Improved Algorithmic Versions of the Lovász Local Lemma*

Aravind Srinivasan†

Abstract

The Lovász Local Lemma is a powerful tool in combinatorics and computer science. The original version of the lemma was nonconstructive, and efficient algorithmic versions have been developed by Beck, Alon, Molloy & Reed, *et al.* In particular, the work of Molloy & Reed lets us automatically extract efficient versions of essentially any application of the symmetric version of the Local Lemma. However, with some exceptions, there is a significant gap between what one can prove using the original Lemma nonconstructively, and what is possible through these efficient versions; also, some of these algorithmic versions run in super-polynomial time. Here, we lessen this gap, and improve the running time of all these applications (which cover all applications in the Molloy & Reed framework) to polynomial. We also improve upon the parallel algorithmic version of the Local Lemma for hypergraph coloring due to Alon, by allowing noticeably more overlap among the edges.

1 Introduction

The Lovász Local Lemma ("LLL") is a powerful tool for proving the existence of discrete structures [9]; see, e.g., [3] for several such applications. Breakthroughs in the last two decades have led to efficient algorithms for *constructing* many families of such structures that are guaranteed to exist by the LLL [4, 2, 12, 13]. In this work, we present further-improved algorithmic versions of the LLL.

Notation. Let e denote the base of the natural logarithm, and $[h]$ denote the set $\{1, 2, \ldots, h\}$. Let $\lg x$ denote the logarithm to the base 2, and $\ln x$ denote the logarithm to the base e.

We start by describing the basic "symmetric" version of the LLL, which provides a sufficient condition for simultaneously avoiding a collection of "bad" events $E_1, E_2, \ldots, E_m$:

THEOREM 1.1. ***(LLL, Symmetric Version [9])*** *Let $E_1, E_2, \ldots, E_m$ be events with $\max_i \Pr[E_i] \leq p$. If there exists $D \geq 1$ such that each E_i is mutually independent of all but at most D of the other events E_j and $(1+1/D)^D \cdot p \cdot (D+1) \leq 1$, then $\Pr[\bigwedge_{i=1}^m \overline{E_i}] \geq \prod_{i=1}^m (1 - e \cdot \Pr[E_i]) > 0$. In particular, if $e \cdot p \cdot (D+1) \leq 1$, then $\Pr[\bigwedge_{i=1}^m \overline{E_i}] \geq \prod_{i=1}^m (1 - e \cdot \Pr[E_i]) > 0$.*

REMARK 1.1. *Although "$e \cdot p \cdot (D+1) \leq 1$" is the usual way in which the sufficient condition is stated, it is easy to check, using the standard proof of the LLL's symmetric version that employs the more-general asymmetric verson, that the condition $(1+1/D)^D \cdot p \cdot (D+1) \leq 1$ also suffices. We use the latter since it helps improve some of our constants. Also, the actual value of the positive probability guaranteed by the LLL is often not used; even when used, it is typically taken to be the lower-bound $(1-ep)^m$ on $\prod_{i=1}^m (1 - e \cdot \Pr[E_i])$. The fact that the LLL gives the probability $\prod_{i=1}^m (1 - e \cdot \Pr[E_i])$ will be very useful for us.*

As observed in [12, 13], most applications of the LLL can be modeled as set-systems, whose ground-set corresponds to a collection of *independent* random variables $X_1, X_2, \ldots, X_n$, and whose sets S_i dictate the subsets of the X_j on which the events E_i depend:

DEFINITION 1.1. ***(Set-system model for applications of the LLL)*** In such a model, we have: (i) *independent* random variables $X_1, X_2, \ldots, X_n$, (ii) a collection of subsets $\{S_i \subseteq [n] : i = 1, 2, \ldots, m\}$, and (iii) a collection of ("bad") events $E_1, E_2, \ldots, E_m$ such that each E_i is completely defined by the random variables $(X_j : j \in S_i)$. Let D denote the maximum number of other S_j with which any S_i intersects, and let $p = \max_i \Pr[E_i]$.

Consider a given set-system model, and suppose $D \geq 1$ (the case $D = 0$ is trivial). Since the X_j are mutually independent, the LLL implies that if $(1+1/D)^D \cdot p \cdot (D+1) \leq 1$, then $\Pr[\bigwedge_i \overline{E_i}] > 0$. As mentioned above, such set-systems conveniently model most applications of the LLL. A prototypical example is hypergraph two-coloring: given a k-uniform hypergraph (V, E) for $k \geq 2$, i.e., a hypergraph with $V = [n]$ and subsets $\{S_i \subseteq [n] : i = 1, 2, \ldots, m\}$ with $|S_i| = k$

*Supported in part by NSF ITR Award CNS-0426683 and NSF Award CNS-0626964.

†Dept. of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. Part of this work was done while on sabbatical at the Network Dynamics and Simulation Science Laboratory of the Virginia Bioinformatics Institute, Virginia Tech. srin@cs.umd.edu.

for each i, when can we guarantee that V can be two-colored so that each S_i is bichromatic? It is a simple exercise for the reader to consider randomly and independently coloring the elements of $[n]$ Red and Blue, writing down the corresponding set-system model, and concluding from the LLL that if D is the "maximum overlap" of this system (i.e., each S_i intersects at most D other S_j) with $D \leq 2^k/(2e) - 1$, then the hypergraph is indeed two-colorable.[1] This example illustrates one extreme, where the maximum support-size γ of the X_j's is "small", i.e., just 2. At the other extreme, one can consider very large γ (say, $(n+m)^{\Omega(1)}$). The set-system model can be considered an abstract "constraint satisfaction" problem where we need to set the variables X_j (whose domains may have some large cardinality γ) so that all the m constraints $\overline{E_i}$ are satisfied.

In order to motivate our results, we start by presenting the work of [12, 13].

The results of [12, 13]. Let $C_0, C_1, \ldots$ denote certain constants. Thus, "$pD \leq C_0$" is a sufficient condition for a set-system model to have some feasible solution. How to construct such a feasible solution efficiently, ideally in (randomized) time $\text{poly}(n, m)$? Molloy & Reed came up with an algorithmic approach for this problem [12, 13]. Suppose:

(a) each X_i has support-size at most γ, and each S_i has $|S_i| \leq \omega$;

(b) $pD^9 \leq C_1$;

(c) each X_j can be generated from its distribution in $\text{poly}(n+m)$ time; and

(d) given values for all the X_j, the truth value of each E_i can be determined in $\text{poly}(n+m)$ time.

Then, there is an algorithm running in time $\text{poly}(\gamma^{wD \log\log m} + n + m)$, which sets values for the X_j in a manner that avoids all the E_i [12, 13]. We make four remarks about this result:

(R1) By a simple optimization of the parameters of [12, 13], we can improve (i.e., weaken) requirement (b) to "$pD^7 \leq C_2$".

