Coloring 3-Colorable Graphs with Less than $n^{1/5}$ Colors

KEN-ICHI KAWARABAYASHI, National Institute of Informatics, Tokyo, Japan
MIKKEL THORUP, University of Copenhagen, Denmark

We consider the problem of coloring a 3-colorable graph in polynomial time using as few colors as possible. We first present a new combinatorial algorithm using $\widetilde{O}(n^{4/11})$ colors. This is the first combinatorial improvement since Blum's $\widetilde{O}(n^{3/8})$ bound from FOCS'90. Like Blum's algorithm, our new algorithm composes immediately with recent semi-definite programming approaches, and improves the best bound for the polynomial time algorithm for the coloring of 3-colorable graphs from $O(n^{0.2072})$ colors by Chlamtac from FOCS'07 to $O(n^{0.2049})$ colors.

Next, we develop a new recursion tailored for combination with semi-definite approaches, bringing us further down to $O(n^{0.19996})$ colors.

CCS Concepts: • **Networks → Network reliability**; • **Computer systems organization → Robotics**; **Embedded systems**; **Redundancy**

Additional Key Words and Phrases: 3-colorable graphs, combinatorial algorithm

ACM Reference Format:
Ken-Ichi Kawarabayashi and Mikkel Thorup. 2017. Coloring 3-colorable graphs with less than $n^{1/5}$ colors. J. ACM 64, 1, Article 4 (March 2017), 23 pages.
DOI: http://dx.doi.org/10.1145/3001582

1. INTRODUCTION

If ever you want to illustrate the difference between what we consider hard and easy to someone not from computer science, use the example of 2-coloring versus 3-coloring: suppose there is too much fighting in a class, and you want to split it so that no enemies end up in the same group. First, you try with a red and a blue group. Put someone in the red group, and everyone he dislikes in the blue group, everyone they dislike in the red group, and so forth. This is an easy systematic approach. Digging a bit deeper, if something goes wrong, you have an odd cycle, and it is easy to see that if you have a necklace with an odd number of red and blue beads, then the colors cannot alternate perfectly. This illustrates both efficient algorithms and the concept of a witness for the case of 2-coloring. Knowing that red and blue do not suffice, we might try introducing green, but this 3-coloring is already beyond what we believe computers can do.

This article combines results presented in preliminary form at the 53rd Symposium on Foundations of Computer Science, 2012 [Kawarabayashi and Thorup 2012] and at the 31st Symposium on Theoretical Aspects of Computer Science, 2014 [Kawarabayashi and Thorup 2014].
Ken-ichi Kawarabayashi's research is partly supported by JST ERATO Kawarabayashi Large Graph Project. Mikkel Thorup's research is partly supported by an Advanced Grant from the Danish Council for Independent Research under the Sapere Aude research carrier programme. Part of this research was done while the he was at AT&T Labs–Research.
Authors' addresses: K. Kawarabayashi, National Institute of Informatics, Tokyo, Japan; email: k_keniti@nii.ac.jp; M. Thorup, Department of Computer Science, University of Copenhagen, Copenhagen, Denmark; email: mikkel2thorup@gmail.com.

DOI: http://dx.doi.org/10.1145/3001582

Formally, a k-coloring of an undirected graph assigns k colors to the vertices. The coloring is only valid if no two adjacent vertices get the same color. The validity of coloring is trivially checked in linear time so deciding if a graph is k-colorable is in NP. However, recognizing 3-colorable graphs is a classic NP-hard problem. It was proved to be NP-hard by Garey, Johnson, and Stockmeyer at STOC'74 [Garey et al. 1976], and was the prime example of NP-hardness mentioned by Karp [1975]. In this article, we also focus on 3-colorable graphs.

Bipartite or 2-colorable graphs are very well-understood. How about tripartite or 3-colorable graphs? How can we reason about them if we cannot recognize them? 3-colorable graphs are obvious targets for any approach to NP-hard problems. With the approximation approach, given a 3-colorable graph, which is a graph with an unknown 3-coloring, we try to color it in polynomial time using as few colors as possible. The algorithm is allowed to fail or give up if the input graph was not 3-colorable. If a coloring is produced, we can always check that it is valid even if the input graph is not 3-colorable. This challenge has engaged many researchers. At STOC'82, Wigderson [1983] got down to $O(n^{1/2})$ colors for a graph with n vertices. Berger and Rompel [1990] improved this to $O((n/(\log n))^{1/2})$. Blum [1994] came with the first polynomial improvements, first to $\widetilde{O}(n^{2/5})$ colors at STOC'89, and then to $\widetilde{O}(n^{3/8})$ colors at FOCS'90.

The next big step from FOCS'94 was by Karger et al. [1998] using semi-definite programming (SDP). This came in the wake of Goemans and Williamson's seminal use of SDP for the max-cut problem in STOC'94 [Goemans and Williamson 1995]. For a graph with maximum degree $\Delta_{\max}$, Karger et al. got down to $O(\Delta_{\max}^{1/3})$ colors. Combining this with Wigderson's algorithm, they got down to $O(n^{1/4})$ colors. Later in 1997, Blum and Karger [1997] combined the SDP from Karger et al. [1998] with the Blum [1994] algorithm, yielding an improved bound of $\widetilde{O}(n^{3/14}) = \widetilde{O}(n^{0.2142})$. Later improvements on SDP have also been combined with Blum's algorithm. At STOC'06, Arora et al. [2006] got down to $O(n^{0.2111})$ colors. The proof in Arora et al. [2006] is based on the seminal construction of Arora et al. [2009] from STOC'04 which gives an $O(\sqrt{\log n})$ approximation algorithm for the sparsest cut problem. Finally, in FOCS'07, Chlamtac [2007] got down to $O(n^{0.2072})$ colors.

On the lower bound side, for general graphs, the chromatic number is inapproximable in polynomial time within factor $n^{1-\epsilon}$ for any constant $\epsilon > 0$, unless $coRP = NP$ [Feige and Kilian 1998; Håstad 1999]. This lower bound is much higher than the above-mentioned upper bounds for 3-colorable graphs. The lower bounds known for the coloring of 3-colorable graphs are much weaker. We know that it is NP-hard to get down to 4 colors [Guruswami and Khanna 2004; Khanna et al. 2000]. Recently, Dinur et al. [2009] proved that it is hard to color 3-colorable graphs with any constant number of colors (i.e., $O(1)$ colors) based on a variant of the Unique Games Conjecture. Some integrality gap results [Feige et al. 2004; Karger et al. 1998; Szegedy 1994] show that the simple SDP relaxation has an integrality gap at least $n^{0.157}$, but such a gap is not known for SDPs using levels of Lasserre lifting [Arora et al. 2006; Arora and Ge 2011; Chlamtac 2007].

In this article, we present the first improvement on the combinatorial side since Blum from FOCS'90 [Blum 1994]. Our first main result is as follows.

THEOREM 1.1. *There is a combinatorial (i.e., not using linear or semi-definite programming) polynomial time algorithm to color 3-colorable graphs with $\widetilde{O}(n^{4/11})$ colors.*

The standard combination of our new combinatorial coloring and the best SDP [Chlamtac 2007] brings us down to $O(n^{0.2049})$ colors. However, we proceed to develop a new recursion tailored for combination with SDP approaches, bringing us further down to $O(n^{0.19996})$ colors, which is our second main result.

THEOREM 1.2. *There is a polynomial time algorithm to color 3-colorable graphs with $O(n^{0.19996})$ colors.*

We note that getting down to $\widetilde{O}(n^{1/5})$ colors (as in the Theorem 1.2) has been considered as a natural and clean milestone for many years (see more details later). Below, we briefly mention our techniques.

Technique. For our basic combinatorial algorithm, we reuse a lot of tools pioneered by Blum [1994], but our overall strategy is more structural. The starting point is a 3-colorable graph $G = (V, E)$. We will be looking for sparse cuts that we can recurse over. When no more sparse cuts can be found (and if we are not done by other means), we will have crystallized a non-trivial vertex set X that we guarantee is monochromatic in every 3-coloring of G.

Next, we present a novel recursion tailored for integrating the above combinatorial approach with SDP. To appreciate it, let us first consider the current interplay between combinatorial and SDP approaches. So far, they have just been balanced via a degree parameter Δ. Using Blum's notion of progress, it suffices to work with graphs that either have minimum degree Δ or maximum degree Δ (this general statement is new to this article, but slightly less general statements are proved in Arora et al. [2006] and Blum and Karger [1997]). High minimum degree is good for combinatorial approaches, while low maximum degree is good for SDP approaches. The best bounds are obtained by choosing Δ to balance between the best SDP and combinatorial algorithms. We note that if a vertex has a degree below the coloring target k, then it is trivially colored after we have colored the rest of the graph, so we can always assume minimum degree $\Delta \geq k$.

On the combinatorial side, with Δ as minimum degree, the coloring bounds have followed the sequence $\widetilde{O}((n/\Delta)^{i/(2i-1)})$ for $i = 1, 2, 3, 4$. Since $\Delta \geq k$, this yields combinatorial colorings with $\widetilde{O}(n^{i/(3i-1)})$ colors. Here, $i = 1$ is by Wigderson in 1982 [Wigderson 1983], while $i = 2$ is by Blum in 1989, and $i = 3$ is by Blum in 1990 (both covered by Blum [1994]). Finally, $i = 4$ is our new combinatorial algorithm using $\widetilde{O}((n/\Delta)^{4/7}) = \widetilde{O}(n^{4/11})$ colors. For $i \to \infty$, the sequence approaches $\widetilde{O}((n/\Delta)^{1/2})$.

With maximum degree Δ, the first SDP solution of Karger et al. [1998] from FOCS'94, got $O(\Delta^{1/3})$ colors. Balancing this with yet to be found $\widetilde{O}((n/\Delta)^{1/2})$ coloring, would yield $\widetilde{O}(n^{1/5})$ colors, which, hence, became a natural and clean milestone.

Later, SDP approaches of Arora et al. [2006] and Chlamtac in FOCS'07 [Chlamtac 2007], respectively, have gotten down to $O(\Delta^{1/3-\varepsilon(n,\Delta)})$ colors, where $\varepsilon(n, \Delta) > 0$ is a small value that decreases as a complicated function of Δ. Thanks to these developments, Chlamtac [personal communication] stated that to get below $n^{1/5}$ colors, we would "only" need to get down to around $\widetilde{O}((n/\Delta)^{12/23})$ colors on the combinatorial side. This, however, is eight steps away in the current sequence where the first four steps have taken 20 years, each bringing in new combinatorial ideas.

