University of Illinois Urbana-Champaign
2024FA
CS598

Final Report

Pingbang Hu, Sean Liu
NetID: pbb, zxliu2

December 9, 2024

1 Introduction

Given an undirected 3-colorable graph $G = (V, E)$, we would like to color G with the least number of colors efficiently. This problem was first proposed by Wigderson [Wig83], and we shall see how balancing combinatorial with semi-definite programming (SDP) methods leads to a natural $O(n^{0.5})$ bound, and how this was first broken [KT17]. We focus on the combinatorial side, specifically Blum's contributions [Blu94] that laid the groundwork for most of Kawarabayashi *et al.*'s later breakthroughs [KT12, KT17, KTY24].

1.1 General Strategy

One of the most important notions in this line of work originates from Blum's *progress towards k-coloring* [Blu94]. While deferring the exact definition, the general idea is that if it is always possible to make progress (towards some fixed k) for any 3-colorable graph, then we can $\widetilde{O}(k)$-color any 3-colorable graph in polynomial time. An interesting aspect of this line of work is that, under different regimes, two different approaches dominate one another. The best bound is obtained by balancing between them via choosing an appropriate Δ: specifically, for any parameter Δ, it suffices to make progress under either a minimum degree $\Delta = \Delta_{\min}$ or maximum degree $\Delta = \Delta_{\max}$ constraint [AC06, BK97, KT17].

1.2 Known Results

Assuming that for a 3-colorable graph with minimum degree $\Delta_{\min} = n^{1-\Omega(1)}$, a series of bounds from past literature for progress follows a sequence of the form:

$$\widetilde{O}\left((n/\Delta_{\min})^{i/(2i-1)}\right), \tag{1}$$

including $\widetilde{O}(n/\Delta_{\min})$ by Wigderson [Wig83] for $i = 1$, $\widetilde{O}((n/\Delta_{\min})^{2/3})$ and $\widetilde{O}((n/\Delta_{\min})^{3/5})$ for $i = 2$ and 3 by Blum [Blu94], and finally $\widetilde{O}((n/\Delta_{\min})^{4/7})$ for $i = 4$ by Kawarabayashi and Thorup [KT12].

Combinatorial Bounds for High Minimum Degree. The first series of bounds implied by Equation 1 stems from the simple observation that we may assume the minimum degree $\Delta_{\min} \geq k$ given any targeted coloring number. This is because it is trivial to color the rest of the graph after coloring vertices with a degree exceeding k. Hence, from Equation 1 for $i = [4]$, $\Delta_{\min} \geq k = (n/\Delta_{\min})^{i/(2i-1)} \Leftrightarrow \Delta_{\min} \geq n^{i/(3i-1)}$, which yields an $\widetilde{O}(n^{i/(3i-1)})$ bound for any $\Delta_{\min} \geq n^{i/(3i-1)}$. Thus, we get $\widetilde{O}(n^{1/2})$ for $i = 1$ [Wig83], $\widetilde{O}(n^{2/5})$ and $\widetilde{O}(n^{3/8})$ for $i = 2$ and 3, respectively [Blu94], and $\widetilde{O}(n^{4/11})$ for $i = 4$ [KT12].

SDP Bounds for Low Maximum Degree. For 3-colorable graphs with maximum degree $\Delta_{\max}$, Karger *et al.* [KMS98] used SDP to achieve $O(\Delta_{\max}^{1/3})$ colors. Combining with Equation 1, one can color 3-colorable graphs with $\widetilde{O}(n^{i/(5i-1)})$ colors for $i \in [4]$ by balancing $\Delta_{\max}^{1/3}$ and $(n/\Delta_{\min})^{i/(2i-1)}$ [BK97, Corollary 3]. In particular, the following general lemma for balancing is known:

> **Lemma 1** (Balancing [KT17]). Suppose for some near-polynomial d and f that for any n, we can make progress towards an $\widetilde{O}(f(n))$ coloring for any 3-colorable graph with either (1) $\Delta_{\min} \geq d(n)$; or (2) $\Delta_{\max} \leq d(n)$. Then we can make progress towards $\widetilde{O}(f(n))$-coloring on any 3-colorable graph.

