2015 IEEE 56th Annual Symposium on Foundations of Computer Science

An Algorithmic Proof of the Lovász Local Lemma via Resampling Oracles

Nicholas J. A. Harvey
University of British Columbia
Vancouver, B.C., Canada
nickhar@cs.ubc.ca

Jan Vondrák
IBM Almaden Research Center
San Jose, CA, USA
jvondrak@gmail.com

Abstract

The Lovász Local Lemma is a seminal result in probabilistic combinatorics. It gives a sufficient condition on a probability space and a collection of events for the existence of an outcome that simultaneously avoids all of those events. Finding such an outcome by an efficient algorithm has been an active research topic for decades. Breakthrough work of Moser and Tardos (2009) presented an efficient algorithm for a general setting primarily characterized by a product structure on the probability space.

In this work we present an efficient algorithm for a much more general setting. Our main assumption is that there exist certain functions, called *resampling oracles*, that can be invoked to address the undesired occurrence of the events. We show that, in *all* scenarios to which the original Lovász Local Lemma applies, there exist resampling oracles, although they are not necessarily efficient. Nevertheless, for essentially all known applications of the Lovász Local Lemma and its generalizations, we have designed efficient resampling oracles. As applications of these techniques, we present new results for packings of Latin transversals, rainbow matchings and rainbow spanning trees.

Keywords

Lovász local lemma, algorithmic proof, general probability spaces, resampling oracles

I. Introduction

The Lovász Local Lemma (LLL) is a powerful tool with numerous uses in combinatorics and theoretical computer science. If a given probability space and collection of events satisfy a certain condition, then the LLL asserts the existence of an outcome that simultaneously avoids those events. The classical formulation of the LLL [13], [32] is as follows.

Let Ω be a discrete probability space with probability measure μ. Let $E_1, \ldots, E_n$ be certain "undesired" events in that space. Let G be an undirected graph with vertex set $[n] = \{1, \ldots, n\}$. The edges of G are denoted $E(G)$. Let $\Gamma(i) = \{\, j \neq i : \{i, j\} \in E(G) \,\}$ be the neighbors of vertex i. Also, let $\Gamma^+(i) = \Gamma(i) \cup \{i\}$ and let $\Gamma^+(I) = \bigcup_{i \in I} \Gamma^+(i)$ for $I \subseteq [n]$.

Theorem I.1 (General Lovász Local Lemma [13], [32])**.** *Suppose that the events satisfy the following condition that controls their dependencies*

$$\Pr_\mu[E_i \mid \cap_{j \in J} \overline{E_j}] \;=\; \Pr_\mu[E_i] \qquad \forall i \in [n],\, J \subseteq [n] \setminus \Gamma^+(i) \tag{Dep}$$

and the following criterion that controls their probabilities

$$\exists x_1, \ldots, x_n \in (0,1) \qquad \textit{such that} \qquad \Pr_\mu[E_i] \;\leq\; x_i \prod_{j \in \Gamma(i)} (1 - x_j) \quad \forall i \in [n]. \tag{GLL}$$

Then $\Pr_\mu[\bigcap_{i=1}^n \overline{E_i}] > 0$.

An equivalent statement of (Dep) is that the event $\mathcal{E}_i$ must be independent of the joint distribution on the events $\{\, \mathcal{E}_j : j \notin \Gamma^+(i) \,\}$. When (Dep) holds, G is called a *dependency graph*. The literature contains several dependency conditions generalizing (Dep) and criteria generalizing (GLL) under which the conclusion of the theorem remains true. We will discuss several such generalizations below.

The LLL can also be formulated [5] in terms of a directed dependency graph instead of an undirected graph, but nearly all applications of which we are aware involve an undirected graph. Accordingly, our work focuses primarily on the undirected case, but we will mention below which of our results extend to the directed case.

0272-5428/15 $31.00 © 2015 IEEE
DOI 10.1109/FOCS.2015.85

Algorithms: Algorithms to efficiently find an outcome in $\bigcap_{i=1}^{n} \overline{E_i}$ have been the subject of research for several decades. In 2008, a nearly optimal result was obtained by Moser [26] for a canonical application of the LLL, the bounded-degree k-SAT problem. Shortly thereafter, Moser and Tardos [27] extended that result to a general scenario called the "variable model" in which Ω consists of independent variables, each $\mathcal{E}_i$ depends on a subset of the variables, and events $\mathcal{E}_i$ and $\mathcal{E}_j$ are adjacent in G if there is a variable on which they both depend. Clearly the resulting graph is a dependency graph. The Moser-Tardos algorithm is extremely simple: after drawing an initial sample of the variables, it repeatedly checks if any undesired event occurs, then *resamples* any such event. Resampling an event means that the variables on which it depends receive fresh samples according to μ. Moser and Tardos prove that, if the (GLL) condition is satisfied, this algorithm will produced the desired outcome after at most $\sum_{i=1}^{n} \frac{x_i}{1-x_i}$ resampling operations.

Numerous extensions of the Moser-Tardos algorithm have been proposed. These extensions can handle more general criteria [21], [29], [1], [22], derandomization [11], exponentially many events [17], distributed scenarios [12], etc. However, all of these results are restricted to the Moser-Tardos variable model. Thus, these results cannot be viewed as algorithmic proofs of the LLL in full generality. There are many known scenarios for the LLL and its generalizations that fall outside the scope of the variable model [23], [24]. Section III discusses several such scenarios, including random permutations, matchings and spanning trees.

