Distributed Algorithms for the Lovász Local Lemma and Graph Coloring[*]

Kai-Min Chung
Academia Sinica
Taipei, Taiwan
kmchung@iis.sinica.edu.tw

Seth Pettie
University of Michigan
Ann Arbor, MI
pettie@umich.edu

Hsin-Hao Su
University of Michigan
Ann Arbor, MI
hsinhao@umich.edu

ABSTRACT

The Lovász Local Lemma (LLL), introduced by Erdős and Lovász in 1975, is a powerful tool of the probabilistic method that allows one to prove that a set of n "bad" events do not happen with non-zero probability, provided that the events have limited dependence. However, the LLL itself does not suggest how to find a point avoiding all bad events. Since the work of Beck (1991) there has been a sustained effort to find a constructive proof (i.e. an algorithm) for the LLL or weaker versions of it. In a major breakthrough Moser and Tardos (2010) showed that a point avoiding all bad events can be found efficiently. They also proposed a distributed/parallel version of their algorithm that requires $O(\log^2 n)$ rounds of communication in a distributed network.

In this paper we provide two new distributed algorithms for the LLL that improve on both the efficiency and simplicity of the Moser-Tardos algorithm. For clarity we express our results in terms of the symmetric LLL though both algorithms deal with the asymmetric version as well. Let p bound the probability of any bad event and d be the maximum degree in the dependency graph of the bad events. When $epd^2 < 1$ we give a truly simple LLL algorithm running in $O(\log_{1/epd^2} n)$ rounds. Under the tighter condition $ep(d+1) < 1$, we give a slightly slower algorithm running in $O(\log^2 d \cdot \log_{1/ep(d+1)} n)$ rounds. Furthermore, we give an algorithm that runs in *sublogarithmic* rounds under the condition $p \cdot f(d) < 1$, where $f(d)$ is an exponential function of d. Although the conditions of the LLL are locally verifiable, we prove that any distributed LLL algorithm requires $\Omega(\log^* n)$ rounds.

In many graph coloring problems the existence of a valid coloring is established by one or more applications of the LLL. Using our LLL algorithms, we give logarithmic-time distributed algorithms for frugal coloring, defective coloring, coloring girth-4 (triangle-free) and girth-5 graphs, edge coloring, and list coloring.

[*]The last two authors are supported by NSF grants CCF-0746673, CCF-1217338, and CNS-1318294 and a grant from the US-Israel Binational Science Foundation. Part of the work was done while visiting MADALGO at Aarhus University, supported by Danish National Research Foundation grant DNRF84.

PODC'14 July 15–18, 2014, Paris, France
Copyright 2014 ACM 978-1-4503-2944-6/14/07 ...$15.00.
http://dx.doi.org/10.1145/2611462.2611465.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*computations on discrete structures*; G.2.2 [**Discrete Mathematics**]: Graph Theory—*network problems*

Keywords

Lovász Local Lemma; distributed graph coloring; constructive algorithm; probabilistic method; locality

1. INTRODUCTION

Consider a system $\mathcal{P}$ of independent random variables and a set $\mathcal{A}$ of n *bad* events, where each $A \in \mathcal{A}$ depends solely on some subset $\mathrm{vbl}(A) \subseteq \mathcal{P}$. For example, in a hypergraph 2-coloring instance, $\mathcal{P}$ represents the vertex colors and $\mathcal{A}$ the events in which an edge is monochromatic. The dependency graph $G_{\mathcal{A}} = (\mathcal{A}, \{(A,B) \mid \mathrm{vbl}(A) \cap \mathrm{vbl}(B) \neq \emptyset\})$ includes edges between events if and only if they depend on at least one common variable. Let $\Gamma(A)$ be A's neighborhood in $G_{\mathcal{A}}$ and $\Gamma^+(A) = \Gamma(A) \cup \{A\}$ be its inclusive neighborhood. The (general, asymmetric) LLL states [11, 31] that if there is a function $x : \mathcal{A} \to (0,1)$ such that

$$\Pr(A) \leq x(A) \cdot \prod_{B \in \Gamma(A)} (1 - x(B))$$

then $\Pr(\bigcap_{A \in \mathcal{A}} \overline{A}) > 0$, that is, there is a *satisfying assignment* to the underlying variables in which no bad events occur. The symmetric LLL is a useful corollary of the general LLL. If p and d are such that $\Pr(A) \leq p$ and $|\Gamma(A)| \leq d$ for all A, and $ep(d+1) < 1$, then $\Pr(\bigcap_{A \in \mathcal{A}} \overline{A}) > 0$. For example, consider a hypergraph in which each edge contains k vertices and intersects at most $d < 2^{k-1}/e - 1$ other edges. Under a uniformly random color assignment $\mathcal{P} \to \{\text{red,blue}\}$ the probability an edge is monochromatic is $p = 2^{-(k-1)}$, so $ep(d+1) < 1$. The symmetric LLL proves the *existence* of a satisfying color assignment but does not yield an efficient algorithm to find one. Beginning with Beck [7], a long line of research has sought to find efficient (and ideally deterministic) algorithms for computing satisfying assignments [7, 1, 19, 9, 32, 22, 23, 24, 8, 12, 14, 25]. Most of these results required a major weakening of the standard symmetric LLL constraint $ep(d+1) < 1$. In many applications we consider, the bad events are that the sum of $d^{\Theta(1)}$ random variables deviates away from its expectation. So the probability they are violated is often bounded by Chernoff-type tail bounds, e.g. $\exp(-d^{\Theta(1)})$.

In a relatively recent breakthrough, Moser and Tardos [24] gave an *algorithmic* proof of the general asymmetric LLL, with no weakening of the parameters. Their algorithm is simple though the analysis is not trivial. At initialization the algorithm chooses a random assignment to the variables $\mathcal{P}$. Call an event $A \in \mathcal{A}$ *violated* if it occurs under the current assignment to the variables. Let $\mathcal{F} \subseteq \mathcal{A}$ be the set of violated events. The algorithm repeatedly chooses some $A \in \mathcal{F}$ and *resamples* the variables in $\mathrm{vbl}(A)$, until $\mathcal{F} = \emptyset$.

The Distributed LLL Problem.

We consider Linial's LOCAL model [26] of distributed computation in which the distributed network is identical to the dependency graph. In other words, each node $A \in \mathcal{A}$ hosts a processor, which is aware of n, the degree bound d, and its neighborhood $\Gamma(A)$. Computation proceeds in synchronized rounds in which each node may send an unbounded message to its neighbors. *Time* is measured by the number of rounds; computation local to each node is free. Upon termination each node A must commit to an assignment to its variables $\mathrm{vbl}(A)$ that is consistent with its neighbors, i.e., the nodes must collectively agree on a satisfying assignment to $\mathcal{P}$ avoiding all bad events. We consider the LOCAL model because we will need to send the assignment of $\mathrm{vbl}(A)$ in one message, which can be large.

Moser and Tardos proposed a parallel version of their resampling algorithm (Algorithm 1), which can easily be implemented in the LOCAL model. Let $G_{\mathcal{F}}$ be the graph induced by the violated events $\mathcal{F}$ under the current variable assignment. They proved that $O(\log_{1/ep(d+1)} n)$ iterations of Algorithm 1 suffice to avoid all bad events with probability $1 - 1/\mathrm{poly}(n)$, i.e., $O(\log n)$ iterations suffice if $ep(d+1)$ is bounded away from 1[1]. (For the sake of a simpler presentation we shall state many results in the symmetric LLL language. Our algorithms and Moser-Tardos work for the asymmetric LLL as well.) Moser and Tardos suggested using Luby's randomized MIS algorithm [18], which runs in $\Theta(\log n)$ rounds w.h.p. (which can also be achieved by [2]), for a total running time of $\Theta(\log n \cdot \log_{1/ep(d+1)} n)$. This is, intuitively, a very wasteful LLL algorithm since nodes spend nearly all their time computing MISs rather than performing resampling steps. For certain values of d the running time can be improved by plugging in an MIS algorithm running in $O(d + \log^* n)$ time [5] or $O(\log d \cdot \sqrt{\log n})$ time w.h.p. [6].[2] However, it is not possible to find an MIS in constant time. Kuhn et al. [15] gave an $\Omega(\min\{\log d, \sqrt{\log n}\})$ lower bound on the complexity of MIS and other symmetry-breaking problems.

New Results.