This gives $\widetilde{O}(n^{1/4})$ for $i = 1$ and $\widetilde{O}(n^{3/14}) = \widetilde{O}(n^{0.2142})$ for $i = 3$. We omit $i = 4$ [KT12] as this appears much later, and further improvements on the SDP bounds have already been achieved. As Equation 1 converges to $\widetilde{O}((n/\Delta_{\min})^{1/2})$ from above, the bound $\widetilde{O}(n^{1/5})$ emerges as a natural barrier. Later improvements in SDP combined with Blum's result [Blu94] suggest a similar barrier: Arora *et al.* [AC06] achieved $O(n^{0.2111})$ colors based on the sparsest cut SDP [ARV09], while Chlamtac [Chl07] further improved it to $O(n^{0.2072})$. Both results rely on bounds of the form $O(\Delta_{\max}^{1/3-\varepsilon(n,\Delta_{\max})})$, where $\varepsilon(n, \Delta) > 0$ is a small value that decreases as a complex function of Δ. With these new SDP results, the combinatorial bound $\widetilde{O}(n^{4/11})$ for $i = 4$ yields a final bound of $\widetilde{O}(n^{0.2049})$ colors [KT12].

Balancing. A more careful treatment of balancing different regimes finally leads to a breakthrough: while Equation 1 converges to $\widetilde{O}((n/\Delta_{\min})^{1/2})$ from above with the natural condition $\Delta_{\min} \geq n^{1/3}$, if our goal is to balance it with the SDP bounds such as $\Delta_{\max}^{1/3}$, Equation 1 only needs to hold when $\Delta_{\min} \geq n^{3/5}$.

This idea is exploited in [KT17] to show that Equation 1 holds for $i = 12$ when $\Delta_{\min} \geq n^{0.61674333}$. Combining the best current SDP bound [Chl07], this yields an overall coloring bound of $\widetilde{O}(n^{0.19996})$, breaking the $\widetilde{O}(n^{1/5})$ barrier. The latest advancement [KTY24] further improves the combinatorial bound, where the limit of Equation 1, i.e., $\widetilde{O}((n/\Delta)^{1/2})$, can be approached arbitrarily when $\Delta_{\min} > \sqrt{n}$:

Theorem 1 ([KTY24])**.** For any 3-colorable graph with $\Delta_{\min} > n^{0.5}$, we can make progress towards a k-coloring for some $k = 2^{(\log\log n)^2}\sqrt{n/\Delta_{\min}}$ in polynomial time.

Combining Theorem 1 with the best SDP bound [Chl07] at $\Delta_{\min} = n^{0.605073}$, an $\widetilde{O}(n^{0.19747})$-coloring can also be found in polynomial time. The goal of this report is to sketch the proof of Theorem 1.

2 Preliminaries

We adopt all the standard notations. In addition, given a graph $G = (V, E)$ and some $X \subseteq V$, we let $d_Y(X) = \{d_Y(v) \mid v \in X\}$ to be the set of degrees to Y from X, and let $\min d_Y(X)$, $\max d_Y(X)$, and $\operatorname{avg} d_Y(X)$ denote the minimum, maximum, and average degree from X to Y. The last non-standard notation is the so-called *near-polynomial* function f, meaning that f is non-decreasing and that there are constants $c, c' > 1$ such that $cf(n) \leq f(2n) \leq c'f(n)$ for all large enough n. This includes any function of the form $f(n) = n^\alpha \log^\beta n$ for constants $\alpha > 0$ and β.

