Recent methods for approximate colorings of 3-colorable graphs

Adam Sealfon (asealfon@mit.edu), Adam Yedidia (adamy@mit.edu),
Melissa Gymrek (mgymrek@mit.edu)

December 12, 2012

1 Introduction

A k-coloring of a graph is an assignment of one of k colors to each vertex such that no two adjacent vertices are assigned the same color. Graph coloring problems have a variety of real-world applications in areas such as scheduling and register allocation (and solving sudoku puzzles). Additionally, they offer a rich set of problems and have led to many fundamental insights in graph theory.

Surprisingly, different graph coloring problems can have wildly different difficulties: while coloring a 2-colorable graph can be done in linear time, 3-coloring is one of the original NP-hard problems described. Therefore, we must resort to approximation. In particular, the question we explore here is:

> Given a 3-colorable graph with n vertices, how many colors k do we need to color it in polynomial time?

Here we will give a brief overview of the evolution of approximation techniques for coloring 3-colorable graphs, beginning with the $O(\sqrt{n})$ solution of Widgerson [4], up to the recent $O(n^{0.2038})$ solution by Kawarabayashi and Thorup [3]. We will then describe in more detail the method of [3] and that of Blum [2] from which the recent solution draws heavily. We will conclude by commenting on ways the combinatorial methods described here have been improved.

2 From $\sqrt{n}$ to $n^{0.2038}$

The first set of approaches to 3-colorable graph approximations work by taking a vertex of interest and looking at its immediate neighborhood. Progress is made on that isolated subgraph before moving on to color the rest of the graph.

The earliest well known approximation for coloring 3-colorable graphs was given by Widgerson [4]. This method (essentially the same as what we were asked to derive in problem set 8) works by repeatedly taking a vertex of degree more than $O(\sqrt{n})$ (else we can greedily color with $O(\sqrt{n})$ colors), and determining a 3-coloring for it (one color) and its immediate neighbors (two additional colors). That vertex can then be collapsed, and we can no longer use the two colors we used for the neighbors. This simple approach uses $3\lceil\sqrt{n}\rceil$ colors. Berger and Rompel [1] improved this approach to $O(\frac{\sqrt{n}}{\sqrt{\log(n)}})$ by choosing $\log(n)$ vertices to start with rather than just one.

The next major improvements came with the new technique of looking not only at the immediate neighbors of a vertex, but also at the neighbors of neighbors, or the *second neighborhood*. The first of these was by

Blum [2], who described an algorithm using $\tilde{O}(n^{2/5})$ colors, which was improved to $\tilde{O}(n^{3/8})$ by paying close attention to dense areas of the graph of interest. His method works by finding isolated subgraphs of second neighborhoods where progress can be made (progress will be defined shortly). Kawarabayashi and Thorup [3] improved this to $\tilde{O}(n^{4/11})$ by looking at subsets of second neighborhoods for sparse cuts that isolate sets of vertices that must share the same color. Although not described here, using the technique of *semi-definite programming* these techniques can be improved to $O(n^{0.2072})$ and $O(n^{0.2038})$, respectively, giving the current best approximation for coloring 3-colorable graphs.

3 Preliminaries

We first provide definitions of terms used throughout this paper, and then introduce the concept of second-order neighborhoods, which are an important component of the combinatorial constructions presented below.

3.1 Definitions

Definition 1 *The **neighborhood** $N(S)$ of a set of vertices S is the set of vertices that are adjacent to a vertex in S.*

Similarly, the **second neighborhood** $N(N(S))$ of a set of vertices S is the set of vertices that are adjacent to a vertex in the neighborhood of S, but that are not themselves in the neighborhood of S. In other words, the set $N(N(S))$ contains the vertices of S and the vertices of distance exactly 2 from S.

Definition 2 *The **chromatic number** of a graph is the smallest number of colors with which that graph can be colored.*

Determining the chromatic number of a graph is known to be NP-hard.

Definition 3 *A set S of vertices is a **monochromatic set** if all vertices in S have the same color in some optimal coloring of the graph. Similarly, S is a **multichromatic set** if in every optimal coloring of the graph some pair of vertices in S is colored with different colors.*

Definition 4 *An **independent set** in a graph is a set S of vertices such that no pair of edges in the set is adjacent.*

Independent sets are useful because they can be colored with a single color in $O(n)$ time. For graph coloring purposes, an independent set is essentially a $O(1)$-colorable set.