We give a new distributed LLL algorithm in the Moser-Tardos resampling framework that avoids the computation of MISs altogether. Due to its simplicity we are happy to display the algorithm in its entirety. We assume that nodes

```
Initialize a random assignment to the variables $\mathcal{P}$.
while $\mathcal{F} \neq \emptyset$ do
    Compute a maximal independent set $\mathcal{I}$ in $G_{\mathcal{F}}$.
    Resample each variable in $\mathrm{vbl}(\mathcal{I}) = \bigcup_{A \in \mathcal{I}} \mathrm{vbl}(A)$.
end while
```

Algorithm 1: The Moser-Tardos Parallel Resampling Algorithm. Here $\mathcal{F}$ is the set of bad events occurring under the current variable assignment and $G_{\mathcal{F}}$ is the dependency graph induced by $\mathcal{F}$.

possess unique IDs, which could be assigned in an adversarial manner. Let $\Gamma_{\mathcal{F}}(A)$ be A's neighborhood in $G_{\mathcal{F}}$.

```
Initialize a random assignment to the variables $\mathcal{P}$

while $\mathcal{F} \neq \emptyset$ do
    Let $\mathcal{I} = \{A \in \mathcal{F} \mid \mathrm{ID}(A) = \min\{\mathrm{ID}(B) \mid B \in \Gamma^+_{\mathcal{F}}(A)\}\}$
    Resample $\mathrm{vbl}(\mathcal{I}) = \bigcup_{A \in \mathcal{I}} \mathrm{vbl}(A)$.
end while
```

Algorithm 2: A Simple Distributed LLL Algorithm

One can see that $\mathcal{I}$ is computed in one round: each node A tells its neighbors whether $A \in \mathcal{F}$ under the current variable assignment. Once A receives messages from all neighbors it can determine if $\mathrm{ID}(A)$ is a local minimum in $G_{\mathcal{F}}$. We prove that under the slightly stronger criterion $epd^2 < 1$, this algorithm halts in $O(\log_{1/epd^2} n)$ steps w.h.p. Most applications of the LLL satisfy the $epd^2 < 1$ criterion, though not all. We give another distributed LLL algorithm in the resampling framework that finds a satisfying assignment in $O(\log^2 d \cdot \log_{1/ep(d+1)} n)$ time under the usual $ep(d+1) < 1$ criterion.

We show that faster algorithms exist when the condition $ep(d+1) < 1$ is replaced by a stronger condition $p \cdot f(d) < 1$, where $f(d)$ is a faster growing function than $e(d+1)$. However, it is not clear whether there exists $f(d)$ so that the LLL can be solved in sublogarithmic time in n, independent of d. Moser and Tardos observed that any parallel algorithm in the resampling framework requires $\Omega(\log_{1/p} n)$ resampling steps, even if the dependency graph has no edges. We combine the resampling framework with a locality approach to give an $O(\log n / \log\log n)$ algorithm for an exponential function $f(d)$. On the other hand, we prove that no constant time distributed LLL algorithm exists and that the LLL for any $f(d)$ requires $\Omega(\log^* n)$ time.

New Applications.

Existential results in graph coloring [20] (those taking the *Rödl nibble* approach) can often be phrased as distributed algorithms in which each step succeeds with some tiny but non-zero probability, as guaranteed by the LLL. By using our distributed LLL algorithms we are able to solve a number of graph coloring problems in $O(\log n)$ time or faster.[3] Some of these applications require minor changes to existing

[1]Note that $\log_{1/ep(d+1)} n$ could be sublogarithmic or superlogarithmic depending on how close $ep(d+1)$ is to 0 or 1.

[2]These MIS algorithms are significantly more complex than Luby's and use larger messages.

[3]Suppose H is both the distributed network and the graph to be colored. When invoking the LLL, the dependency graph $G_{\mathcal{A}}$ is not *identical* to H. Typically bad events in $\mathcal{A}$ are associated with H-vertices and two bad events are adjacent in $G_{\mathcal{A}}$ only if the corresponding vertices are at distance $O(1)$ in H. Thus, a distributed LLL algorithm for $G_{\mathcal{A}}$ can be simulated in H with an $O(1)$ slowdown.

algorithms while others are quite involved. Below Δ is the maximum degree, and $\epsilon > 0$ an arbitrarily small parameter.

Frugal Coloring A k-frugal vertex coloring is one in which each color appears at most k times in the neighborhood of any vertex. Pemmaraju and Srinivasan [27] showed the existence of $(\Delta+1)$-colorings that are $O(\log^2\Delta/\log\log\Delta)$-frugal, and proved that $(\log\Delta\cdot\log n/\log\log n)$-frugal colorings could be computed in $O(\log n)$ time. With some modifications to their proof we show that a $O(\log^2\Delta/\log\log\Delta)$-frugal $(\Delta+1)$-coloring can be computed in $O(\log n)$ time. Notice that the best existential bound on the frugality for $(\Delta+1)$-coloring is $O(\log\Delta/\log\log\Delta)$ by Molloy and Reed [21].

Hind et al. [13] showed there exist β-frugal, $O(\Delta^{1+\frac{1}{\beta}})$-colorings by using the asymmetric LLL. We show how to turn their proof into a distributed algorithm that runs in $O(\log n\cdot\log^2\Delta)$ time.

Girth 4 and 5 In prior work [28] we proved that triangle-free graphs have $(4+\epsilon)\Delta/\ln\Delta$-colorings and gave $O(\log^{1+o(1)} n)$ time algorithms for $(4+\epsilon)\Delta/\ln\Delta$-coloring triangle-free graphs and $(1+\epsilon)\Delta/\ln\Delta$-coloring girth-5 graphs. Here we prove that both problems can be solved in $O(\log n)$ time.

Edge Coloring Dubhashi et al. [10] gave a $(1+\epsilon)\Delta$ edge-coloring algorithm running in $O(\log n)$ time, provided that $\Delta = (\log n)^{1+\Omega(1)}$ is sufficiently large relative to n. We give a logarithmic time algorithm for the same problem that works for any $\Delta > \Delta_\epsilon$, where Δ_ϵ is a sufficiently large constant depending on ϵ.

List-Coloring Suppose each vertex is issued a list of $(1+\epsilon)D > D_\epsilon$ colors such that each color appears in at most D lists in the neighborhood of any vertex, where D_ϵ is a sufficiently large constant depending on ϵ. (D need not be close to the degree Δ.) Reed and Sudakov [29] proved that $(1+\epsilon)D$-list-colorings exist. We show how to construct them in $O(\log D + \log_D n + \log\log D\cdot\log n/D^{1/2}) = O(\log n)$ time. Significant modifications are needed to adapt their approach to a distributed network. Furthermore, for *any* D and any constant $\epsilon > 0$, we show that $(2e+\epsilon)D$ list coloring can be solved in $O(\log n)$ time.

Defective Coloring An f-defective coloring is one in which a vertex may share its color with up to f neighbors. Barenboim and Elkin [4], and implicitly, Kuhn and Wattenhofer [16] gave an $O(1)$ time procedure to compute a $O(\log n)$-defective $O(\Delta/\log n)$-coloring. We prove that for any $f = \Omega(\log\Delta)$, an f-defective $O(\Delta/f)$-coloring can be computed in $O((\log n)/f)$ time.

2. PRELIMINARIES

Let $\Gamma^r(A)$ be the r-neighborhood of A (the set of nodes at distance at most r from A, excluding A) and $\Gamma^{r+}(A) = \Gamma^r(A)\cup\{A\}$ be its inclusive r-neighborhood. A node set in the subscript indicates a restriction of the neighborhood to that set, e.g., $\Gamma^{2+}_{\mathcal{F}}(A) = \Gamma^{2+}(A)\cap\mathcal{F}$.

Consider an execution of a Moser-Tardos-type resampling algorithm. Let $C:\mathbb{N}\to\mathcal{A}$ be such that $C(i)$ is the ith event selected by the algorithm for resampling; C is called the *record* of the execution. (If the algorithm selects events in independent batches then the events in each batch can be listed arbitrarily.) A *witness tree* $\tau = (T,\sigma_T)$ is a finite rooted tree where $\sigma_T : V(T)\to\mathcal{A}$ labels each vertex in T with an event such that the children of $u\in T$ receive labels from $\Gamma^+(\sigma_T(u))$. A *2-witness tree* $\tau = (T,\sigma_T)$ is defined in the same way except that the children of $u\in T$ may receive labels from $\Gamma^{2+}(\sigma_T(u))$. A witness tree (or 2-witness tree) is *proper* if the children of a vertex receive distinct labels.

Given a record C, the witness tree $\tau_C(t)$ is constructed as follows. First, create a root node labelled $C(t)$. Looking backward in time, for each $i = t-1, t-2,\ldots,1$, check if an existing node is labeled with an event from $\Gamma^+(C(i))$. If so, let u be one of the *deepest* such nodes. Create a new node v labeled $C(i)$ and make it a child of u. Given a witness tree τ, we say τ *occurs in* C if there exists an index t such that $\tau_C(t) = \tau$. Moser and Tardos proved the following lemma:

Lemma 2.1. *Let τ be a fixed witness tree and C be the record produced by the algorithm.*

1. If τ occurs in C, then τ is proper.
2. The probability that τ appears in C is at most $\prod_{v\in V(\tau)}\Pr(\sigma_T(v))$.

Similarly, for $r\ge 2$, we can define an r-witness tree $\tau^r_C(t)$ in the same way except that in each step we attach a node labelled $C(i)$ to the deepest node among nodes labelled $\Gamma^{r+}(C(i))$. Also, we say τ r-occurs in C if there exists $t\in\mathbb{N}$ such that $\tau^r_C(t) = \tau$. Then Lemma 2.1 holds analogously:

Lemma 2.2. *Let τ be a fixed r-witness tree and C be the record produced by the algorithm.*

1. If τ r-occurs in C, then τ is proper.
2. The probability that τ appears in C is at most $\prod_{v\in V(\tau)}\Pr(\sigma_T(v))$.