Recently two efficient LLL algorithms have been developed that go beyond the variable model. Harris and Srinivasan [18] extend the Moser-Tardos algorithm to a scenario involving random permutations; this scenario originates in work of Erdős and Spencer [14]. Achlioptas and Iliopoulos [2] define an abstract "flaw correction" framework that captures many known applications of the LLL outside the variable model, and some results beyond the LLL itself. Their framework eliminates the need for an underlying measure μ but on the other hand has other limitations; in particular the authors do not claim a formal connection with Theorem I.1. We discuss this further in Section I-E.

A. Our contributions

We present an algorithmic framework for the Lovász Local Lemma based on a new notion of *resampling oracles*. We then present an algorithm that uses resampling oracles to yield an algorithmic proof of the LLL. Our analysis yields a new proof of Theorem I.1, and several generalizations as described below.

1) Algorithmic assumptions: In order to provide algorithmic access to the probability space Ω, we will need to assume the existence and efficiency of certain subroutines that are use by our algorithm. These assumptions will suffice to make our algorithmic proof efficient.

Before proceeding, let us introduce further notation. An atomic event ω in the probability space Ω will be called a *state*. We write $\omega \sim \mu$ to denote that a random state ω is distributed according to μ, and $\omega \sim \mu|_{E_i}$ to denote that the distribution is μ conditioned on E_i. The resampling oracles are defined with respect to a graph G on $[n]$ with neighborhood structure Γ. We emphasize that G is an arbitrary undirected graph, and it is not necessarily a dependency graph satisfying (Dep).

The three subroutines required by our algorithm are as follows.

- *Sampling from μ:* There is a subroutine that can provide an independent random state $\omega \sim \mu$.
- *Checking events:* For each $i \in [n]$, there is a subroutine that can determine whether $\omega \in \mathcal{E}_i$.
- *Resampling oracles:* For each $i \in [n]$, there is a randomized subroutine $r_i : \Omega \to \Omega$ with the following properties.
  - (R1) If E_i is an event and $\omega \sim \mu|_{E_i}$, then $r_i(\omega) \sim \mu$. (The oracle r_i removes conditioning on E_i.)
  - (R2) For any $j \notin \Gamma^+(i)$, if $\omega \notin E_j$ then also $r_i(\omega) \notin E_j$. (Resampling an event cannot cause new non-neighbor events to occur.)

  When these conditions are satisfied, we say that r_i is a resampling oracle for events $E_1, \dots, E_n$ and graph G.

If efficiency concerns are ignored, the first two subroutines trivially exist. It is not hard to see that resampling oracles also exist if (Dep) is satisfied. This will be proven in greater generality in Section II-A.

2) Main Result: Our main result is that, if the three subroutines described above have efficient implementations, then there is an efficient algorithmic proof of the LLL.

Theorem (Informal)**.** *Consider any probability space, any events $E_1, \dots, E_n$, and any undirected graph G on vertex set $[n]$. If* (GLL) *is satisfied and if the three subroutines described above exist and have efficient implementations, then our algorithm efficiently finds a state in $\bigcap_{i=1}^{n} \overline{E_i}$.*

This theorem does not assume that (Dep) holds, and it turns out that the existence of resampling oracles is a strictly weaker condition. Thus, our algorithm yields a new proof of Theorem I.1 under its original assumptions. However, in order for our algorithm to be efficient, we must additionally assume that the three subroutines are efficient.

B. Our algorithm: MaximalSetResample

A striking aspect of the work of Moser and Tardos [27] is the simplicity and flexibility of their algorithm — in each iteration, *any* event E_i that occurs can be resampled. We propose a different algorithm that is somewhat less flexible, but whose analysis seems to be simpler in our scenario. Roughly speaking, our algorithm executes the standard greedy algorithm for computing a maximal independent set, except that a vertex i is only added if E_i occurs in the current state, and those conditions change dynamically. This is repeated until no events occur. Pseudocode for this procedure is shown in Algorithm 1. Nearly identical algorithms have been proposed before, particularly parallel algorithms [27], [21] although our interest lies not in the parallel aspects but rather with handling resampling oracles as cleanly as possible.

Algorithm 1 MaximalSetResample uses resampling oracles to output a state $\omega \in \bigcap_{i=1}^{n} \overline{E_i}$. It requires the three subroutines described in Section I-A1: sampling $\omega \sim \mu$, checking if an event E_i occurs, and the resampling oracles r_i.

```
Initialize ω with a random state sampled from μ;
t := 0;
repeat
    t := t + 1;
    J_t := ∅
    while there is i ∉ Γ⁺(J_t) such that ω ∈ E_i do
        Pick the smallest such i;
        J_t := J_t ∪ {i};
        ω := r_i(ω);    ▷ Resample E_i
    end while
until J_t = ∅;
return ω.
```

Our algorithmic proof of the LLL amounts to showing that MaximalSetResample terminates, at which point $\omega \in \bigcap_{i=1}^{n} \overline{E_i}$ clearly holds. That is shown by the following theorem.

Theorem I.2. *Suppose that the events $E_1, \ldots, E_n$ satisfy* (GLL) *and that the three subroutines described above in Section I-A1 are available. Then the expected number of calls to the resampling oracles by MaximalSetResample is*

$$O\left(\sum_{i=1}^{n} \frac{x_i}{1-x_i} \sum_{j=1}^{n} \log \frac{1}{1-x_j}\right).$$

Each execution of the outer loop in Algorithm 1 is called an *iteration*. As remarked above, the set J_t computed in the t^{th} iteration is an *independent set* in the graph, meaning that $J_t \cap \Gamma(i) = \emptyset$ for all $i \in J_t$.