(R2) An approach of [2, 12, 13] shows that the running time of $\text{poly}(\gamma^{wD \log\log m} + n + m)$ can be reduced to $\text{poly}(n+m)$ if γ and ω are bounded by $\text{poly}(D)$.

(R3) The algorithms of [12, 13] and of the present paper appear at first to also require certain conditional probabilities to be efficiently evaluable. However, as mentioned in Remark 2.2, these values can be approximated sufficiently well (by sampling) for the algorithms to work, as long as the running-time bounds (c) and (d) above hold.

(R4) For certain problems (e.g., coloring k-uniform hypergraphs with constant k, and with a number of colors that is a function of the maximum degree of the hypergraph), a clever approach of [12, 13] shows that "$pD \leq C_3$" suffices; however, most known applications using the approach of [12, 13] indeed need "$pD^9 \leq C_1$" – more precisely, its improvement "$pD^7 \leq C_2$" pointed out in (R1) above. Also see [7, 8, 16] for algorithmic versions of certain weighted versions of the LLL.

We will pay a good deal of attention to the *efficient* 2-colorability of k-uniform hypergraphs, so let us summarize its status. It follows from (R1) above that we can efficiently 2-color such hypergraphs if $D \leq C_4 2^{k/7}$; this is the best known prior to our work. Thus, the goal here is to get close to "$D \leq O(2^k)$" algorithmically. Motivated by this problem, the works [5, 6] consider an important special class of the k-uniform hypergraphs – the *almost-disjoint* or *linear* hypergraphs in which no two distinct edges intersect at more than one vertex – and allow D to be up to certain values of the form $2^{(1-o(1))k}$. In terms of parallel algorithms, there is an NC algorithm for 2-coloring k-uniform hypergraphs when $D \leq 2^{k/500}$ [2]. We present our contributions next.

(i) Better upper-bound on D, polynomial-time for all (γ, ω, D). Let $c_0 < 1/(8e)$ be any positive constant. Given any set-system model for the LLL that satisfies $pD^4 \leq c_0$ as well as the standard run-time assumptions (c) and (d) above, we present a randomized $\text{poly}(n+m)$-time algorithm to construct $\{X_j\}$ that avoid all the E_i. Thus, we improve on both (R1) and (R2): we replace "pD^7" by "pD^4", and also make the running-time polynomial irrespective of the values of γ, ω, and D. (In particular, γ could be $\exp(n+m)$ or even unbounded as in the case of continuous-valued X_j.) For hypergraph 2-coloring, we improve the sufficient condition to $D \leq 0.38 \cdot 2^{k/4}$. Again, we find it useful that our running-time is polynomial for all (γ, ω, D).

(ii) Hypergraph coloring in parallel. Recall there is an NC algorithm for hypergraph 2-coloring, if $D \leq 2^{k/500}$ (Alon [2]). While Alon did not attempt to optimize the constant "500" and it can certainly be improved, the approach of [2] will lead to a constant that is several tens. We improve this constant to 10.34. More precisely, let $H(x) = -x \lg x - (1-x)\lg(1-x)$ be the usual binary entropy function, defined for $x \in (0,1)$; $H(0)$ and $H(1)$ are zero. Define $c^* \sim 1/10.331$ to be the unique solution of the equation $3x + H(2x) = 1$. We

[1] It is also known that $D \leq c'\sqrt{k/\ln k} \cdot 2^k$ suffices, for some constant $c' > 0$ [15].

prove that for any constant $\delta > 0$, there is a constant $c_1 = c_1(\delta) > 0$ such that $D \leq c_1 \cdot 2^{k(c^*-\delta)}$ suffices.

Three ideas underlie our results. First, we make use of the explicit lower-bound $\prod_{i=1}^{m}(1 - e \cdot \Pr[E_i])$ that the LLL places on the probability of avoiding all the events E_i. While this value is typically too low when we start, we use it fruitfully after running the first phase of [12, 13]. Second, a fundamental idea used in [4, 2, 12, 13] is that of constructing certain *2-3 trees* to show that after an initial phase, the residual problem decomposes into "small" components that can be handled independently of each other. In order to obtain our improvements, we augment these trees with certain additional nodes, which turns out to be crucial. This augmentation is much more involved in our parallel hypergraph-coloring as compared to our sequential improvement over [12, 13]. Our third contribution is this augmentation approach for parallel hypergraph-coloring. In particular, Lemma 3.1 shows that any vertex-weighted graph has an independent set I such that: (i) the total weight of I is "large", and (ii) the average weight of the vertices in I is almost as large as that of the vertices of the graph (set $\epsilon \to 0$ in Lemma 3.1, so that $1 - \epsilon/2 \to 1$). It is critical for our results that the former be tunable to within any desired factor smaller than one (such as our "$1 - \epsilon/2$" term), of the latter. To our knowledge, standard schemes such as probabilistic ones, can get the former only to within $(1 - \Omega(1))$ of the latter, especially when d is very close to N and/or when the values v_i are disparate.

The next two sections describe our two main results. We conclude in Section 4.

2 The improved sequential version

We now present our improved sequential algorithm for set-system models of LLL applications. Our main theorem here is Theorem 2.1. A previous version of this paper had the stronger requirement in condition (P1) of the theorem – that $pD^4 \leq c_0$, where c_0 is any constant strictly smaller than $1/(8e)$. We have now included the better (i.e., weaker) but more-complicated-looking "$pD^4 \leq (1-\epsilon) \cdot (2e(1+1/D)^{D+1})^{-1}$" in (P1) in order to get slightly-better constants in typical applications where D is "large" – note that $(1+1/D)^{D+1}$ decreases from the value 4 at $D = 1$, to an asymptotic value of e. See also Remark 2.1.

THEOREM 2.1. *Let ϵ be any constant in $(0, 1)$. Suppose we have a collection of events $E_1, E_2, \ldots, E_m$ conforming to Definition 1.1, such that the following three conditions hold:*

(P1) $pD^4 \leq (1-\epsilon) \cdot (2e(1+1/D)^{D+1})^{-1}$;

(P2) *each X_j can be generated from its distribution in poly$(n+m)$ time; and*

(P3) *given values for all the X_j, the truth value of each E_i can be determined in poly$(n+m)$ time.*

Then, there is a randomized poly$(n+m)$-time algorithm that with high probability sets values for the X_j in a manner that avoids all the E_i; the exponent in the running time depends on ϵ.

