2012 IEEE 53rd Annual Symposium on Foundations of Computer Science

Combinatorial coloring of 3-colorable graphs

Ken-ichi Kawarabayashi
National Institute of Informatics
Tokyo, Japan
Email: k_keniti@nii.ac.jp

Mikkel Thorup
University of Copenhagen and AT&T Labs—Research
Copenhagen, Denmark Florham Park, USA
Email: mikkel2thorup@gmail.com

***Abstract*— We consider the problem of coloring a 3-colorable graph in polynomial time using as few colors as possible. We present a combinatorial algorithm getting down to $\widetilde{O}(n^{4/11})$ colors. This is the first combinatorial improvement of Blum's $\widetilde{O}(n^{3/8})$ bound from FOCS'90. Like Blum's algorithm, our new algorithm composes nicely with recent semi-definite programming approaches. The current best bound is $O(n^{0.2072})$ colors by Chlamtac from FOCS'07. We now bring it down to $O(n^{0.2049})$ colors.**

***Keywords*-Graph Coloring; Approximation Algorithms;**

I. 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. Knowing that red and blue do not suffice, we might try introducing green, but this is already beyond what we believe computers can do.

Formally a k-coloring of an undirected graph assigns k colors to the vertices. The coloring is only valid if no two neighbors get the same color. The validity of coloring is trivially checked in linear time so the deciding if a graph is k-colorable is in NP.

Three-coloring is a classic NP-hard problem. It was proved hard by Garey, Johnson, and Stockmeyer at STOC'74 [1], and was the prime example of NP-hardness mentioned by Karp in 1975 [2]. It is an obvious target for any approach to NP-hard problems. With the approximation approach, given a 3-colorable graph, that 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 [3] got down to $O(n^{1/2})$ colors for a graph with n vertices. Berger and Rompel [4] improved this to $O((n/(\log n))^{1/2})$. Blum [5] 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 at FOCS'94 was by Karger, Motwani, Sudan [6] using semi-definite programming (SDP). This came in the wake of Goemans and Williamson's seminal use of SDP for max-cut at STOC'94 [7]. 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 Blum and Karger [8] combined the SDP from [6] with Blum's [5] algorithm, yielding an improved bound of $\widetilde{O}(n^{3/14}) = \widetilde{O}(n^{0.2142})$. Later improvements in semi-definite programming have also been combined with Blum's algorithm. At STOC'06, Arora, Chlamtac, and Charikar [9] got down to $O(n^{0.2111})$ colors. The proof in [9] is based on the seminal result of Arora, Rao and Vazirani [10] which gives an $O(\sqrt{\log n})$ algorithm for the sparsest cut problem. The last improvement was at FOCS'07 by Chlamtac [11] who got down to $O(n^{0.2072})$ colors.

Only a few lower bounds are known for the coloring of 3-colorable graphs. We know that it is NP-hard to get down to 5 colors [12], [13]. Recently, Dinur, Mossel and Regev [14] showed that it's hard to color with any constant number of colors (i.e., $O(1)$ colors) based on a variant of the Unique Games Conjecture.

Some integrality gap results [6], [15], [16] show that the simple SDP relaxation has integrality gap at least $n^{0.157}$. It is therefore natural to go back and see if we can improve things combinatorially.

In this paper, we present the first improvement on the combinatorial side since Blum at FOCS'90 [5]. With a purely combinatorial approach, we get down to $\widetilde{O}(n^{4/11})$ colors. Combining it with Chlamtac's SDP [11], we get down to $O(n^{0.2049})$ colors..

Technique: In the details we reuse a lot of the techniques pioneered by Blum [5], 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

0272-5428/12 $26.00 © 2012 IEEE
DOI 10.1109/FOCS.2012.16

non-trivial vertex set X that we guarantee is monochromatic in every 3-coloring of G.

Below we focus on our combinatorial algorithm looking for a monochromatic set. The integration with SDP is essentially explained in [8] and is sketched in Section VIII.