C. Generalizing the dependency condition

Erdős and Spencer [14] showed that Theorem I.1 still holds when (Dep) is generalized to[1]

$$\Pr_{\mu}[E_i \mid \cap_{j \in J} \overline{E_j}] \;\leq\; \Pr_{\mu}[E_i] \qquad \forall i \in [n],\, J \subseteq [n] \setminus \Gamma^+(i). \tag{Lop}$$

They playfully called this the "lopsidependency" condition, and called G a "lopsidependency graph". This more general condition enables several interesting uses of the LLL in combinatorics and theoretical computer science, e.g., existence of Latin transversals [14] and optimal thresholds for satisfiability [15].

Recall that Theorem I.2 did not assume (Dep) and instead assumed the existence of resampling oracles. It is natural to wonder how the latter assumption relates to lopsidependency. We show that the existence of resampling oracles is equivalent to a condition that we call *lopsided association*, and whose strength lies strictly between (Dep) and (Lop). The lopsided association condition is

$$\Pr_{\mu}[E_i \cap F] \;\geq\; \Pr_{\mu}[E_i] \cdot \Pr_{\mu}[F] \qquad \forall i \in [n], \forall F \in \mathcal{F}_i \tag{LopA}$$

where $\mathcal{F}_i$ contains all events F whose indicator variable is a monotone non-decreasing function of the indicator variables of $(E_j : j \notin \Gamma^+(i))$. We call a graph satisfying (LopA) a *lopsided association graph* for events $E_1, \ldots, E_n$.

[1]More precisely, (Lop) should be restricted to J for which $\Pr_{\mu}[\cap_{j \in J} \overline{E_j}] > 0$. However that restriction is ultimately unnecessary because, in the context of the LLL, the theorem of Erdős and Spencer implies that $\Pr_{\mu}[\cap_{j \in [n]} \overline{E_j}] > 0$.

Theorem (Informal)**.** *Resampling oracles exist for events $E_1, \ldots, E_n$ and graph G if and only if G is a lopsided association graph for events $E_1, \ldots, E_n$.*

This equivalence follows essentially from LP duality: The existence of a resampling oracle can be formulated as a *transportation problem* for which the lopsided association condition is exactly the necessary and sufficient condition for a feasible transportation to exist. Section II proves this result in detail.

As remarked above, the dependency conditions are related by (Dep) $\Rightarrow$ (LopA) $\Rightarrow$ (Lop). The first implication is obvious since E_i is independent of F in (LopA). To see the second implication, simply take $F = \bigcup_{j \in J} E_j$ for any $J \subseteq [n] \setminus \Gamma^+(i)$ to obtain that $\Pr_\mu[E_i \mid \cup_{j \in J} E_j] \geq \Pr_\mu[E_i]$. Although lopsided association is formally a stronger assumption than lopsidependency, every use of the LLL with lopsidependency that we have studied actually satisfies the stronger lopsided association condition. We demonstrate this in Section III by designing efficient resampling oracles for those scenarios. Consequently, Theorem I.2 makes the LLL efficient in those scenarios.

The equivalence between (LopA) and resampling oracles comes with no efficiency guarantees. Nevertheless in all lopsidependency scenarios that we have considered, efficient implementations of the resampling oracles arise naturally from existing work, or can be devised with modest effort. In particular this is the case for random permutations, perfect matchings in complete graphs, and spanning trees in complete graphs, as discussed in Section III.

D. Generalizing the LLL criterion

In the early papers on the LLL [13], [32], the (GLL) criterion relating the dependency graph G and the probabilities $\Pr_\mu[E_i]$ was shown to be a sufficient condition to ensure that $\Pr_\mu[\bigcap_{i=1}^n \overline{E_i}] > 0$. Shearer [31] discovered a more general criterion that ensures the same conclusion. In fact, Shearer's criterion is the best possible: whenever his criterion is violated, there exist a corresponding measure μ and events $E_1, \ldots, E_n$ for which $\Pr_\mu[\bigcap_{i=1}^n \overline{E_i}] = 0$. In the full version of the paper [19], we state Shearer's criterion and prove the following result.

Theorem (Informal)**.** *Suppose that a graph G and the probabilities $\Pr_\mu[E_1], \ldots, \Pr_\mu[E_n]$ satisfy Shearer's criterion with ϵ slack, and that the three subroutines described in Section I-A1 exist. Then the expected number of calls to the resampling oracles by MaximalSetResample is bounded by a polynomial in $1/\epsilon$ and the logarithm of Shearer's coefficients.*

Unfortunately Shearer's criterion is unwieldy and has not seen much use in applications of the LLL. Recently several researchers have proposed criteria of intermediate strength between (GLL) and Shearer's criterion [7], [22]. The first of these, called the *cluster expansion* criterion, was originally devised by Bissacot et al. [7], and is based on insights from statistical physics. This criterion has given improved results in several applications of the local lemma [8], [18], [28]. Previous algorithmic work has also used the cluster expansion criterion in the variable model [1], [29] and for permutations [18].

The following theorem is another generalization of Theorem I.2, replacing (GLL) with the cluster expansion criterion, given below in (CLL). To state the result, we require additional notation: let Ind denote the family of independent sets in the graph G.