REMARK 2.1. *Since $(1+1/D)^{D+1}$ is a decreasing function of D starting from $D = 1$, we can take (P1) to be "$pD^4 \leq c_0$, where c_0 is some constant strictly smaller than $1/(8e)$". In fact, it is left as an easy exercise for the reader that as long as p is smaller than and bounded away from $1/2$, there is a simple and efficient algorithmic version for the case $D = 1$. Thus, we may assume that $D \geq 2$, in which case a sufficient condition for (P1) to hold is that "$pD^4 \leq c_0$, where c_0 is some constant strictly smaller than $4/(27e)$".*

We now present the algorithm and its proof of correctness. Define

$$c_0 = c_0(\epsilon, D) \doteq (1-\epsilon) \cdot (2e(1+1/D)^{D+1})^{-1}.$$

Although ϵ is viewed as a fixed constant but D is not, we can still essentially take c_0 to be a constant, since it lies between $\frac{1-\epsilon}{8e}$ and $\frac{1}{2e^2}$. Also define

$$K_0 \doteq ((1+1/D)^{D+1} \cdot c_0^{1/4})^{-1}. \tag{2.1}$$

Thus we have from (P1) that

$$(1+1/D)^D \cdot K_0 p^{1/4} \cdot (D+1) \leq 1. \tag{2.2}$$

Note that K_0 is also lower- and upper-bounded by constants that depend only on ϵ.

Our algorithm has two phases discussed next; the complete algorithm is summarized in § 2.3.

2.1 Phase I We proceed as in the first phase of [12, 13]. Initially, all of the X_j are unfrozen. We process the X_j in the order $X_1, X_2, \ldots, X_n$. When we come to X_j, if it is frozen now, we skip over it (i.e., do nothing and go to the next iteration, wherein we will process X_{j+1}). Suppose X_j is not frozen now. Then, we generate a random value, say v_j, for X_j from X_j's distribution. Let $\mathcal{A}_j$ denote the assignment of all values to the variables among $\{X_1, X_2, \ldots, X_j\}$ that are *currently unfrozen*. We check, for each E_i that depends on X_j, whether $\Pr[E_i \mid \mathcal{A}_j] \leq K_0 p^{1/4}$ or not. (In the calculation of this conditional probability, all variables not included in $\mathcal{A}_j$ are viewed as yet-unassigned, and in

particular as mutually independent of the assignment $\mathcal{A}_j$.) If any such E_i has

$$\Pr[E_i \mid \mathcal{A}_j] \geq K_0 p^{1/4}, \tag{2.3}$$

we label each such E_i "bad", and freeze all yet-unassigned variables in each such E_i, and **also** freeze X_j. Thus, we "undo" the assignment v_j we made for X_j; so, the assignment $\mathcal{A}_j$ now has X_j removed from it. A moment's reflection shows that since X_j was not frozen when we came to consider it, we have ensured the property "$\Pr[E_i \mid \mathcal{A}_j] < K_0 p^{1/4}$ for all i", by conducting such a freezing.

REMARK 2.2. *It appears at first sight from the test (2.3), that we will need to be able to compute conditional probabilities such as* $\Pr[E_i \mid \mathcal{A}_j]$. *However, we only need to estimate such terms to within* $(1 \pm \psi)$ *for some sufficiently-small constant* $\psi > 0$, *say; furthermore, the target of comparison "*$K_0 p^{1/4}$*" in (2.3) is at least inverse-polynomial,* $\Omega(m^{-1/4})$, *without loss of generality. Hence, approximate versions of tests such as (2.3) can be done in polynomial time by sampling multiple times (i.e., by randomly generating the variables not assigned by* $\mathcal{A}_j$*). We will present the details of this simple scheme in the full version, and establish that conditions (P1), (P2) and (P3) are indeed sufficient. We assume here for convenience that terms such as* $\Pr[E_i \mid \mathcal{A}_j]$ *can be computed efficiently.*

Thus, at the end of n steps, we arrive at a partial assignment $\mathcal{A} = \mathcal{A}_n$ of the variables. All notions of being frozen will from now on be w.r.t. $\mathcal{A}$.

2.2 Phase II Before giving the basic idea of the algorithm of Phase II, we start with a definition of "weight", and of an intersection-graph G.

DEFINITION 2.1. ***(Weights*** w_i***)*** *For all* $i \in [m]$, *the* weight w_i *is* $\Pr[E_i \mid \mathcal{A}]$.

Importantly, our freezing process ensures that

$$\forall i \in [m],\ w_i < K_0 p^{1/4}. \tag{2.4}$$

Let $F = \{j : X_j \text{ is frozen}\}$. The already-set variables will not be changed; our goal is to assign values to the frozen variables so that all the E_i are avoided. Let us formalize the remaining problem.

Define $R = \{i \in [m] : \exists j \in S_i \text{ such that } X_j \text{ is frozen}\}$. Note that for any E_i such that $i \notin R$, its truth-value is completely determined now (since all X_j on which E_i depends, have been fixed); thus, $w_i = 0$ or 1 for such an i. We have from (2.4) and (2.2) that $w_i < K_0 p^{1/4} < 1$, so since $w_i \in \{0, 1\}$, $w_i = 0$. Thus, we do not have to worry about those E_i for which $i \notin R$. Define a set $B \subseteq [m]$ by setting $i \in B$ iff E_i was declared "bad" at some point; note that $B \subseteq R$. (B denotes "bad" and R denotes "remaining".) Construct an intersection-graph G on vertex-set R as follows:

- first put an edge between distinct vertices $i, j \in R$ iff $(S_i \cap F) \cap (S_j \cap F) \neq \emptyset$. That is, i and j are made adjacent if the restrictions of S_i and S_j to F, intersect each other;
- next, within each *connected component* of G, **add** an edge between distinct i and j if $S_i \cap S_j \neq \emptyset$ (if they were not already adjacent); we emphasize that no edges are added between different connected components here.

Algorithm: basic idea. Since the X_j's are all independent, it is easy to see that all connected components in G can be handled separately. Consider any one such component $\mathcal{C}$. By an abuse of notation, we will use "$i \in \mathcal{C}$" to mean "i lies in the vertex-set of $\mathcal{C}$". The basic algorithmic idea is to generate all X_j such that $j \in F \cap (\bigcup_{i \in \mathcal{C}} S_i)$ independently from their distributions. We next develop an analysis of this algorithm (specifically, what happens when we repeat it sufficiently many times), a key part of which is Theorem 2.2.