II. Preliminaries including ingredients from Blum

We are given a 3-colorable graph $G = (V, E)$ with $n = |V|$ vertices. The (unknown) 3-colorings are with 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 $G|X$ be the subgraph induced by X.

For some color target k depending on n, we wish to find a $\widetilde{O}(k)$ coloring of G in polynomial time. We are going to reuse several ideas and techniques from Blum's approach [5].

Progress: Blum has a general notion of *progress towards $\widetilde{O}(k)$ 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 towards $\widetilde{O}(k)$ 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 $cf(n) \leq f(2n) \leq c'f(n)$ for all n. As described in [5], this includes any function of the form $f(n) = n^\alpha \log^\beta n$ for constants $\alpha > 0$ and β.

Lemma 1 ([5, Lemma 1]): Let f be near-polynomial. If we in time polynomial in n can make progress towards $\widetilde{O}(f(n))$ 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 1 in Section VIII 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 1 and get a coloring, we also need to bound k with 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 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 [5]. We include the simple proofs for completeness.

Observation 2: Both for the large independent set progress and for the small neighborhood progress, it suffices with 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 $|N(X_1)|/|X_1| \leq |N(X)|/|X_1| \leq 2|N(X)|/|X| = \widetilde{O}(k)$. ■

Observation 3: If the minimum degree in the graph is $\Delta_{\min}$, we can make progress towards $\widetilde{O}(\Delta_{\min})$ coloring.

Proof: Consider a vertex v of degree $\Delta_{\min}$. The graph is 3-colorable so $N(v)$ is 2-colorable. By Observation 2, we can use $N(v)$ for large independent set progress. ■

Some of our progress will be made via results of Blum presented below using a common parameter

$$\Psi = n/k^2. \tag{1}$$

A very useful tool we get from Blum is the following multichromatic test:

Lemma 4 ([5, Corollary 4]): Given a vertex set $X \subseteq V$ of size at least $\Psi = n/k^2$, in polynomial time, we can either make progress towards $\widetilde{O}(k)$-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. Otherwise, if $|N(X)| < n/k = k\,\Psi$, we have a small neighborhood for X, and if $|N(X)| \geq n/k$, by Observation 2, we have large independent set from $N(X)$. ■

In fact Blum has a stronger lemma [5, 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 4 he proves:

Lemma 5 ([5, Theorem 3]): If two vertices have more then Ψ common neighbors, we can make progress towards $\widetilde{O}(k)$ 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 4 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 6: 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 5 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 [5] is a certain regular second neighborhood structure. Let $\Delta_{\min}$ be the smallest degree in the graph G. By Observation 3, we can make progress towards $\widetilde{O}(\Delta_{\min})$ coloring, so we can assume $k \leq \Delta_{\min}$.

For some $\Delta_0 = \widetilde{\Omega}(\Delta_{\min})$, Blum [5, 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 may 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 within a factor $(1 \pm o(1))$ around an average $\delta_0 \geq \Delta_0^2\, k/n$.

Note that [5, 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 towards a $\widetilde{O}(k)$ coloring as described in [5, 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 reductions which either make progress, find a good cut recursing on one side, or identify a monochromatic set. This is why we already now used the subscript $_0$ to indicate the original structure provided by Blum [5].

III. Our coloring algorithm

We will use Blum's 2-level neighborhood structure $H_0 = (r_0, S_0, T_0)$ with a color target

$$k = \widetilde{\Theta}((n/\Delta_{\min})^{4/7}). \tag{2}$$

We are going to work on induced subproblems $(S, T) \subseteq (S_0, T_0)$ defined in terms of a 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 average degree δ_0 from T_0 to S_0), and we will make sure that any subproblem (S, T) considered satisfies:

(i) We have more than Ψ vertices of high S-degree in T.

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

- Reporting a "sparse cut around a subproblem $(S', T') \subseteq (S, T)$" with no cut edges between S' and $T \setminus T'$ and only few cut edges between T' and $S \setminus S'$. 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 the different colors in a 3-coloring C_3 of G, then S is monochromatic in C_3.