3. DISTRIBUTED CONSTRUCTIVE LOVÁSZ LOCAL LEMMA

Recall that the parallel/distributed Moser-Tardos algorithm iteratively selects maximal independent sets (MIS) of violated events for resampling. They proved that if there is some slack in the general LLL preconditions then the algorithm terminates in $O(\log n)$ rounds of MIS.

Theorem 3.1. *(Moser and Tardos) Let $\mathcal{P}$ be a finite set of mutually independent random variables in a probability space. Let $\mathcal{A}$ be a finite set of events determined by these variables. If there exists an assignment of reals $x:\mathcal{A}\to(0,1)$ such that*

$$\forall A\in\mathcal{A} : \Pr(A) \le (1-\epsilon)x(A)\prod_{B\in\Gamma(A)}(1-x(B)),$$

then the probability any bad event occurs after k resampling rounds of Algorithm 1 is at most $(1-\epsilon)^k\sum_{A\in\mathcal{A}}\frac{x(A)}{1-x(A)}$.

In other words, if $x(A)$ is bounded away from 1 then $O(\log_{\frac{1}{1-\epsilon}} n)$ resampling rounds suffice, w.h.p. A distributed implementation of this algorithm takes $O(\log_{\frac{1}{1-\epsilon}} n\cdot \mathrm{MIS}(n,d))$, where d is the maximum degree of $G_{\mathcal{A}}$ and $\mathrm{MIS}(n,d)$ is the time needed to find an MIS in an n-vertex degree-d graph. It is known that $\mathrm{MIS}(n,d) =$

$\Omega(\min\{\sqrt{\log n}, \log d\})$ [15]. Our algorithms avoid the computation of MISs. In Section 3.1 we analyze the simple distributed LLL algorithm presented in the introduction, which requires slightly weakening the general LLL conditions. In Section 3.2 we present an algorithm that works for the standard LLL conditions but is slower by a $O(\log^2 d)$ factor.

3.1 A Simple Distributed Algorithm

Recall that in each round of Algorithm 2, a violated event $A \in \mathcal{F}$ is selected for resampling if $\mathrm{ID}(A)$ is a local minimum in the violated subgraph $G_{\mathcal{F}}$. In order to analyze this algorithm in the witness-tree framework we must establish some connection between the depth of witness trees and the number of rounds of resampling. Lemma 3.2 will let us make such a connection.

LEMMA 3.2. *Suppose an event A is resampled in round $j > 1$ of Algorithm 2. There must exist some $B \in \Gamma^{2+}(A)$ resampled in round $j-1$.*

PROOF. Let $\mathcal{F}'$ and $\mathcal{F}$ be the violated event sets just before and after the resampling step at round $j-1$. If A is not in $\mathcal{F}'$ but is in $\mathcal{F}$ then its variables $\mathrm{vbl}(A)$ must have been changed in round $j-1$, which could only occur if some $B \in \Gamma(A)$ were resampled. Now suppose A is in both $\mathcal{F}'$ and $\mathcal{F}$. It was not resampled in round $j-1$ but was in round j, meaning $\mathrm{ID}(A)$ is not a local minimum in $\Gamma_{\mathcal{F}'}(A)$ but is a local minimum in $\Gamma_{\mathcal{F}}(A)$. This implies that some neighbor $B \in \Gamma(A)$ with $\mathrm{ID}(B) < \mathrm{ID}(A)$ is in $\mathcal{F}'$ but not $\mathcal{F}$, which could only occur if some $C \in \Gamma^{+}(B) \subseteq \Gamma^{2+}(A)$ were resampled in round $j-1$. □

We can now proceed to bound the number of rounds of Algorithm 2 needed to find a satisfying assignment.

THEOREM 3.3. *(Asymmetric LLL) Let $\mathcal{P}$ be a finite set of mutually independent random variables in a probability space. Let $\mathcal{A}$ be a finite set of events determined by these variables. If there exists an assignment of reals $x : \mathcal{A} \to (0,1)$ such that*

$$\forall A \in \mathcal{A} : \Pr(A) \le (1-\epsilon)x(A) \prod_{B \in \Gamma^2(A)} (1 - x(B)),$$

then the probability any bad event occurs after k resampling rounds of Algorithm 2 is at most $(1-\epsilon)^k \sum_{A\in\mathcal{A}} \frac{x(A)}{1-x(A)}$.

Note the difference with Theorem 3.1 is that the product is over all $B \in \Gamma^2(A)$ not $B \in \Gamma(A)$.

COROLLARY 3.4. *(Symmetric LLL) Let $\mathcal{P}$ be a finite set of mutually independent random variables in a probability space. Let $\mathcal{A}$ be a finite set of events determined by these variables, such that for $\forall A \in \mathcal{A}$*

1. *$Pr(A) \le p < 1$, and*
2. *A shares variables with at most d of the other events.*

If $epd^2 < 1$, then w.h.p. none of the bad events occur after $O(\log_{\frac{1}{epd^2}} n)$ rounds of Algorithm 2.

PROOF. Setting $x(A) = 1/d^2$ and $\epsilon = 1 - epd^2$ in Theorem 3.3, we have

$$(1-\epsilon)x(A) \prod_{B\in\Gamma^2(A)} (1-x(B)) \ge \frac{1-\epsilon}{d^2}(1-1/d^2)^{|\Gamma^2(A)|}$$
$$\ge \frac{1-\epsilon}{d^2}(1-1/d^2)^{(d^2-1)} \ge \frac{1-\epsilon}{ed^2} \ge p \ge \Pr(A).$$

Therefore, the probability a bad event occur after k rounds of resampling is at most $(1-\epsilon)^k \sum_{A\in\mathcal{A}} \frac{x(A)}{1-x(A)} = (1-\epsilon)^k n/(d^2-1)$, which is $1/\mathrm{poly}(n)$ if $k = O(\log_{\frac{1}{1-\epsilon}} n) = O(\log_{\frac{1}{epd^2}} n)$. □

Following Moser and Tardos [24], we analyze the following Galton-Watson process for generating an r-witness tree and prove the following lemma. See Appendix B for the proof.

LEMMA 3.5. *If for all $A \in \mathcal{A}$, we have $\Pr(A) \le (1-\epsilon)x(A) \cdot \prod_{B\in\Gamma^r(A)}(1-x(B))$, then the probability that any r-witness tree of size at least k occurs is at most $(1-\epsilon)^k \cdot \sum_{A\in\mathcal{A}} \frac{x(A)}{1-x(A)}$.*

Let C be the record of Algorithm 2 and S_j be the segment of the record corresponding to resamplings in round j. The following lemma relates the number of resampling rounds with the occurence of 2-witness trees.

LEMMA 3.6. *If there is still a violated event after k resampling rounds in Algorithm 2 then some 2-witness tree of size at least k occurs in C.*

PROOF. Let A_k be any event in S_k and t be its position in the record C. By Lemma 3.2 there exist events $A_{k-1}, \ldots, A_1$ in $S_{k-1}, \cdots, S_1$ such that for all $j < k$, $A_j \in \Gamma^{2+}(A_{j+1})$. This implies that $A_{k-1}, \ldots, A_1$ are mapped to distinct nodes in the 2-witness tree $\tau_C(t)$, whose root is labeled A_k. □

Therefore, by Lemma 3.6, if there is a violated event after k resampling rounds, then a 2-witness tree of size at least k occurs. However, by Lemma 3.5, it happens with probability at most $(1-\epsilon)^k \cdot \sum_{A\in\mathcal{A}} \frac{x(A)}{1-x(A)}$. Thus, Theorem 3.3 holds. Note that if $x(A)$ is bounded away from 1, then after $O(\log_{\frac{1}{1-\epsilon}} n)$ rounds, w.h.p. no bad event occurs.

3.2 Resampling by Weak MIS

In this section we analyze the efficiency of Moser and Tardos's Algorithm 1 when a new *weak MIS* procedure (Algorithm 3) is used in lieu of an actual MIS. The Weak-MIS procedure produces, in $O(\log^2 d)$ time, an independent set S such that the probability that a node is *not* in $\Gamma^+(S) = S \cup \Gamma(S)$ is $1/\mathrm{poly}(d)$. The procedure consists of $O(\log d)$ iterations where the probability that a vertex avoids $\Gamma^+(S)$ is constant per iteration. Each iteration consists of $\log d$ phases where, roughly speaking, the goal of phase i is to eliminate vertices with degree at least $d/2^i$ with constant probability. Each phase is essentially one step of Luby's MIS algorithm, though applied only to a judiciously chosen subset of the vertices. See Algoirthm 3.

Our main results are as follows.

THEOREM 3.7. *(Assymmetric LLL) Let $\mathcal{P}$ be a finite set of mutually independent random variables in a probability space. Let $\mathcal{A}$ be a finite set of events determined by these variables. If there exists an assignment of reals $x : \mathcal{A} \to (0,1)$ such that*

$$\forall A \in \mathcal{A} : \Pr(A) \le (1-\epsilon)x(A) \prod_{B\in\Gamma(A)} (1-x(B)),$$

then the probability any bad event occurs after k resampling rounds using the Weak-MIS algorithm is at most $n(\frac{1}{d+1})^k + (1-\epsilon)^{k/2} \sum_{A\in\mathcal{A}} \frac{x(A)}{1-x(A)}$.