Theorem I.3. *Suppose that the events $E_1, \ldots, E_n$ satisfy the following criterion*

$$\exists y_1, \ldots, y_n > 0 \qquad \text{such that} \qquad \Pr_\mu[E_i] \;\leq\; \frac{y_i}{\sum_{J \subseteq \Gamma^+(i), J \in \mathsf{Ind}} \prod_{j \in J} y_j} \tag{CLL}$$

and that the three subroutines described in Section I-A1 exist. Then the expected number of calls to the resampling oracles by MaximalSetResample is $O\big(\sum_{i=1}^n y_i \sum_{j=1}^n \ln(1 + y_j)\big)$.

In Section V, we prove slightly weaker versions of Theorem I.2 and Theorem I.3, in which we assume that the (GLL) or (CLL) criteria hold with some slack. The analysis without slack is more involved, relying on analysis in Shearer's setting, and we defer it to the full version of the paper [19].

E. Techniques and related work

The breakthrough work of Moser and Tardos [27] stimulated a string of results on algorithms for the LLL. This section reviews the results that are most relevant to our work. Several interesting techniques play a role in the analyses of these previous algorithms. These can be roughly categorized as the *entropy method* [25], [2], *witness trees* or *witness sequences* [27], [18], [21] and *forward-looking combinatorial analysis* [16].

As mentioned above, Moser and Tardos [27] showed that a very simple algorithm will produce a state in $\bigcap_{i=1}^n \overline{E_i}$, assuming the variable model and that the (GLL) criterion holds. This paper is primarily responsible for the development of witness trees, and proved the "witness tree lemma", which yields an extremely elegant analysis in the variable model. The witness tree lemma has further implications. In particular, it allows them analyze separately for each event its expected number

of resamplings. Moser and Tardos also extended the variable model to incorporate a limited form of lopsidependency, and showed that their analysis still holds in that setting.

The main advantage of our result over the Moser-Tardos result is that we address the occurrence of an event through the abstract notion of resampling oracles rather than directly resampling the variables of the variable model. Furthermore we give efficient implementations of resampling oracles for essentially all known probability spaces to which the LLL has been applied. A significant difference with our work is that we do not have an analogue of the witness tree lemma; our approach provides a simpler analysis when the LLL criterion has slack but requires a more complicated analysis to remove the slack assumption. As a consequence, our bound on the number of resampling oracle calls is larger than the Moser-Tardos bound.

Earlier work of Moser [25] had developed the entropy method to analyze an algorithm for the "symmetric" LLL [13], which incorporates the maximum degree of G and a uniform bound on $\Pr_\mu[E_i]$. The entropy method roughly shows that, if the algorithm runs for a long time, a transcript of the algorithm's actions provides a compressed representation of the algorithm's random bits, which is unlikely due to entropy considerations.

The Moser-Tardos algorithm is known to terminate under criteria more general than (GLL), while still assuming the variable model. Pegden [29] showed that the cluster expansion criterion suffices, whereas Kolipaka and Szegedy [21] showed more generally that Shearer's condition suffices. We show that the cluster expansion criterion suffices in the more general context of resampling oracles. Our result is weaker than Pegden's only in that our bound on the number of resampling oracle calls is larger, due to our lack of a witness tree lemma. We also show that Shearer's criterion suffices in the context of resampling oracles. Our result is weaker than the Kolipaka-Szegedy result in that we require Shearer's condition to hold with a small slack, once again due to the lack of a witness tree lemma. In Shearer's context one may actually assume slack without loss of generality, since Shearer's criterion defines an open set. Nevertheless, quantitatively our result requires more resamplings than the Kolipaka-Szegedy result.

Kolipaka and Szegedy [21] present another algorithm, called GeneralizedResample, whose analysis proves the LLL under Shearer's condition for arbitrary probability spaces. GeneralizedResample is similar to MaximalSetResample in that they both work with abstract distributions and that they repeatedly choose a maximal independent set J of undesired events to resample. However, the way that the bad events are resampled is different: GeneralizedResample needs to sample from $\mu|_{\cap_{j \notin \Gamma^+(J)} \overline{E_j}}$, which is a complicated operation that seems difficult to implement efficiently. Thus MaximalSetResample can be viewed as a variant of GeneralizedResample that makes it efficient in all known scenarios.

Harris and Srinivasan [18] show that the Moser-Tardos algorithm can be adapted to handle certain events in a probability space involving random permutations. Their method for resampling an event is based on the Fischer-Yates shuffle. This scenario can also be handled by our framework; their resampling method perfectly satisfies the criteria of a resampling oracle. The Harris-Srinivasan's result is stronger than ours in that they do prove an analog of the witness tree lemma. Consequently their algorithm requires fewer resamplings than ours, and they are able to derive parallel variants of their algorithm. The work of Harris and Srinivasan is technically challenging, and generalizing it to a more abstract setting seems daunting.

Achlioptas and Iliopoulos [2], [3] proposed a general framework for finding "flawless objects", based on actions for addressing flaws. They show that, under certain conditions, a random walk over such actions rapidly converges to a flawless object. This naturally relates to the LLL by viewing each event E_i as a flaw. At the same time, their framework is not tied to the probabilistic formulation of the LLL and can derive results believed to be beyond the reach of the LLL such as the greedy algorithm for vertex coloring. The framework has other restrictions and does not currently claim to recover any particular form of the LLL. Nevertheless, their framework can accommodate applications of the LLL where lopsidependency plays a role, such as rainbow matchings and rainbow Hamilton cycles. In contrast, our framework can recover the LLL in full generality, even up to Shearer's criterion, but it does not recover results outside the scope of the LLL such as greedy vertex coloring. The analysis is inspired by Moser's entropy method, but in our view the approach is similar to forward-looking combinatorial analysis, which we discuss next.