2.1 Making Progress

Blum has a general notion of *progress towards an $\widetilde{O}(k)$ coloring* [Blu94], or simply *progress* if k is understood, with the basic idea being that such progress eventually leads to a full $\widetilde{O}(k)$ coloring of a graph. There are three types of progress towards $\widetilde{O}(k)$ coloring:

Type 0: Finding vertices u and v that have the same color in every 3-coloring.

Type 1: Finding an independent or 2-colorable vertex set X of size $\widetilde{\Omega}(n/k)$.

Type 2: Finding a non-empty independent or 2-colorable vertex set X such that $|N(X)| = \widetilde{O}(k|X|)$.

As shown in Lemma 2, if we can always make progress towards an $\widetilde{O}(f(n))$ coloring in polynomial time, then we may color G with $\widetilde{O}(f(n))$ colors in polynomial time, reducing the coloring problem into one of finding progress. It turns out that one can exploit the 3-colorability property to guarantee progress that is otherwise not afforded to us.

Lemma 2 ([Blu94])**.** Let f be a near-polynomial. If we in time polynomial in n can make progress towards an $\widetilde{O}(f(n))$ 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.

Proof sketch. If there is Type 0 progress and we find that u and v have the same color in every 3-coloring, then we may shrink u, v into a new node, with arcs previously connecting to u or v connecting into this new node, and we may recurse on this new graph. Thus this is an easy case, and we shall assume that this does not happen for the sake of argument.

Firstly, if we can always find an independent or 2-colorable set of size $O(n/k)$, then we can always achieve an $O(k)$-coloring of G. We will omit the calculation, but in essence, every time we find such a set, we can color that set with 1 or 2 colors, remove that set from the graph, and recurse. Then we shall prove that if we can always make Type 1 or 2 progress, we will always be able to find an independent or 2-colorable set of size $O(n/k)$. We shall maintain sets V', U with the invariant that U has no neighbors in V'. V' starts with V, while U is empty at the start. Then while $|V'| \geq n/2$, if we make Type 1 progress then we just return; thus we will always assume that we make Type 2 progress in V'. In each iteration, if you find an independent or 2-colorable set S with r vertices, then you add S to U and remove $S \cup N(S)$ from V'. At the end, $|V'| < n/2$ and we return U.

We see that U is 2-colorable because at every step you find a set from V', which at every step has no neighbors in U. Hence you can use the same 2 colors to color every S. We also see that U is large, because at every step we remove $O(|S| + k|S|) = O(k|S|)$ vertices, with $|S|$ of that going to U. Since $|V'| < n/2$, we thus have $|U| = O(n/k)$, so we have a 2-colorable set with $O(n/k)$ vertices. ∎

2.2 Monochromatic Progress and Multichromatic Test

With Lemma 2, our goal now becomes to identify a small k such that we can guarantee the progress of some near-polynomial $f(n) \geq k$. As we will see, the key subroutine, *monochromatic subroutine*, is to find a vertex set X with $|X| > 1$ that is guaranteed to be monochromatic in every 3-coloring if no other progress is made along the way. If we get to this point, any pair of vertices in X will give us Type 0 progress, and hence we're done. We refer to this as *monochromatic progress*. A useful tool to obtain monochromatic progress is the following *multichromatic test* with a common parameter $\Psi = n/k^2$:

Lemma 3 (Multichromatic test [Blu94])**.** Given a vertex set $X \subseteq V$ of size at least Ψ, in polynomial time, we can either make progress towards an $\widetilde{O}(k)$-coloring of G, or else guarantee that under *every* 3-coloring of G, X is multichromatic.