3.2 Second-order neighborhoods

In the algorithms presented below, we are interested in searching the second neighborhood of each vertex rather than the neighborhood of each vertex for large independent sets. This is due to the difference in impact of the color of a vertex v on its first neighborhood vs. its second neighborhood. This section is devoted to explaining, in a very general sense, why this difference exists, thereby giving the reader an intuition at the most basic level for why the algorithm represents an improvement over the $\sqrt{n}$-approximation algorithm.

When we are looking to find a large independent set, we would like to show that a large fraction of the second neighborhood of vertex v whose color is green (for example) is also green. Although this is no proof, a good heuristic for how easy it will be to show that a large fraction of the first or second neighborhood of a vertex is a single color is the probability that a large fraction of that neighborhood turns out to be that color given

an optimal coloring on a random graph. When we take a green vertex in an optimal coloring of a random graph, its first neighborhood is in expectation half-red, half-blue. Its second neighborhood, however, is in expectation half-green, and the variance will be much lower, because if a typical vertex degree on the graph is d, there will be in expectation $\Theta(d^2)$ vertices in the second neighborhood while there were $\Theta(d)$ in the first. For this reason, "More likely," of course, was in quotes because Blum's algorithm is not probabilistic; but by restricting the degree of each vertex, and by restricting the number of neighbors that can be shared by two vertices, Blum and later Kawarabayashi and Thorup, restrict the problem enough to use the above intuition. This concept is depicted in **Figure 1**.

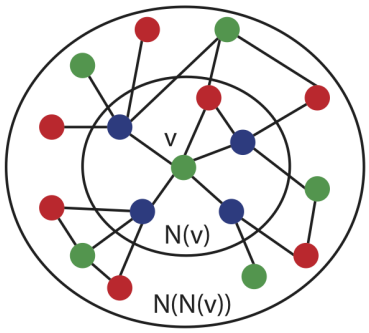

Figure 1: The second-order neighborhood of a vertex v. Roughly half of the second neighborhood will be green.

4 Progress toward k-coloring

Second-neighborhood structures are useful because they help find large independent sets, which help us make "progress" towards a k-coloring of the graph. In order to formalize this concept of "progress," Blum divides it into three types and defines each in his paper. We reuse these definitions as a way of helpfully describing the possible steps taken by a hypothetical graph coloring algorithm.

Progress Type 1: Identify an independent or 2-colorable set of size $\Omega(n/k)$. If we find such a set, we can color it quickly with a constant number of colors, and proceed to the rest of the graph. If used repeatedly, this progress type will result in a k-coloring of the graph, since we will have used $O(1)$ colors for each $\Omega(n/k)$ vertices, resulting in the use of $O(1)/\Omega(n/k) = O(k/n)$ colors per vertex and $O(k)$ colors for the whole graph.

Progress Type 2: Identify an independent or 2-colorable set of size $|S|$ with a neighborhood of size $|S|O(k/n)$. If we find such a set, the neighborhood of that set will be small enough that we can "set it aside" and always be able to find colors with which to color it later. If used repeatedly, this progress type will also result in a k-coloring of the graph, since we will have been using $O(k/n)$ colors per vertex amortized over each vertex in S.

Progress Type 3: Identify two vertices that must have the same color. If we find two such vertices, we can safely "merge them" (i.e. take the neighbors of one and join it to the neighbors of the other). Note that this type of progress reduces the size of the graph but uses no colors, so we need not worry about the coloring that would result from repeated application of this type of progress.

It follows that if we can prove that we can always make progress of one of types 1, 2, or 3, we can reach an $O(k)$ coloring. The three types of progress are depicted in **Figure 2**.

Blum proves an important theorem regarding progress, which we will make use of later. We will refer to this

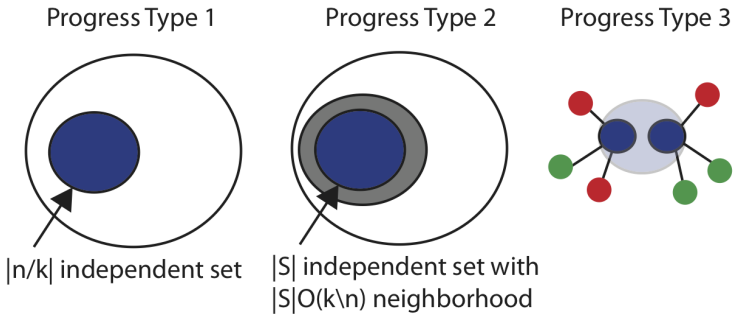

Figure 2: The three types of progress toward k-coloring.