Corollary 3.8. *(Symmetric LLL) Let $\mathcal{P}$ be a finite set of mutually independent random variables in a probability space. Let $\mathcal{A}$ be a finite set of events determined by these variables, such that for $\forall A \in \mathcal{A}$,*

1. *$Pr(A) \le p < 1$, and*

2. *A shares variables with at most d of the other events.*

If $ep(d+1) < 1$, then w.h.p. none of the bad events occur after $O(\max(\log_{d+1} n, \log_{\frac{1}{ep(d+1)}} n))$ Weak-MIS resampling rounds.

Corollary 3.8 follows directly by plugging in $x(A) = 1/(d+1)$ for all $A \in \mathcal{A}$ and $k = O(\max(\log_{d+1} n, \log_{\frac{1}{ep(d+1)}} n))$. Notice that if $\frac{1}{ep(d+1)} > d+1$, we can apply the faster simple distributed algorithm, so the running time in Corollary 3.8 will be dominated by $O(\log_{\frac{1}{ep(d+1)}} n \cdot \log^2 d)$.

```
$S \leftarrow \emptyset$
**for** iteration $1 \ldots, t = 4e^2 \ln(2e(d+1)^4)$ **do**
  $G' \leftarrow G_{\mathcal{F}} \setminus \Gamma^+(S)$
  **for** phase $i = 1 \ldots \lceil \log d \rceil$ **do**
    $V_i \leftarrow \{v \in G' \mid \deg_{G'}(v) \ge d/2^i\}$.
    For each vertex $v \in G'$, set $b(v) = 1$ with probability
      $p_i = 1/(\frac{d}{2^{i-1}} + 1)$ and 0 otherwise.
    For each vertex $v \in G'$, if $b(v) = 1$ and $b(w) = 0$ for all
      $w \in \Gamma_{G'}(v)$, then set $S \leftarrow S \cup \{v\}$.
    $G' \leftarrow G' \setminus (\Gamma^+(S) \cup V_i)$ (i.e., remove both $\Gamma^+(S)$ and $V_i$
      from $G'$.)
  **end for**
  Let $S'$ be the (isolated) vertices that remain in $G'$.
  Set $S \leftarrow S \cup S'$
**end for**
**return** $S$
```

Algorithm 3: Weak-MIS

Consider the first iteration of the Weak-MIS algorithm. For each phase i, G' is the subgraph of $G_{\mathcal{F}}$ containing vertices with degree at most $d/2^i$ and not adjacent to the independent set S. Let $V_i = \{v \in G' \mid \deg_{G'}(v) \ge d/2^i\}$. Note that every vertex in $G_{\mathcal{F}}$ must end up isolated in S' or one of the V_i's. Let (u, v) be an edge in G'. Following Peleg's analysis [26], define $\mathcal{E}(u, w)$ to be the event that at phase i, $b(u) = 0$ and $b(w) = 1$ and for all other neighbors x of u and w, $b(x) = 0$. Define $\mathcal{E}(u) = \bigcup_{w \in \Gamma_{G'}(u)} \mathcal{E}(u, w)$ to be the event that exactly one neighbor joins S in this phase. Since these events are disjoint, we have $\Pr(\mathcal{E}(u)) = \sum_{w \in \Gamma_{G'}(u)} \Pr(\mathcal{E}(u, w))$.

Lemma 3.9. *If $v \in V_i$, then $\Pr(\mathcal{E}(u)) \ge \frac{1}{4e^2}$.*

Proof. $\Pr(\mathcal{E}(u, w)) \ge p_i(1 - p_i)^{\deg_{G'}(u) + \deg_{G'}(w)} \ge p_i(1 - p_i)^{2d/2^{i-1}} \ge p_i e^{-2}$. Since $\deg_{G'}(u) \ge d/2^i$, $Pr(\mathcal{E}(u)) \ge \frac{d}{2^i} p_i e^{-2} \ge \frac{1}{4e^2}$ □

Therefore, if $v \in G_{\mathcal{F}} \setminus \Gamma^+(S)$ at the beginning of iteration l, the probability that $v \in \Gamma^+(S)$ at the end of iteration l is at least $1/(4e^2)$. We say a vertex in $G_{\mathcal{F}}$ *fails* if, after all $t = 4e^2 \ln(2e(d+1)^4)$ iterations, it is still not in $\Gamma^+(S)$.

Lemma 3.10. *Let S be an independent set selected by Weak-MIS. If $v \in \mathcal{F}$ then $\Pr(\Gamma^+(v) \cap S = \emptyset) \le \frac{1}{2e(d+1)^4}$.*

Proof. By Lemma 3.9, the probability that v survives iteration ℓ conditioned on it surviving iterations 1 through $\ell - 1$ is at most $1 - 1/(4e^2)$. Over $t = 4e^2 \ln(2e(d+1)^4$ iterations the probability of failure is at most $(1 - 1/(4e^2))^t \le e^{-\ln(2e(d+1)^4)} = \frac{1}{2e(d+1)^4}$. □

The next step is to relate the number of rounds of Weak-MIS resampling with the size of witness trees.

Lemma 3.11. *Suppose a bad event is violated after k rounds of Weak-MIS resampling and the maximum depth of the witness trees is t, then there exists a sequence of not necessarily distinct vertices $v_1, \ldots, v_k$ such that the following hold:*

(1) $v_i \in G_i$, where G_i is the violated subgraph $G_{\mathcal{F}}$ the beginning of round i.

(2) $v_{i+1} \in \Gamma^+(v_i)$ for $1 \le i \le k - 1$.

(3) For at least $k - t$ indices $1 < l \le k$, v_l failed in the call to Weak-MIS in round $l - 1$.

Proof. For $1 \le i \le k$, let S_i be the segment of the record C corresponding to events resampled at round i. Suppose that an event A is violated after k resampling rounds. Build a witness tree τ with root labeled A, adding nodes in the usual fashion, by scanning the record C in time-reversed order. For each j, in decreasing order, attach a node labelled $C(j)$ to the deepest node in τ whose label is in $\Gamma^+(C(j))$, if such a node in τ exists. Let $v_{k+1} = A$. We will build $v_k, v_{k-1}, \ldots, v_1$ in backward manner. For $k \ge i \ge 1$, we claim there is an event $v_i \in \Gamma^+(v_{i+1})$ such that either $v_i \in S_i$ or $v_i \in G_i$ and v_i failed at round i. If $v_{i+1} \notin G_i$ is not violated at the beginning of round i, then it must be the case that there exists an event $v_i \in \Gamma^+(v_{i+1})$ resampled at round i to cause $v_{i+1} \in G_{i+1}$. On the other hand, if $v_{i+1} \in G_i$ *is* violated at the beginning of round i, then either there exists $v_i \in \Gamma^+(v_{i+1})$ resampled at round i or v_{i+1} failed at round i. In the latter case, we let $v_i = v_{i+1}$. Notice that τ (excluding its artificial root labeled A) is a witness that occured and thus has depth at most t. Since in each of the k rounds, either the depth of our witness tree grows or a vertex is failed, at least $k - t$ vertices must have failed in their respective rounds. □

Notice that the total possible number of sequences satisfying (2) in Lemma 3.11 is at most $n(d+1)^{k-1}$. Given a sequence of vertices $P = (v_1, \ldots, v_k)$ satisfying (2), define $X_P^{(i)}$ to be 1 if $v_i \in G_i$ and v_i failed, 0 otherwise. Let $X_P = \sum_{i=1}^k X_P^{(i)}$. If a sequence satisfying (1–3) occured, then there exists P such that $X_P \ge k - t$. Since $X_P^{(1)}, \ldots, X_P^{(i-1)}$ are determined by $S_1, \ldots, S_{i-1}$ and $G_1, \ldots, G_{i-1}$, $\mathrm{E}(X_P^i \mid X_1, \ldots, X_{i-1}) = \mathrm{E}(X_P^i \mid S_1, \ldots, S_{i-1}, G_1, \ldots, G_{i-1}) \le q \overset{\text{def}}{=} \frac{1}{2e(d+1)^4}$ by Lemma 3.10. Fixing $t = k/2$, we have $k - t = k/2 = kq \cdot e(d+1)^4 \le \mathrm{E}[X_P] \cdot e(d+1)^4$. By Corollary A.4 (Conditional Chernoff Bound):

$$\Pr(X_P \ge k/2) \le \left(\frac{e^{e(d+1)^4 - 1}}{(e(d+1)^4)^{e(d+1)^4}} \right)^{\frac{k}{2e(d+1)^4}}$$
$$\le \left(\frac{1}{(d+1)^2} \right)^k .$$

By the union bound over all possible P satsfying (2), the probability that any such sequence in Lemma 3.11 occurs is at most

$$n\,(d+1)^{k-1} \cdot \left(\frac{1}{(d+1)^2}\right)^k \leq n \cdot \left(\frac{1}{d+1}\right)^k.$$

Moser and Tardos showed that the probability that any witness tree of size at least t occurs is at most $(1-\epsilon)^t \sum_{A\in\mathcal{A}} \frac{x(A)}{1-x(A)}$. Thus, either a witness tree of depth at least $t = k/2$ occurs or there exists a sequence of vertices (as in Lemma 3.11) such that $t-k = k/2$ of them failed. The probability either of these occurs is at most $n \cdot \left(\frac{1}{d+1}\right)^k + (1-\epsilon)^{k/2} \sum_{A\in\mathcal{A}} \frac{x(A)}{1-x(A)}$ by the union bound.