2.3 Two Level Structure

The most complex ingredient from Blum [Blu94] is a certain regular second neighborhood structure. Specifically, unless other progress is made, for some $\Delta_1 = \widetilde{\Omega}(\Delta_{\min})$, in polynomial time [Blu94, KT17], we can identify a 2-level neighborhood structure $H_1 = (r_0, S_1, T_1)$ in G from a root $r_0 \in V$, that bound the number of edges between two neighborhoods S_1 and T_1:

- S_1: a first neighborhood $S_1 \subseteq N(r_0)$ of size at least Δ_1;
- T_1: a second neighborhood $T_1 \subseteq N(S_1)$ of size at most $\Psi k = n/k$ (S_1 and T_1 may overlap);
- Δ_1: the vertices in S_1 all have degrees at least Δ_1 into T_1, considering edges $E(S_1, T_1)$ in G;
- δ_1: there exists some δ_1 such that the degrees from T_1 to S_1 are all between δ_1 and $5\delta_1$.

3 Methodology

In this section, we present the recursive combinatorial coloring algorithm [KTY24] building upon the previous work [KT17] that recurses through a sequence of *nested cuts* until it finds progress.

3.1 Overview of Recursive Combinatorial Coloring

Given a 3-colorable graph with $\Delta_{\min} \geq \sqrt{n}$, Algorithm 1 makes progress towards a k-coloring for $k = 2^{(\log\log n)^2}\sqrt{n/\Delta_{\min}}$. Algorithm 1 utilizes the two-level neighborhood structure $H_1 = (r_0, S_1, T_1)$ in Section 2.3, and recurses on induced sub-problems $(S, T) \subseteq (S_1, T_1)$ with edges $E(S, T)$ between S and T in G. Specifically, there are two loops, the inner and the outer, both are combinatorial.

Algorithm 1: Seeking Progress Towards $\widetilde{O}(k)$ Coloring

Data: A 3-colorable graph $G = (V, E)$, coloring target k

$(S_1, T_1, \Delta_1, \delta_1) \leftarrow$ initial two-level structure // Section 2.3
for $j = 1, 2, \ldots$ **do**
  $(S, T) \leftarrow (S_j, T_j)$
  do // Monochromatic subroutine
    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"*
    if Check-Multichromatic(G, U) $=$ *False* **then** **return** *"Progress made"* // Lemma 3
    if $\exists v \in U$ *s.t.* Cut-or-Color(G, S, T, v) $=$ *"cut around (X, Y)"* **then** // Algorithm 2
      $(S, T) \leftarrow (X, Y)$
    else
      return *"S is monochromatic in every 3-coloring"* // Progress found
  while $|E(S, T)| < \delta_j |T|/2$
  $(X', Y') \leftarrow$Best-Side-Cut(G, S, T, S_j, T_j) // Algorithm 4
  if $|Y'| < |T|$ **then** $(S, T) \leftarrow (X', Y')$
  $(S_{j+1}, T_{j+1}, \Delta_{j+1}, \delta_{j+1}) \leftarrow$Regularize($G$, S, T) // Algorithm 3

Outer Loop. When we start an outer loop at iteration j, it is with a quadruple $(S_j, T_j, \Delta_j, \delta_j)$ where $(S_j, T_j) \subseteq (S_1, T_1)$ such that degrees from S_j to T_j are at least Δ_j, and degrees from T_j to S_j are between δ_j and $5\delta_j$, i.e., the invariants described in Section 2.3 are maintained with the fixed vertex root r_0.

Inner Loop. On the other hand, the inner loop, known as the *monochromatic subroutine*, tries to find a monochromatic progress while maintaining the following invariant for (S, T):

(i) At least Ψ vertices of high S-degree ($\geq \delta_j/4$) in T.

We define the set U as the set of high-degree nodes in T to S. Because of (i), U is large enough, and we may always either make progress or guarantee that it is multichromatic with Lemma 3.

3.2 Inner Loop: Monochromatic Subroutine

Cut-or-Color. Given sets $S, T \subseteq V$ and a an arbitrary seed vertex $t \in T$ with high S-degree, Algorithm 2 either (1) makes some progress; (2) reports that S is monochromatic in every 3-coloring such that r_0 and t have different colors; (3) finds a "cut around a sub-problem $(X, Y) \subseteq (S, T)$" satisfying:

(ii) The original high S-degree vertex t has all its neighbors to S in X, i.e., $N_S(t) \subseteq X$.