as **Blum's Theorem**. The theorem states that:

Theorem 5 *If there is a set of vertices that are neighbors with both of two vertices u and v of size at least $\Omega(n/k^2)$, then we can make one of the three types of progress. Formally, if $|N(u) \cap N(v)| = \Omega(\frac{n}{k^2})$, we can make progress.*

This proof has an important lemma that is also used throughout, and referred to below as **Blum's Corollary**.

Corollary 6 *Given a vertex set $X \subset V$ of size at least $\Psi = n/k^2$, in polynomial time we can either make progress towards a $\tilde{O}(k)$-coloring of G or can guarantee that X must be multichromatic.*

Details of proofs of these are not given here, but we may come to understand this theorem as follows: define the set S to be equal to $N(u) \cap N(v)$. If S is not 1-colorable, then we can make Type 3 Progress; u and v must be the same color, for obvious reasons (2 colors must be used to color S; u and v must share the final color). We may therefore assume that S is monochromatic; then, if $|N(S)| \leq n/k$, we have made Type 2 Progress, since the set S is 2-colorable and has size n/k^2. If $|N(S)| \geq n/k$, then we make Type 1 Progress. We know we can make Type 1 Progress in this case, since the set $N(S)$ is 2-colorable (it must be; S is monochromatic) and large.

5 Combinatorial 3-coloring approximations

Below we describe two combinatorial 3-coloring approximations that make use of second neighborhoods. We first describe the framework for Blum's $\tilde{O}(n^{2/5})$ result, and then describe and analyze the work of Kawarabayashi and Thorup which builds upon Blum's techniques resulting in an algorithm using $\tilde{O}(n^{4/11})$ colors.

5.1 Blum's $\tilde{O}(n^{2/5})$ approximation

Blum's algorithm attempts to make progress by looking at second neighborhoods. It either makes trivial progress, or finds independent sets inside of this neighborhood. This approach was inspired by thinking about the structure of random graphs, but Blum restricts the structure of these second neighborhoods to have the desired properties even for general graphs. The details come in dealing with these structures.

5.1.1 Preliminaries

In order to understand Blum's preliminary $\tilde{O}(n^{2/5})$ combinatorial algorithm, define a parameter $\delta = 1/(5 \log n)$. Additionally, we divide the vertices into bins I_j based on their degree; each bin I_j contains those vertices whose degree is at least $(1+\delta)^j$ and less than $(1+\delta)^{j+1}$. Similarly, we define a "specific neighborhood" $N_i(S)$, which is like the previous definition of a "neighborhood", except only including vertices for whom the number of edges connected to S is at least $(1+\delta)^i$ and less than $(1+\delta)^{i+1}$.

5.1.2 The algorithm

Now, we explain the algorithm. First, we can assume that all vertices have at least k neighbors, since we can otherwise make progress of Type 2. Additionally, we can assume that no two vertices have a neighborhood of size $O(n/k^2)$, since that would allow us to make progress using Blum's Theorem.

We take the remaining graph, and for each vertex v and each $i, j \in 1, 2, 3, ...5 \log^2 n$, we define $T_{v,i,j} = N_i(N(v) \cap I_j)$. In other words, $T_{v,i,j}$ is the set of distance-2 neighbors of v, considering only neighbors that have degree in bin I_j and only distance-2 neighbors that have between $(1+\delta)^i$ and $(1+\delta)^{i+1}$ connections to those neighbors.

On each of these $T_{v,i,j}$, Blum gives a method to determine an independent set in polynomial time. We do not describe this method here, but it relies on the BE/MS Vertex-Cover approximation algorithm. The resulting independent set will have size $\Omega(n^{3/5}/(\log n)^{8/5})$, resulting in Type 1 Progress.