Our goal is to improve the overall coloring bound in terms of n, and we will indeed get down to $o(n^{1/5})$ colors. To do so, we specifically target the connection between combinatorial and SDP approaches. Using our new combinatorial algorithm as a subroutine, we present a novel recursion that gets us down to $\widetilde{O}((n/\Delta)^{12/23})$ colors, but only for the large values of Δ needed for an optimal combination with SDP. In combination with Chlamtac's SDP [Chlamtac 2007], we get a polynomial time algorithm that colors any 3-colorable graph on n vertices with $O(n^{0.19996})$ colors.

We note that for smaller values of Δ, our new recursion does not offer any improvement over our new combinatorial bound $\widetilde{O}((n/\Delta)^{4/7})$. Instead of adding another independent dot, the new recursion connects the dots, improving the combinatorial side only in the parameter range of relevance for combination with SDP.

2. PRELIMINARIES INCLUDING INGREDIENTS FROM BLUM

To hide $\log n$ factors, we use the notation that $\widetilde{O}(x) \le x \log^{O(1)}(n)$, $\widetilde{\Omega}(x) \ge x/\log^{O(1)}(n)$, $\widetilde{o}(x) \le x/\log^{\omega(1)}(n)$, and $\widetilde{\omega}(x) \ge x\log^{\omega(1)}(n)$.

We are given a 3-colorable (simple undirected) graph $G = (V, E)$ with $n = |V|$ vertices. The (unknown) 3-colorings are red, green, and blue. For a vertex v, we let $N(v)$ denote its set of neighbors. For a vertex set $X \subseteq V$, let $N(X) = \bigcup_{v\in X} N(v)$ be the neighborhood of X. If Y is a vertex set, we use N_Y to denote neighbors in Y, so $N_Y(v) = N(v) \cap Y$ and $N_Y(X) = N(X) \cap Y$. We let $d_Y(v) = |N(v) \cap Y|$ and $d_Y(X) = \{d_Y(v) \mid v \in X\}$. Then, $\min d_Y(X)$, $\max d_Y(X)$, and $\operatorname{avg} d_Y(X)$ denote the minimum, maximum, and average degree from X to Y. Here, in our combinatorial algorithms, we shall use Δ as a lower bound for all degrees in the graph.

For some color target k depending on n, we wish to find an $\widetilde{O}(k)$ coloring of G in polynomial time. We are going to reuse several ideas and techniques from Blum's approach [Blum 1994]. So, let us summarize below Blum's techniques.

Progress. Blum has a general notion of *progress toward $\widetilde{O}(k)$ coloring* (or *progress* for short if k is understood). The basic idea is that such progress eventually leads to a full $\widetilde{O}(k)$ coloring of a graph. Blum presents three types of progress toward $\widetilde{O}(k)$ coloring:

Type 0: *Same color*. Finding vertices u and v that have the same color in every 3-coloring.
Type 1: *Large independent set*. Finding an independent vertex set X of size $\widetilde{\Omega}(n/k)$.
Type 2: *Small neighborhood*. Finding a non-empty independent vertex set X such that $|N(X)| = \widetilde{O}(k|X|)$.

In order to get from progress to actual coloring, we want k to be bounded by a *near-polynomial* function f of the vertex number n, where near-polynomial means that f is non-decreasing and that there are constants $c, c' > 1$ such that $cf(n) \le f(2n) \le c' f(n)$ for all n. As noted in Blum [1994], this includes any function of the form $f(n) = n^\alpha \log^\beta n$ for constants $\alpha > 0$ and β.

LEMMA 2.1 ([BLUM 1994, LEMMA 1]). *Let f be near-polynomial. If we, in time polynomial in n, can make progress toward $\widetilde{O}(f(n))$ coloring of either Type 0, 1, or 2, on any 3-colorable graph on n vertices, then, in time polynomial in n, we can $\widetilde{O}(f(n))$ color any 3-colorable graph on n vertices.*

We shall review the proof of Lemma 2.1 in Section 7 when we integrate the above types of progress with progress via SDP. Until then, all progress is understood to be of Type 0, 1, or 2.

Our general strategy now is to identify a small parameter k for which we can guarantee progress. To apply Lemma 2.1 and get a coloring, we want k to be a near-polynomial function of n. As soon as we find progress of one of the above types, we are done; so, generally, whenever we see a condition that implies progress, we may assume that the condition is not satisfied.

Our algorithm will focus on finding a set X, $|X| > 1$, that is guaranteed to be monochromatic in every 3-coloring. This will happen, assuming that we do not get other progress on the way. When we have the set X, we get same-color progress for any pair of vertices in X. We shall refer to this as *monochromatic progress*.

Below, we review some of the basic results that we need from Blum [1994] and Wigderson [1983], including simple proofs for completeness. First, as a trivial example of Type 2 progress, take the neighborhood of a degree d vertex. It gives progress toward

$\widetilde{O}(d)$ coloring. With degree lower bound Δ, we may, therefore, assume

$$k \leq \Delta / \log^a n \text{ for any constant } a. \tag{1}$$

Observation 2.2. *Both for the large independent set progress (Type 1) and for the small neighborhood progress (Type 2), it suffices to find a 2-colorable set X.*

Proof. If X is 2-colorable, we can find a 2-coloring in linear time. Let X_1 and X_2 be the two color classes, each being an independent set. Assume X_1 is the larger set. It is an independent set of size $|X_1| \geq |X|/2 = \widetilde{\Omega}(n/k)$. For the small neighborhood, we note $|N(X_1)|/|X_1| \leq |N(X)|/|X_1| \leq 2|N(X)|/|X| = \widetilde{O}(k)$. □

Consider a vertex v of degree d. The neighborhood of v is 2-colorable, so by Observation 2.2, we get Type 1 progress toward $\widetilde{O}(n/d) = \widetilde{O}(n/\Delta)$ coloring. Balancing with the color bound from Statement (1) implies the $\widetilde{O}(\sqrt{n})$ coloring from Wigderson [1983]. We only consider near-polynomial color targets $k = n^{\Omega(1)}$, so we may assume

$$\Delta = n^{1-\Omega(1)} \tag{2}$$

Much of our progress will be made via subroutines of Blum, presented below using a common parameter

$$\Psi = n/k^2. \tag{3}$$

We will only consider color targets k such that

$$k = (n/\Delta)^{1/2+\Omega(1)}. \tag{4}$$

Combining Equations (2) and (4), we get

$$\Delta = \Psi n^{\Omega(1)}. \tag{5}$$

The gap between Δ and Ψ will be the driving force behind most progress. A very useful tool from Blum [1994] is the following multichromatic test:

Lemma 2.3 ([Blum 1994, Corollary 4]). *Given a vertex set $X \subseteq V$ of size at least Ψ, in polynomial time, we can either make progress toward $\widetilde{O}(k)$-coloring of G, or else guarantee that under every legal 3-coloring of G, the set X is multichromatic.*

Proof. Note that if X is monochromatic in some 3-coloring, then X is independent and $N(X)$ is 2-colored. To prove the lemma, we check that X is independent and that $N(X)$ is 2-colorable. If either test fails, we know that X is multichromatic in every 3-coloring. Suppose no test fails. If $|N(X)| < n/k = k\Psi$, we have a small neighborhood of the independent set X. If $|N(X)| \geq n/k$, since $|N(N(X))| \leq n$, we have a small neighborhood of the 2-colored set $N(X)$, hence, Type 2 progress by Observation 2.2. □

In fact, Blum has a stronger lemma [Blum 1994, Lemma 12] guaranteeing not only that X is multichromatic, but that no single color is used by more than a fraction $(1 - 1/(4\log n))$ of the vertices in X. This stronger version is not needed here. Using Lemma 2.3, he proves:

Lemma 2.4 ([Blum 1994, Theorem 3]). *If two vertices have more then Ψ common neighbors, we can make progress toward $\widetilde{O}(k)$ coloring. Hence, we can assume that no two vertices have more than Ψ common neighbors.*

Proof. Suppose v and w have two different colors in a 3-coloring. Then, $M = N(v) \cap N(w)$ must be monochromatic with the third color. We apply Lemma 2.3 to M. If no progress is made, we conclude that M is multichromatic in every 3-coloring, hence, that v and w have the same color, and then we can make same color progress. □

Using this bound on joint neighborhoods, Blum proves the following lemma (which he never states in this general quotable form):

LEMMA 2.5. *If the vertices in a set Z on the average have d neighbors in U, then the whole set Z has at least* $\min\{d/\Psi, |Z|\}\, d/2$ *distinct neighbors in U.*

PROOF. If $d/\Psi \leq 2$, the result is trivial, so $d/\Psi \geq 2$. It suffices to prove the lemma for $|Z| \leq d/\Psi$, for if Z is larger, we restrict our attention to the d/Ψ vertices with most neighbors in U. Let the vertices in Z be ordered by decreasing degree into U. Let d_i be the degree of vertex v_i into U. We now study how the neighborhood of Z in U grows as we include the vertices v_i. When we add v_i, we know from Lemma 2.4 that its joint neighborhood with any (previous) vertex v_h, $h < i$ is of size at most Ψ. It follows that v_i adds at least $d_i - (i-1)\Psi$ new neighbors in U, so $|N(Z) \cap U| \geq \sum_{i=0}^{|Z|-1}(d_i - (i-1)\Psi) > |Z|d/2$. □

Two-Level Neighborhood Structure. The most complex ingredient we get from Blum [1994] is a certain regular second neighborhood structure. Let Δ be the smallest degree in the graph G.

For some $\Delta_0 = \widetilde{\Omega}(\Delta)$, Blum [1994, Theorems 7 and 8 and the Proof of Theorem 5] identifies, in polynomial time, a 2-level neighborhood structure $H_0 = (r_0, S_0, T_0)$ in G consisting of:

—A root vertex r_0. We assume r_0 is colored red in any 3-coloring.
—A first neighborhood $S_0 \subseteq N(r_0)$ of size at least Δ_0.
—A second neighborhood $T_0 \subseteq N(S_0)$ of size at most n/k. The sets S_0 and T_0 could overlap.
—The edges between vertices in H_0 are the same as those in G.
—The vertices in S_0 have average degree Δ_0 into T_0.
—The degrees from T_0 to S_0 are all between δ_0 and $(1+o(1))\delta_0$, where $\delta_0 \geq \Delta_0^2 k/n$.

Note that Blum [1994, Theorems 7 and 8] does not have the n/k size bound on T_0. Instead, there is a large set R of red vertices, leading to a large identifiable independent set constituting a constant fraction of T_0. If this set is of size $\widetilde{\Omega}(n/k)$, then Blum makes Type 1 progress toward a $\widetilde{O}(k)$ coloring as described in Blum [1994, Proof of Theorem 5], and we are done. Assuming that this did not happen, we have the size bound n/k on T_0.

Blum seeks progress directly in the above structure, but we are going to apply a series of pruning steps which either make progress, find a good cut recursing on one side, or identify a monochromatic set. That is why we already now used the subscript $_0$ to indicate the original structure provided by Blum [1994].