(iii) All edges from X to T go to Y, so there are no edges between X and $T \setminus Y$.

(iv) Each vertex $s' \in S \setminus X$ has $|N_Y(s')| < \Psi$.

(v) Each vertex $t' \in T \setminus Y$ has $|N_Y(N_{N(r_0)}(t'))| < \Psi$.

Algorithm 2: Cut-or-Color

Data: A 3-colorable graph $G = (V, E)$, $S \subseteq S_j$, $T \subseteq T_j$, high S-degree vertex $t \in T$

1 $X \leftarrow N_S(t)$, $Y \leftarrow N_T(X)$
2 **while** *True* **do**
3   **if** $X = S$ **then**
4   **return** *"S is monochromatic in every 3-coloring where t and r_0 have different colors"*
5   **else if** $\exists s \in S \setminus X$ *s.t.* $|N_Y(s)| \geq \Psi$ **then** // X-extension
6   **if** Check-Multichromatic(G, $N_Y(s)$) = *False* **then** // Lemma 3
7   **return** *"Progress made"*
8   **else**
9       $X \leftarrow X \cup \{s\}$, $Y \leftarrow Y \cup N_T(s)$
10   **else if** $\exists t' \in T \setminus Y$ *s.t.* $|N_Y(N_{N(r_0)}(t'))| \geq \Psi$ **then** // Y-extension
11   **if** Check-Multichromatic(G, $N_Y(N_{N(r_0)}(t'))$) = *False* **then** // Lemma 3
12   **return** *"Progress made"*
13   **else**
14       $Y \leftarrow Y \cup \{t'\}$
15   **else** // $X \neq S$ and neither an X nor a Y-extension is possible
16     $(X(t), Y(t)) \leftarrow (X, Y)$
17   **return** *"cut around $(X(t), Y(t))$"*

From the algorithm, (ii) to (v) are easy to see. The non-trivial invariant is the following:

(vi) If r_0 was red and t was green in a 3-coloring C_3, then X is all blue and Y has no blue in C_3.

To see (vi), initially, if r_0 is red and t is green, then $X = N_S(t)$ is incident to both r_0 and t, so it must be blue; $Y = N_T(X)$ is incident to X, so it cannot have blue. The high-level idea of Algorithm 2 is to extend X and Y while maintaining (vi). Specifically, the X-*extension* first makes sure that $N_Y(s)$ is multichromatic in G for some $s \in S \setminus X$ such that $d_Y(s) \geq \Psi$ (hence Lemma 3 applies). From (vi), $Y \supseteq N_Y(s)$ has no blue, so $N_Y(s)$ is red and green (must use both as it's multichromatic), so s is blue. Clearly, line 9 preserves (vi). Y-extension follows a similar idea, and hence (vi) is an invariant.

The important point is that if we end up with $X = S$, (vi) implies line 4; otherwise, if neither extension can be made further, then a "cut around (X, Y)" can be used for recursion in the inner loop, or monochromatic subroutine, of Algorithm 1. We refer to these cuts as *inner cuts* from now on.

Recursion Towards a Monochromatic Set The monochromatic subroutine seeks a monochromatic set, starting from a sub-problem $(S, T) = (S_j, T_j)$ with $|S| > 1$. Assuming (i), Lemma 3 can be applied to either make progress or certify that U is multichromatic in every 3-coloring. Suppose no progress is made, then we apply Algorithm 2 to each $v \in U$, leading to three potential outcomes: (a) some "inner cut around $(X(v), Y(v))$" is found; (b) some progress is made; (c) otherwise. Note that if some progress is made, then we're done; if an "inner cut around $(X(v), Y(v))$" is found, we can then recurse. The interesting case is the last: when all $v \in U$, S is monochromatic in every 3-coloring when v and r_0 have different colors. It turns out that this implies that S is monochromatic in every 3-coloring (line 12) [KT17, Lemma 3.1]. The key idea is that we've already established U is multichromatic in every 3-coloring (Algorithm 1, line 8), and so for any 3-coloring, there must exist some $x \in X$ with a different color to r_0 in that coloring. Thus in every case S must be monochromatic.