Giotis et al. [16] show that a variant of Moser's algorithm gives an algorithmic proof in the variable model of the symmetric LLL. While this result is relatively limited when compared to the results above, their analysis is a clear example of forward-looking combinatorial analysis. Whereas Moser and Tardos use a *backward-looking* argument to find witness trees in the algorithm's "log", Giotis et al. analyze a *forward-looking* structure: the tree of resampled events and their dependences, looking forward in time. This viewpoint seems more natural and suitable for extensions.

Our approach can be roughly described as *forward-looking analysis* with a careful modification of the Moser-Tardos algorithm, formulated in the framework of resampling oracles. Our main conceptual contribution is the simple definition of the resampling oracles, and a small modification of the Moser-Tardos algorithm (described in Section I-B) so that the resamplings can be readily incorporated into the forward-looking analysis. The simplicity of analysis allows it accommodate various LLL criteria, including Shearer's criterion, which plays a fundamental role in the full proof of Theorem I.2. One

drawback of the forward-looking analysis is that it naturally leads to an exponential bound on the number of resamplings, unless there is some slack in the criterion that controls the probabilities; this same issue arises in [2], [16].

Our second contribution is an idea that eliminates the need for slack in (GLL) and (CLL), and seems interesting in its own right. We prove that, even if (GLL) or (CLL) are tight, we can instead perform our analysis using Shearer's criterion, which is never tight because it defines an open set. For example, consider the familiar case of Theorem I.1, and suppose that (GLL) holds with equality, i.e., $\Pr_\mu[E_i] = x_i \prod_{j\in\Gamma(i)}(1-x_j)$ for all i. We show that the conclusion of the LLL remains true even if each event E_i actually had the larger probability $\Pr_\mu[E_i]\cdot\big(1+(2\sum_i \frac{x_i}{1-x_i})^{-1}\big)$. The proof of this fact crucially uses Shearer's criterion and it does not seem to follow from more elementary tools [13], [32]. All proofs involving Shearer's criterion are omitted from this extended abstract, but can be found in the full version of the paper [19].

II. Resampling oracles and lopsided association

One of the main contributions of this paper is the notion of resampling oracles. These are randomized functions that address the occurrence of the undesired events E_i. A formal definition appears in Definition II.1 below.

Whereas most formulations of the LLL involve a dependency condition, such as (Dep) or (Lop), our results will assume neither of these conditions and instead assume the existence of resampling oracles. However, there is a close relationship between these assumptions: the existence of a resampling oracle for each event is equivalent to the condition (LopA), which is a strengthening of (Lop). Once again, we emphasize that the *efficiency of an implementation* of a resampling oracle is a separate issue. There is no general guarantee that resampling oracles can be implemented efficiently. Nevertheless, this is not an issue in practice. In all applications of the LLL of which we are aware (with or without lopsidependency), resampling oracles exist and can be implemented efficiently. Section III presents several efficient resampling oracles that we have developed.

A. Existence of resampling oracles

In this section we define the notion of lopsided association and prove that it is equivalent to the existence of resampling oracles. We begin with a formal definition of resampling oracles.

Definition II.1. *Let $E_1,\dots,E_n$ be events on a space Ω with a probability measure μ. Let $G=([n],E)$ be a graph with neighbors of $i\in[n]$ denoted by $\Gamma(i)$, and $\Gamma^+(i)=\Gamma(i)\cup\{i\}$. Let r_i be a randomized procedure that takes a state $\omega\in\Omega$ and outputs a state $r_i(\omega)\in\Omega$. We say that r_i is a resampling oracle for E_i with respect to G, if*

(R1) *For $\omega\sim\mu|_{E_i}$, we obtain $r_i(\omega)\sim\mu$. (The resampling oracle removes conditioning on E_i.)*

(R2) *For any $j\notin\Gamma^+(i)$, if $\omega\notin E_j$ then also $r_i(\omega)\notin E_j$. (Resampling E_i cannot cause new non-neighbor events.)*

Next, let us define the notion of a lopsided association graph. We denote by $E_i[\omega]$ the $\{0,1\}$-valued function indicating whether E_i occurs at a state $\omega\in\Omega$.

Definition II.2. *A graph G with neighborhood function Γ is a lopsided association graph for events $E_1,\dots,E_n$ if*

$$\Pr_\mu[E_i\cap F] \;\geq\; \Pr_\mu[E_i]\cdot\Pr_\mu[F] \qquad \forall i\in[n],\forall F\in\mathcal{F}_i \tag{LopA}$$

where $\mathcal{F}_i$ contains all events F such that $F[\omega]$ is a monotone non-decreasing function of $(\,E_j[\omega]\,:\,j\notin\Gamma^+(i)\,)$.

Lemma II.3. *Consider a fixed $i\in[n]$ and assume $\Pr_\mu[E_i]>0$. The following statements are equivalent:*

- *There exists a resampling oracle r_i for event E_i with respect to a graph G (ignoring issues of computational efficiency).*
- *$\Pr_\mu[E_i\cap F]\geq\Pr_\mu[E_i]\Pr_\mu[F]$ for any event F such that $F[\omega]$ is a non-decreasing function of $(\,E_j[\omega]\,:\,j\notin\Gamma^+(i)\,)$.*

Corollary II.4. *Resampling oracles $r_1,\dots,r_n$ exist for events $E_1,\dots,E_n$ with respect to a graph G if and only if G is a lopsided association graph for $E_1,\dots,E_n$. Both statements imply that the lopsidependency condition* (Lop) *holds.*