It is still true that the "dependency" among the events in $\mathcal{C}$ is at most D; it is also apparent from (2.4) and (2.2) that for any E_i such that $i \in \mathcal{C}$,

$$(1 + 1/D)^D \cdot \Pr[E_i \mid \mathcal{A}] \cdot (D + 1) \leq 1.$$

So, we have from Theorem 1.1 that with positive probability, all events E_i such that $i \in \mathcal{C}$ will be avoided, by the basic algorithm of the previous paragraph. However, we can go further, and make use of the explicit probability-lower-bound given by Theorem 1.1: the probability that all bad events in $\mathcal{C}$ are avoided is at least

$$\prod_{i \in \mathcal{C}} (1 - e \cdot w_i) \geq \exp(-\sum_{i \in \mathcal{C}} 2e \cdot w_i), \tag{2.5}$$

where $\exp(x)$ denotes e^x. The inequality here follows from basic calculus since, by (2.4) and (2.2), $w_i \leq 1/4$ for all $i \in \mathcal{C}$. (That is, we are using the fact that $1 - x \geq \exp(-2x)$ for $x \in [0, e/4]$.) Define $w(\mathcal{C}) = \sum_{i \in \mathcal{C}} w_i$. Thus, our goal will be to show that for a constant $K = K(\epsilon)$, we have with high probability that *for all* components $\mathcal{C}$ of G, $w(\mathcal{C}) \leq K \ln m$. In this case, (2.5) guarantees that each $\mathcal{C}$ has at least an inverse-polynomial (i.e., m^{-2eK}) probability of success, which can be boosted appropriately by repeating polynomially

many times; and, as mentioned above, the different components can be dealt with separately. Directly employing (2.5) and upper-bounding the total weight of any component, is the first key to our improved results.

Thus, proving Theorem 2.2 will help complete the proof of Theorem 2.1 as seen a few lines above; we will also set the value of K_0 during the course of proving Theorem 2.2.

THEOREM 2.2. *There is an explicitly-computable constant $K = K(\epsilon)$ such that with high probability, $w(\mathcal{C}) \leq K \ln m$ holds simultaneously for all components $\mathcal{C}$ of G.*

Proof. As witnesses for some component $\mathcal{C}$ having large weight, we will construct *2-3 trees* as in [4, 2, 12, 13], but will critically augment these with certain other nodes.

Let $B_{\mathcal{C}}$ denote the set of vertices of $\mathcal{C}$ that lie in B, and let $\overline{B}_{\mathcal{C}}$ denote all the other vertices in $\mathcal{C}$. Given $\mathcal{C}$, we will construct a tree T' whose vertices are a subset of those of $\mathcal{C}$. First, let T be a *maximal* 2-3 tree for $\mathcal{C}$ as defined in [4, 2, 12, 13]. That is: (i) each vertex of T lies in $B_{\mathcal{C}}$; (ii) the distance between any two of these vertices in G is at least two, and (iii) if we connect each pair of these vertices whose distance in G is either two or three, we get a connected graph. Further, this structure is maximal in the sense that no further vertex from $B_{\mathcal{C}}$ can be added to it while preserving these three properties. T is obtained by arbitrarily deleting edges in cycles from this connected graph, to obtain a tree.

We now augment T with some vertices from $\overline{B}_{\mathcal{C}}$, in order to obtain T'. Given a vertex $i \in \overline{B}_{\mathcal{C}}$, call i "heavy" if $w_i \geq 4c_0/D^2$, and "light" otherwise. Let S denote the set of **heavy** vertices that are **not** adjacent in $\mathcal{C}$ to any node of T, and let $\ell = |S|$. Since the maximum degree of $\mathcal{C}$ is at most D, there exists some independent set of the heavy vertices, with cardinality $\lceil \ell/(D+1) \rceil$; call this I. A moment's reflection about out freezing process shows that every vertex of $\overline{B}_{\mathcal{C}}$ is adjacent to some vertex of $B_{\mathcal{C}}$ in $\mathcal{C}$. As proved in [4], this property can be used to show that since T is a *maximal* 2-3 tree, every vertex in $B_{\mathcal{C}}$ that does not lie in T, has some neighbor in T. Next, these two properties together imply:

PROPOSITION 2.1. Every vertex $i \in S$ is at distance two from some vertex $h(i)$ in T.

Augment T with the vertices I and edges $\{(i, h(i)) : i \in I\}$. This yields T'.

We first count the number of ways in which we can construct T' from the given set-system, and then bound its probability. Fix t, the number of nodes in T, and $r = \lceil \ell/(D+1) \rceil$. It is known that choosing the shape, root, and remaining nodes of T can be done in at most $m \cdot (eD^3)^t$ ways: see, e.g., the proof of Lemma 2.1 in [2]. Fix a choice for T. Next, let us choose how many augmenting edges of the form $(i, h(i))$ we will add to each node of T. A standard combinatorial argument shows that the number of ways of making this choice is $\binom{t+r-1}{t-1} \leq 2^{t+r}$. Finally, given this sequence of numbers, the actual choice of added nodes can be done in $(D^2)^r$ ways: an added edge to some $u \in T$ can be selected in at most D^2 ways, since the dependency graph has degree at most D, and from Proposition 2.1. Thus, the total number of ways of constructing (T, T') with parameters t and $r = \lceil \ell/(D+1) \rceil$, is at most

$$m \cdot (2eD^3)^t \cdot (2D^2)^r. \tag{2.6}$$

Note that this is a "global" count, given by the original collection of events $E_1, E_2, \ldots, E_m$, and is not derived from some particular choice of $\mathcal{C}$.

Next, given such a choice of (T, T'), we will bound its probability of appearance. The inequality $\Pr[U \mid V] \leq \Pr[U]/\Pr[V]$ has two consequences: (i) for all i, $\Pr[i \in B]$ is at most $p/(K_0 p^{1/4}) = p^{3/4}/K_0$, and similarly (ii) for all i, the probability that i is heavy is at most $p/(4c_0/D^2) \leq 1/(4D^2)$, where the inequality follows from (P1). We note that by construction, all events E_i corresponding to the nodes i in T' are mutually independent. Thus, our bounds (i) and (ii) here show that the probability of a given pair (T, T') emerging, is at most $(p^{3/4}/K_0)^t \cdot (4D^2)^{-r}$. Combining with (2.6) and simplifying, we get from a union bound that $\Pr[\exists\ (T, T')$ with parameters $(t, \ell)]$ is at most

$$\begin{aligned} m \cdot (2eD^3 p^{3/4}/K_0)^t \cdot (1/2)^{\ell/(D+1)} &\leq \\ m \cdot (2ec_0^{3/4}/K_0)^t \cdot (1/2)^{\ell/(D+1)} &= \\ m \cdot (2e(1+1/D)^{D+1} c_0)^t \cdot (1/2)^{\ell/(D+1)} &\leq \\ m \cdot (1-\epsilon)^t \cdot (1/2)^{\ell/(D+1)}. & \end{aligned} \tag{2.7}$$

Now let us derive some necessary conditions on t and ℓ in order for some component $\mathcal{C}$ to have "large" total weight. Recall from (2.4) and (2.2) that $w_i \leq ((1+1/D)^D \cdot (D+1))^{-1}$ for all i with probability one, and hence in particular that $w_i \leq 1/(2(D+1))$. Given (t, ℓ), how large a weight can $\mathcal{C}$ have? Since the maximum degree in $\mathcal{C}$ is at most D (with probability one), the total weight of the nodes in T and their adjacent nodes, is at most $t(D+1) \cdot ((1+1/D)^D \cdot (D+1))^{-1} \leq t/2$. Our discussion above (e.g., the argument that led to Proposition 2.1) shows that all nodes of $\mathcal{C}$ that are not adjacent to T, are at distance at most two from T; further, all of these lie in $\overline{B}_{\mathcal{C}}$. These nodes are either heavy or light. The light nodes are at most tD^2 in number, and thus contribute a total weight of at most $tD^2 \cdot (4c_0/D^2) = 4c_0 t$. Each heavy node has weight at most $1/(2(D+1))$, and so the total weight of these is at most $\ell/(2(D+1))$. Thus, $w(\mathcal{C}) \leq t(1/2 + 4c_0) +$

$\ell/(2(D+1)) \leq t\lambda + \ell/(2(D+1))$, where λ denotes the constant $1/2 + 2/e^2$.