5.1.3 Independent sets can always be found

It remains simply to prove that such a set will be found each time. Blum does this using the following logic:

Theorem 7 *Given a graph with average vertex degree d and an independent set R such that $\Omega(1)$ of the edges in the graph are incident to vertices of R and the average degree of R is at least d, there exists some vertex v and some bin I_j satisfying the following two conditions: (1) The number of neighbors of v that are in bin I_j is large $(\Omega(\delta^2 d/\log_{1+\delta} n))$. (2) A $\Omega(1-3\delta)$-fraction of the edges from neighbors of v in bin I_j are incident to R.*

Theorem 8 *Consider any set S for which an $\Omega(1)$-fraction of the set is incident to another set R. For some j, a significant fraction $(\Omega(\delta/\log_{1+\delta} n))$ of edges (s, r) between S and R involve a vertex $s \in S$ in bin I_j. Furthermore, a significant fraction $(\Omega(1-2\delta))$ of the neighbors of the elements of $S \cap I_j$ are in R.*

This makes a good deal of sense–since the degrees inside degree bins increase exponentially, one of them must hold many of the edges. Blum proves these results by means of the pigeonhole principle. Having shown these two theorems, Blum combines them. In the optimally-colored graph, he knows that there exists some color (call it red) for which it is true that most of the edges going out of vertices of other colors are going into vertices of that color (this must be true of the most common color.)

For some set S, he analyzes the number of edges between the red vertices with bounded degree and that set. By Theorem 7, there must be many edges like these, because the number of edges between red vertices with bounded degree and the set S is asymptotically equivalent to the number of edges between all red vertices and S, and some constant fraction of those edges are incident on a large independent set.

By Theorem 8, he will be able to find some vertex and some degree-bound for which there are a lot of vertices with that degree who are neighbors of that vertex. This all together implies that when he searches the second neighborhood of every vertex for potential large independent sets, he will find one.

This logic is the basis for Blum's combinatorial $n^{0.4}$-coloring algorithm. He later explains a better algorithm–$n^{0.375}$–an improvement that gains using much the same ideas as above, but using a different method for coloring high density regions of the graph.

5.2 Kawarabayashi and Thorup - an $\tilde{O}(n^{4/11})$ coloring algorithm

Like Blum, the basic approach of the method of [3] is to find induced subgraphs of the second neighborhood of some vertex with a specific structure. The method then either makes progress toward a k-coloring, or finds a cut giving a smaller section of the second neighborhood graph on which to recurse. This is done until all vertices are assigned a color. Here, the goal number of colors, k, is:

$$k = \tilde{\theta}((n/\Delta_{min})^{4/7}) \tag{1}$$

where Δ_{min} is the minimum vertex degree. We will discuss in the analysis how this color bound was obtained.

Their method heavily relies on **Blum's Corollary** which is reiterated here: Given a vertex set $X \subset V$ of size at least $\Psi = n/k^2$, in polynomial time we can either make progress towards an $O(k)$-coloring of G or can guarantee that X must be multichromatic. A repeated pattern of the algorithm described below is: any time we have a set of vertices this size, we use this lemma to say that if we make progress, we're done, and so we assume that we did not make progress.

5.2.1 Second neighborhood structure

Before describing the algorithm, we describe the second neighborhood structure used by [3]. This structure consists of a root vertex r_0, labeled red, along with subsets of its first and second neighbors $S_0 \subset N(r_0)$ and $T_0 \subset N(S_0)$, respectively. The structure has the following constraints:

- $|S_0| \geq \Delta_0$
- $|T_0| \leq \frac{n}{k}$
- All edges are between r_0 and S_0 or between S_0 and T_0.
- All vertices in S_0 have average degree Δ_0 into T_0.
- The degrees from T_0 to S_0 are within a factor of $(1 \pm o(1))$ around an average $\delta_0 \geq \Delta_0^2 \frac{k}{n}$.
- The method looks at subgraphs $(S, T) \subseteq (S_0, T_0)$ and will always maintain that there are more than $\Psi = n/k^2$ vertices of degree more than $\delta_0/16$ in T. Call these vertices **high-degree vertices**.

where $\Delta_0 = \tilde{\Omega}(\Delta_{min})$. In [2] Blum gives a method using vertex cover approximation to find such a structure in polynomial time, and we do not present that here. The size and degree bounds will be important later in describing how we can make progress and in deriving the bound on the number of colors. For now we are just concerned about the general structure of the graph. This is visualized in **Figure 3**.