Proof: First, let us assume that the resampling oracle exists. Consider the coupled states (ω,ω') where $\omega\sim\mu|_{E_i}$ and $\omega'=r_i(\omega)$. By (R1), $\omega'\sim\mu$. By (R2), for any event F monotone in $\{E_j : j\notin\Gamma^+(i)\}$, if F does not occur at ω then it does not occur at ω' either. This establishes that

$$\Pr_\mu[F] \;=\; \mathbf{E}_{\omega'\sim\mu}[F[\omega']] \;\leq\; \mathbf{E}_{\omega\sim\mu|E_i}[F[\omega]] \;=\; \Pr_\mu[F\mid E_i]$$

which implies $\Pr_\mu[F\cap E_i]\geq\Pr_\mu[F]\Pr_\mu[E_i]$. In particular this implies (Lop), by taking $F=\bigcup_{j\in J}E_j$.

The reverse implication follows essentially from LP duality. We can formulate the existence of a resampling oracle as the following *transportation problem*. We have a bipartite graph $G=(U\cup W,E)$, where U and W are disjoint, U represents all the states $\omega\in\Omega$ satisfying E_i, and W represents all the states $\omega\in\Omega$. Edges represent the possible actions of the

resampling oracle: $(u, w) \in E$ if u satisfies every event among $\{ E_j : j \notin \Gamma^+(i) \}$ that w satisfies. Each vertex has an associated weight: For $w \in W$, we define $p_w = \Pr_\mu[w]$, and for $u \in U$, $p_u = \Pr_\mu[u] / \Pr_\mu[E_i]$ (i.e, p_u is the probability of u conditioned on E_i). We claim that the resampling oracle r_i exists if and only if there is an assignment of non-negative values on the edges $f_{uw} \geq 0$ (a feasible transportation) such that

- $\forall u \in U$, $\sum_{w:(u,w)\in E} f_{uw} = p_u$,
- $\forall w \in W$, $\sum_{u:(u,w)\in E} f_{uw} = p_w$.

This is true because for any such transportation, the resampling oracle is defined naturally by following each edge from $u \in U$ with probability f_{uw}/p_u, and the resulting distribution is p_w. Conversely, for a resampling oracle which, for a given state $u \in U$, generates $w \in W$ with probability q_{uw}, we define $f_{uw} = p_u q_{uw}$ and this satisfies the transportation conditions above.

A necessary and sufficient condition for the existence of such a transportation can be determined from LP duality (see, e.g., Theorem 21.11 in [30]): A feasible transportation exists if and only if $\sum_{u\in U} p_u = \sum_{w\in W} p_w$ and for every subset $A \subseteq U$ and its neighborhood $\Gamma(A) = \{ w \in W : \exists u \in A \text{ s.t. } (u, w) \in E \}$, we have $\sum_{w\in\Gamma(A)} p_w \geq \sum_{u\in A} p_u$. (This is an extension of Hall's condition for the existence of a perfect matching.)

Observe that if $u \in A$, then we might as well include in A all the vertices $u' \in U$ that satisfy no more events among $\{ E_j : j \in \Gamma^+(i) \}$ than those satisfied by u. This does not create any new neighbors in $\Gamma(A)$, because if the set of events among $\{E_j : j \in \Gamma^+(i)\}$ satisfied by u' is a subset of those satisfied by u, then $\Gamma(u') \subseteq \Gamma(u)$. This makes A a set of states corresponding to an event F' such that $F'[\omega]$ is a *non-increasing* function of $(E_j[\omega] : j \notin \Gamma^+(i))$. The neighborhood $\Gamma(A)$ consists of states satisfying at most those events among $\{E_j : j \notin \Gamma^+(i)\}$ satisfied by some state in A. Consequently, $\Gamma(A)$ corresponds to exactly the same event F'. As we argued, it is sufficient to satisfy the conditions for such pairs of sets $(A, \Gamma(A))$.

Suppose now that $\Pr_\mu[F \cap E_i] \geq \Pr_\mu[F] \Pr_\mu[E_i]$ for every event F monotone in $(E_j : j \notin \Gamma^+(i))$. This is equivalent to $\Pr_\mu[\overline{F} \cap E_i] = \Pr_\mu[E_i] - \Pr_\mu[F \cap E_i] \leq \Pr_\mu[E_i] - \Pr_\mu[F] \Pr_\mu[E_i] = \Pr_\mu[\overline{F}] \Pr_\mu[E_i]$. Assuming $\Pr[E_i] > 0$, we can rewrite this as $\Pr_\mu[\overline{F} \mid E_i] \leq \Pr_\mu[\overline{F}]$. We take F to be the event complementary to the event F' defined by the states in $\Gamma(A)$. Using the above connection, we have $\sum_{u\in A} p_u = \Pr_\mu[F' \mid E_i] \leq \sum_{w\in\Gamma(A)} p_w = \Pr_\mu[F']$. This verifies the condition for the existence of the transportation, and hence the existence of the resampling oracle. ∎

B. Example: monotone events on lattices

To illustrate the (LopA) condition, let us discuss an example in which Lemma II.3 implies the existence of a non-trivial resampling oracle, even though the lopsided association graph is empty. This example extends ideas of Lu et al. [23, Section 2.4].