Thus, a necessary condition for $w(\mathcal{C}) > K \ln m$ to hold is that (t, ℓ) lies in the set

$$U \doteq \{(t,\ell) : \ t\lambda + \ell/(2(D+1)) \geq K \ln m\}.$$

Let $\alpha = \alpha(\epsilon)$ be given by $(1-\epsilon)^{1/\lambda}$; note that for any fixed $\epsilon \in (0,1)$, α also lies in $(0,1)$. Therefore, bound (2.7) shows that $\Pr[\exists$ comp. $\mathcal{C}$ with $w(\mathcal{C}) > K \ln m]$ is at most

$$\begin{aligned} m \cdot \sum_{(t,\ell):\ (t,\ell)\in U} (1-\epsilon)^t \cdot (1/2)^{\ell/(D+1)} \quad &= \\ m \cdot \sum_{(t,\ell):\ (t,\ell)\in U} \alpha^{t\lambda} \cdot (1/4)^{\ell/(2(D+1))} \quad &\leq \\ m \cdot \sum_{(t,\ell):\ (t,\ell)\in U} (\max\{\alpha, 1/4\})^{t\lambda + \ell/(2(D+1))}. \quad &(2.8) \end{aligned}$$

For any $i \geq K \ln m$, there are at most $O(i)$ pairs (t,ℓ) lying in U that satisfy $t\lambda + \ell/(2(D+1)) = i$. So, (2.8) gives that $\Pr[\exists$ comp. $\mathcal{C}$ with $w(\mathcal{C}) > K \ln m]$ is at most

$$m \cdot \sum_{i \geq K \ln m} O(i) \cdot (\max\{\alpha, 1/4\})^i;$$

we can take $K = K(\epsilon)$ large enough so that this bound is at most $1/3$.

Thus, with probability at least $2/3$, all components will have weight at most $K \ln m$. This probability can be amplified by repetition. □

2.3 Algorithm summary Thus, the complete algorithm has two basic phases:

1. Repeat Phase I – i.e., generate the partial assignment $\mathcal{A} = \mathcal{A}_n$ – until all the components $\mathcal{C}$ defined by $\mathcal{A}$ have $w(\mathcal{C}) \leq K \ln m$.
2. For each component $\mathcal{C}$ separately: repeat

   generate all X_j such that $j \in F \cap (\bigcup_{i\in\mathcal{C}} S_i)$ independently

   until no E_i such that $i \in \mathcal{C}$ is true.

As discussed above, the expected running-time is $\text{poly}(n, m)$.

An additional sample improvement follows from the work of [10]: given any undirected graph with δ and Δ denoting its minimum and maximum degrees respectively, we can efficiently construct a *domatic partition* for it (see [10]) with at least $(1/4 - o(1))\delta/\ln\Delta$ blocks in the partition, where the "$o(1)$" term is a function of Δ that tends to zero as $\Delta \to \infty$.

3 Hypergraph-coloring in parallel

Recall that $c^* \sim 1/10.331$ is the unique solution of the equation $3x + H(2x) = 1$. We are given a k-uniform hypergraph (V, E) with $V = [n]$, $E = \{S_1, S_2, \ldots, S_m\}$, and overlap at most $D = c_1 \cdot 2^{kc}$, where $c_1 = c_1(\delta)$ and $c = c^* - \delta$ for some positive constant δ. We now present an RNC algorithm to two-color H if the positive value $c_1(\delta)$ is small enough.

We can make the following assumptions w.l.o.g.: $c_1 \leq 1/8, k \geq 3/c$, and $D \geq 2$. We will choose $c_1(\delta)$ small enough (and at most $1/8$). Now if $k < 3/c$ or $D < 2$, then $D < 2$, which means that the overlap among the S_i is at most one, in which case it is easy to 2-color H in NC; so we may assume that $k \geq 3/c$ and $D \geq 2$.

The following combinatorial lemma will be useful:

LEMMA 3.1. *Suppose we are given a graph with vertex-set $\mathcal{N}$ of cardinality N, maximum degree at most d, and with a value $v_i \geq 0$ for each vertex i. Then, for any parameter $\epsilon \in (0,1]$, the graph has an independent set I of cardinality at most $\lceil N\epsilon/d \rceil$ such that $\sum_{i\in I} v_i \geq \epsilon(1-\epsilon/2) \cdot (\sum_{i\in\mathcal{N}} v_i)/d$.*

Proof. We employ an approach of [17] to generate a random bit X_i for each vertex i of $\mathcal{N}$ such that:

(i) $\Pr[X_i = 1] = \epsilon/d$ for each i;

(ii) With probability one, $\sum_i X_i$ lies in $\{\lfloor N\epsilon/d \rfloor, \lceil N\epsilon/d \rceil\}$; and

(iii) the X_i are *negatively correlated* with each other: in particular, for any distinct i and j, $\Pr[X_i = X_j = 1] \leq (\epsilon/d)^2$.

(Such a distribution can also be obtained as a convex combination of two hypergeometric distributions.) See [1, 11] for related ideas. In addition, following [14], we generate a random permutation π of the vertices, and let Y_{uv} be the indicator random variable for vertex u appearing before vertex v in π.

We construct an independent set by the following alteration process:

(a) tentatively select each i with $X_i = 1$; and

(b) unselect each such selected i for which some neighbor j that *precedes i in π* was also selected in step (a).

Clearly, the finally-selected vertices constitute an independent set I, and it easily follows from property (ii) above that $|I| \leq \lceil N\epsilon/d \rceil$ with probability one (note that while we may unselect some selected vertices, we do not add any further to the set of selected vertices). Next, let

Z_i be the indicator random variable for i finally getting selected, and let Γ_i denote the set of neighbors of i. We have

$$\begin{aligned}\mathbf{E}[Z_i] &\geq \mathbf{E}[X_i] - \sum_{j\in\Gamma_i} \mathbf{E}[Y_{ji}]\cdot\mathbf{E}[X_iX_j] \\ &\geq \epsilon/d - \sum_{j\in\Gamma_i}(1/2)\cdot(\epsilon/d)^2 \\ &\geq \epsilon(1-\epsilon/2)/d,\end{aligned}$$

where the second inequality follows from (iii). Thus we have that the expected total value of I is at least $\epsilon(1-\epsilon/2)\cdot(\sum_{i\in\mathcal{N}} v_i)/d$. Since the bound $|I| \leq \lceil N\epsilon/d\rceil$ holds with probability one, we are done. □

Choice of constants. We will pick five positive constants in the following order (thus, each choice may depend on the choices made already): a sufficiently-large absolute constant K_1, a sufficiently small $\epsilon = \epsilon(\delta)$, a sufficiently small $c_1 = c_1(\delta)$ (our main constant), a sufficiently large $K_2 = K_2(\delta)$, and a sufficiently large $K_3 = K_3(\delta)$.