3.3 Outer Loop: Regularization and Side Cut

Regularization. Put the side cuts aside, the iteration j of the outer loop is finished by *regularizing* the degrees of vertices, as described in Algorithm 3. One can show that Algorithm 3 maintains the invariants described in Section 2.3; specifically, given input $(S, T) \subseteq (S_j, T_j)$, Algorithm 3 outputs $(S^r, T^r, \Delta^r, \delta^r)$ such that degrees from S^r to T^r are at least $\Delta^r \geq \operatorname{avg} d_T(S)/(30 \log n)$, and degrees from T^r to S^r are between δ^r and $5\delta^r$ with $\delta_j \geq \operatorname{avg} d_S(T)/8$ [KT17, Lemma 9.1].

Algorithm 3: Regularize

Data: A 3-colorable graph $G = (V, E)$, $S \subseteq S_j$, $T \subseteq T_j$

Partition T into sets $U_\ell = \{v \in T \mid d_S(v) \in [d_\ell, d_{\ell+1})\}$ where $d_\ell = (4/3)^\ell$
$\ell^* \leftarrow \arg\max_{\ell:\, d_\ell \geq \operatorname{avg} d_S(T)} |E(U_\ell, S)|$
$\delta^r \leftarrow d_{\ell^*}/4$, $\Delta^r \leftarrow \operatorname{avg} d_{U_{\ell^*}}(S)/4$
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 $(S^r, T^r, \Delta^r, \delta^r)$

Side Cut. The final ingredient is the concept of *side cuts* that can be used as an alternative to the inner cuts identified by the monochromatic subroutine [KTY24]. In outer round j, at the end of the inner loop, we have got to the **last** inner cut $(X, Y) \subseteq (S_j, T_j)$ such that $|E(X, Y)| < \delta_j |Y|/2$. Then, consider a family of *side cuts* $(X'(u), Y'(u))$, one for each $u \in Y$ with $d_{S_j \setminus X}(u) \geq \delta_j/3$ such that $X'(u) = N_{S_j}(u) \setminus X$ and $Y'(u) = N_{T_j}(X'(u)) \setminus Y$. Note that $(X'(u), Y'(u))$ is disjoint from (X, Y), as illustrated in Figure 1. Among these side cuts, the *best* side cut is the one with the smallest $Y'(u)$.

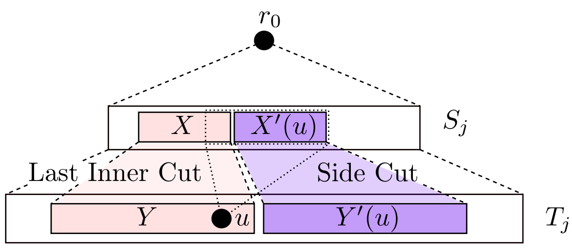

Figure 1: Side cuts and the last inner cuts.

Algorithm 4 implements the above construction. Given the best side cut $(X'(u), Y'(u))$, it then replaces the final inner cut (X, Y) in Algorithm 1 if Y' is indeed smaller than Y. Incidentally, the final inner cut (X, Y) found by the monochromatic subroutine is also the one with the smallest Y, so (X, Y) we end up using is the one minimizing Y among all inner cuts and side cuts considered.