The probability space is $\Omega = \{0, 1\}^M$. Let $\mu : \{0, 1\}^M \to [0, 1]$ be a probability distribution over $\{0, 1\}^M$ that is *log-supermodular*, meaning that

$$\mu(x \vee y)\mu(x \wedge y) \;\geq\; \mu(x)\mu(y) \qquad \forall x, y \in \{0, 1\}^M.$$

In particular, any product distribution is log-supermodular. Consider monotone increasing events E_i, i.e. such that $x' \geq x \in E_i \Rightarrow x' \in E_i$. (Here $x' \geq x$ is the partial ordering determined by the Boolean lattice, i.e., $x'_i \geq x_i$ for all $i \in M$.) Note that any monotone increasing function of such events is again monotone increasing. It follows from the FKG inequality [5] that the condition of Lemma II.3 is satisfied for such events with an *empty* lopsided association graph. Therefore, resampling oracles exists in this setting. However, the explicit description of its operation might be complicated and we do not know whether it can be implemented efficiently in general.

Alternatively, the existence of the resampling oracle can be proved directly, using the following theorem of Holley [20, Theorem 6].

Theorem II.5. *Let μ_1 and μ_2 be probability measures on $\{0, 1\}^M$ satisfying*

$$\mu_1(x \vee y)\mu_2(x \wedge y) \;\geq\; \mu_1(x)\mu_2(y) \qquad \forall x, y \in \{0, 1\}^M. \tag{1}$$

Then there exists a probability distribution $\nu : \{0, 1\}^M \times \{0, 1\}^M \to \mathbb{R}$ satisfying

$$\begin{aligned} \mu_1(x) &= \textstyle\sum_y \nu(x, y) \\ \mu_2(y) &= \textstyle\sum_x \nu(x, y) \\ \nu(x, y) &= 0 \;\; \textit{unless} \;\; x \geq y. \end{aligned}$$

The resampling oracle is described in Algorithm 2. The reader can verify that this satisfies the assumptions (R1) and (R2), using Holley's Theorem.

Algorithm 2 Resampling oracle for a monotone increasing event E. Let ν be the function guaranteed by Theorem II.5, when $\mu_1(x) = \frac{\mu(x)\mathbf{1}_{x\in E}}{\sum_{e\in E}\mu(e)}$ and $\mu_2(y) = \mu(y)$.

1: **Function** $r_E(x)$:
2: Verify that $x \in E$, otherwise **return** x.
3: Randomly select y with probability $\frac{\nu(x,y)}{\sum_{y'}\nu(x,y')}$.
4: **return** y.

III. Implementation of resampling in specific settings

In this section, we present efficient implementations of resampling oracles in four application settings: the variable model (which was the setting of Moser-Tardos [27]), random permutations (consider by Harris-Srinivasan [18]), perfect matchings in complete graphs (some of whose applications are made algorithmic by Achlioptas-Iliopoulos [2]), and spanning trees in complete graphs (which was not considered by previous algorithmic work).

To be more precise, resampling oracles also depend on the types of events and dependencies that we want to handle. In the setting of independent random variables, we can handle arbitrary events with dependencies defined by overlapping relevant variables, as Moser-Tardos did [27]. In the setting of permutations, we handle the appearance of patterns in permutations as Harris-Srinivasan did [18]. In the settings of matchings and spanning trees, we consider the "canonical events" defined by Lu et al. [23], characterized by the appearance of a certain subset of edges.

We also show in Section III-E how resampling oracles for a certain probability space can be extended in a natural way to products of such probability spaces; for example, how to go from resampling oracles for one random permutation to a collection of independent random permutations. These settings cover nearly all the applications of the lopsided LLL of which we are aware.

A. Independent random variables

This is the most common setting, considered originally by Moser and Tardos [27]. Here, Ω has a product structure corresponding to independent random variables $\{ X_a : a \in \mathcal{U} \}$. The probability measure μ here is a product measure. Each bad event E_i depends on a particular subset of variables A_i, and two events are independent iff $A_i \cap A_j = \emptyset$.

Here our algorithmic assumptions correspond exactly to the Moser-Tardos framework [27]. Sampling from μ means generating a fresh set of random variables independently. The resampling oracle r_i takes a state ω and replaces the random variables $\{ X_a : a \in A_i \}$ by fresh random samples. It is easy to see that the assumptions are satisfied: in particular, a random state sampled from μ conditioned on E_i has all variables outside of A_i independently random. Hence, resampling the variables of A_i produces the distribution μ. Clearly, resampling $\{ X_a : a \in A_i \}$ does not affect any events that depend on variable sets disjoint from A_i.

This resampling oracle is also consistent with the notion of lopsidependency on product spaces considered by Moser and Tardos. They call two events E_i, E_j lopsidependent if $A_i \cap A_j \neq \emptyset$ and it is possible to cause E_j to occur by resampling A_i in a state where E_i holds but E_j does not. (The definition in [27] is worded differently but equivalent to this). This is exactly condition (R2)in the definition of resampling oracles.

B. Random permutations

The probability space Ω here is the space of all permutations π on a set $[n]$, with a uniform measure μ. Bad events are assumed here to be "simple" in the following sense: Each bad event E_i is defined by a "pattern"

$$P(E_i) \;=\; \left\{(x_1, y_1), \ldots, (x_{t(i)}, y_{t(i)})\right\}.$$

The event E_i occurs if $\pi(x_j) = y_j$ for each $1 \leq j \leq t(i)$. Let $\mathrm{vbl}(E_i) = \{ x : \exists y \text{ s.t. } (x, y) \in P(E_i) \}$ denote the variables of π relevant to event E_j. Let us define a relation $i \sim i'$ to hold iff there are pairs $(x, y) \in P(E_i)$ and $(x', y') \in P(E_{i'})$ such that $x = x'$ or $y = y'$; i.e., the two events involve the same value in either the range or domain. This relation defines a lopsidependency graph. It is known that the lopsided LLL holds in this setting.