Our *RNC* algorithm conceptually consists of two phases, and is as follows.

Phase I. This phase is basically the same as in [2], but with a different choice of a constant. Color the vertices Red or Blue uniformly at random and independently. Call $i \in [m]$ *bad* if at most ck of S_i's vertices are of any one color (Red or Blue). Freeze (i.e., uncolor) all vertices in all S_i for which i is bad. Let $\mathcal{A}$ denote this partial coloring, and let F denote the set of frozen vertices.

Phase II. The already-colored vertices will not be recolored; we will now color the vertices in F. The algorithm will basically be a parallelization of that of Phase II of § 2; the analysis will be substantially different. Given the coloring $\mathcal{A}$ from Phase I, we define the weights w_i just as in Definition 2.1, where E_i is the bad event that edge S_i becomes monochromatic (if all vertices in F get colored randomly). Call $i \in [m]$ *dangerous* iff:

(Q1) all of S_i's already-colored vertices (if any) have the same color, and

(Q2) S_i has at least ck frozen vertices.

Condition (Q2) is as in [2]; Condition (Q1) is an addition that will be crucial to our analysis. Note that if i is bad, then it is also dangerous. Also, the reader can verify that if i is *not* dangerous, then S_i will remain bichromatic under any coloring of F. Thus, we need only focus on the dangerous i. Analogously to § 2, let R be the set of dangerous i, and define a graph G with vertex-set R just as in § 2. Consider any component $\mathcal{C}$ of G. It is easy to check the following two inequalities:

$$\forall i \in \mathcal{C},\ w_i \leq \max\{2^{-kc}, 2^{1-k}\} = 2^{-kc}; \tag{3.9}$$

$$e\cdot 2^{-kc}\cdot(D+1) \leq 1, \tag{3.10}$$

where (3.10) holds since $c_1 \leq 1/8$ and $k \geq 3/c$. Thus, by Theorem 1.1, there exists some way of coloring the frozen vertices corresponding to $\mathcal{C}$, so that all edges corresponding to $\mathcal{C}$ become two-colored. Our key result will be, just as in Theorem 2.2, that all components $\mathcal{C}$ of G simultaneously have weight at most $K_3 \ln m$ with high probability; hence, (2.5) helps show that each $\mathcal{C}$ has a probability at least m^{-2eK_3} of becoming successful in one random coloring of F in Phase II.

Overall algorithm. The algorithm is a natural parallelization of that of § 2.3. We repeat Phase I in parallel until all components have weight at most $K_3 \ln m$. Then, we handle each component $\mathcal{C}$ in parallel, and randomly color its frozen vertices separately $m^{2eK_3}\ln(2m)$ times in parallel; if there is some $\mathcal{C}$ on which none of these $m^{2eK_3}\ln(2m)$ independent attempts was successful, we again handle all such unsuccessful components. The expected running time is polylogarithmic, and the number of processors needed is polynomial.

Analysis. We now show that for an appropriate choice of $(K_1, \epsilon, c_1, K_2, K_3)$, all components resulting from Phase I have weight at most $K_3 \ln m$, with high probability. Let B be the set of bad indices. Given a component $\mathcal{C}$, we define the 2-3 tree T (with some t nodes) just as in § 2. For all i,

$$\Pr[i \in B] = 2\cdot\sum_{j\leq kc}\binom{k}{j}2^{-k} \leq p_0 \doteq 2\cdot 2^{-k(1-H(c))}.$$

By the argument of § 2, we have the following proposition:

PROPOSITION 3.1. *Let t denote the size of a possible 2-3 tree T. Then, (a) the number of choices for T is at most $m\cdot(eD^3)^t$, and (b) the probability that a specific T_0 of size t becomes a 2-3 tree is at most p_0^t.*

Note that if there is a 2-3 tree of size t_2, then there is also one of any size $t_1 \leq t_2$. Thus, there is an absolute constant K_1 such that the probability of existence of some 2-3 tree of size more than $K_1(\ln m)/k$, is at most

$$m\cdot(2ec_1^3 2^{-k(1-H(c)-3c)})^{\lceil K_1(\ln m)/k\rceil} \leq 1/m; \tag{3.11}$$

the inequality follows from the fact that $1-H(c)-3c$ is lower-bounded by the positive constant $H(2c^*)-H(c^*)$

for $c \le c^*$ (recall that c^* is the unique solution of the equation $3x + H(2x) = 1$, and note that the function $x \mapsto 3x + H(x)$ increases for $x \le c^*$).

Thus, we need only focus on the case $t \le K_1(\ln m)/k$. Fix such a t, and a 2-3 tree T of size t. Let us bound $w(\mathcal{C})$. The weight of T and its neighboring nodes in $\mathcal{C}$ is, by (3.9) and (3.10), at most $t(D+1)2^{-kc} \le t/e$. As in § 2, the set of other nodes S_0 in $\mathcal{C}$ has all its elements at distance 2 from T. We now use much more care in partitioning S_0 into "heavy" and "light" nodes. Let j_0 be the largest integer such that $2^{-j_0} \ge 1/(e(D+1))$, and j_1 be the largest integer such that $2^{-j_1-1} \ge 1/(5D^2)$. Since $D \ge 2$, we have $1/(2.5D^2) \le 1/(e(D+1))$, and so $j_0 \le j_1$. For integers $j \in [j_0, j_1]$, let f_j denote the interval $(2^{-j-1}, 2^{-j}]$; also define $f_{j_1+1} = [0, 2^{-j_1-1}]$. For integers $j \in [j_0, j_1+1]$, let $F_j = \{i \in S_0 : w_i \in f_j\}$. It follows from (3.9) and (3.10) that these sets F_j partition S_0. We declare $i \in S_0$ to be "light" if $i \in F_{j_1+1}$, or if ($i \in F_j$ for $j \le j_1$, and $|F_j| \le t2^{j+1}$). All other elements of S_0 define the set S of heavy elements. Now, since $|S_0| \le tD^2$, the total weight of the elements in F_{j_1+1} is at most $2t$; and more importantly, the definition of "light" and the fact that the number of integers in $[j_0, j_1]$ is at most $\lg D + O(1)$, imply that the total weight of the remaining light elements is at most $2t(\lg D + O(1))$, which is at most $2.5K_1 \ln m$ for large enough K_1, since $t \le K_1(\ln m)/k$. And since T and its neighbors add at most t/e weight as seen above, we get the following:

PROPOSITION 3.2. *The vertices of $\mathcal{C}$ that lie outside S, contribute a total weight of at most $3K_1 \ln m$.*

Bounding the total weight of S is more involved. As in § 2, we will augment T with some nodes from S in order to get a tree T', and argue that if $w(\mathcal{C})$ is "large", then the probability of obtaining a corresponding T' is very small. Suppose $|S| = \ell$, and that the range of weights of its elements is $[a_0, a_1]$, where $1/(5D^2) \le a_0 \le a_1 \le 1/(e(D+1))$. Note that each element of S is dangerous but not bad, and recall conditions (Q1) and (Q2) for becoming dangerous; suppose that for each $i \in S$, S_i has kx_i frozen vertices, for $x_i \ge c$. Define $v_i = k(1 - x_i - H(x_i))$. Construct an independent set I of S by using Lemma 3.1 on the subgraph of $\mathcal{C}$ induced by S, and with parameter $\epsilon = \epsilon(\delta)$; we get

PROPOSITION 3.3. *$r = |I|$ is at most $\lceil \ell\epsilon/D \rceil$; also, $\sum_{i \in I} v_i$ is at least $\epsilon(1 - \epsilon/2) \cdot (\sum_{i \in S} v_i)/D$.*

As in § 2, we obtain T' by augmenting T with the vertices I and edges $\{(i, h(i)) : i \in I\}$, where each $h(i)$ lies in T and is at distance two from i in $\mathcal{C}$. We will now count the number of choices for (T, T') with parameters t and $r = |I|$ more carefully than in (2.6). Recall that T has been fixed. Suppose node $\#j$ of T connects to y_j nodes of I in our augmentation, where $\sum_j y_j = r$. As in § 2, this sequence $\langle y_j \rangle$ can be chosen in at most 2^{t+r} ways. Fix this sequence, and suppose $s \le \min\{t, r\}$ nodes of T actually connect to S in our augmentation, i.e., have $y_j \ge 1$. Then, given these s nodes of T, their augmenting edges can be chosen in at most

$$\left(\prod_{j=1}^{s} \binom{D^2}{y_j} \right) \le \left(\prod_{j=1}^{s} (D^2 e/y_j)^{y_j} \right) \le e^t (D^2 s/r)^r$$

ways, where the last inequality is shown using the convexity of the function $x \mapsto x \ln x$ for $x > 0$. Combining with Propositions 3.1 and 3.3, the number of choices of (T, T') with parameters t and ℓ is at most the following, for some $r \le \lceil \ell\epsilon/D \rceil$:

$$m \cdot (2e^2 D^3)^t \cdot (2D^2 \min\{t, r\}/r)^r. \tag{3.12}$$

Since $v_i = k(1 - x_i - H(x_i))$, we have for all i and for any given $x_i \ge c$ that the probability of i being dangerous with $|S_i \cap F| = kx_i$, is at most

$$2 \cdot \binom{k}{kx_i} 2^{-k(1-x_i)} \le 2^{1-v_i};$$

we remind the reader that this probability is taken over the random choices made in Phase I. Recall Propositions 3.1 and 3.3. Since all nodes of T' form an independent set, the probability of getting a particular (T, T') with parameters t, ℓ and $\{x_i\}$, is at most

$$\begin{aligned} p_0^t \cdot \left(\prod_{i \in I} 2^{1-v_i} \right) &\le p_0^t 2^{\lceil \ell\epsilon/D \rceil} \cdot 2^{-\epsilon(1-\epsilon/2) \cdot (\sum_{i \in S} v_i)/D} \\ &\le 2p_0^t \cdot 2^{\ell\epsilon/D} \cdot 2^{-\epsilon(1-\epsilon/2) \cdot (\sum_{i \in S} v_i)/D}. \end{aligned} \tag{3.13}$$

Now, if $i \in S$ is dangerous and S_i has kx_i frozen vertices, then since $w_i \le 1/(e(D+1))$, we can verify that $x_i < 1$; therefore, $w_i = 2^{-kx_i}$. Let $\Phi(a_0, a_1, \ell; W)$ denote the minimum possible value of $\sum_{i \in S} v_i$ for given (a_0, a_1, ℓ), assuming $W = \sum_{i \in S} 2^{-kx_i}$. (The variables in this minimization are the x_i, which are constrained by $a_0 \le w_i \le a_1$.) Combining (3.11), Proposition 3.2, (3.12) and (3.13), $\Pr[w(\mathcal{C}) \ge 3K_1 \ln m + K_2 \ln m]$ is at most the following, for some $r \le \lceil \ell\epsilon/D \rceil$: $1/m$ plus the sum, over all $t \le K_1(\ln m)/k$, of

$$2m \cdot (2e^2 D^3 p_0)^t \cdot (2D^2 \min\{t, r\}/r)^r \cdot 2^{(\epsilon/D) \cdot (\ell - (1-\epsilon/2)\Phi(a_0, a_1, \ell; K_2 \ln m))}. \tag{3.14}$$

The following lemma helps lower-bound Φ. The facts that $k \ge 3/c$ and $w(x)$ is "small" are critical, since the function g is not uniformly concave:

LEMMA 3.2. *Let g be the function such that given $y = w(x) = 2^{-kx}$ and $v(x) = k(1 - x - H(x))$, $g(w(x)) = v(x)$. That is, $g(y) = k(1 + (\lg y)/k - H(-\lg y/k))$. Then, for $k \geq 3/c$, any a_0, a_1 such that $1/(5D^2) \leq a_0 \leq a_1 \leq 1/(e(D+1))$, and in the range of x for which $a_0 \leq w(x) \leq a_1$, i.e., $a_0 \leq y \leq a_1$, $g(y)$ is a concave function of y.*

Proof. Define $G_1(z) = z + H(z)$, $G_2(y) = \lg(1/y)/k$, and $G(y) = G_1(G_2(y))$. We need to show that $G(y)$ is convex for the range of y as in the lemma. We have $G'(y) = G_2'(y) \cdot G_1'(G_2(y))$; so,

$$G''(y) = G_2''(y) \cdot G_1'(G_2(y)) + (G_2'(y))^2 \cdot G_1''(G_2(y)).$$

Let us calculate this. Since

$$G_1(z) = z - \frac{1}{\ln 2} \cdot (z \ln z + (1-z)\ln(1-z)),$$

we have

$$\begin{aligned} G_1'(z) &= 1 - \frac{1}{\ln 2} \cdot (\ln z - \ln(1-z)); \\ G_1''(z) &= -\frac{1}{z(1-z)\ln 2}. \end{aligned}$$

Also, $G_2'(y) = -1/(ky \ln 2)$ and $G_2''(y) = 1/(k \ln 2 \cdot y^2)$. Set $x = -\ln y/(k \ln 2)$, and define

$$F(x) = 1 - \frac{\ln x - \ln(1-x)}{\ln 2} - \frac{1}{k(\ln 2)^2 x(1-x)}.$$

Thus, $G''(y) = ((k \ln 2)y^2)^{-1} \cdot F(x)$.