3. OUR COMBINATORIAL COLORING ALGORITHM

We will use Blum's 2-level neighborhood structure $H_0 = (r_0, S_0, T_0)$. For our combinatorial coloring, we will use a color target

$$k = \widetilde{\Theta}((n/\Delta)^{4/7}). \tag{6}$$

Trivially, this target satisfies the constraint on k from Equation (4). We are going to work on induced subproblems $(S, T) \subseteq (S_0, T_0)$ defined in terms of subsets $S \subseteq S_0$ and $T \subseteq T_0$. The edges considered in the subproblem are exactly those between S and T in G. This edge set is denoted $E(S, T)$.

With r_0 red in any 3-coloring, we know that all vertices in $S \subseteq S_0 \subseteq N(r_0)$ are blue or green. We say that a vertex in T has *high S-degree* if its degree to S is bigger than $\delta_0/16$ (almost a factor 16 below the minimum degree δ_0 from T_0 to S_0), and we will make sure that any subproblem (S, T) considered satisfies:

(i) We have at least Ψ vertices of high S-degree in T.

We note that (ii)–(vii) will be introduced later.

Cut-or-Color. We are going to implement a subroutine `cut-or-color`(t, S, T), which, for a problem $(S, T) \subseteq (S_0, T_0)$, starts with an arbitrary high S-degree vertex $t \in T$. It will have one of the following outcomes:

—Reporting a "sparse cut around a subproblem $(X, Y) \subseteq (S, T)$" with no cut edges between X and $T \setminus Y$ and only few cut edges between Y and $S \setminus X$. The exact definition of a sparse cut is complicated, but at this point, all we need to know is that `cut-or-color` may declare a sparse cut.
—Some progress toward k-coloring. If that happens, we are done, so we typically assume that this does not happen.
—A guarantee that if r and t have different colors in a 3-coloring C_3 of G, then S is monochromatic in C_3.

Recursing Toward a Monochromatic Set. Assuming an implementation of `cut-or-color`, we now describe our main recursive algorithm, `monochromatic`, which takes as input a subproblem (S, T). The pseudo-code is presented in Algorithm 1.

ALGORITHM 1: `monochromatic`(S, T)

```
let $U$ be the set of high $S$-degree vertices in $T$;
check that $U$ is multichromatic in $G$ with Lemma 2.3;
if there is a $t \in U$ such that cut-or-color$(S, T, t)$ returns "sparse cut around $(X, Y)$" then
    recursively call monochromatic$(X, Y)$
end
else
    return "$S$ is monochromatic in every 3-coloring"
end
```

Let U be the set of high S-degree vertices in T. By invariant (i), we have $|U| \geq \Psi$, so we can apply Blum's multichromatic test from Lemma 2.3 to U in G. Assuming we did not make progress, we know that U is multichromatic in every valid 3-coloring. We now apply `cut-or-color` to each $t \in U$, stopping only if a sparse cut is found or progress is made. If we make progress, we are done. If a sparse cut around a subproblem (X, Y) is found, we recurse on (X, Y).

The most interesting case is if we get neither progress nor a sparse cut.

LEMMA 3.1. *If `cut-or-color` does not find progress or a sparse cut for any high S-degree $t \in U$, then S is monochromatic in every 3-coloring of G.*

PROOF. Consider any 3-coloring C_3 of G. With Lemma 2.3, we checked that U is multichromatic in every 3-coloring of G, including C_3, so there is some $t \in U$ that has a different color than r_0 in C_3. With this t, `cut-or-color`(S, T, t) guarantees that S is monochromatic in C_3. Note that different 3-colorings may use a different t for the guarantee, and our algorithm does not need to know which t are used. □

Thus, unless other progress is made, `monochromatic` ends up with a set S that is monochromatic in every 3-coloring. We note that the high-degree vertices from invariant (i) imply that S has more than one vertex, so monochromatic progress can be made with S (see Lemma 6.2 below). However, the correctness demands that we respect invariant (i) and never apply `monochromatic` to a subproblem (S, T) where T has less than Ψ high S-degree vertices (otherwise, Lemma 2.3 cannot be applied to U). The proof that invariant (i) is respected will be based on a global analysis that can only be described after all the details of our algorithm are in place.

4. IMPLEMENTING CUT-OR-COLOR

We will now implement `cut-or-color`(S, T, t). The pseudo-code is presented as Algorithm 2. When we present and motivate our implementation of `cut-or-color`, we assume an arbitrary 3-coloring C_3 of G. The coloring C_3 is not known to the algorithm, and, in fact, it may not even exist. However, based on the assumption, if the algorithm can prove that S is monochromatic in C_3, then it can correctly declare that "S is monochromatic in every 3-coloring where t and r_0 have different colors." Below, we assume that r_0 is red and t is green in C_3. The last color is blue. The pseudo-code for `cut-or-color` in Algorithm 2 shows the raw code that will be executed regardless of what 3-colorings of G are possible.

ALGORITHM 2: `cut-or-color`(S, T, t)

```
$X = N_S(t)$; $Y = N_T(X)$;
loop
    if $X = S$ then
        return "$S$ is monochromatic in every 3-coloring where $t$ and $r_0$ have different colors"
    end
    else if there is $s \in S \setminus X$ with $|N_Y(s)| \geq \Psi$
    then // $X$-extension
        check that $N_Y(s)$ is multichromatic in $G$ with Lemma 2.3;
        add $s$ to $X$ and $N_T(s)$ to $Y$
    end
    else if there is $t' \in T \setminus Y$ with $|N_Y(N_S(t'))| \geq \Psi$
    then // $Y$-extension
        check that $N_Y(N_S(t'))$ is multichromatic in $G$ with Lemma 2.3;
        add $t'$ to $Y$
    end
    else // $X \neq S$ and no $X$- or $Y$-extension possible
        return "sparse cut around $(X, Y)$"
    end
```

The first part of `cut-or-color` is essentially the coloring that [Blum 1994, §5.2] uses for dense regions. We shall describe how we bypass the limits of his approach as soon as we have presented his part.

Let X be the neighborhood of t in S and let Y be the neighborhood of X in T. As in Blum [1994], we note that all of X must be blue, and that no vertex in Y can be blue. We are going to expand $X \subseteq S$ and $Y \subseteq T$, preserving the following invariant:

(ii) if r_0 was red and t was green in C_3, then X would be all blue and Y would have no blue.

If we end up with $X = S$, then invariant (ii) implies that S is monochromatic in any 3-coloring where r_0 and t have different colors.

X-Extension. Now consider any vertex $s \in S$ whose degree into Y is at least Ψ. Using Lemma 2.3, we can check that $N_Y(s)$ is multichromatic in G. Since $Y \supseteq N_Y(s)$ has no blue, we conclude that $N_Y(s)$ is red and green, and, hence, that s is blue. Note, conversely, that if s was green, then all its neighbors in Y would have to be red, and then the multichromatic test from Lemma 2.3 would have made progress. Preserving invariant (ii), we now add the blue s to X and all neighbors of s in T to Y. We shall refer to this as an X-extension.

Relation to Blum's Algorithm. Before continuing, let us briefly relate to the algorithm presented by Blum [1994]. The above X-extension is essentially the coloring Blum [1994, §5.2] uses for dense regions. He applies it directly to his structure $H_0 = (r_0, S_0, T_0)$ from Section 2 from some arbitrary $t \in T_0$, which is presumed green, that is, any color different from that of r_0. The initial set $X = N_{S_0}(t)$ is then of size at least $\delta_0 \geq \Delta_0^2 k/n$. By counting, he then proves that there are at least Ψ vertices $s \in S_0$ with degree at least Ψ into $Y = N_{t_0}(X)$. He performs the above X-extensions on all these s. If none of these X-extensions make direct progress, he ends up with a set X of size at least Ψ that is all blue by invariant (ii), assuming that r_0 was red and t was green. He then applies Lemma 2.3 to X. If Lemma 2.3 does not make progress, he concludes that X is multichromatic in all colorings, contradicting the assumption that r_0 and t had different colors. He can then make Type 0 progress, identifying r_0 and t.

In order to use fewer colors than Blum [1994], our algorithm has to work for degrees below $\Delta_0^2 k/n$ from T to S. As a result, our extended X will typically be of size below Ψ and then Lemma 2.3 cannot be applied to X. In fact, as we recurse, we will get sets S that themselves are much smaller than Ψ. Otherwise, we would be done with Lemma 2.3 if S was monochromatic.

Below, we introduce Y-extensions. They are similar in spirit to X-extensions, and would not help us if we, like Blum, worked directly with H_0. The important point will be that if we do not end up with $X = S$, and if neither extension is possible, then we have identified a sparse cut that we can use for recursion. We are, thus, borrowing from Blum's proof [Blum 1994] in some technical details, but the overall strategy, seeking sparse cuts for recursion to crystallize a small monochromatic set S, is entirely different and new. Indeed, this idea is the most critical in our combinatorial proof.

Y-Extension. We now describe a Y-extension, which is similar in spirit to the X-extension, but which will cause more trouble in the analysis. Consider a vertex t' from $T \setminus Y$. Let $X' = N_S(t')$ be its neighborhood in S. Suppose $|N_Y(X')| \geq \Psi$. Using Lemma 2.3, we check that $N_Y(X')$ is multichromatic in G. We now claim that t' cannot be blue, for suppose it was. Then, its neighborhood has no blue and S is only blue and green, so $X' = N_S(t')$ must be all green. Then, the neighborhood of X' has no green, but Y has no blue, so $N_Y(X')$ must be all red, contradicting that $N_Y(X')$ is multichromatic. We conclude that t' is not blue. Preserving invariant (ii), we now add t' to Y.

Closure. We are going to extend X and Y as long as possible or till $X = S$. Suppose we end up with $X = S$. By invariant (ii), X is blue, so `cut-or-color` declares that S is monochromatic in any 3-coloring where r and t have different colors.

Otherwise, we are in a situation where no X-extension nor Y-extension is possible, and then `cut-or-color` will declare a sparse cut around (X, Y). A *sparse cut around* (X, Y) is simply defined as being obtained this way. It has the following properties:

(iii) The original high S-degree vertex t has all its neighbors from S in X, that is, $N_S(t) \subseteq X$.
(iv) All edges from X to T go to Y, so there are no edges between X and $T \setminus Y$. To see this, recall that when an X-extension adds s' to X, it includes all its neighbors in Y. The Y-extension does not change X.
(v) Each vertex $s' \in S \setminus X$ has $|N_Y(s')| < \Psi$.
(vi) Each vertex $t' \in T \setminus Y$ has $|N_Y(N_S(t'))| < \Psi$.