Recursing towards 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 4;
if there is a t ∈ U such that cut-or-color(S, T, t)
returns "sparse cut around (X, Y)" then
    recursively call monochromatic(X, Y)
else
    return "S is monochromatic in every 3-coloring"
```

Let U be the set of high S-degree vertices in T. By (i) we have $|U| \geq \Psi$, so we can apply Blum's multichromatic test from Lemma 4 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, so assume that this does not happen. 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 7: 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 4 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, and then monochromatic progress can be made. However, the correctness demands that we respect (i) and never apply `monochromatic` to a subproblem (S, T) where T has less than Ψ high S-degree vertices (otherwise Lemma 4 cannot be applied to U). The proof that (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.

IV. 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

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"
    else if there is s ∈ S \ X with |N_Y(s)| ≥ Ψ
    then // X-extension
        check that N_Y(s) is multichromatic in G with
        Lemma 4;
        add s to X and N_T(s) to Y
    else if there is t' ∈ T \ Y with |N_Y(N_S(t'))| ≥ Ψ
    then // Y-extension
        check that N_Y(N_S(t')) is multichromatic in G
        with Lemma 4;
        add t' to Y
    else // X ≠ S and no X- or Y-extension possible
        return "sparse cut around (X, Y)"
```

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, if based on the assumption, 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.

The first part of `cut-or-color` is essentially the coloring that Blum [5, §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 [5] 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 is all blue and Y has no blue.

If we end up with $X = S$, then (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 4 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, 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 4 would have made progress. Preserving (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 Blum's [5] algorithm. The above X-extension is essentially the coloring Blum [5, §5.2] uses for dense regions. He applies it directly to his structure $H_0 = (r_0, S_0, T_0)$ from Section II. He needs a larger degree $\delta_0 \geq \Delta_0^2 k/n$ than we do, but then he proves that the set of vertices s with degree at least Ψ into Y is more than Ψ. This means that either he finds progress with a green vertex s, or he ends up with a blue set X of size Ψ, and gets progress applying Lemma 4 to X.

Our algorithm works for a smaller δ_0 and thereby for a smaller color target k. Our extended X is typically too small for Lemma 4. In fact, as we recurse, we will get sets S that themselves are much smaller than Ψ. Otherwise we would be done with Lemma 4 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 [5] in the technical details, but the overall strategy, seeking sparse cuts for recursion to crystallize a small monochromatic set S, is entirely different (and new).

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 4 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 (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 (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 $|N_Y(N_S(t'))| < \Psi$.

A 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.

V. Correctness

It should be noted that the correctness of `cut-or-color` follows from (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 (i), hence where we cannot apply Lemma 4 to ensure that the high S-degree vertices do not all have the same color as r_0.

VI. Degree constraints

Before we can start our recursive algorithm, we need some slightly different degree constraints from those provided by Blum [5] described in Section II:

- The vertices in S_0 have average degree Δ_0 into T_0.
- The degrees from T_0 to S_0 are all within a factor $(1 \pm o(1))$ around the average $\delta_0 \geq \Delta_0^2\, k/n$.

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 $E(S_1, T_1)$. We also note that the average degree from T to S remains above $\delta_0/2 = 2\delta_1$. 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 $(1 + o(1))\delta_0 < 5\delta_1$, with an average above $2\delta_1$. Note that $\delta_1 = \delta_0/4 \geq \Delta_0^2\, k/(4n) = 4\Delta_1^2 k/n$.

Lemma 8: We may assume that $k < \Delta_1$.

Proof: As noted earlier, by Observation 3, we can make progress towards $\widetilde{O}(\Delta_{\min})$ coloring. However, $\Delta_0 = \widetilde{\Omega}(\Delta_{\min})$ and $\Delta_1 = \Delta_0/4$, so $\widetilde{O}(\Delta_{\min}) = \widetilde{O}(\Delta_1)$. We are therefore done if $k \geq \Delta_1$. ∎

A *high S-degree* in T can now be restated as a degree to S above $\delta_1/4 = \delta_0/16 = \Delta_1^2 k/n$.

Lemma 9: $\Delta_1 = \Psi n^{\Omega(1)}$.