5.3 Cut-or-color method

The algorithm first determines a subgraph of the structure described above by choosing an S and T such that the degree constraints are met. The core method that acts on this subgraph is called **cut-or-color**. Let U be the set of high degree vertices in T. The method then takes an arbitrary $t \in U$ and finds either:

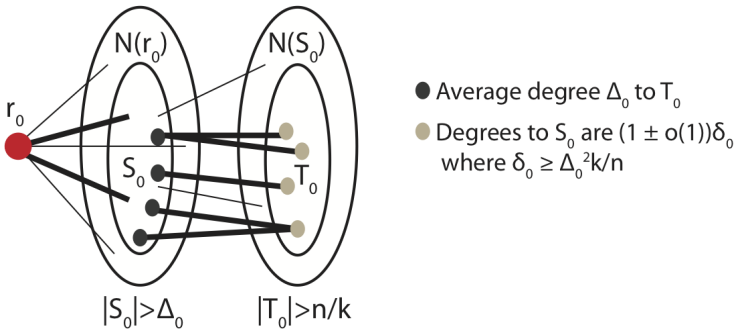

Figure 3: The second neighborhood structure used by [3]. The structure gives a subgraph of the second neighborhood of a red vertex r_0 with degree constraints on vertices in each subset.

- (1) A *sparse cut* giving a subproblem $(S', T') \subseteq (S, T)$ on which to recurse.
- (2) Progress toward k-coloring.
- (3) A guarantee that if r and t have different colors in some 3-coloring, then S is monochromatic in that 3-coloring.

These three outcomes are show in **Figure 4**.

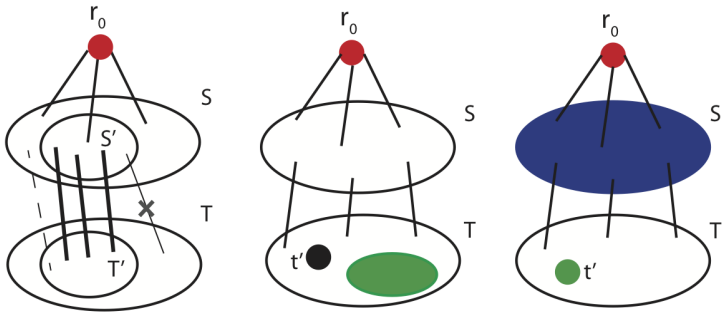

Figure 4: Three outcomes of the cut-or-color method. Either a sparse cut, progress, or a guarantee of monochromicity of S results from an application of cut-or-color.

If cut-or-color finds a sparse cut, we can then just recurse. If it makes progress, we are done. If we do neither after looking at all high degree vertices, we know S is monochromatic: If r_0 is red, S must be blue and green, and U is assumed multichromatic, which we can test using Blum's Corollary. So there must be one colored different than r_0, and by (3) of cut-or-color S must be monochromatic.

We now describe the cut-or-color method. The basic idea is: pick a high degree vertex t which we assume has a different color than r_0, say green. Let X be the neighborhood of t in S and Y be the neighborhood of X in T. We will expand both X in S and Y in T as far as we can while maintaining that: (1) X is all blue and (2) Y has no blue (**Figure 5A**). If we get that $X = S$, we get outcome (3) that says S is monochromatic. Else, we declare a sparse cut (outcome (1)).

Now we describe how X and Y are extended. First we look at X-extensions. Consider a vertex s with at least Ψ edges into Y. We test the neighbors of s in Y using Blum's Corollary, and assume they are multichromatic

(else we make progress and have outcome (2)). Y has no blue, and so s must have both red and green neighbors, and so s must be blue. Add s to X and all neighbors of s in T to Y (**Figure 5B**).

Now we describe Y-extensions. Consider a vertex t' in $T \backslash Y$. Let X' be its neighborhood in S and Y' its neighorhood in T. If $|Y \cap Y'| \geq \Psi$, we can use Blum's Corollary to check if it is multichromatic. If not, we make progress and get outcome (2). If so, we prove by contradiction that t' cannot be blue: if it were, then X' would be all green, and so $Y' \cap Y$ would be all red, which is a contradiction, since we said it is multichromatic. Therefore, t' is not blue and we can add it to Y (**Figure 5C**).