The most important point here is that this characterization of a sparse cut does not depend on the assumption that t and r have different colors in some 3-coloring. It only assumes that X and Y cannot be extended further.

Remark on Correctness of `cut-or-color`. It should be noted that the correctness of `cut-or-color` follows from invariant (ii), which is immediate from the construction. The only issue that remains is to ensure that we never end up considering a subproblem with too few high S-degree vertices for invariant (i), hence, where we cannot apply Lemma 2.3 to ensure that the high S-degree vertices have multiple colors (hence, not all the same color as r_0).

Below we describe our recursive algorithm. The above point will be made clearer in Lemma 6.4.

5. STARTING THE RECURSION

Before we can start our recursive algorithm, we need some slightly different degree constraints from those provided by Blum [1994] described in Section 2:

—The vertices in S_0 have average degree $\Delta_0 = \widetilde{\Omega}(\Delta)$ into T_0.
—The degrees from T_0 to S_0 are all between δ_0 and $(1+o(1))\delta_0$, where $\delta_0 \geq \Delta_0^2 k/n$.

We need some initial degree lower bounds, which are obtained simply by removing low-degree vertices creating our first induced subproblem $(S_1, T_1) \subseteq (S_0, T_0)$. Starting from $(S_1, T_1) = (S_0, T_0)$, we repeatedly remove vertices from S_1 with degree to T_1 below $\Delta_0/4$ and vertices from T_1 with degree to S_1 below $\delta_0/4$ until there are no such low-degree vertices left. The process eliminates less than $|S_0|\Delta_0/4+|T_0|\delta_0/4 = |E(S_0, T_0)|/2$ edges, so half the edges of $E(S_0, T_0)$ remain in $E(S_1, T_1)$. The point is that the average on the T-side only goes down when we remove low-degree vertices from S_0, and that can take away at most 1/4 of the edges. With $\Delta_1 = \Delta_0/4$ and $\delta_1 = \delta_0/4$, we get

—The degrees from S_1 to T_1 are at least Δ_1.
—The degrees from T_1 to S_1 are between δ_1 and $(1+o(1))\delta_0 < 5\delta_1$.

Note that $\delta_1 = \delta_0/4 \geq \Delta_0^2 k/(4n) = 4\Delta_1^2 k/n$, and that *high S-degree* in T can now be restated as a degree to S above $\delta_1/4 = \delta_0/16 = \Delta_1^2 k/n$.

Our recursion will start from $(S, T) = (S_1, T_1) \subseteq (S_0, T_0)$, that is, the first call to Algorithm 1 is `monochromatic`(S_1, T_1). This completes the description of our combinatorial coloring algorithm, which we know by the above remark is correct if invariant (i) is satisfied by all recursive calls.

6. ANALYSIS

We are now going to analyze our recursion. We are going to show that there is a $k = \widetilde{\Theta}((n/\Delta)^{4/7})$, as in Equation (6), such that `monochromatic`(S_1, T_1) will never consider a recursive subproblem $(S, T) \subseteq (S_1, T_1)$, violating invariant (i) with less than Ψ high S-degree in T; for then, `monochromatic`(S_1, T_1) must find progress toward an $\widetilde{O}(k)$ coloring.

Preparing for a later, more elaborate algorithm in Section 9, we consider here a general case where the starting point is any quadruple $(S_j, T_j, \Delta_j, \delta_j)$ satisfying certain *pre-conditions* presented below. To understand our first algorithm, it suffices to think of the case $j = 1$, and we will show $(S_1, T_1, \Delta_1, \delta_1)$ satisfies the pre-conditions.

As our first pre-condition, we want $(S_j, T_j) \subseteq (S_0, T_0)$ to be a *regular* subproblem satisfying:

—The degrees from S_j to T_j are at least Δ_j.
—The degrees from T_j to S_j are between δ_j and $5\delta_j$.

In addition, we have the following two pre-conditions:

$$\Delta_j = \widetilde{\Omega}(\Delta) \tag{7}$$

$$\delta_j \geq 4\Delta_j/\Psi. \tag{8}$$

Note that Pre-condition (7) together with the color bound from Statement (1) imply

$$k = \widetilde{o}(\Delta_j). \tag{9}$$

LEMMA 6.1. $(S_1, T_1, \Delta_1, \delta_1)$ *satisfies the above pre-conditions.*

PROOF. From the description in Section 5, it follows immediately that $(S_1, T_1, \Delta_1, \delta_1)$ is regular and satisfies Pre-condition (7), which also implies Equation (9). We also have $\delta_1 \geq 4\Delta_1^2 k/n$, and by Equation (9), $\Delta_1^2 k/n = \omega(\Delta_1 k^2/n) = \omega(\Delta_1/\Psi)$, so Pre-condition (8) is also satisfied. □

When analyzing the general case, we do not assume $k = \widetilde{\Theta}((n/\Delta)^{4/7})$, but only the general parameter constraints Equation (2) that $\Delta = n^{1-\Omega(1)}$ and Equation (5) that $\Delta = \Psi n^{\Omega(1)}$. The latter together with Pre-condition (7) implies

$$\Psi = \widetilde{o}(\Delta_j). \tag{10}$$

We now make the call `monochromatic`(S_j, T_j). For each recursive subproblem $(S, T) \subseteq (S_j, T_j)$ considered, we define *high S-degree* in T as being at least $\delta_j/4$. This agrees with the original recursion from (S_1, T_1) where high degree was defined as $\delta_0/16 = \delta_1/4$. For every subproblem $(S, T) \subseteq (S_j, T_j)$ considered, we need to make sure that invariant (i) is satisfied with at least Ψ high S-degree vertices in T.

LEMMA 6.2. *Invariant (i) implies $|S| > 1$, so if S is monochromatic in all 3-colorings, then monochromatic progress can be made.*

PROOF. By invariant (i), we have at least one high S-degree vertex in T. It has at least $\delta_j/4$ neighbors in S. By Pre-condition (8) and Relation (10), $\delta_j = \widetilde{\omega}(1)$, so $|S| = \widetilde{\omega}(1)$. □

LEMMA 6.3. *Each surviving vertex $v \in S$ preserves all its neighbors from T_j in T, so degrees from S to T remain at least Δ_j.*

PROOF. The statement follows inductively from the sparse cut invariant (iv), which says that every vertex included in X includes all its neighbors from T in Y. □

For every subproblem (S, T) considered, we will make sure that

(vii) The average degree from T to S is at least $\delta_j/2$.

The following lemma shows our new invariant (vii) implies our old invariant (i).

LEMMA 6.4. *(vii) implies (i).*

PROOF. If h is the fraction of high S-degree vertices in T, the average degree in T is at most $h5\delta_j + (1-h)\delta_j/4$, which by invariant (vii) is at least $\delta_j/2$. Hence, $h = \Omega(1)$. By Lemma 6.3, we have $|T| \geq \Delta_j$, so we have $\Omega(\Delta_j)$ high S-degree vertices in T. By Relation (10), this is much more than the Ψ high S-degree vertices required for invariant (i). □

Correctness is thus satisfied as long as each subproblem considered satisfies invariant (vii).

Inductively, when a recursive subproblem (S, T) is considered, we assume that invariant (vii) is satisfied. Suppose a sparse cut is declared around a new subproblem $(X, Y) \subseteq (S, T)$. In the general case, we will terminate if (X, Y) violates invariant (vii). We will show that this implies that Y is small compared with T_j. For the specific case, when we start from (S_1, T_1), we will show that Y is not small enough for a violation of invariant (vii).

As in Lemma 6.3, we note that the sparse cut invariant (iv) inductively implies that each vertex in X includes all its neighbors from T_j in Y, so

$$\min d_Y(X) \geq \Delta_j. \tag{11}$$

In our original problem (S_j, T_j), each vertex $v \in Y$ had at least δ_j edges to S_j, so we have $\delta_j|Y|$ edges from Y to S_j. Hence, invariant (vii) follows if at least half of these go to X. To prove this, we seek a global bound on the number of edges cut by the recursion.

The following main technical lemma relates the number of new cut edges around the subproblem (X, Y) to the reduction $|T \setminus Y|$ in the size of the T-side:

LEMMA 6.5. *The number of cut edges from Y to $S \setminus X$ is bounded by*

$$|T \setminus Y| \frac{40\delta_j n^2}{\Delta_j^2 k^4}. \tag{12}$$

PROOF. First, we note that from invariant (v), we get a trivial bound of $\Psi|S \setminus X|$ on the number of new cut edges, but this is not strong enough for the Bound (12). Here, using invariant (vi), we get

$$\begin{aligned}\sum_{y\in Y} |N_T(N_S(y)) \setminus Y| = |\{y \in Y, t' \in T \setminus Y \mid N_S(y) \cap N_S(t') \neq \emptyset\}| = \sum_{t'\in T\setminus Y} |N_Y(N_S(t'))| \\ \leq |T \setminus Y|\Psi = |T \setminus Y|n/k^2. \end{aligned} \tag{13}$$

We will now, for any $y \in Y$, relate $|N_T(N_S(y)) \setminus Y|$ to the number $|N_S(y) \setminus X|$ of edges cut from y to $S \setminus X$. Let $Z = N_S(y) \setminus X$. By invariant (iv), we have that $N_T(N_S(y)) \setminus Y = N_T(Z) \setminus Y$. Consider any vertex $v \in Z$. By Inequality (11), the degree from v to T is at least Δ_j. Since $v \notin X$, by invariant (v), the degree from v to Y is at most Ψ, and by Relation (10), $\Psi = o(\Delta_j)$. The degree from v to $T \setminus Y$ is, therefore, at least $(1 - o(1))\Delta_j \geq \Delta_j/2$. This holds for every $v \in Z$. It follows from Pre-condition (8) that $|N_T(Z) \setminus Y| \geq \min\{(\Delta_j/2)/\Psi, |Z|\}\, \Delta_j/4$. Relative to $|Z|$, this is

$$\frac{|N_T(Z) \setminus Y|}{|Z|} \geq \min\left\{\frac{\Delta_j/(2\Psi)}{|Z|}, 1\right\} \Delta_j/4.$$

From our original configuration (S_j, T_j), we know that all degrees from T to S are bounded by $5\delta_j$ and this bounds the size of $Z \subseteq N_S(y)$. Therefore,

$$\frac{\Delta_j/(2\Psi)}{|Z|} \geq \frac{\Delta_j k^2/(2n)}{5\delta_j} = \frac{\Delta_j k^2}{10\delta_j n}.$$

By Pre-condition (8) $\delta_j \geq 4\Delta_j/\Psi = 4\Delta_j k^2/n$, so

$$\frac{\Delta_j k^2}{10\delta_j n} \leq 1/40 < 1.$$

Therefore,

$$\begin{aligned}\frac{|N_T(Z) \setminus Y|}{|Z|} &\geq \min\left\{\frac{\Delta_j/(2\Psi)}{|Z|}, 1\right\} \Delta_j/4 \\ &\geq \frac{\Delta_j k^2}{10\delta_j n} \Delta_j/4 = \frac{\Delta_j^2 k^2}{40\delta_j n}.\end{aligned}$$