Proof: Recall that $\Delta_1 = \Delta_0/4 = \widetilde{\Omega}(\Delta_{\min})$, so from (2) we get $k = \widetilde{O}((n/\Delta_1)^{4/7})$, or equivalently, $\Delta_1 = \widetilde{\Omega}(n/k^{7/4})$. Since $\Psi = n/k^2$, we conclude that $\Delta_1 = \widetilde{\Omega}(\Psi k^{1/4}) = \Psi n^{\Omega(1)}$. ∎

Recursion: 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). We will prove that every subproblems (S, T) considered recursively satisfies the following degree invariants:

(vii) Each vertex $v \in S$ has all its neighbors from T_1 in T, so the degrees from S to T remain at least Δ_1. Since the next subproblem is $(S', T') = (X, Y)$, this invariant follows inductively from the sparse cut condition (iv).

(viii) The average degree in T to S is at least $\delta_1/2$.

The following key lemma shows that the above degree constraints imply invariant (i) which states that we Ψ high S-degree vertices in T, as needed for the correctness of the recursive call.

Lemma 10: (vii) and (viii) imply (i).

Proof: If h is the fraction of high S-degree vertices in T, the average degree in T is at most $h\, 5\delta_1 + (1 - h)\delta_1/4$ which by (viii) is at least $\delta_1/2$. Hence $h = \Omega(1)$. By (vii) we have $|T| \geq \Delta_1$, so we have $\Omega(\Delta_1)$ high S-degree vertices in T. By Lemma 9 this is much more than Ψ high S-degree vertices. ∎

VII. Maintaining degrees recursively

All that remains is to prove that invariant (viii) is preserved, i.e., that the average degree from T to S does not drop below half the original minimum degree δ_1 from T_1 to S_1. Inductively, when a sparse cut is declared around a new subproblem $(S', T') = (X, Y) \subseteq (S, T)$, we can assume that (vii) and (viii) are satisfied for (S, T) and that (vii) is satisfied for (X, Y). It remains to prove (viii) for (X, Y).

Below we first show that when a sparse cut is declared around $(X, Y) \subseteq (S, T)$, then X cannot be too small. We later complement this by showing that the total number of edges cut in the recursion cannot be too large.

Lemma 11: $|Y| \geq \Delta_1^2 k^2/(2n)$.

Proof: If t is the high S-degree vertex we started with in T, then by (iii), we have the whole neighborhood $Z = N_S(t)$ of t in S preserved in X. By definition of high S-degree, $|Z| \geq \delta_1/4 = \Delta_1^2 k/n$. By (iv) and (vii) the degrees from Z to Y are at least Δ_1 and by Lemma 8, $k < \Delta_1$. It follows that $\Delta_1/\Psi = \Delta_1 k^2/n < |Z|$, so by Lemma 6, we have $|N_Y(Z)| \geq \Delta_1^2/(2\Psi) = \Delta_1^2 k^2/(2n)$. ∎

In our original problem (S_1, T_1), each vertex $v \in Y$ we had δ_1 edges to S_1, so by Lemma 11, the original number of edges from Y to S_1 was at least

$$\Delta_1 |Y| \geq \delta_1 \Delta_1^2 k^2/(2n). \tag{3}$$

To prove (viii), we argue that at least half of these edges are between Y and X. This follows if we can prove that the total number of edges cut is only half the number in (3).

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 12: The number of cut edges from Y to $S \setminus X$ is bounded by

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

Proof: First we note that from (v) we get a trivial bound of $\Psi|S \setminus X|$ on the number of new cut edges, but is not strong enough for (4). Here we use (vi) we get

$$\sum_{y \in Y} |N_T(N_S(y)) \setminus Y| = \sum_{t' \in T \setminus Y} |N_Y(N_S(t'))| \leq |T \setminus Y| \Psi = |T \setminus Y| n/k^2. \tag{5}$$