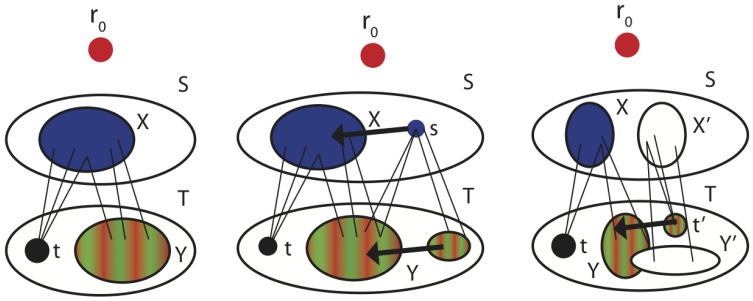

Figure 5: Extensions in cut-or-color. (A) The goal of extentions is to increase the size of the all blue set X and the non-blue set Y. (B) X-extensions look for a vertex that can be added to X (which also results in adding vertices to Y). (C) Y-extensions look for vertices that can be added to Y.

We extend X and Y as far as possible. Eventually, either we have made progress (outcome 2), found that $X = S$ and make monochromatic progress (outcome 3), or find a sparse cut, (X, Y) (outcome 1) defined loosely as having no cut edges between X and $T \backslash Y$ and few cut edges between Y and $S \backslash X$. We then call cut-or-color recursively on (X, Y) and will make further progress in the recursive calls.

5.4 Analysis

The crux of the analysis lies in demonstrating that the subproblems we consider when recursively applying the **cut-or-color** method always satisfy the desired conditions. In particular we need to ensure that the subproblems (S, T) considered have sufficiently many high-degree vertices in T. We construct a first induced subproblem $(S_1, T_1) \subseteq (S_0, T_0)$ by considering only high degree vertices in S_0 and T_0 by initially taking $(S_1, T_1) = (S_0, T_0)$ and repeatedly removing vertices from S_1 that have degree to T_1 less than $\Delta_1 := \Delta_0/4$ and vertices from T_1 with degree to S_1 below $\delta_1 := \delta_0/4$. This process removes at most $|S_0|\Delta_1 + |T_0|\delta_1 = |S_0|\Delta_0/4 + |T_0|\delta_0/4$ edges, which is half the number of edges between S_0 and T_0 since $|S_0|\Delta_0 = |T_0|\delta_0$ is the number of edges between S_0 and T_0. Consequently half of the edges between S_0 and T_0 are between S_1 and T_1. In addition, the average degree of edges in T_1 remains at least $2\delta_1$. This will ensure that the degrees from S_1 to T_1 are at least Δ_1 and that the degrees from T_1 to S_1 are between δ_1 and $(1 + o(1))\delta_1 < 5\delta_1$. We will guarantee inductively that whenever we consider a subproblem (S, T), the following pair of contraints is satisfied:

(a) For all vertices $v \in S$, $N(v) \cap T_1 \subset T$. This ensures that all vertices in S have average degree at least Δ_1 into T.

(b) The average degree from T into S is at least $\delta_1/2$.

If these conditions are satisfied for a problem (S, T) and **cut-or-color** invokes a new subproblem $(X, Y) \subseteq (S, T)$, then it is clear from the **cut-or-color** procedure that the first condition is satisfied for the subproblem

(X, Y). It remains to show that the second condition is satisfied for (X, Y). Kawarabayashi and Thorup prove this by first showing that Y is sufficiently large and then that not too many edges from Y to $S \setminus X$ are cut. Since Y is large enough and vertices of Y have sufficiently many neighbors in S, there are many edges between Y and S. Since not too many of these edges are between Y and $S \setminus X$, it follows that there are many edges between Y and X, so the average degree from Y to X is sufficiently high. These results of Kawarabayashi and Thorup are encapsulated in the pair of lemmas stated below. The proofs of these lemmas are fairly technical and are omitted here.

Lemma 9 $|Y| \geq \Delta_1^2 k^2/(8n)$

Since each vertex $v \in Y \subseteq T_1$ has at least δ_1 edges to S_1, it follows from this that the number of edges from Y to S_1 is at least $\delta_1 \Delta_1^2 k^2/(8n)$.