Recalling $Z = N_S(y) \setminus X$ and $N_T(Z) \setminus Y = N_T(N_S(y)) \setminus Y$, we rewrite the inequality as

$$|N_S(y) \setminus X| \leq \frac{40\delta_j n}{\Delta_j^2 k^2} |N_T(N_S(y)) \setminus Y|.$$

Using Inequality (13), we now get the desired bound on the number of cut edges from Y to $S \setminus X$:

$$\sum_{y \in Y} |N_S(y) \setminus X| \leq \frac{40\delta_j n}{\Delta_j^2 k^2} \sum_{y \in Y} |N_T(N_S(y)) \setminus Y|$$

$$\leq \frac{40\delta_j n}{\Delta_j^2 k^2} |T \setminus Y| \, n/k^2 = |T \setminus Y| \frac{40\delta_j n^2}{\Delta_j^2 k^4}. \quad \square$$

From Lemma 6.5, it immediately follows that the total number of edges cut in the whole recursion is at most $|T_j|(40\delta_j n^2)/(\Delta_j^2 k^4)$. In particular, this bounds the number of edges cut from Y, that is,

$$|E(Y, S_j \setminus X)| \leq |T_j| \frac{40\delta_j n^2}{\Delta_j^2 k^4}. \tag{14}$$

We conclude:

THEOREM 6.6. *Suppose*

$$|Y| \geq |T_j| \frac{80 n^2}{\Delta_j^2 k^4}, \tag{15}$$

then the average degree from Y to X is at least $\delta_j/2$, so $(S', T') = (X, Y)$ satisfies invariants (vii) and (i).

PROOF. Each vertex from Y starts with at least δ_j edges to S_j, so by Inequality (14), the average degree to X is at least $\delta_j/2$ if

$$\delta_j |Y|/2 \geq |T_j| \frac{40\delta_j n^2}{\Delta_j^2 k^4} \iff |Y| \geq |T_j| \frac{80 n^2}{\Delta_j^2 k^4}. \quad \square$$

We will now consider lower bounds on $|Y|$ implying Condition (15) of Theorem 6.6. Let t be the high S-degree vertex we started with in T. Then, $\delta_S(t) \geq \delta_j/4$. By invariant (iii), X includes the whole neighborhood $N_S(t)$ of t in S. By Pre-condition (8)

$$|X| \geq \delta_j/4 \geq \Delta_j/\Psi. \tag{16}$$

With Inequalities (16) and (11), Lemma 2.5 immediately implies

$$|Y| \geq \Delta_j^2/(2\Psi) = \Delta_j^2 k^2/(2n). \tag{17}$$

Thus, Condition (15) is satisfied if

$$\Delta_j^2 k^2/(2n) \geq |T_j| \frac{80 n^2}{\Delta_j^2 k^4} \iff \Delta_j^4 k^6 \geq 160\, |T_j|\, n^3. \tag{18}$$

If condition (18) is satisfied, then we will never get a violation of invariants (i) and (vii).

Finally, we return to our original starting point $(S_1, T_1, \Delta_1, \delta_1)$ from Section 5, which, by Lemma 6.1, follows the general case. Since $T_1 \subseteq T_0$ and $|T_0| \leq n/k$, we get Condition (18) satisfied if

$$\Delta_1^4 k^6 \geq 160\,(n/k) n^3 \iff k \geq 160^{1/7}\,(n/\Delta_1)^{4/7}.$$

By Pre-condition (7), we have $\Delta_1 = \widetilde{\Theta}(\Delta)$, so setting

$$k = 160^{1/7}\,(n/\Delta_1)^{4/7} = \widetilde{\Theta}((n/\Delta)^{4/7}),$$

we satisfy Condition (18). This is the k from (6) that we use in our purely combinatorial coloring algorithm, which just calls monochromatic(S_1, T_1). Thus, we conclude

THEOREM 6.7. *If a 3-colorable graph has minimum degree Δ, then we can make progress of Type 0, 1, or 2, toward $\widetilde{O}((n/\Delta)^{4/7})$ coloring in polynomial time.*

The corresponding result of Blum [1994] was that we could make progress toward $\widetilde{O}((n/\Delta)^{3/5})$ coloring.

Finally, we obtain the corollary below, which is the same as Theorem 1.1 from the introduction.

COROLLARY 6.8. *A 3-colorable graph on n vertices can be colored with $\widetilde{O}(n^{4/11})$ colors in polynomial time.*

PROOF. Since $n^{4/11}$ is a near-polynomial function in n, by Lemma 2.1, it suffices to prove progress toward $\widetilde{O}(n^{4/11})$ coloring. If $\Delta = \widetilde{O}(n^{4/11})$, then, as argued in the paragraph preceding Equation (1), a single low-degree vertex and its neighbors give Type 2 progress. Otherwise, $\Delta = \widetilde{\Omega}(n^{4/11})$, and then by Theorem 6.7, we get progress toward $\widetilde{O}((n/\Delta)^{4/7}) = \widetilde{O}((n/n^{4/11})^{4/7}) = \widetilde{\Omega}(n^{4/11})$ coloring. □

7. INTEGRATION WITH SDP: ALL SMALL OR ALL LARGE DEGREES

In the introduction, we claimed, for any parameter Δ, that it suffices to consider either *low-degree graphs* with all degrees below Δ, which is good for current semi-definite approaches, or *high-degree graphs* with all degrees above Δ, which is good for current combinatorial approaches. Similar, but somewhat more specialized statements have been proved in Arora et al. [2006] and Blum and Karger [1997]. Below, we briefly discuss the main ideas from the constructions in Arora et al. [2006] and Blum and Karger [1997].

In Blum and Karger [1997], they combine Blum's combinatorial algorithm [Blum 1994] with the now classic SDP algorithm of Karger et al. [1998]. They observe that to apply the algorithm of Karger et al. [1998], we can orient the edges, and then we only need a bound Δ on the out-degree. As they look for progress with Blum's algorithm (c.f. Section 2), whenever they encounter a vertex of degree below Δ, they remove it. This ensures that they are always looking for progress in a graph with degrees above Δ. Now, if half the vertices get removed, they consider instead the graph induced by the deleted vertices. With edges oriented toward later deletions, their out-degrees are bounded by Δ. Applying the algorithm of Karger et al. [1998], they get a large independent set for Type 1 progress. We note that this sketch is far from a real proof because Type 2 progress on the reduced high-degree graph may not translate into Type 2 progress on the original graph.

For the SDP approach in Arora et al. [2006], they want the standard undirected degree to be bounded, so they suggest a complimentary approach. They state that Blum's algorithm is good for progress as long as the average degree is high, so they run it as long as the average degree is above Δ. If the average degree drops below Δ, they delete all vertices of degrees bigger than 2Δ, leaving them a graph with at least half as many vertices, and all degrees below 2Δ. Now they can apply their SDP approach.

As described above, Blum and Karger [1997] tailor the small-degree graph for Karger et al. [1998], exploiting the fact that a small out-degree suffices, and Arora et al. [2006] tailor the high-degree graph for Blum [1994], exploiting the fact that a high average degree suffices. Our reduction makes a clean split between low- and high-degree graphs:

PROPOSITION 7.1. *Let f and d be near-polynomial functions and assume $d(n/2) = (1/2 + \Omega(1))d(n)$. Suppose that in time polynomial in n, we can make progress toward $\widetilde{O}(f(n))$ coloring of Type 0, 1, or 2 on any graph from the following two classes:*

—*3-colorable "large-degree" graphs on n vertices with minimum degree $\geq d(n)$.*
—*3-colorable "small-degree" graphs on n vertices with maximum degree $\leq d(n)$.*

Then, in time polynomial in n, we find an $\widetilde{O}(f(n))$-coloring of any 3-colorable graph.

Generally, the idea is to use combinatorial algorithms to find progress with the large degrees in Proposition 7.1 and SDP algorithms to find progress with the small degrees. The rest of this section is devoted to the proof of Proposition 7.1. The basic idea is the same as Blum and Karger [1997]. As we look for progress toward coloring, whenever we meet a vertex v of degree below $d(n)$, we move it to a small-degree graph H. This includes all edges from v to vertices already in H. Nevertheless, the average degree in H remains bounded by $2d(n)$.

Our next step is simply to apply the following lemma twice.

LEMMA 7.2. *Consider a graph on n vertices with average degree d. Repeatedly delete a highest-degree vertex until $n/2 + 1$ vertices remain. The resulting induced subgraph has maximum degree at most d and average degree at most $d/2$.*

PROOF. Concerning the maximum degree, when we start, the sum of degrees is nd. For a contradiction, if a vertex of degree d remains, then all vertices removed had degree at least d, and when we remove such a vertex, the degree sum is reduced by $2d$. The remaining degree d vertex implies that the degree sum is at least $2d$, so we have removed at most $n/2 - 1$ vertices.

Concerning the average degree, we exploit the fact that each vertex removed has degree no less than the average. The first vertex, thus, has degree at least d. Removing it reduces the average degree of the remaining $n-1$ vertices to at most $d(1-1/(n-1)) = d(n-2)/(n-1)$. Inductively, when x vertices have been removed, we are down to an average degree of at most $d(n-x-1)/(n-1)$, which is at most $d/2$ for $x \geq n/2 - 1$. $\square$

This does not, however, suffice for the full generality of Proposition 7.1. Currently, Blum and Karger [1997] assume that SDP is applied to the small-degree graph, yielding a large independent set for Type 1 progress. However, in Proposition 7.1, we allow arbitrary progress on both the small and the large-degree graph. The non-obvious issue here is Type 2 progress based on a small neighborhood; for if we find such a small neighborhood in one of these induced subgraphs, then it may not correspond to a small neighborhood in the original graph. Therefore, we cannot apply Lemma 2.1 as a black box. To prove Proposition 7.1, we need to study the inner workings of the proof of Lemma 2.1.

Lemma 2.1 states that if we in time polynomial in n can make progress toward $\widetilde{O}(f(n))$ coloring of either Type 0, 1, or 2 on any 3-colorable graph on n vertices, then in time polynomial in n, we can $\widetilde{O}(f(n))$ color any 3-colorable graph on n vertices.

The simplest case is the same color progress of Type 0 with two vertices u and v that have the same color in all 3-colorings of G. We can then identify u and v in a new vertex w, removing u and v from the graph. Any edges that ended in u or v now end in w. The 3-colorings of G and the new graph G' are isomorphic, so G' is still 3-colorable, and when we have colored G', we color G, transferring the color of w to u and v. Working with G' is an advantage because $f(n)$ is non-decreasing in n.