3.3 A Sublogarithmic Algorithm

We have seen a faster algorithm for LLL when the general condition $ep(d+1) < 1$ is replaced by a stronger condition $p \cdot f(d) < 1$, where $f(d)$ is a faster growing function than $e(d+1)$. The question of how fast we can do for a stronger condition arises. Does there exists a sublogarithmic algorithm for faster growing $f(d)$, independent of n? We answer this affirmatively for an exponential function of d.

Inspired by [3], our approach is a two-stage approach. In the first stage, we run Algorithm 2 for k rounds. Then we identify the *dangerous* events, who are likely to become violated if some subset of its neighbors are resampled. We will show there is a feasible solution by re-assigning the variables belonging to dangerous events. Moreover, we show the components induced by the dangerous events are likely to have *weak diameter* at most k. The weak diameter of a component is the maximum distance w.r.t. the original graph of any pair in the component. In the second stage, each component computes the answer independent of others in time proportional to its weak diameter.

Consider an event A. Let $P_1(A), P_2(A)$ be probabilities such that $P_1(A)P_2(A) = 2^d \cdot \Pr(A)$. Given an assignment of the random variables, we say A is dangerous w.r.t. the current assignment if resampling of some subset of neighbors causes A to become violated with probability more than $P_2(A)$. We will show that the probability for A to become dangerous is at most $P_1(A)$.

Given that $P_2(A)$ is small enough for all $A \in \mathcal{A}$, we can find a feasible solution by resassiging the variables belonging to the dangerous vertices. Also, given that $P_1(A)$ is small enough, we will show that the weak diameter of each component after the first stage is at most k w.h.p. We explain the idea roughly. If we build a 2-witness tree rooted at a dangerous vertex after the first stage, the 2-witness tree has size at least k. If there exists a path consisting of dangerous vertices of length k after the first stage, we will show the union of the witness trees rooted at these vertices has size at least $\Omega(k \log k)$. Then, we will glue them together into a 3-witness tree. By choosing $k = \Theta(\log n / \log\log n)$, we would have a 3-witness tree with size $\Theta(\log n)$, which does not occur w.h.p. We defer the proof for the following theorem and corollary to the full version.

Theorem 3.12 (Asymmetric LLL). *Let* $\Pr(A) \leq P_2(A) \leq 1$ *and* $P_1(A) = 2^d \cdot \frac{\Pr(A)}{P_2(A)}$, *where d is the maximum degree of the dependency graph. If there exists an assgiments of reals* $x_1, x_2 : \mathcal{A} \to (0, 0.99]$ *such that for all* $A \in \mathcal{A}$

1. $P_1(A) \leq (1-\epsilon)x_1(A)\prod_{B\in\Gamma^3(A)}(1-x_1(B))$
2. $P_2(A) \leq x_2(A)\prod_{B\in\Gamma(A)}(1-x_2(B))$

then the LLL problem can be solved in $O\left(\log_{1/(1-\epsilon)} n / \log\log_{1/(1-\epsilon)} n\right)$ *rounds.*

Corollary 3.13 (Symmetric LLL). *Suppose that for all* $A \in \mathcal{A}$, $\Pr(A) \leq p$ *and A is dependent with at most d other events in $\mathcal{A}$. Let* $z = 4ep2^dd^4$. *If* $z < 1$, *then a satisfying assignment can be found in* $O(\log_{1/z} n / \log\log_{1/z} n)$ *rounds.*

3.4 Lower Bound

Linial [17] proved that in an n-vertex ring, any distributed $(\log^{(k)} n)$-coloring algorithm requires $\Omega(k)$ rounds of communication, even if randomization is used. In particular, $O(1)$-coloring a ring requires $\Omega(\log^* n)$ time. We prove that Linial's lower bound implies that even weak versions of the Lovász Local Lemma cannot be computed in constant time.

Theorem 3.14. *Let $\mathcal{P}$, $\mathcal{A}$, and $G_{\mathcal{A}}$ be defined as usual. Let d be the maximum degree of any vertex in $G_{\mathcal{A}}$,* $p = \max_{A\in\mathcal{A}} \Pr(A)$ *be the maximum probability of any bad event, and* $f : \mathbb{N} \to \mathbb{N}$ *be an arbitrarily quickly growing function, where* $f(d) \geq e(d+1)$. *If* $p \cdot f(d) < 1$ *then* $\Pr(\bigcap_{A\in\mathcal{A}} \overline{A}) > 0$. *However,* $\Omega(\log^* |\mathcal{A}|)$ *rounds of communication are required for the vertices of $G_{\mathcal{A}}$ to agree on a point in* $\bigcap_{A\in\mathcal{A}} \overline{A}$.

The purpose of the function f is to show that our lower bound is insensitive to significant weakening of the standard criterion "$ep(d+1) < 1$." We could just as easily substitute $e^{e^d}p < 1$ or any similar criterion, for example.

Proof. Consider the following coloring procedure. Each vertex in an n-vertex ring selects a color from $\{1, \ldots, c\}$ uniformly at random. An edge is *bad* if it is monochromatic, an event that holds with probability $p = 1/c$. Let $\mathcal{A}$ be the dependency graph for these events having maximum degree $d = 2$ and choose c to be (the constant) $f(2)+1$, for any quickly growing function f. It follows from the LLL that a good c-coloring exists since $p \cdot f(2) < 1$. However, by [17], the vertices of $G_{\mathcal{A}}$ require $\Omega(\log^* n - \log^* c) = \Omega(\log^* n)$ time to find a good c-coloring. $\square$

It is also possible to obtain *conditional* lower bounds on distributed versions of the LLL. For example, the best known randomized $O(\Delta)$-coloring algorithm takes $\exp(O(\sqrt{\log\log n}))$ time [6], though better bounds are possible if $\Delta \gg \log n$ [30]. If LLL could be solved in less than $\exp(O(\sqrt{\log\log n}))$ time then we could improve on [6], as follows. Each vertex in G selects a color from a palette of size $c \geq 2e\Delta$ uniformly at random. As usual, an edge is bad if it is monochromatic. The dependency graph of these bad events corresponds to the line graph of G, which has maximum degree $d = 2\Delta - 2$. Since $e(1/c)(d+1) < 1$, a valid coloring can be found with one invocation of an LLL algorithm.

4. APPLICATIONS

The Lovász Local Lemma has applications in many coloring problems, such as list coloring, frugal coloring, total coloring and coloring triangle-free graphs [20]. We give a few examples of constructing these colorings distributively. In these applications, the existential bounds are usually achieved by the so called "Rödl Nibble" method or the semi-random method. The method consists of one or more iterations. Each iteration is a random process and some local properties are maintained in the graph. The properties depend on the randomness within a constant radius. Each property is associated with a bad event, which is the event that the property fails to hold. The Lovász Local Lemma can then be used to show the probability none of the bad events hold is positive, though it may be exponentially small in the size of the graph. This probability can then be amplified in a distributed fashion using a Moser-Tardos-type resampling algorithm. Notice that we will need to find an independent set (e.g., an MIS or Weak-MIS or set of events with locally minimal IDs) *in the dependency graph* induced by the violated local properties. Since we assumed the LOCAL model, the violated local properties can be identified in constant time and the algorithms for MIS/Weak-MIS can be simulated with a constant factor overhead, where each property is taken care by one of the processors nearby (within constant distance). The important point here is that the dependency graph and the underlying distributed network are sufficiently similar so that distributed algorithms on one topology can be simulated on the other with $O(1)$ slowdown.

Most applications of the LLL demand $epd^2 < 1$ or even weaker bounds. In this case, the efficient simple distributed algorithm can be applied. (The local properties are often that some quantities do not deviate too much from their expectations. Thus, the the failure probability of each local property is often bounded via standard Chernoff-type concentration inequalities.)

In the following, we present the distributed defective coloring and the frugal coloring algorithms. The other applications mentioned in the introduction will be deferred to the full version due to the page limit.

