}(n,p)$, then the third part of Spencer's argument carries over directly and we need not prove it again here. (Recall that $\mu = \Theta(n^{\mathbf{v}} p^{\mathbf{e}})$ so for any $\epsilon' > 0$ there exists $\epsilon > 0$ such that $(\mu \leq n^{\epsilon}) \Rightarrow (n^{\mathbf{v}} p^{\mathbf{e}} \leq n^{\epsilon'})$.) We focus now on the second part. The analysis here is taken directly the proof of Spencer [35].

Let us first calculate some basic quantities. First, the number of X_i in K_n is at most $n^{\mathbf{v}}$ so we can loosely upper bound the number of families $\mathcal{F} \subseteq \text{Im}(H, K_n)$ of t pairwise disjoint images X_i, by $\binom{n^{\mathbf{v}}}{t}$. Also, for any fixed such family $\mathcal{F}$, the probability that $\mathcal{F} \subseteq \text{Im}(H, G)$ (that is, that the X_i in $\mathcal{F}$ are all in G) is $(\hat{p}^{\mathbf{e}})^t$ since the $X_i \in \mathcal{F}$ are all disjoint so the corresponding events A_i are mutually independent.

For a given family $\mathcal{F}$ of t pairwise disjoint X_i, we now upper bound the probability that *no* image X_j disjoint from all $X_i \in \mathcal{F}$ exists in G; that is, the probability that $\mathcal{F}$ is a maximal family of disjoint images given that $\mathcal{F} \subseteq \text{Im}(H, G)$. Let $X_{i_1}, \ldots, X_{i_r}$ be all the images in $\text{Im}(H, K_n)$ disjoint from $\mathcal{F}$. We know that:

$$r = (n - t\mathbf{v} - |R|)_{\mathbf{v}}$$

since there are $(n - t\mathbf{v} - |R|)$ non-root vertices not inside $\mathcal{F}$, and H has no automorphisms. By Theorem 8.3, for ϵ sufficiently small, $\sum_{i \sim j} \mathbf{Pr}[A_i \wedge A_j] = o(1)$ where $i \sim j$ if $i \neq j$ and $E(X_i) \cup E(X_j) \neq \phi$. So, certainly the summation restricted to just the $i, j \in \{i_1, \ldots, i_r\}$ equals $o(1)$ as well. Thus, by Janson's inequality, (noting that the "λ" term is $o(1)$) we have:

$$\mathbf{Pr}\left[\bigwedge_{j=1}^{r} \bar{A}_{i_j} \right] = [1 + o(1)] \prod_{j=1}^{r} \mathbf{Pr}\left[\bar{A}_{i_j} \right]$$
$$= [1 + o(1)](1 - \hat{p}^{\mathbf{e}})^r. \qquad \text{(by definition of } \hat{p})$$

Given the above facts, we can upper bound the probability P_t that there exists any maximal family $\mathcal{F}$ of t pairwise disjoint X_i's within G by the quantity:

- $\alpha_t \;=\; [1 - o(1)]\binom{n^{\mathbf{v}}}{t}(\hat{p}^{\mathbf{e}})^t(1 - \hat{p}^{\mathbf{e}})^{(n - t\mathbf{v} - |R|)_{\mathbf{v}}}.$

Consider now two cases. First, suppose $t \geq n^{2\epsilon}$. We may upper bound $\binom{n^{\mathbf{v}}}{t}$ by $\frac{n^{\mathbf{v}t}}{t!} \leq \left[\frac{3n^{\mathbf{v}}}{t}\right]^t$. (Using $3 > 2.718...$ to avoid confusion with $\mathbf{e}$.)   So, since $n^{\mathbf{v}}\hat{p}^{\mathbf{e}} = O(n^\epsilon)$ we have:

$$\alpha_t \;\leq\; \left[\frac{3n^{\mathbf{v}}}{t}\right]^t(\hat{p}^{\mathbf{e}})^t \;\leq\; \left[\frac{3n^{\mathbf{v}}\hat{p}^{\mathbf{e}}}{t}\right]^t \;=\; O\!\left(\frac{n^\epsilon}{n^{2\epsilon}}\right)^{n^{2\epsilon}} \;=\; o(n^{-(c+1)}).$$

Thus, the probability there exists within G a maximal family $\mathcal{F}$ of *any* size $t \geq n^{2\epsilon}$ is at most $o(n^{-c})$.

The second case is $t \leq n^{2\epsilon}$. For ϵ sufficiently small (at most $1/4$) we have $t \leq n^{1/2}$ so: $(n - t\mathbf{v} - |R|)_{\mathbf{v}} \;=\; n^{\mathbf{v}} - \Theta(t\mathbf{v}n^{\mathbf{v}-1}) \;\geq\; n^{\mathbf{v}} - \Theta(n^{\mathbf{v}-1/2})$. Thus,

$$
\begin{aligned}
(1 - \hat{p}^{\mathbf{e}})^{(n - t\mathbf{v} - |R|)_{\mathbf{v}}} \;&\leq\; (1 - \hat{p}^{\mathbf{e}})^{n^{\mathbf{v}}}/(1 - \hat{p}^{\mathbf{e}})^{\Theta(n^{\mathbf{v}-1/2})} \\
&=\; (1 - \hat{p}^{\mathbf{e}})^{n^{\mathbf{v}}}/(1 - \Theta(\hat{p}^{\mathbf{e}}n^{\mathbf{v}-1/2})) \\
&\leq\; (1 - \hat{p}^{\mathbf{e}})^{n^{\mathbf{v}}}/(1 - \Theta(n^\epsilon n^{-1/2})) \\
&=\; (1 - \hat{p}^{\mathbf{e}})^{n^{\mathbf{v}}}[1 + o(1)].
\end{aligned}
$$