When looking for other types of progress, we will be working with decreasing induced subgraphs of G. We note that if u and v have some color in all 3-colorings of a subgraph of G, then they must also have the same color in all 3-colorings of G, so Type 0 can be made on G. When this happens, we simply abandon any other progress we might be aiming for. Type 0 progress can happen at most n times, so this strategy cannot violate polynomial time. The fact that we do a complete restart with Type 0 progress means that we do not have to mix the Type 0 identification of vertices with other types of progress.

The other simple type of progress is Type 1 progress with an independent set X of size $\widetilde{\Omega}(n/f(n))$. The set X gets its own color (unless we find Type 0 progress and restart), and then we recurse on $G \setminus X$ using $\widetilde{O}(f(n-|X|))$ other colors. Because f is near-polynomial, we end up with at most $\widetilde{O}(f(n))$ colors by Lemma 2.1. Note that if an independent set X is found in an induced subgraph, then it is also independent in the original graph.

We would now be done if all the progress was of Type 0 or 1, and in fact, it is, for all other progress will lead to Type 1 progress with a large independent set. This includes both Type 2 progress with a small neighborhood and progress with SDP. However, we can no longer just study each type of progress on its own, for we will work iteratively, switching between different types of progress. To get the right interaction, we define it as a game played on a decreasing induced subgraph $G' = (V', E')$, $n' = |V'|$ of a 3-colorable graph $G = (V, E)$, $n = |V|$. We start with $G' = G$, and while we play, G' will always have at least $n/2$ vertices. Because G' is induced, an independent set of G' is also independent in G.

We have a constant number of players $i = 1, \dots, \ell = O(1)$ that may make progress on G'. Each player i starts with an empty vertex set V_i. When player i makes progress, he removes some vertices from G' and places them in his set V_i. The game stops when less than $n/2$ vertices remain. The conditional guarantee of player i is that if he ends up with $n_i = |V_i| \geq n/(2\ell) = \Omega(n)$ vertices, then he can find an independent set I_i of size $\widetilde{\Omega}(n/f(n))$ in the subgraph $G|V_i$ of G induced by V_i.

When we play, we always need someone to make progress on G', as long as it has $n' \geq n/2$ vertices. When done, the players all together removed more than $n/2$ vertices, so some player i ends up with $n_i \geq n/(2\ell) = \Omega(n)$ vertices. The condition of player i is satisfied so his independent set I_i is of size $\widetilde{\Omega}(n/f(n))$. This is the desired Type 1 progress for G.

We will now first play the game with a Player 1 and 2 using progress on G' of Type 1 and 2, respectively. If we find Type 0 progress on G', we restart as described above, so we may assume this does not happen.

Player 1 looks directly for a Type 1 progress on G', that is, an independent set X of size $\widetilde{\Omega}(n'/f(n')) = \widetilde{\Omega}(n/f(n))$. If he finds it, he just deletes the rest of the vertices, terminating the game with $I_1 = X$ and $V_1 = V'$.

More interestingly, we have a Player 2 looking for Type 2 progress on G' with a small neighborhood. Player 2 claims progress when he finds a non-empty independent vertex set X such that $|N(X)| = \widetilde{O}(f(n')|X|) = \widetilde{O}(f(n)|X|)$. He adds X to his independent set I_i and removes $X \cup N(X)$ from the graph, adding them to his set V_i. Because all neighbors of X are removed, the successive independent sets he gets are all independent of each other, so I_2 remains independent with $|V_2| = \widetilde{O}(f(n)|I_2|)$. Thus, if Player 2 ends up with $|V_2| = \Omega(n)$, then $|I_2| = \widetilde{\Omega}(n/f(n))$, as promised.

This completes our review of the proof of Lemma 2.1 from Blum [1994], based on progress of Type 0, 1, and 2. However, we are free to add more players $i = 3, 4, .., \ell = O(1)$ to the game, as long as player i guarantees that if he ends up with $n_i = |V_i| = \Omega(n)$ vertices, then he can find an independent set I_i of size $\widetilde{\Omega}(n/f(n))$ in $G|V_i$.

Now, as in Blum and Karger [1997], we introduce Player 3 with a parameter Δ looking for the following type of progress.

Type 3: Small degree. Finding a vertex of degree at most $\Delta = d(n)$, where n is the number of vertices in the initial graph G. When Player 3 finds such a vertex of degree at most Δ, he removes it from G' and places it in his set V_3.

We now have a complete game in the sense that for any subgraph G' of G considered, there will always be one of our players that can make progress. The point is that if Player 3 cannot make progress, then all degrees are above $\Delta = d(n) \geq d(n')$. We are then in the high-degree case of Proposition 7.1, where progress of Type 0, 1, or 2 is assumed possible.

To complete the proof of Proposition 7.1, we need to show that if Player 3 "wins" with $n_3 = \Omega(n)$ vertices, then, either we can make Type 0 progress, identify vertices, or we can find an independent set I_3 of size $\widetilde{\Omega}(n/f(n))$ in $G_3 = G|V_3$.

First we argue that the average degree in G_3 is bounded by 2Δ. To see this, note that when a vertex v is added to V_3, it has at most Δ edges to vertices w currently outside V_3. If such a w is later added to V_3, the edge (v, w) is included in $G|V_3$. This is the only way edges can end in $G_3 = G|V_3$, so we end up with at most $|V_3|\Delta$ edges, hence, with an average degree of at most 2Δ.

We now apply Lemma 7.2 repeatedly to G_3 until we get an induced subgraph G^* with n^* vertices such that the maximum degree is at most $d(n^*/2)$. The condition $d(n/2) = (1/2 + \Omega(1))d(n)$ from Proposition 7.1 implies that a constant number of applications suffice and that we end up with $n^* = \Omega(n)$ vertices.

We now start a new game on G^*, but looking only for progress of Type 0, 1, and 2, that is, Player 3 is no longer involved. We only play on induced subgraphs G' of G^* with $n' \geq n^*/2$ vertices. The maximal degree, therefore, remains bounded by $d(n^*/2) \leq d(n')$, so we will always be in the low-degree case of Proposition 7.1 where progress of Type 0, 1, or 2 is assumed possible. This is the case we analyzed before introducing Player 3, the difference being that progress is always possible. Either we get progress of Type 0, and then we are done by identification of two vertices, or we continue with progress of Type 1 or 2, until we have a winner providing us with an independent set I of G^* of size $\widetilde{\Omega}(n^*/f(n^*)) = \widetilde{\Omega}(n/f(n))$. Then, I is also an independent set in G_3, so this completes the proof of Proposition 7.1.

8. CHLAMTAC'S SDP

Contrasting the combinatorial approaches that benefit from a large minimum degree (c.f. Theorem 6.7), SDP coloring of 3-colorable graphs works best for graphs of low maximum degree bound $\Delta_{\max}$. The original result of Karger et al. [1998] was that we, in polynomial time, can find an independent set of size $\widetilde{\Omega}(n/\Delta_{\max}^{1/3})$, and, hence, make Type 1 progress toward an $\widetilde{O}(\Delta_{\max}^{1/3})$ coloring. The bound has been improved [Arora et al. 2006; Chlamtac 2007] to $O(\Delta^{1/3-\varepsilon(n,\Delta_{\max})})$, where $\varepsilon(n, \Delta) > 0$ is a small value that decreases as a complicated function of Δ. The strongest current bounds are due to the work of Chlamtac [2007, Theorem 15]. In Chlamtac [2007, Corollary 16] the author provided an instantiation of Theorem 15 from his work in 2007 Chlamtac [2007, Theorem 15], which was optimized for combination with Blum [1994] coloring. Chlamtac [personal communication] has provided us a corresponding instantiation of Chlamtac [2007, Theorem 15], optimized for combination with our Theorem 6.7:

THEOREM 8.1. *For any $\tau > \frac{6}{11}$, there is a $c > 0$ such that there is a polynomial time algorithm that for any 3-colorable graph G with n vertices, all degrees below $\Delta = n^\tau$ find an independent set of size $\widetilde{\Omega}(n/\Delta^{1/(3+3c)})$. Hence, we can make Type 2 progress toward an $\widetilde{O}(\Delta^{1/(3+3c)}) = \widetilde{O}(n^{\tau/(3+3c)})$-coloring.*

The requirement on τ and c is that $c < 1/2$ and $\lambda_{c,\tau}(\alpha) = 7/3 + c + \alpha^2/(1-\alpha^2) - (1+c)/\tau - (\sqrt{(1+\alpha)/2} + \sqrt{c(1-\alpha)/2})^2$ is positive for all $\alpha \in [0, \frac{c}{1+c}]$.

PROOF BY TRANSLATION FROM CHLAMTAC [2007]. Chlamtac [personal communication] helped us interpreting his Theorem 15 in Chlamtac [2007]. To see how our

Theorem 8.1 follows, we need to make reference to Theorem 15 with surrounding texts and definitions in Chlamtac [2007]. First, we note that Chlamtac [2007] wants the above requirements to be satisfied by some $c(\tau) > c$. It is, however, easy to see that if the constants τ and c satisfy the above requirements, then the requirements are also satisfied with a slightly larger constant $c' > c$. Also, Chlamtac [2007, Definition 14] defines the c-inefficiency of a parameter r (which looks a bit like τ but plays an entirely different role). We pick r to be exactly c-inefficient if the maximal degree is exactly $\Delta = \lceil n^{\tau} \rceil$. Then, as described in the paragraph above Theorem 15 in Chlamtac [2007], we get $N(r) = \widetilde{\Theta}(\Delta^{1/(3+3c)})$ and an independent set of size $\Omega(N(r)n) = \widetilde{\Omega}(n/\Delta^{1/(3+3c)})$ as stated in Theorem 8.1 above. $\square$

Using Mat-lab® to check the conditions in Theorem 8.1, we get the following concrete bounds:

COROLLARY 8.2. *For any 3-colorable graph G on n vertices,*

(i) if all degrees are below $\Delta_{\max} = O(n^{0.6415})$, then in polynomial time, we can find an independent set of size $\Omega(n^{0.7951})$, and hence, make Type 1 progress toward $\widetilde{O}(n^{0.2049})$ coloring. Here, $\widetilde{O}(n^{0.2049})$ is (slightly) bigger than $(n/\Delta_{\max})^{4/7}$.

(ii) if all degrees are below $O(n^{0.61673832})$, then in polynomial time, we can find an independent set of size $\Omega(n^{0.80004})$, and hence, make Type 1 progress toward $\widetilde{O}(n^{0.19996})$ coloring. Here, $\widetilde{O}(n^{0.19996})$ is bigger than $(n/\Delta_{\max})^{13/25}$.