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 (iv) we have that $N_T(N_S(y)) \setminus Y = N_T(Z) \setminus Y$. Consider any vertex $v \in Z$. By (vii) the degree from v to T is at least Δ_1. Since $v \notin X$, by (v), the degree from v to Y is at most Ψ, and by Lemma 9, $\Psi = o(\Delta_1)$. The degree from v to $T \setminus Y$ is therefore at least $(1 - o(1))\Delta_1 \geq \Delta_1/2$. This holds for every $v \in Z$. It follows by Lemma 6 that $|N_T(Z) \setminus Y| \geq \min\{(\Delta_1/2)/\Psi, |Z|\} \, \Delta_1/4$. Relative to $|Z|$, this is

$$\frac{|N_T(Z) \setminus Y|}{|Z|} \geq \min\left\{\frac{\Delta_1/(2\Psi)}{|Z|}, 1\right\} \Delta_1/4$$

From our original configuration (S_1, T_1), we know that all degrees from T to S are bounded by $5\delta_1$ and this bounds the size of $Z \subseteq N_S(y)$. Therefore

$$\frac{\Delta_1/(2\Psi)}{|Z|} \geq \frac{\Delta_1 k^2/(2n)}{5\delta_1} = \frac{\Delta_1 k^2}{10\delta_1 n}.$$

Since $\delta_1 \geq 4\Delta_1^2 k/n$, we have

$$\frac{\Delta_1 k^2}{10\delta_1 n} \leq \frac{\Delta_1 k^2}{10(4\Delta_1^2 k/n)n} = \frac{k}{40\Delta_1} < 1.$$

Therefore

$$\frac{|N_T(Z) \setminus Y|}{|Z|} \geq \min\left\{\frac{\Delta_1/(2\Psi)}{|Z|}, 1\right\} \Delta_1/4 \geq \frac{\Delta_1 k^2}{10\delta_1 n} \Delta_1/4 = \frac{\Delta_1^2 k^2}{40\delta_1 n}.$$

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

$$|N_S(y) \setminus X| \leq \frac{40\delta_1 n}{\Delta_1^2 k^2} |N_T(N_S(y)) \setminus Y|.$$

Using (5) we now get the desired bound on the number of cut edges from Y to $S \setminus X$:

$$\sum_{y \in Y} |N_S(y) \setminus X| \leq \frac{40\delta_1 n}{\Delta_1^2 k^2} \sum_{y \in Y} |N_T(N_S(y)) \setminus Y| = \frac{40\delta_1 n}{\Delta_1^2 k^2} |T \setminus Y| \, n/k^2 = |T \setminus Y| \, \frac{40\delta_1 n^2}{\Delta_1^2 k^4}.$$

∎

From Lemma 12 it immediately follows that the total number of edges cut in the whole recursion is at most

$$|T_1| \, \frac{40\delta_1 n^2}{\Delta_1^2 k^4} \leq \frac{40\delta_1 n^3}{\Delta_1^2 k^5}. \tag{6}$$

This should be at most half the original number of edges from (3). Thus we maintain (vii) with an average degree of $\delta_1/2$ from Y as long as

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

or equivalently

$$k^7 \geq 160 \, (n/\Delta_1)^4. \tag{7}$$

Thus, if k satisfies (7), then all our degree constraints are maintained, which means that we will keep recursing over sparse cuts until we either make progress towards a $\widetilde{O}(k)$ coloring, or end up with a provably monochromatic set S on which we can make monochromatic progress towards $\widetilde{O}(k)$ coloring. Since $\Delta_1 = \Omega(\Delta_0) = \widetilde{\Omega}(\Delta_{\min})$, we can pick k as a function of $\Delta_{\min}$ and n such that $k = \widetilde{\Theta}((n/\Delta_{\min})^{4/7})$ and such that (7) will be satisfied. Thus we conclude

Theorem 13: If a 3-colorable graph has minimum degree Δ_{min}, then we can make progress of Type 0, 1, or 2, towards $\widetilde{O}((n/\Delta_{min})^{4/7})$ coloring in polynomial time.
The corresponding result of Blum [5] was that we could make progress towards $\widetilde{O}((n/\Delta_{\min})^{3/5})$ coloring.