We therefore want to prove that $F(x)$ is non-negative when $1/(5D^2) \leq 2^{-kx} \leq 1/(e(D+1))$. We have

$$\begin{aligned} F'(x) &= \frac{-1}{\ln 2} \cdot \left(\frac{1}{x(1-x)} - \frac{1-2x}{(k \ln 2) \cdot (x(1-x))^2} \right) \\ (3.15) \quad &= \frac{-1}{(\ln 2) \cdot x(1-x)} \cdot \left(1 - \frac{1-2x}{(k \ln 2) \cdot x(1-x)} \right). \end{aligned}$$

Recall that $e(c_1 2^{kc} + 1) \leq 2^{kx} \leq 5c_1^2 2^{2kc}$. Since $c_1 \leq 1/8$, we have $e \leq 2^{kx} \leq 2^{2kc}$, i.e.,

$$kx \ln 2 \geq 1 \text{ and } x \leq 2c.$$

As $0 \leq x \leq 2c < 1/2$, all of x, $1-x$, and $1-2x$ are non-negative. Thus, from (3.15),

$$F'(x) \leq \frac{-1}{(\ln 2) \cdot x(1-x)} \cdot \left(1 - \frac{1-2x}{1-x} \right) = \frac{-1}{(\ln 2)(1-x)^2},$$

which is negative for the entire interval $0 \leq x \leq 2c$.

So, for any x in our domain,

$$\begin{aligned} F(x) &\geq F(2c) \\ &\geq 1 - \frac{1}{k(\ln 2)^2 2c(1-2c)} \\ &\geq 1 - \frac{1}{6(\ln 2)^2(1-2c)} \quad (\text{since } k \geq 3/c) \\ &\geq 1 - \frac{1}{6(\ln 2)^2(1-2c^*)} \quad (\text{since } c \leq c^*) \\ &\geq 0, \end{aligned}$$

as required. □

Thus, an adversary who wants to minimize $\sum_{i \in S} v_i$ for given (a_0, a_1, ℓ), will set b_0 of the w_i's to be a_0 and b_1 of the w_i's to be a_1, where b_0 and b_1 are non-negative integers such that $b_0 + b_1 = \ell$. Further work helps upper-bound the term "$(2D^2 \min\{t, r\}/r)^r$" in (3.14). Using these ideas, we can show that by choosing ϵ small enough, then c_1 small enough, and then K_2 large enough, the bound (3.14) can be made at most $2/m$, say. Finally, we set $K_3 = 3K_1 + K_2$. We will present the details in the full version.

4 Conclusion

The main open question is to make further progress toward a goal such as "$pD^{1+o(1)} \leq O(1)$" being sufficient for algorithmic versions; even making "$pD^{3-\Omega(1)} \leq O(1)$" a sufficient condition seems challenging using currently-known methods, with the 2-3 trees-based approach being the bottleneck. It would also be very useful to further develop the theory of algorithmic versions of the Asymmetric version of the Local Lemma. Nothing is known about constructive versions of the Lopsided Local Lemma [3]; some initial steps in this direction would be of interest.

Finally, our algorithm is fundamentally randomized, due to the random process of Phase II. Can it be derandomized?

References

[1] A. Ageev and M. Sviridenko, *Pipage rounding: a new method of constructing algorithms with proven performance guarantee*, Journal of Combinatorial Optimization, 8 (2004), pp. 307–328.

[2] N. Alon, *A parallel algorithmic version of the Local Lemma*, Random Structures & Algorithms, 2 (1991), pp. 367–378.

[3] N. Alon and J. H. Spencer, *The Probabilistic Method*, Second Edition, Wiley, 2000.

[4] J. Beck, *An algorithmic approach to the Lovász Local Lemma*, Random Structures & Algorithms, 2 (1991), pp. 343–365.

[5] J. Beck and S. Lodha, *Efficient proper* 2*-coloring of almost disjoint hypergraphs*, Proc. ACM-SIAM Symposium on Discrete Algorithms, pp. 598–605, 2002.

[6] A. Chattopadhyay and B. Reed, *Properly* 2*-colouring linear hypergraphs*, Proc. RANDOM 2007, Lecture Notes in Computer Science 4627, pages 395–408, 2007.

[7] A. Czumaj and C. Scheideler, *Coloring non-uniform hypergraphs: A new algorithmic approach to the General Lovász Local Lemma*, Random Structures & Algorithms, 17 (2000), pp. 213–237.

[8] A. Czumaj and C. Scheideler, *A new algorithmic approach to the General Lovász Local Lemma with applications to scheduling and satisfiability problems*, In Proc. ACM Symposium on Theory of Computing, pages 38–47, 2000.

[9] P. Erdős and L. Lovász, *Problems and results on* 3*-chromatic hypergraphs and some related questions*, In *Infinite and Finite Sets*, A. Hajnal et. al., eds., Colloq. Math. Soc. J. Bolyai 11, North Holland, Amsterdam, 1975, pp. 609–627.

[10] U. Feige, M. M. Halldórsson, G. Kortsarz and A. Srinivasan, *Approximating the Domatic Number*, SIAM Journal on Computing, 32 (2002), pp. 172–195.

[11] R. Gandhi, S. Khuller, S. Parthasarathy, and A. Srinivasan, *Dependent Rounding and its Applications to Approximation Algorithms*, Journal of the ACM, 53 (2006), pp. 324–360.

[12] M. Molloy and B. Reed, *Further algorithmic aspects of the Local Lemma*, In Proc. ACM Symposium on Theory of Computing, pages 524–529, 1998.

[13] M. Molloy and B. Reed, *Graph Colouring and the Probabilistic Method*, Springer, 2000.

[14] S. Pemmaraju and A. Srinivasan, *The Randomized Coloring Procedure with Symmetry-Breaking*, submitted.

[15] J. Radhakrishnan and A. Srinivasan, *Improved Bounds and Algorithms for Hypergraph* 2*-Coloring*, Random Structures & Algorithms, 16 (2000), pp. 4–32.

[16] M. R. Salavatipour, *A* $(1+\epsilon)$*-approximation algorithm for partitioning hypergraphs using a new algorithmic version of the Lovász Local Lemma*, Random Structures & Algorithms, 25 (2004), pp. 68–90.

[17] A. Srinivasan, *Distributions on level-sets with applications to approximation algorithms*, In Proc. IEEE Symposium on Foundations of Computer Science, pages 588–597, 2001.