(iii) if all degrees are below $O(n^{0.600309})$, then in polynomial time, we can find an independent set of size $\Omega(n^{0.80015})$, and hence, make Type 1 progress toward $\widetilde{O}(n^{0.19985})$ coloring. Here, $\widetilde{O}(n^{0.19985})$ is (slightly) bigger than $(n/\Delta_{\max})^{1/2}$.

Combining Corollary 8.2(i) with Proposition 7.1, Theorem 6.7, and Lemma 2.1, we immediately get the following corollary:

COROLLARY 8.3. *We can color any 3-colorable graph G on n vertices in polynomial time using $\widetilde{O}(n^{0.2049})$ colors.*

We are going to use Corollary 8.2 (ii) in the next section in conjunction with a new combinatorial recursion tailored for combination with SDP so as to get down below $n^{1/5}$ colors.

9. A NEW RECURSION FOR HIGH-DEGREE GRAPHS

We will now present a combinatorial recursion tailored for the high-degree graphs we get by combination with SDP. Complementing Corollary 8.2(ii), we set the color target k and the minimum degree bound Δ as follows.

$$k = n^{0.19996} \text{ and } \Delta = n^{0.61673832} \tag{19}$$

We note that $(n/\Delta)^{13/25} < k < (n/\Delta)^{12/23}$. With such a small value of k in the call `monochromatic`(S_1, T_1) (Algorithm 1), we can no longer guarantee that the average degree from Y to X remains above $\delta_1/2$ as required for invariant (vii).

To preserve the correctness, we simply stop our recursive Algorithm 1 if we get to a subproblem (X, Y) where the average degree from Y to X is less than $\delta_1/2$. Inside (X, Y), we find a new regular subproblem (S_2, T_2) where degrees are between δ_2 and $5\delta_2$ for some δ_2. Again, we apply Algorithm 1 until the average drops below $\delta_2/2$. We continue this new outer loop, generating a sequence of regular subproblems $(S_1, T_1) \supset (S_2, T_2) \supset (S_3, T_3) \supset \cdots$, until we somehow end up either making progress, or some error event happens. In combination with SDP, our analysis will show that this outer loop can be used to give error-free progress toward $\tilde{O}(n^{0.19996})$ coloring.

The regularization is described in Algorithm 3, and it is, in itself, fairly standard. Blum [1994] used several similar regularizations.

ALGORITHM 3: `regularize`(S, T)

```
Let $d_\ell = (4/3)^\ell$;
Partition the vertices of $T$ into sets $U_\ell = \{v \in T \mid d_S(v) \in [d_\ell, d_{\ell+1})\}$;
Subject to $d_\ell \geq \operatorname{avg} d_S(T)/2$ let $\ell$ maximize $|E(U_\ell, S)|$;
$\delta^r \leftarrow d_\ell/3$; $\Delta^r \leftarrow \operatorname{avg} d_{U_\ell}(S)/3$;
Repeatedly remove vertices $v \in S$ with $d_{U_\ell}(v) \leq \Delta^r$ and $w \in U_\ell$ with $d_S(w) \leq \delta^r$;
$S^r \leftarrow S$; $T^r \leftarrow U_\ell$;
return $(S^r, T^r, \Delta^r, \delta^r)$
```

LEMMA 9.1. *When* `regularize`(S, T) *in Algorithm 3 returns* $(S^r, T^r, \Delta^r, \delta^r)$*, then* $\Delta^r \geq \operatorname{avg} d_T(S)/(30 \lg n)$ *and* $\delta^r \geq \operatorname{avg} d_S(T)/8$*. The sets* S^r *and* T^r *are both non-empty. The degrees from* S^r *to* T^r *are at least* Δ^r *and the degrees from* T^r *to* S^r *are between* δ^r *and* $5\delta^r$*.*

PROOF. Below S and U_ℓ refers to the sets before vertices are removed. We have S^r and T^r denoting the sets after the vertices have been removed.

To prove $\Delta^r \geq \operatorname{avg} d_T(S)/(30 \lg n)$, we first note that the sets U_ℓ with $d_\ell < \operatorname{avg} d_S(T)/2$ only contain vertices of degrees below $(4/3)\operatorname{avg} d_S(T)/2 = (2/3)\operatorname{avg} d_S(T)$, so at least $1/3$ of the edges from $E(S, T)$ leave vertices from sets U_ℓ satisfying the condition $d_\ell \geq \operatorname{avg} d_S(T)/2$. There are only $\log_{4/3} n < (5/2)\lg n$ possible values of ℓ, and subject to the condition, we picked ℓ maximizing $E(S, U_\ell)$. Therefore, $|E(S, U_\ell)| > (1/3)|E(S, T)|/((5/2)\lg n) = (2/15)|E(S, T)|/\lg n$. It follows that $\Delta^r \geq \operatorname{avg} d_{U_\ell}(S)/4 > \operatorname{avg} d_T(S)/(30 \lg n)$.

The only other slightly non-trivial statement is that the sets S^r and T^r do not end up empty. When we remove vertices from S, we remove at most $|S|\operatorname{avg} d_{U_\ell}(S)/3 \leq |E(S, U_\ell)|/3$ edges, and likewise for the vertices removed from U_ℓ, so these removals take away at most two-thirds the edges. It follows that some edges remain, and hence, that $S^r, T^r \neq \emptyset$. □

Our new coloring is described in Algorithm 4. Except for the possible regularization, each round j is an iterative version of the recursive Algorithm 1. Moreover, we have made it self-checking in the sense that we report an error if the set U of high-degree vertices is too small for invariant (i) ("Error B" below). Also, we report an error if the set S is too small for monochromatic progress, which requires at least two same-color vertices ("Error A" below). With $k = \Theta((n/\Delta)^{4/7})$, our previous analysis shows that we never get an error and that we never get $|E(S, T)| \leq \delta_1|T|/2$, so the regularization never happens.

The outer loop in Algorithm 4 continues until it either makes an error or makes progress. The progress can either be explicit with a monochromatic set, or it can happen implicitly as part of the multichromatic test from Lemma 2.3. Ensuring that we make progress and no errors happen will require a very careful choice of parameters, and we will only gain over our combinatorial result (i.e., Theorem 1.1) when large minimum degree vertices are guaranteed from SDP as in Equation (19).

10. ANALYSIS OF OUTER LOOP

In this section, we will identity parameter settings for which our recursive Algorithm 4 will not make errors. Therefore, it will eventually make progress toward the desired coloring.

ALGORITHM 4: Seeking Progress Toward $\widetilde{O}(k)$ Coloring

```
let $(S_1, T_1, \Delta_1, \delta_1)$ be the initial two-level structure from Section 2;
for $j \leftarrow 1, 2, \ldots$ do    // outer loop, round $j$
    $(S, T) \leftarrow (S_j, T_j)$;
    repeat    // iterative version of recursive monochromatic$(S_j, T_j)$
        if $|S| \leq 1$ then return "Error A";
        $U \leftarrow \{v \in T \mid d_S(v) \geq \delta_j/4\}$;
        if $|U| < \Psi$ then return "Error B";
        check $U$ multichromatic with Lemma 2.3;    // if not, progress was found and we are done
        if $\exists t \in U$ such that cut-or-color$(S, T, t)$ returns "sparse cut around $(X, Y)$" then
            $(S, T) \leftarrow (X, Y)$
        end
        else return "$S$ is monochromatic in every 3-coloring, so monochromatic progress found"
    until $|E(S, T)| < \delta_j|T|/2$;
    $(S_{j+1}, T_{j+1}, \Delta_{j+1}, \delta_{j+1}) = $ regularize$(S, T)$;
end
```

Theorem 10.1. *Consider a 3-colorable graph on n vertices with all degrees above Δ where $\Delta = n^{1-\Omega(1)}$. Suppose for some integer $c = O(1)$ that*

$$k = \widetilde{\omega}\left((n/\Delta)^{\frac{2c+2}{4c+3}}\right), \tag{20}$$

and for all $j = 1, \ldots, c-1$,

$$(\Delta/k)(\Delta k/n)^j(\Delta k^2/n)^{j(j+1)} = \Delta^{j^2+2j+1}k^{2j^2+3j-1}/n^{j^2+2j} = \widetilde{\omega}(1). \tag{21}$$

Then, Algorithm 4 will make progress toward an $\widetilde{O}(k)$ coloring no later than round c.

If we did not have the j-bound Condition (21), we would just make c very large, with the bound in Condition (20) converging to $(n/\Delta)^{1/2}$.

The j-bound Condition (21) is rather unattractive, but we need it to make sure that no errors are made when $c > 1$. As an example, our previous bound $k = \widetilde{O}((n/\Delta)^{4/7})$ from Theorem 6.7 corresponds to the case $c = 1$ in Condition (20). To improve this bound, we need $c > 1$. In particular, we need to satisfy Condition (21) for $j = 1$, which becomes $\Delta^4k^4/n^3 = \widetilde{\omega}(1)$. Now, $k \leq (n/\Delta)^{4/7}$ implies $\Delta^4(n/\Delta)^{4/7\cdot 4}/n^3 = \Delta^{12/7}/n^{5/7} = \widetilde{\omega}(1) \iff \Delta = \widetilde{\omega}(n^{5/12})$. Thus, we can only make improvements over Theorem 6.7 if we restrict ourselves to sufficiently high-degree vertices, e.g., relying on SDP for lower-degree vertices.

Proof of Theorem 10.1. First, we note that the parameters in Theorem 10.1 satisfy the general parameter constraints of Equation (2) that $\Delta = n^{1-\Omega(1)}$ and Equation (5) that $\Delta = \Psi n^{\Omega(1)}$.

We now consider round $j = O(1)$, satisfying the Pre-conditions (7) and (8). We know from Lemma 6.1 that round $j = 1$ satisfies the pre-conditions. Inductively, for $j > 1$, we assume that all the previous rounds satisfied the pre-conditions, made no errors, and ended up regularizing.

Based on our previous analysis, it is easy to see that round j does not make errors; for Lemma 6.4 states that we cannot violate invariant (i) unless the average degree from T to S has dropped below $\delta_j/2$, but this is when round j got stopped. Error B is the violation of invariant (i), so Error B cannot happen. Also, by Lemma 6.2, invariant (i) implies that $|S| > 1$, so Error A cannot happen either.