Corollary 14: 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 1, it suffices to prove progress towards $\widetilde{O}(n^{4/11})$ coloring. By Observation 3, we get this progress (Type 1) if $\Delta_{\min} = \widetilde{O}(n^{4/11})$. Otherwise, $\Delta_{\min} = \widetilde{\Omega}(n^{4/11})$, and then by Theorem 13, we get progress towards $\widetilde{O}((n/\Delta_{min})^{4/7}) = \widetilde{O}((n/n^{4/11})^{4/7}) = \widetilde{\Omega}(n^{4/11})$ coloring. ■

VIII. Integration with SDP

SDP coloring of 3-colorable graphs works best for graphs of low maximum degree $\Delta_{\max}$. The original result of Karger et al. [6] was that we, in polynomial time, can find an independent of size $\widetilde{\Omega}(n/\Delta_{\max}^{1/3})$, and hence make progress towards an $\widetilde{O}(\Delta_{\max}^{1/3})$ coloring. There has been substantial improvements [9], [11], but the number of colors used is a more complicated function of $\Delta_{\max}$ and n. The strongest current bounds are due to Chlamtac [11, Theorem 15]. In [11, Corollary 16] Chlamtac provided an instantiation of [11, Theorem 15] which was optimized for combination with Blum's [5] coloring. Chlamtac [personal communication] has provided us a corresponding instantiation of [11, Theorem 15] optimized for combination with our Theorem 13:

Lemma 15 (Chlamtac [Personal Communication]): For any 3-colorable graph G on n vertices with maximum degree $\Delta_{\max} = O(n^{0.6415})$, in polynomial time, we can find an independent set of size $\Omega(n^{0.7951})$, and hence make progress towards $O(n^{0.2049})$ coloring.
Note that with minimum degree $\Delta_{\min} = \Omega(n^{0.6415})$ in Theorem 13, we also make progress towards $\widetilde{O}((n/\Delta_{min})^{4/7}) = \widetilde{O}(n^{0.2049})$ coloring. Unfortunately it is only for regular graphs that $\Delta_{\min} = \Delta_{\max}$.

To handle mixed degrees, we follow the approach of Karger and Blum [8] who showed how to combine the original SDP of Karger et al. [6] with Blum's algorithm [5]. To describe the combination in general, we first need to elaborate a bit on Blum's progress from Section II, essentially reproving Lemma 1 from [5, Lemma 1].

We need to argue that if we in time polynomial in n can always make progress towards $\widetilde{O}(f(n))$ 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. Here k is a non-decreasing function that is near-polynomial which formally means that there are constants $c, c' > 1$ such that for all n, $cf(n) \leq f(2n) \leq c'f(n)$.

The simplest case is that of same color progress in Type 0 where we identify two vertices u and v with 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 w. 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 subgraphs of G. We note here 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 progress is made. When that 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 other simple type of progress is Type 1 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.

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 $i = 1, ..., \ell = O(1)$ 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 place 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, combined the players 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 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 1 which assumes that we can find progress of Type 0, 1, or 2, on every graph G'. 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 $G|V_i$.

We now introduce Player 3 that for a parameter Δ looks for the following type of progress.

Type 3: Small degree. Finding a vertex of degree at most Δ.

From Observation 3 we know that this gives Type 1 progress if $\Delta = \widetilde{\Omega}(n/f(n))$. However, using the SDP from Lemma 15, we will get progress for a much smaller Δ.

When Player 3 finds a vertex v of degree at most Δ, he removes v from G' and places it in his set V_3. Player 3 will have an induced subgraph $G_3 = G|V_3$ of G where the average degree is below 2Δ.

Suppose Player 3 ends up with $n_3 = |V_3| = \Omega(n)$. We delete from V_3 all vertices with degree at least 4Δ. The resulting vertex set V_3^- is of size $n_3^- > n_3/2 = \Omega(n)$, and now the induced graph $G_3^- = G|V_3^-$ has maximum degree $\Delta_{\max} < 4\Delta$.