4.1 Distributed Defective Coloring

We begin with a simple single-iteration application that uses the local lemma. Given a k-coloring $\phi : V \to \{1, 2, \dots, k\}$. Define $\text{def}_\phi(v)$ to be the number of neighbors $w \in N(v)$ such that $\phi(v) = \phi(w)$. ϕ is said to be f-defective if $\max_v \text{def}_\phi(v) \le f$. Barenboim and Elkin ([4], Open Problem 10.7) raised the problem of devising an efficient distributed algorithm for computing an f-defective $O(\Delta/f)$-coloring. Here we extend one of their procedures for obtaining a $O(\log n)$-defective $O(\Delta/\log n)$-coloring to obtaining an f-defective $O(\Delta/f)$-coloring in $O(\log n/f)$ time w.h.p., for $f \ge 7 + 28 \ln \Delta$. Suppose each vertex colors itself with a color selected from $\{1, 2, \dots, \lceil 2\Delta/f \rceil\}$ uniformly at random. For every $v \in N(u)$, let X_v be 1 if v is colored the same as u, 0 otherwise. Let $X = \sum_{v \in \Gamma(u)} X_v$ denote the number of neighbors colored the same as v. Let A_u denote the bad event that $X > f$ at u. Clearly, whether A_u occurs is locally checkable by u in a constant number of rounds. Moreover, the event A_u only depends on the the random choices of u's neighbors. If A_u occured and is selected for resampling, the colors chosen by A_u and its neighbors will be resampled. The dependency graph $G_{\mathcal{A}}$ has maximum degree $d = \Delta^2$, because two events share variables only if they are within distance two. Now we will calculate the probability that A_u occurs. If we expose the choice of u first, then $\Pr(X_v = 1) \le f/(2\Delta)$ and it is independent among other $v \in N(u)$. Letting $M = f/2$, we have $E[X] \le f/2 = M$. By Corollary A.3 (a useful form of the Chernoff Bound), $\Pr(X > f) \le e^{-f/6}$. Let A_u denote the bad event that $X > f$ at u. Therefore, $epd^2 \le e^{-(f/6 - 1 - 4\ln\Delta)} \le e^{-(f/42)}$, since $f \ge 7 + 28 \ln \Delta$. By using the simple distributed algorithm, it takes $O(\log_{1/epd^2} n) = O(\log n/f)$ rounds to avoid the bad events w.h.p.

4.2 Distributed Frugal Coloring

A *β-frugal coloring* of a graph G is a proper vertex-coloring of G such that no color appears more than β times in any neighborhood. Molloy and Reed [20] showed the following by using an asymmetric version of the local lemma:

THEOREM 4.1. *For any constant integer $\beta \ge 1$, if G has maximum degree $\Delta \ge \beta^\beta$ then G has a β-frugal proper vertex coloring using at most $16\Delta^{1+\frac{1}{\beta}}$ colors.*

Here we outline their proof and show how to turn it into a distributed algorithm that finds such a coloring in $O(\log n \cdot \log^2 \Delta)$ rounds. If $\beta = 1$, then simply consider the square graph of G, which is obtained by adding the edges between vertices whose distance is 2. A proper coloring in the square graph is a 1-frugal coloring in G. Since the square graph has maximum degree Δ^2, it can be (Δ^2+1)-colored by simulating distributed algorithms for $(\Delta+1)$-coloring.

For $\beta \ge 2$, let $k = 16\Delta^{1+\frac{1}{\beta}}$. Suppose that each vertex colors itself with one of the k colors uniformly at random. Consider two types of bad events. For each edge uv, the Type I event $A_{u,v}$ denotes that u and v are colored the same. For each subset $\{u_1, \dots, u_{\beta+1}\}$ of the neighborhood of a vertex, Type II event $A_{u_1,\dots,u_{\beta+1}}$ denotes that $u_1, \dots, u_{\beta+1}$ are colored the same. If none of the events occur, then the random coloring is a β-frugal coloring. For each Type I event $A_{u,v}$, $\Pr(A_{u,v})$ is at most $1/k$. For each Type II event $A_{u_1,\dots,u_{\beta+1}}$, $\Pr(A_{u_1,\dots,u_{\beta+1}}) \le 1/k^\beta$. For each bad event A, let $x(A) = 2\Pr(A)$. Notice that $x(A) \le 1/2$, we have:

$$\begin{aligned}
x(A) \prod_{B \in \Gamma(A)} (1 - x(B)) &\ge x(A) \prod_{B \in \Gamma(A)} \exp\left(-x(B) \cdot 2\ln 2\right) \\
&\qquad \{(1-x) \ge e^{-x \cdot 2\ln 2} \text{ for } x \le 1/2\} \\
&= x(A) \exp\left(-2\ln 2 \cdot \sum_{B \in \Gamma(A)} 2\Pr(B)\right)
\end{aligned}$$

Since A shares variables with at most $(\beta+1)\Delta$ Type I events and $(\beta+1)\Delta\binom{\Delta}{\beta}$ Type II events,

$$\begin{aligned}
\sum_{B \in \Gamma(A)} \Pr(B) &\le (\beta+1)\Delta \cdot \frac{1}{k} + (\beta+1)\Delta\binom{\Delta}{\beta} \cdot \frac{1}{k^\beta} \\
&< \frac{(\beta+1)\Delta}{k} + \frac{(\beta+1)\Delta^{\beta+1}}{\beta! k^\beta} \\
&= \frac{\beta+1}{16\Delta^{\frac{1}{\beta}}} + \frac{\beta+1}{\beta!(16)^\beta} \\
&< 1/8 \qquad\qquad (\text{for } \Delta \ge \beta^\beta \text{ and } \beta \ge 2)
\end{aligned}$$

Therefore,

$$\begin{aligned}
x(A) \prod_{B \in \Gamma(A)} (1 - x(B)) &\ge x(A) \exp\left(-\frac{\ln 2}{2}\right) \\
&= \sqrt{2}\Pr(A).
\end{aligned}$$

By letting $1 - \epsilon = 1/\sqrt{2}$ in Theorem 3.7, we need at most $O(\log_{\sqrt{2}} n)$ rounds of weak MIS resampling. In each resampling round, we have to identify the bad events first.

Type I events $A_{u,v}$ can be identified by either u or v in constant rounds, where ties can be broken by letting the node with smaller ID check it. If $\{u_1, \dots, u_{\beta+1}\}$ is in the neighborhood of u, then the Type II event $A_{u_1,\dots,u_{\beta+1}}$ will be checked by u. If $\{u_1, \dots, u_{\beta+1}\}$ is in the neighborhood of multiple nodes, we can break ties by letting the one having the smallest ID to check it. All Type II events in the neighborhood of u can be identified from the colors selected by the neighbors of u. Next we will find a weak MIS induced by the bad events in the dependency graph. Each node will simulate the weak MIS algorithm on the events it is responsible to check. Each round of the weak MIS algorithm in the dependency graph can be simulated with constant rounds. The maximum degree d of the dependency graph is $O((\beta+1)\Delta\binom{\Delta}{\beta})$. Therefore, we need at most $O(\log n \cdot \log^2 d) = O(\log n \cdot \log^2 \Delta)$ rounds, since β is a constant and $(\beta+1)\Delta\binom{\Delta}{\beta} \le (\beta+1)\Delta^{\beta+1} = \mathrm{poly}(\Delta)$.

4.2.1 β-frugal, $(\Delta+1)$-coloring

The frugal $(\Delta+1)$-coloring problem for general graphs is studied by Hind et al. [13], Pemmaraju and Srinivasan [27], and Molloy and Reed [21]. In particular, the last one gave an upper bound of $O(\log\Delta/\log\log\Delta)$ on the frugality of $(\Delta+1)$-coloring. This is optimal up to a constant factor, because it matches lower bound of $\Omega(\log\Delta/\log\log\Delta)$ given by Hind et al. [21]. However, it is not obvious whether it can be implemented efficiently in a distributed fashion, because they used a structural decomposition computed by a sequential algorithm. Pemmaraju and Srinivasan [27] showed an existential upper bound of $O(\log^2\Delta/\log\log\Delta)$. Furthermore, they gave a distributed algorithm that computes an $O(\log\Delta\cdot\frac{\log n}{\log\log n})$-frugal, $(\Delta+1)$-coloring in $O(\log n)$ rounds. We show how to improve it to find a $O(\log^2\Delta/\log\log\Delta)$-frugal, $(\Delta+1)$-coloring also in $O(\log n)$ rounds. They proved the following theorem:

Theorem 4.2. *Let G be a graph with maximum vertex degree Δ. Suppose that associated with each vertex $v \in V$, there is a palette $P(v)$ of colors, where $|P(v)| \ge \deg(v)+1$. Furthermore, suppose $|P(v)| \ge \Delta/4$ for all vertices v in G. Then, for some subset $C \subseteq V$, there is a list coloring of the vertices in C such that:*

(a) $G[C]$ is properly colored.

(b) For every vertex $v \in V$ and for every color x, there are at most $9\cdot\frac{\ln\Delta}{\ln\ln\Delta}$ neighbors of v colored x.

(c) For every vertex $v \in V$, the number of neighbors of v not in C is at most $\Delta(1-\frac{1}{e^5}) + 27\sqrt{\Delta\ln\Delta}$.

(d) For every vertex $v \in V$, the number of neighbors of v in C is at most $\frac{\Delta}{e^5} + 27\sqrt{\Delta\ln\Delta}$.