Since no errors happen, we make progress in round j unless we regularize. The challenge is to show that if we regularize, then $j < c$ and then the pre-conditions will be satisfied for round $j+1$. Thus, assume that we end up regularizing in round j. Let X_j and Y_j denote the last values of X and Y. Then,

$$(S_{j+1}, T_{j+1}, \Delta_{j+1}, \delta_{j+1}) = \texttt{regularize}(X_j, Y_j).$$

By Lemma 9.1, this quadruple is regular. We need to show that it satisfies Pre-condition (7) that $\Delta_{j+1} = \widetilde{\Omega}(\Delta)$ and Pre-condition (8) that $\delta_{j+1} \geq 4\Delta_{j+1}/\Psi$. To do so, we derive inductive bounds on Δ_{j+1}, $|T_{j+1}|$, and δ_{j+1}. From Inequality (11) and Lemma 9.1, with $(S, T) = (X_j, Y_j)$, we get

$$\begin{aligned}\Delta_{j+1} &\geq \operatorname{avg} d_{Y_j}(X_j)/(30 \lg n) \geq \Delta_j/(30 \lg n) \\ &\geq \Delta_1/(30 \lg n)^j = \widetilde{\Omega}(\Delta).\end{aligned} \tag{22}$$

In particular, we conclude that Pre-condition (7) is satisfied for round $j+1$.

Round j terminates with (X_j, Y_j) because the average degree from Y_j to X_j is below $\delta_j/2$. Hence, it follows from Theorem 6.6 that $|Y_j| \leq |T_j|(80n^2)/(\Delta_j^2 k^4)$. Since $T_{j+1} \subseteq Y_j$, we get

$$\begin{aligned}|T_{j+1}| \;\leq\; |Y_j| \;&\leq\; |T_j|(80n^2)/(\Delta_j^2 k^4) \\ &=\; \widetilde{O}(|T_j| n^2/(\Delta^2 k^4)) \\ &=\; \widetilde{O}\left(|T_1| \left(\frac{n^2}{\Delta^2 k^4}\right)^j\right) \\ &=\; \widetilde{O}\left((n/k) \left(\frac{n^2}{\Delta^2 k^4}\right)^j\right).\end{aligned} \tag{23}$$

From Inequality (17), we know that any Y considered is of size at least $\Delta_j^2/(2\Psi) = \widetilde{\Omega}(\Delta^2/\Psi) = \widetilde{\Omega}(\Delta^2 k^2/n)$, so we must have

$$\begin{aligned}\Delta^2 k^2/n = \widetilde{O}\left((n/k)\left(\frac{n^2}{\Delta^2 k^4}\right)^j\right) \\ \iff k = \widetilde{O}\left((n/\Delta)^{\frac{2j+2}{4j+3}}\right).\end{aligned}$$

By our choice of k in Condition (20), $j < c$.

Our last challenge is to make sure that Pre-condition (8) is satisfied with $\delta_{j+1} \geq 4\Delta_j/\Psi = \widetilde{\Theta}(\Delta k^2/n)$. Applying Inequality (11) for the first inequality below, and Inequality (16), Relation (22), and Relation (23) for the second, we get

$$\begin{aligned}\operatorname{avg} d_{X_j}(Y_j) \;&\geq\; \Delta_j |X_j|/|Y_j| \\ &=\; \widetilde{\Omega}(\Delta)(\delta_j/4)/\widetilde{O}\left((n/k)\left(\frac{n^2}{\Delta^2 k^4}\right)^j\right). \\ &=\; \widetilde{\Omega}\left(\Delta\delta_j(k/n)\left(\frac{\Delta^2 k^4}{n^2}\right)^j\right).\end{aligned}$$

By Lemma 9.1, $\delta_{j+1} \geq \operatorname{avg} d_{X_j}(Y_j)/8$, so $\delta_{j+1} = \delta_j\, \widetilde{\Omega}((\Delta k/n)(\frac{\Delta^2 k^4}{n^2})^j)$. Since $j = O(1)$, it follows inductively that

$$\delta_{j+1} = \widetilde{\Omega}\left(\delta_1(\Delta k/n)^j \left(\frac{\Delta^2 k^4}{n^2}\right)^{j(j+1)/2}\right).$$

Here, $5\delta_1 \geq |S_1|\Delta_1/|T_1| = \widetilde{\Omega}(\Delta^2 k/n)$, so we get

$$\delta_{j+1} = \widetilde{\Omega}\left(\Delta(\Delta k/n)^{j+1} \left(\frac{\Delta k^2}{n}\right)^{j(j+1)}\right)$$

By Condition (21), we have $(\Delta/k)(\Delta k/n)^j(\Delta k^2/n)^{j(j+1)} = \widetilde{\omega}(1)$, so

$$\delta_{j+1} = \widetilde{\omega}(\Delta k^2/n) = \widetilde{\omega}(\Delta/\Psi) > 4\Delta_j/\Psi,$$

so Pre-condition (8) is indeed satisfied for round $j + 1$. This completes our proof of Theorem 10.1. □

Our main coloring result (Theorem 1.2) follows.

THEOREM 10.2. *In polynomial time, we can color any 3-colorable n vertex graph using* $\widetilde{O}(n^{0.19996})$ *colors.*

PROOF. Combining Corollary 8.2 (ii) with Proposition 7.1, for progress toward $k = \widetilde{O}(n^{0.19996})$ coloring, we may assume the minimum degree is at least $\Delta = n^{0.61674333}$, as stated in Equation (19). Then, $(n/\Delta)^{14/27} < k < (n/\Delta)^{12/23}$, so to satisfy Condition (20) in Theorem 10.1, we set $c = 6$. It is easily verified that Condition (21) is satisfied for $j = 1, \ldots, 5$. By Theorem 10.1, we conclude that progress is made within the first $c = 6$ rounds.

Incidentally, with our particular values of k and Δ, the j-bound Condition (21) reaches its minimum with $j = 5$. This implies that our bounds also hold with any larger c. □

11. CONCLUDING REMARKS

We have shown that we can color any 3-colorable n vertex graph using less than $n^{1/5}$ colors. Recall here that $n^{1/5}$ colors could have been obtained as a simple balancing with Proposition 7.1 between the clean $O(\Delta^{1/3})$ coloring from Karger et al. [1998] when all degrees are below Δ, and a yet to be discovered $O((n/\Delta)^{1/2})$ coloring when all degrees are above Δ. With the first combinatorial improvement in 23 years, we managed here to find a clean $O((n/\Delta)^{4/7})$ coloring. To get below $n^{1/5}$ colors, we made a new recursion tailored for combination with SDP, focusing only on the large-degree vertices that make SDP perform purely. This was then combined with the latest improvements on the SDP side by Chlamtac [2007]. For this combination, the general bounds on either side, that is, Theorem 8.1 and Theorem 10.1, are rather complicated, illustrating just how delicate the combination is. It is, however, this recursion, tailored for combination with SDP, which yields the biggest nominal improvement in the coloring bound in 17 years, illustrating that carefully targeting the combination of techniques can be far more powerful than the development of new individual techniques: the difference between adding dots and connecting the dots.

In some sense, this article marks the end of what we can currently hope to achieve combinatorially. Almost all combinatorial ideas for coloring 3-colorable graphs break down if we try to use less than $(n/\Delta)^{1/2}$ colors. Even getting down to $(n/\Delta)^{1/2}$ colors seems quite elusive at this point, and even if we did, the improvement in terms of n would be minimal. More precisely, in combination with Corollary 8.2 (iii), we would get down to $\widetilde{O}(n^{0.19985})$ colors, as compared with our current $\widetilde{O}(n^{0.19996})$ colors.

ACKNOWLEDGMENTS

We would like to thank a very thorough reviewer from *J. ACM* for many useful comments, which improve our presentation.

REFERENCES

S. Arora, E. Chlamtac, and M. Charikar. 2006. New approximation guarantee for chromatic number. In *Proc. 38th STOC*. 215–224.

S. Arora and R. Ge. 2011. New tools for graph coloring. In *Proc. APPROX-RANDOM*. 1–12.

S. Arora, S. Rao, and U. Vazirani. 2009. Expanders, geometric embeddings and graph partitioning. *J. ACM* 56, 2 (2009), 1–37. Announced at STOC'04.

B. Berger and J. Rompel. 1990. A better performance guarantee for approximate graph coloring. *Algorithmica* 5, 3 (1990), 459–466.

A. Blum. 1994. New approximation algorithms for graph coloring. *J. ACM* 41, 3 (1994), 470–516. Combines announcements from STOC'89 and FOCS'90.

A. Blum and D. R. Karger. 1997. An $\tilde{O}(n^{3/14})$-coloring algorithm for 3-colorable graphs. *Inf. Process. Lett.* 61, 1 (1997), 49–53.

E. Chlamtac. 2007. Approximation algorithms using hierarchies of semidefinite programming relaxations. In *Proc. 48th FOCS*. 691–701.

I. Dinur, E. Mossel, and O. Regev. 2009. Conditional hardness for approximate coloring. *SIAM J. Comput.* 39, 3 (2009), 843–873. Announced at STOC'06.

U. Feige and J. Kilian. 1998. Zero-knowledge and the chromatic number. *J. Comput. Syst. Sci.* 57 (1998), 187–199.

U. Feige, M. Langberg, and G. Schechtman. 2004. Graphs with tiny vector chromatic numbers and huge chromatic numbers. *SIAM J. Comput.* 33, 6 (2004), 1338–1368. Announced at FOCS'02.

M. R. Garey, D. S. Johnson, and L. J. Stockmeyer. 1976. Some simplified NP-complete graph problems. *Theor. Comput. Sci.* 1, 3 (1976), 237–267. Announced at STOC'74.

M. X. Goemans and D. P. Williamson. 1995. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM* 42, 6 (1995), 1115–1145. Announced at STOC'94.

V. Guruswami and S. Khanna. 2004. On the hardness of 4-coloring a 3-colorable graph. *SIAM J. Discrete Math.* 18, 1 (2004), 30–40.

J. Håstad. 1999. Clique is hard to approximate within $n^{1-\varepsilon}$. *Acta Math.* 182 (1999), 105–142.

D. R. Karger, R. Motwani, and M. Sudan. 1998. Approximate graph coloring by semidefinite programming. *J. ACM* 45, 2 (1998), 246–265. Announced at FOCS'94.

R. M. Karp. 1975. On the computational complexity of combinatorial problems. *Networks* 5 (1975), 45–68.

K. Kawarabayashi and M. Thorup. 2012. Combinatorial coloring of 3-colorable graphs. In *Proc. 53rd FOCS*. 68–75.

Ken-Ichi Kawarabayashi and Mikkel Thorup. 2014. Coloring 3-colorable graphs with $o(n^{1/5})$ colors. In *Proc. 31st STACS, LIPIcs 25*. 458–469. http://drops.dagstuhl.de/opus/volltexte/2014/4479

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

M. Szegedy. 1994. A note on the Θ number of Lovász and the generalized Delsarte bound. In *Proc. 35th FOCS*. 36–39.

A. Wigderson. 1983. Improving the performance guarantee for approximate graph coloring. *J. ACM* 30, 4 (1983), 729–735. Announced at STOC'82.

Received December 2014; revised November 2016; accepted November 2016