Algorithm 4: Best Side Cut

Data: A 3-colorable graph $G = (V, E)$, $X \subseteq S_j$, $Y \subseteq T_j$, $S_j \subseteq V$, $T_j \subseteq V$

$(X', Y') \leftarrow (S_j, T_j)$
for $u \in Y$ **do**
  if $d_{S_j \setminus X}(u) \geq \delta_j/3$ **then**
    $X'(u) \leftarrow N_{S_j}(u) \setminus X$, $Y'(u) \leftarrow N_{T_j}(X'(u)) \setminus Y$
    if $|Y'(u)| < |Y'|$ **then** $(X', Y') \leftarrow (X'(u), Y'(u))$
return (X', Y')

Finally, in Algorithm 1, the best inner cut or side cut, denoted as (X_j, Y_j) at iteration j, gets assigned to (S, T) before we regularize it, obtaining the new quadruple $(S_{j+1}, T_{j+1}, \Delta_{j+1}, \delta_{j+1})$.

3.4 Proof of Theorem 1

It suffices to show that Algorithm 1 is correct and efficient. From the discussion, what is left to prove is that the invariant (i) holds (implying "Error B" will not occur), "Error A" won't happen (i.e., S is non-trivial such that $|S| > 1$), and Algorithm 1 terminates in $\mathsf{poly}(n)$ rounds. It boils down to showing that the following two preconditions hold for all j before Algorithm 1 terminates [KTY24, Lemma 15]:

(a) $\Delta_j = \Delta_{\min}/(\log n)^{O(j)} = \widetilde{\omega}(\Psi)$, and (b) $\delta_j \geq 4\Delta_j/\Psi$.

For a particular iteration j, one can show that $|Y_j| \leq O(\sqrt{\Psi|T_j|})$ [KTY24, Lemma 14], and since $|T_{j+1}| \leq |Y_j|$ from regularization (Algorithm 3), this can be used recursively. With this bound, both (a) and (b) can be proved for all $j \leq \lfloor \log\log n \rfloor$. Let's first see how these imply the correctness of Algorithm 1, where we implicitly assume that Algorithm 1 terminates in $\lfloor \log\log n \rfloor$ rounds for now.

Correctness: Condition (i), "Error A", and "Error B". By exploiting (a) and the fact that $\max d_S(T) \leq 5\delta_j$ (from Algorithm 3), one can show that (i) will be satisfied in the monochromatic subroutine if $\operatorname{avg} d_X(Y) \geq \delta_j/2$ [KT17, KTY24]. Note that this holds during the entire monochromatic subroutine since line 13, hence, "Error B" will not occur. This further implies that there is at least one (in fact, $\geq \Psi$) high S-degree vertex in T that has at least $\delta_j/4$ neighbors in S, i.e., $|S| \geq \delta_j/4 \geq \Delta_j/\Psi = \widetilde{\omega}(1)$ from (a) and (b). This means that $|S| > 1$, and hence "Error A" will also not occur.

This proves the correctness of Algorithm 1, if we can show that Algorithm 1 indeed terminates in $\lfloor \log\log n \rfloor$ rounds, which will also imply efficiency of Algorithm 1, hence proving Theorem 1.

Efficiency: Termination after $\lfloor \log\log n \rfloor$ Rounds. Assume no progress has been made in the first j rounds. Applying $|Y_j| \leq O(\sqrt{\Psi|T_j|})$ recursively, with the initial condition $|T_1| \leq n/k$, we conclude that $|Y_j| = O(\Psi k^{1/2^j})$. On the other hand, a simple lower bound for $|Y_j|$ is given by $|Y_j| = \Omega(\Delta_j^2/\Psi)$ [KTY24, Lemma 11]. We hence see that there exist constants A, B such that $A\Delta_j^2/\Psi < B\Psi k^{1/2^j} \Leftrightarrow (\Delta_j/\Psi)^2 < Bk^{1/2^j}/A$. As $\Delta_j/\Psi = \widetilde{\omega}(1)$ for all $j < \log\log n$ from (a) and $k^{1/2^j} < 2$ for $j = \lceil \log\log k \rceil < \log\log n$, forcing $j < \log\log n$. Hence, Algorithm 1 will always make progress in $\lfloor \log\log n \rfloor$ rounds.