In our case, we set $\Delta = n^{0.6415}$. If Player 3 ends up with $n_3 = |V_3| = \Omega(n)$, he applies Lemma 15 to G_3^- and get an independent set of size $\Omega(n^{0.7951}) = \widetilde{\Omega}(n/f(n))$ for $f(n) = n^{0.2049}$.

To claim $\widetilde{O}(f(n))$ coloring for every 3-colorable graph, we need to argue that we can always find progress of Type 0, 1, 2, or 3. We have Type 3 progress if there is any vertex of degree below $\Delta = n^{0.6415}$. Otherwise we have a 3-colorable graph with min-degree $\Delta_{\min} \geq \Delta$. By Theorem 13, we can find progress of Type 0, 1, or 2, towards $\widetilde{O}((n/\Delta_{\min})^{4/7}) = \widetilde{O}(n^{0.2049}) = \widetilde{O}(f(n))$ coloring. Thus we conclude:

Theorem 16: A 3-colorable graph on n vertices can be colored with $O(n^{0.2049})$ colors in polynomial time.

REFERENCES

[1] M. Garey, D. Johnson, and L. Stockmeyer, "Some simplified NP-complete graph problems," *Theor. Comput. Sci.*, vol. 1, no. 3, pp. 237–267, 1976, announced at STOC'74.

[2] R. M. Karp, "On the computational complexity of combinatorial problems," *Networks*, vol. 5, pp. 45–68, 1975.

[3] A. Wigderson, "Improving the performance guarantee for approximate graph coloring," *J. ACM*, vol. 30, no. 4, pp. 729–735, 1983, announced at STOC'82.

[4] B. Berger and J. Rompel, "A better performance guarantee for approximate graph coloring," *Algorithmica*, vol. 5, no. 3, pp. 459–466, 1990.

[5] A. Blum, "New approximation algorithms for graph coloring," *J. ACM*, vol. 41, no. 3, pp. 470–516, 1994, announced at STOC'89 and FOCS'90.

[6] D. Karger, R. Motwani, and M. Sudan, "Approximate graph coloring by semidefinite programming," *J. ACM*, vol. 45, no. 2, pp. 246–265, 1998, announced at FOCS'94.

[7] M. Goemans and D. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *J. ACM*, vol. 42, no. 6, pp. 1115–1145, 1995, announced at STOC'94.

[8] A. Blum and D. Karger, "An $\tilde{O}(n^{3/14})$-coloring algorithm for 3-colorable graphs," *Inf. Process. Lett.*, vol. 61, no. 1, pp. 49–53, 1997.

[9] S. Arora, E. Chlamtac, and M. Charikar, "New approximation guarantee for chromatic number," in *Proc. 38th STOC*, 2006, pp. 215–224.

[10] S. Arora, S. Rao, and U. Vazirani, "Expanders, geometric embeddings and graph partitioning," *J. ACM*, vol. 56, no. 2, pp. 1–37, 2009, announced at STOC'04.

[11] E. Chlamtac, "Approximation algorithms using hierarchies of semidefinite programming relaxations," in *Proc. 48th FOCS*, 2007, pp. 691–701.

[12] V. Guruswami and S. Khanna, "On the hardness of 4-coloring a 3-colorable graph," *SIAM Journal on Discrete Mathematics*, vol. 18, no. 1, pp. 30–40, 2004.

[13] S. Khanna, N. Linial, and S. Safra, "On the hardness of approximating the chromatic number," *Combinatorica*, vol. 20, no. 3, pp. 393–415, 2000.

[14] I. Dinur, E. Mossel, and O. Regev, "Conditional hardness for approximate coloring," *SIAM J. Comput.*, vol. 39, no. 3, pp. 843–873, 2009, announced at STOC'06.

[15] U. Feige, M. Langberg, and G. Schechtman, "Graphs with tiny vector chromatic numbers and huge chromatic numbers," in *Proc. 43rd FOCS*, 2002, pp. 283–292.

[16] M. Szegedy, "A note on the θ number of Lovász and the generalized Delsarte bound," in *Proc. 35th FOCS*, 1994, pp. 36–39.