So, we can upper bound the probability P_t by:

$$
\begin{aligned}
P_t \;&\leq\; [1 + o(1)]\binom{n^{\mathbf{v}}}{t}(\hat{p}^{\mathbf{e}})^t(1 - \hat{p}^{\mathbf{e}})^{n^{\mathbf{v}}} \\
&\leq\; [1 + o(1)]\binom{n^{\mathbf{v}}}{t}(\hat{p}^{\mathbf{e}})^t(1 - \hat{p}^{\mathbf{e}})^{n^{\mathbf{v}}-t}.
\end{aligned}
$$

Thus, $P_t \leq (1 + o(1))\mathbf{Pr}[Y = t]$ where Y has the binomial distribution $B(n^{\mathbf{v}}, \hat{p}^{\mathbf{e}})$. Let $\mu^* = n^{\mathbf{v}}\hat{p}^{\mathbf{e}}$. We know for such a distribution, for any $\delta > 0$ we have $\mathbf{Pr}[|Y - \mu^*| > \frac{\delta}{2}\mu^*] = o(n^{-c})$, so long as $\mu^* > K \log n$ for sufficiently large K. Thus, the probability there exists any maximal family $\mathcal{F}$ of disjoint images X_i of size *not* within $\delta\mu$ of μ, and so *not* within $\frac{\delta}{2}\mu^*$ of μ^*, is at most $o(n^{-c})$. This finishes the second part of Spencer's argument. Since as noted above, the third part follows immediately from the result for $\mathcal{G}(n,p)$, we have proved the theorem.   ∎

Bibliography

[1] N. Alon and J. Spencer. *The Probabilistic Method.* Wiley, 1991.

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

[3] K. Appel and W. Haken. Every planar map is four colorable. *Illinois J. Math.*, 21:429–490, 491–567, 1979.

[4] R. Bar-Yehuda and S. Even. A $2 - \frac{\log \log n}{2 \log n}$ performance ratio for the weighted vertex cover problem. Technical Report Technical Report #260, Technion Haifa, January 1983.

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

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

[7] P. Berman and G. Schnitger. On the complexity of approximating the independent set problem. In *6th STACS. Lecture Notes in Computer Science # 349*, pages 256–267, 1989.

[8] Avrim 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.

[9] Avrim Blum. Some tools for approximate 3-coloring. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, St. Louis, October 1990.

[10] B. Bollobás. The chromatic number of random graphs. *Combinatorica*, 8:49–55, 1988.

[11] R. Boppana and J. Spencer. A useful elementary correlation inequality. *J. Combin. Theory Ser. A*, 50:305–307, 1989.

[12] 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.

[13] 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.

[14] R. L. Brooks. On colouring the nodes of a network. *Prov. Cambridge Phil. Soc.*, 37:194–197, 1941.

[15] 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.

[16] 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.

[17] B. Chor and O. Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM J. Computing*, 17(2):230–261, April 1988.

[18] 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.

[19] U. Feige, S. Goldwasser, L. Lovasz, M. Safra, and M. Szegedy. Approximating clique is almost NP-Complete. In preparation, 1991.

[20] M. R. Garey and D. S. Johnson. The complexity of near-optimal graph coloring. *JACM*, 23:43–49, 1976.

[21] B. Grünbaum. Grötzsche's theorem on 3-colorings. *Michigan Math J.*, 10:303–310, 1963.

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

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

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

[25] N. Linial and U. Vazirani. Graph products and chromatic numbers. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, Research Triangle Park, October 1989.

[26] L. Lovasz. Three short proofs in graph theory. *Journal of Combinatorial Theory, Series B*, 19:111–113, 1973.

[27] D. W. Matula. Expose-and-merge exploration and the chromatic number of a random graph. *Combinatorica*, pages 275–284, 1987.

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

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

[30] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 229–234, Chicago, May 1988.

[31] A. D. Petford and D. J. A. Welsh. A randomised 3-colouring algorithm. *Discrete Mathematics*, 74:253–261, 1989.

[32] P. Raghavan. personal communication.

[33] G. Ringel. *Map Color Theorem*. Springer-Verlag, 1974.

[34] M. Santha and U. V. Vazirani. Generating quasi-random sequences from semi-random sources. *JCSS*, 33:75–87, 1986.

[35] Joel Spencer. Counting extensions. *J. Combin. Theory Ser. A*, 55:247–255, 1990.

[36] Joel Spencer. Threshold functions for extension statements. *J. Combin. Theory Ser. A*, 53:286–305, 1990.

[37] Richard Steinberg. The state of the three color problem. *Annals of Discrete Mathematics*. To appear.

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

[39] U. Vazirani and V. Vazirani. Random polynomial time ie equal to slightly-random polynomial time. In *Proceedings of the 26th Annual IEEE Symposium on Foundations of Computer Science*, pages 417–428, Portland, October 1985.

[40] U. V. Vazirani. Towards a strong communication complexity theory, or generating quasi-random sequences from two communicating slightly-random sources. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, pages 366–378, Providence, 1985.

[41] Sundar 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.

[42] D. J. A. Welsh and M. B. Powell. An upper bound on the chromatic number of a graph and its application to timetabling problems. *The Computer Journal*, 10:85–87, 1967.

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

[44] D. Zuckerman. General weak random sources. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, pages 534–543, St. Louis, October 1990.