Lemma 10 *The number of edges from Y to $S \setminus X$ is at most*

$$|T \setminus Y| \cdot \frac{40\delta_1 n^2}{\Delta_1^2 k^4} \tag{2}$$

The sets $T \setminus Y$ considered at the various levels of recursion are disjoint, since we recurse on the cut (X, Y) after processing the cut (S, T). Each set $T \setminus Y$ is a subset of T_1, so since they are disjoint, the sum of the sizes of the sets considered is at most $|T_1|$. Consequently we have that the number of edges cut in the entire recursion is at most

$$|T_1| \frac{40\delta_1 n^2}{\Delta_1^2 k^4} \leq \frac{40\delta_1 n^3}{\Delta_1^2 k^5}$$

since $|T_1| \leq |T_0| \leq n/k$. We wish to obtain that the number of edges cut is at most half the original number of edges, that is, that

$$\frac{40\delta_1 n^3}{\Delta_1^2 k^5} \leq \frac{\delta_1 \Delta_1^2 k^2}{16n} \iff k^7 \geq 640(n/\Delta_1)^4 \tag{3}$$

But since $\Delta_1 = \Omega(\Delta_0) = \tilde{\Omega}(\Delta_{\min})$, we can take $k = \tilde{O}((n/\Delta_{\min})^{4/7})$ so that this is satisfied. Consequently for such k we have that at least half of the edges from Y to S_1 are between Y and X. Since all vertices of $Y \subseteq T_1$ have degree at least δ_1, this implies that the average degree from Y into X is at least $\delta_1/2$, so condition (b) is satisfied for (X, Y) as well. By induction we have that conditions (a) and (b) are satisfied for each subproblem (S, T) we consider. We now show that these conditions imply that there are sufficiently many vertices of high degree in T.

Lemma 11 *If a cut (S, T) satisfies conditions (a) and (b), then there are more than $\Psi = n/k^2$ vertices of high degree in T.*

Proof: Let h be the proportion of high degree vertices in T. Since all vertices in T have degree at most $5\delta_1$ and vertices which are not high degree have degree at most $\delta_1/4$, we have that the average degree in T is at most $h5\delta_1 + (1-h)\delta_1/4$, which is at least $\delta_1/2$ by condition (b). Therefore we have that $h = \Omega(1)$. By condition (a) we have that a vertex in S has at least Δ_1 neighbors in T, so in particular $|T| \geq \Delta_1$. Therefore the number of high degree vertices in T is $h|T| = \Omega(\Delta_1)$. But $\Delta_1 = \Delta_0/4 = \tilde{\Omega}(\Delta_{\min})$, so since $k = \tilde{O}((n/\Delta_{\min})^{4/7})$ it follows that $\Delta_1 = \tilde{\Omega}(n/k^{7/4}) = \tilde{\Omega}(\Psi k^{1/4})$, so the number of high degree vertices in T is $\tilde{\Omega}(\Phi k^{1/4})$. Consequently there are at least Ψ vertices of high degree in T. ■

6 Conclusion

The **cut-or-color** procedure makes progress in polynomial time toward k-coloring a 3-colorable graph for $k = \tilde{O}((n/\Delta_{\text{min}})^{4/7})$. Since we can trivially make progress toward a k-coloring if there are any vertices of degree less than k, we can assume that $\Delta_{\text{min}} \geq k$. Consequently we can make progress toward k-coloring a 3-colorable graph $k = \tilde{O}((n/k)^{4/7})$, which holds for $k = \tilde{O}(n^{4/11})$. Therefore, the combinatorial method of Kawarabayashi and Thorup can color a 3-colorable graph on n vertices with $\tilde{O}(n^{4/11})$ colors in polynomial time. This is an improvement over the combinatorial method of Blum, which used $\tilde{O}(n^{3/8})$ colors. Like Blum's algorithm, Kawarabayashi's can be combined with semidefinite programming to obtain an improved coloring algorithm. Combining Kawarabayashi's algorithm with the method of semidefinite programming yields an algorithm that can color a 3-colorable graph with $O(n^{0.2038})$ colors in polynomial time. This is only an incremental improvement over the previous best bound of $O(n^{0.2072})$ colors for coloring a 3-colorable graph in polynomial time. Blum and others have postulated that using different methods such as looking at the third neighborhoods could be useful, but to our knowledge no progress has been made on such methods.

References

[1] Bonnie Berger and John Rompel. A better performance guarantee for approximate graph coloring. *Algorithmica*, 5:459–466, 1990.

[2] Avrim Blum. New approximation algorithms for graph coloring. *Journal of the ACM*, 41:470–516, 1994.

[3] Ken ichi Kawarabayashi and Mikkel Thorup. Combinatorial coloring of 3-colorable graphs. *CoRR*, abs/1205.1254, 2012.

[4] Avi Wigderson. Improving the performance guarantee for approximate graph coloring. *Journal of the ACM*, 30(4):729–735, October 1983.