They showed by iteratively applying the theorem for $O(\log\Delta)$ iterations, an $O(\log^2\Delta/\log\log\Delta)$-frugal, $(\Delta+1)$-coloring can be obtained. Let G_i be the graph after round i obtained by deleting already colored vertices and Δ_i be the maximum degree of G_i. The palette $P(u)$ for each vertex u contains colors that have not been used by its neighbors. It is always true that $|P(v)| \ge \deg(v)+1$. Notice that to apply Theorem 4.2, we also need the condition $|P(v)| \ge \Delta/4$. The worst case behavior of Δ_i and p_i is captured by the recurrences:

$$\Delta_{i+1} = \Delta_i(1-\frac{1}{e^5}) + 27\sqrt{\Delta_i\ln\Delta_i}$$
$$p_{i+1} = p_i - \frac{\Delta_i}{e^5} - 27\sqrt{\Delta_i\ln\Delta_i}. \quad (1)$$

They showed the above recurrence can be solved to obtain the following bounds on Δ_i and p_i:

Lemma 4.3. *Let $\alpha = (1-1/e^5)$. There is a constant C such that for all i for which $\Delta_i \ge C$, $\Delta_i \le 2\Delta_0\alpha^i$ and $p_i \ge \frac{\Delta_0}{2}\alpha^i$.*

Therefore, $|P(v)| \ge \Delta/4$ always holds. The two assumptions of Theorem 4.2 are always satisfied and so it can be applied iteratively until $\Delta_i < C$, which takes at most $\log_{1/\alpha}\left(\frac{2\Delta_0}{C}\right) = O(\log\Delta)$ iterations. Since each iteration introduces at most $O(\log\Delta/\log\log\Delta)$ neighbors of the same color to each vertex, the frugality will be at most $O(\log^2\Delta/\log\log\Delta)$. In the end, when $\Delta_i < C$, one can color the remaining graph in $O(\Delta_i + \log^* n)$ time using existing (Δ_i+1)-coloring algorithms [5]. This will only add $O(1)$ copies of each color to the neighborhood, yielding a $O(\log^2\Delta/\log\log\Delta)$-frugal, $(\Delta+1)$-coloring. In order to make it suitable for our simple distributed algorithm and achieve the running time of $O(\log n)$, we will relax the criteria of (b),(c),(d) in Theorem 4.2:

(b') For every vertex $v \in V$ and for every color x, there are at most $18\cdot\frac{\ln\Delta_0}{\ln\ln\Delta_0}$ neighbors of v colored x.

(c') For every vertex $v \in V$, the number of neighbors of v not in C is at most $\Delta(1-\frac{1}{e^5}) + 40\sqrt{\Delta}\ln\Delta$.

(d') For every vertex $v \in V$, the number of neighbors of v in C is at most $\frac{\Delta}{e^5} + 40\sqrt{\Delta}\ln\Delta$.

In (b'), Δ is replaced by Δ_0, which is the maximum degree of the initial graph. Also, the constant 9 is replaced by 18. In (c') and (d'), the constant 27 is replaced by 40 and $\sqrt{\ln\Delta}$ is replaced by $\ln\Delta$. It is not hard to see that Lemma 4.3 still holds and an $O(\log^2\Delta/\log\log\Delta)$-frugal coloring is still obtainable. Originally, by Chernoff Bound and Azuma's Inequality, they showed

$$\Pr\left(\#\text{ neighbors of } v \text{ colored } x \text{ exceeds } 9\cdot\frac{\ln\Delta}{\ln\ln\Delta}\right) < \frac{1}{\Delta^6} \quad (2)$$

and

$$\Pr\left(\left|P_v - \frac{\deg(v)}{e^5}\right| > 27\sqrt{\Delta\ln\Delta}\right) < \frac{2}{\Delta^{4.5}} \quad (3)$$

where P_v is the number of colored neighbors of v. Theorem 4.2 can be derived from (2) and (3). The relaxed version (b'), (c'), and (d') can be shown to fail with a lower probability.

$$\Pr\left(\#\text{ neighbors of } v \text{ colored } x \text{ exceeds } 18\cdot\frac{\ln\Delta_0}{\ln\ln\Delta_0}\right) < \frac{1}{\Delta_0^{12}} \quad (4)$$

and

$$\Pr\left(\left|P_v - \frac{\deg(v)}{e^5}\right| > 40\sqrt{\Delta}\ln\Delta\right) < \frac{2}{\Delta^{9\ln\Delta}} \quad (5)$$

The bad event A_v is when the neighbors of v colored x exceeds $18\cdot\frac{\ln\Delta_0}{\ln\ln\Delta_0}$ for some color x or $|P_v - \frac{\deg(v)}{e^5}| > 40\sqrt{\Delta}\ln\Delta$ happens. By (4), (5), and the union bound, $\Pr(A_v) \leq (\Delta+1)/\Delta_0^{12} + 2/\Delta^{9\ln\Delta}$. In their random process, they showed A_v depends on variables up to distance two. Thus, the dependency graph $G_{\mathcal{A}}$ has maximum degree d less than Δ^4. Note that $epd^2 = e\Delta^8((\Delta+1)/(2\Delta_0^{12}) + 2/\Delta^{9\ln\Delta}) \leq 1/(2\Delta_0) + 1/(2\Delta^{\ln\Delta}) < 2 \cdot \max(1/(2\Delta_0), 1/(2\Delta^{\ln\Delta})) = \max(1/\Delta_0, 1/\Delta^{\ln\Delta})$. The number of resampling rounds needed is at most $O(\log_{\frac{1}{epd^2}} n)$, which is at most $\frac{\ln n}{\min(\ln\Delta_0, \ln^2\Delta)} \leq \frac{\ln n}{\ln\Delta_0} + \frac{\ln n}{\ln^2\Delta}$. Therefore, the total number of rounds needed is at most:

$$\begin{aligned}
&\sum_{i=1}^{c\ln\Delta_0}\left(\frac{\ln n}{\ln\Delta_0} + \frac{\ln n}{\ln^2\Delta_i}\right) \\
&\leq \sum_{i=1}^{c\ln\Delta_0}\left(\frac{\ln n}{\ln\Delta_0} + \frac{\ln n}{\ln^2(2\Delta_0\alpha^i)}\right) \\
&= c\ln\Delta_0 \cdot \frac{\ln n}{\ln\Delta_0} + \ln n \sum_{i=1}^{c\ln\Delta_0} \frac{1}{(\ln\Delta_0 - i\ln\frac{1}{\alpha} + \ln 2)^2} \\
&\leq c\ln n + \ln n \cdot O\left(\sum_{i=1}^{\infty}\frac{1}{i^2}\right) = O(\log n)
\end{aligned}$$

where $c > 0$ is some constant, and $\alpha = (1 - 1/e^5)$.

5. REFERENCES

[1] N. Alon. A parallel algorithmic version of the local lemma. *Random Structures & Algorithms*, 2(4):367–378, 1991.

[2] N. Alon, L. Babai, and A. Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of Algorithms*, 7(4):567 – 583, 1986.

[3] N. Alon, M. Krivelevich, and B. Sudakov. Coloring graphs with sparse neighborhoods. *Journal of Combinatorial Theory, Series B*, 77(1):73 – 82, 1999.

[4] L. Barenboim and M. Elkin. *Distributed Graph Coloring: Fundamentals and Recent Developments.* Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2013.

[5] L. Barenboim, M. Elkin, and F. Kuhn. Distributed $(\Delta+1)$-coloring in linear (in Δ) time. *SIAM Journal on Computing*, 43(1):72–95, 2014.

[6] L. Barenboim, M. Elkin, S. Pettie, and J. Schneider. The locality of distributed symmetry breaking. In *Proc. IEEE 53rd Symposium on Foundations of Computer Science (FOCS)*, pages 321 – 330, oct. 2012.

[7] J. Beck. An algorithmic approach to the Lovász local lemma. I. *Random Structures & Algorithms*, 2(4):343–365, 1991.

[8] K. Chandrasekaran, N. Goyal, and B. Haeupler. Deterministic algorithms for the Lovász local lemma. *SIAM Journal on Computing*, 42(6):2132–2155, 2013.

[9] A. Czumaj and C. Scheideler. A new algorithm approach to the general Lovász local lemma with applications to scheduling and satisfiability problems (extended abstract). In *Proc. 32nd ACM Symposium on Theory of Computing (STOC)*, pages 38–47, 2000.

[10] D. Dubhashi, D. A. Grable, and A. Panconesi. Near-optimal, distributed edge colouring via the nibble method. *Theor. Comput. Sci.*, 203(2):225–251, August 1998.

[11] P. Erdős and L. Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. In A. Hanjal, R. Rado, and V. T. Sós, editors, *Infinite and Finite Sets*, volume 11, pages 609–627. North-Holland, 1975.

[12] B. Haeupler, B. Saha, and A. Srinivasan. New constructive aspects of the Lovász local lemma. *J. ACM*, 58(6):28, 2011.

[13] H. Hind, M. Molloy, and B. Reed. Colouring a graph frugally. *Combinatorica*, 17(4):469–482, 1997.

[14] K. Kolipaka and M. Szegedy. Moser and Tardos meet Lovász. In *Proceedings 43rd ACM Symposium on Theory of Computing (STOC)*, pages 235–244, 2011.