Balancing. With Theorem 1, we can show the following:

Theorem 2. In polynomial time, we can color any 3-colorable graph using $\widetilde{O}(n^{0.19747})$ colors.

Proof. Using Chlamtac's SDP result [Chl07, Theorem 15] for low degrees, for $\Delta_{\max} \leq \Delta = n^\tau$ with $\tau = 0.605073$ and $c = 0.0213754$, we can make progress towards $k = \widetilde{O}(n^{0.19747})$ coloring. By Lemma 1, we may therefore assume that Δ is the minimum degree bound, i.e., $\Delta_{\min} \geq \Delta$. This is easily above $\sqrt{n}$, hence by Theorem 1, we get progress towards $\sqrt{n/\Delta} \cdot 2^{(\log\log n)^2} = n^{(1-0.605073)/2} \cdot 2^{(\log\log n)^2} < n^{0.19747}$. With Lemma 2, in polynomial time, we can color any 3-colorable graph with $\widetilde{O}(n^{0.19747})$ colors. ■

4 Conclusion

We have followed a line of inquiry that was first posed by Wigderson [Wig83], and have focused on how combinatorial methods have led to a fruitful line of study [Blu94, KT12, KT17, KTY24]. Combined with SDP methods [KMS98, Chl07], we can make progress in both cases when the minimum degree is large and when the maximum degree is small. Defining progress as either finding two vertices such that in every 3-coloring they are the same color, finding a large 2-colorable set, or finding a 2-colorable set with a small neighborhood, we show how finding a 2-level structure [Blu94] guarantees that we can always find progress, invoking a key lemma (Lemma 3) that guarantees either progress or multichromaticity of a set S provided it is large enough. Next, a novel recursion and the introduction of "side cuts" provide the tiny but significant boost to break the $O(n^{1/5})$ barrier and push the boundary further.

If Kawarabayashi *et al.*'s observation holds that combinatorial improvements always follow the sequence of $O(n^{i/(2i-1)})$ colors, then it is bounded below by $O(n^{0.5})$, and [KTY24] seems to hit that bound—or at least arbitrarily approach it. Thus it may be reasoned that the next improvement may most likely result from improved SDP bounds, relaxation of guarantees (for example, allowing for approximation or probabilistic algorithms), or even a radical approach that abandons the current balancing paradigm.

References

[AC06] Sanjeev Arora and Eden Chlamtac. New approximation guarantee for chromatic number. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 215–224, 2006.

[ARV09] Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM (JACM)*, 56(2):1–37, 2009.

[BK97] Avrim Blum and David Karger. An $\widetilde{O}(n^{3/14})$-coloring algorithm for 3-colorable graphs. *Information processing letters*, 61(1):49–53, 1997.

[Blu94] Avrim Blum. New approximation algorithms for graph coloring. *Journal of the ACM (JACM)*, 41(3):470–516, 1994.

[Chl07] Eden Chlamtac. Approximation algorithms using hierarchies of semidefinite programming relaxations. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 691–701. IEEE, 2007.

[KMS98] David Karger, Rajeev Motwani, and Madhu Sudan. Approximate graph coloring by semidefinite programming. *Journal of the ACM (JACM)*, 45(2):246–265, 1998.

[KT12] Ken-ichi Kawarabayashi and Mikkel Thorup. Combinatorial coloring of 3-colorable graphs. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 68–75. IEEE, 2012.

[KT17] Ken-ichi Kawarabayashi and Mikkel Thorup. Coloring 3-colorable graphs with less than $n^{1/5}$ colors. *Journal of the ACM (JACM)*, 64(1):1–23, 2017.

[KTY24] Ken-ichi Kawarabayashi, Mikkel Thorup, and Hirotaka Yoneda. Better coloring of 3-colorable graphs. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 331–339, 2024.

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