[15] F. Kuhn, T. Moscibroda, and R. Wattenhofer. Local computation: Lower and upper bounds. *CoRR*, abs/1011.5470, 2010.

[16] F. Kuhn and R. Wattenhofer. On the complexity of distributed graph coloring. In *Proc. 25th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 7–15, 2006.

[17] N. Linial. Locality in distributed graph algorithms. *SIAM J. Comput.*, 21(1):193–201, February 1992.

[18] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM Journal on Computing*, 15(4):1036–1053, 1986.

[19] M. Molloy and B. Reed. Further algorithmic aspects of the local lemma. In *Proc. 30th ACM Symposium on Theory of Computing (STOC)*, pages 524–529, 1998.

[20] M. Molloy and B. Reed. *Graph Colouring and the Probabilistic Method.* Algorithms and Combinatorics. Springer, 2001.

[21] M. Molloy and B. Reed. Asymptotically optimal frugal colouring. *J. Comb. Theory Ser. B*, 100(2):226–246, March 2010.

[22] R. A. Moser. Derandomizing the Lovász local lemma more effectively. *CoRR*, abs/0807.2120, 2008.

[23] R. A. Moser. A constructive proof of the Lovász local lemma. In *Proc. 41st ACM symposium on Theory of computing (STOC)*, pages 343–350, 2009.

[24] R. A. Moser and G. Tardos. A constructive proof of the general Lovász local lemma. *J. ACM*, 57(2):11, 2010.

[25] W. Pegden. An extension of the Moser-Tardos algorithmic local lemma. *CoRR*, abs/1102.2853, 2011.

[26] D. Peleg. *Distributed Computing: A Locality-Sensitive Approach.* Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, 2000.

[27] S. Pemmaraju and A. Srinivasan. The randomized coloring procedure with symmetry-breaking. In *Proc. 35th Int'l Colloq. on Automata, Languages, and Programming (ICALP)*, volume 5125 of *LNCS*, pages 306–319. 2008.

[28] S. Pettie and H.-H. Su. Fast distributed coloring algorithms for triangle-free graphs. In *Proc. 40th Int'l Colloq. on Automata, Languages, and Programming (ICALP)*, volume 7966 of *LNCS*, pages 687–699, 2013.

[29] B. Reed and B. Sudakov. Asymptotically the list colouring constants are 1. *Journal of Combinatorial Theory, Series B*, 86(1):27 – 37, 2002.

[30] J. Schneider and R. Wattenhofer. A new technique for distributed symmetry breaking. In *Proc. 29th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 257–266, 2010.

[31] J. Spencer. Asymptotic lower bounds for Ramsey functions. *Discrete Mathematics*, 20:69–76, 1977.

[32] A. Srinivasan. Improved algorithmic versions of the Lovász local lemma. In *Proc. 19th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 611–620, 2008.

APPENDIX

A. TOOLS

The following lemma shows when conditioning on a very likely event B, the probability of an event can only be affected by a small amount.

Lemma A.1. *Let A, B be two events, $|\Pr(A) - \Pr(A|B)| \leq \Pr(\overline{B})$.*

Proof. $\Pr(A) = \Pr(B)\Pr(A|B) + \Pr(\overline{B})\Pr(A|\overline{B}) = \Pr(A|B) + \Pr(\overline{B})(\Pr(A|\overline{B}) - \Pr(A|B))$. Therefore, $|\Pr(A) - \Pr(A|B)| \leq \Pr(\overline{B})$. $\square$

Lemma A.2. (Chernoff Bound) *Let $X_1, \ldots, X_n$ be independent trials such that $\Pr(X_i) = p$. Let $X = \sum_{i=1}^n X_i$. Then, for $\delta > 0$:*

$$\Pr(X > (1+\delta)\,\mathrm{E}[X]) \leq \left[\frac{e^\delta}{(1+\delta)^{(1+\delta)}}\right]^{\mathrm{E}[X]}$$

$$\Pr(X < (1-\delta)\,\mathrm{E}[X]) \leq \left[\frac{e^\delta}{(1-\delta)^{(1-\delta)}}\right]^{\mathrm{E}[X]}$$

The two bounds above imply that for $0 < \delta < 1$, we have:

$$\Pr(X > (1+\delta)\,\mathrm{E}[X]) \leq e^{-\delta^2\,\mathrm{E}[X]/3}$$

$$\Pr(X < (1-\delta)\,\mathrm{E}[X]) \leq e^{-\delta^2\,\mathrm{E}[X]/2}.$$

The proof for the following corollaries will be included in the full version.

Corollary A.3. *Let $X_1, \ldots, X_n$ be independent trials such that $\Pr(X_i) = p_i$. Let $X = \sum_{i=1}^n X_i$. If $M \geq \mathrm{E}[X]$ and $0 < \delta \leq 1$, then*

$$\Pr(X > \mathrm{E}[X] + \delta M) \leq e^{-\delta^2 M/3}$$

Corollary A.4. *Let $X_1, \ldots, X_n$ be trials such that for each $1 \leq i \leq n$,*

$$\Pr(X_i \mid X_1, \ldots, X_{i-1}) \leq p$$

Then the upper tail of $X = \sum_{i=1}^n X_i$ can be bounded by the upper tail Chernoff estimate for an independent set of variables $X'_1, X'_2, \ldots, X'_n$ with $\mathrm{E}[X'_i] = p$. In particular, for $\delta > 0$:

$$\Pr(X > (1+\delta)np) \leq \left[\frac{e^\delta}{(1+\delta)^{(1+\delta)}}\right]^{np}$$

B. PROOF OF LEMMA 3.5

Lemma 3.5. *If for all $A \in \mathcal{A}$, we have $\Pr(A) \leq (1-\epsilon)x(A) \cdot \prod_{B \in \Gamma^r(A)}(1 - x(B))$, then the probability that any r-witness tree of size at least k occurs is at most $(1-\epsilon)^k \cdot \sum_{A \in \mathcal{A}} \frac{x(A)}{1-x(A)}$.*

Following Moser and Tardos [24] we analyze the following Galton-Watson process for generating a r-witness tree T. Fix an event $A \in \mathcal{A}$. Begin by creating a root for T labelled A. To shorten the notation, we let $[v] := \sigma_T(v)$. In each subsequent step, consider each vertex v created in the previous step. For each $B \in \Gamma^{r+}([v])$, independently, attach a child labelled B with probability $x(B)$ or skip it with probability $1 - x(B)$. Continue the process until no new vertices are born. We prove a lemma analogous to one in [24].

Lemma B.1. *Let τ be a fixed proper r-witness tree with its root vertex labelled A. The probability p_τ that the Galton-Watson process yields exactly the tree τ is*

$$p_\tau = \frac{1 - x(A)}{x(A)} \prod_{v \in V(\tau)} x'([v])$$

where $x'(B) = x(B) \cdot \Pi_{C \in \Gamma^r(B)}(1 - x(C))$.

Proof. Let $W_v \subseteq \Gamma^{r+}([v])$ denote the set of inclusive r-neighbors of $[v]$ that do not occur as a label of some child node of v. Then,

$$\begin{aligned}
p_\tau &= \frac{1}{x(A)} \cdot \prod_{v \in V(\tau)} \left(x([v]) \cdot \prod_{u \in W_v} (1 - x([u]) \right) \\
&= \frac{1 - x(A)}{x(A)} \cdot \prod_{v \in V(\tau)} \left(\frac{x([v])}{1 - x([v])} \cdot \prod_{u \in \Gamma^{r+}([v])} (1 - x([u])) \right) \\
&= \frac{1 - x(A)}{x(A)} \cdot \prod_{v \in V(\tau)} \left(x([v]) \cdot \prod_{u \in \Gamma^{r}([v])} (1 - x([u])) \right) \\
&= \frac{1 - x(A)}{x(A)} \cdot \prod_{v \in V(\tau)} x'([v])
\end{aligned}$$

$\square$

Let $\mathcal{T}^r_A(k)$ denote the infinite set of r-witness trees having root labelled A and containing at least k vertices. By Lemma 3.6 and the union bound, the probability there exists a violated event after k resampling rounds is at most

$$\begin{aligned}
&\sum_{A \in \mathcal{A}} \sum_{\tau \in \mathcal{T}^r_A(k)} \Pr(\tau \ r\text{-occurs in } C) \\
&\leq \sum_{A \in \mathcal{A}} \sum_{\tau \in \mathcal{T}^r_A(k)} \prod_{v \in V(\tau)} \Pr([v]) && \text{by Lemma 2.2} \\
&\leq \sum_{A \in \mathcal{A}} \sum_{\tau \in \mathcal{T}^r_A(k)} \prod_{v \in V(\tau)} (1-\epsilon)x'([v]) && \text{cond. of Thm 3.3} \\
&\leq (1-\epsilon)^k \sum_{A \in \mathcal{A}} \frac{x(A)}{1 - x(A)} \sum_{\tau \in \mathcal{T}^r_A(k)} p_\tau && \text{by Lemma B.1} \\
&\leq (1-\epsilon)^k \sum_{A \in \mathcal{A}} \frac{x(A)}{1 - x(A)}
\end{aligned}$$

The last inequality follows since the Galton-Watson process grows exactly one tree.