Harris and Srinivasan [18] showed that a permutation avoiding all bad events can be found algorithmically, assuming any of the usual LLL criteria: (GLL), (CLL), etc. We show that their resampling process, based on the Fischer-Yates shuffle, satisfies the conditions of an resampling oracle. Pseudocode is shown in Algorithm III-B. To prove the correctness of this resampling oracle within our framework, we need the following lemma.

Algorithm 3 Resampling oracle for permutations. The input is a permutation π and an event E_i. It is assumed that E_i occurs in permutation π.

Function $\mathrm{Shuffle}(\pi, E_i)$:
$X := \mathrm{vbl}(E_i)$, i.e., the variables in π affecting event E_i;
Fix an arbitrary order $X = (x_1, x_2, \ldots, x_t)$;
for $i = t$ down to 1 **do**
  Swap $\pi(x_i)$ with $\pi(z)$ for z chosen uniformly at random among $[n] \setminus \{x_1, \ldots, x_{i-1}\}$;
end for
return π;

Lemma III.1. *Let X be the variables upon which event E depends. Suppose that a permutation π has some arbitrary fixed assignment on the variables in X, $\pi|_X = \phi$, and it is uniformly random among all permutations satisfying $\pi|_X = \phi$. Then the output of $\mathrm{Shuffle}(\pi, E)$ is a uniformly random permutation.*

This resampling oracle is a partial version of the Fischer-Yates shuffle. In contrast to the full shuffle, we assume that some part of the permutation has been shuffled already: X is the remaining portion that still remains to be shuffled, and conditioned on its assignment the rest is uniformly random. This would be exactly the distribution achieved after performing the Fisher-Yates shuffle on the complement of X. The resampling oracle performs the rest of the Fisher-Yates shuffle, which produces a uniformly random permutation. For completeness we give a self-contained proof.

Proof: Let $X = \{x_1, \ldots, x_t\}$. By induction, after performing the swap for x_i, the permutation is uniform among all permutations with a fixed assignment of $\{x_1, \ldots, x_{i-1}\}$ (consistent with ϕ). This holds because, before the swap, the permutation was by induction uniform conditioned on the assignment of $\{x_1, \ldots, x_i\}$ being consistent with ϕ, and we choose a uniformly random swap for x_i among the available choices. This makes every permutation consistent with ϕ on $\{x_1, \ldots, x_{i-1}\}$ equally likely after this swap. ∎

This verifies condition (R1)for the function $\mathrm{Shuffle}$. The condition (R2)states that resampling an event cannot cause non-neighbor events to occur. This is true because of the following lemma.

Lemma III.2. *The resampling oracle $\mathrm{Shuffle}(\pi, E_i)$ applied to a permutation satisfying E_i does not cause any new event outside of $\Gamma^+(i)$ to occur.*

Proof: Suppose E_j changed its status during a call to $\mathrm{Shuffle}(\pi, E_i)$. This means that something changed among its relevant variables $\mathrm{vbl}(E_j)$. This could happen in two ways:

(1): either a variable $z \in \mathrm{vbl}(E_j)$ was swapped because $z \in X = \mathrm{vbl}(E_i)$; then clearly $j \in \Gamma^+(i)$.

(2): a variable in $\mathrm{vbl}(E_j)$, although outside of X, received a new value by a swap with some variable in $X = \mathrm{vbl}(E_i)$. Note that in the $\mathrm{Shuffle}$ procedure, every time a variable z outside of X changes its value, it is by a swap with a fresh variable of X, i.e., one that had not been processed before. Therefore, the value that z receives is one that previously caused E_i to occur. If it causes E_j to occur, it means that E_i and E_j share a value in the range space and we have $j \in \Gamma^+(i)$ as well. ∎

C. Perfect matchings

Here, the probability space Ω is the set of all perfect matchings in K_{2n}, with the uniform measure. This is a setting considered by Achlioptas-Iliopoulos [2], and it is also related to the setting of permutations. (Permutations on $[n]$ can be viewed as perfect matchings in $K_{n,n}$.) A state here is a perfect matching in K_{2n}, which we denote by $M \in \Omega$. We consider bad events of the following form: E_A for a set of edges A occurs if $A \subseteq M$. Obviously, $\Pr_\mu[E_A] > 0$ only if A is a (partial) matching. Let us define $A \sim B$ iff $A \cup B$ is *not* a matching. It was proved by Lu et al. [23] that this defines a lopsidependency graph.

Algorithm 4 describes an resampling oracle in this setting.

Lemma III.3. *Let A be a matching in K_{2n} and let M be distributed uniformly among perfect matchings in K_{2n} such that $A \subseteq M$. Then after calling the resampling oracle, $r_A(M)$ is a uniformly random perfect matching.*

Proof: We prove by induction that at any point, M' is a uniformly random perfect matching conditioned on containing A'. This is satisfied at the beginning: $M' = M, A' = A$ and M is uniformly random conditioned on $A \subseteq M$.

Assume this is true at some point, we pick $(u, v) \in A'$ arbitrarily and $(x, y) \in M' \setminus A'$ uniformly at random. Denote the vertices covered by $M' \setminus A'$ by $V(M' \setminus A')$. Observe that for a uniformly random perfect matching on $V(M' \setminus A') \cup \{u, v\}$,

Algorithm 4 Resampling oracle for perfect matchings

Function $r_A(M)$:
Check that $A \subseteq M$, otherwise **return** M.
$A' := A$;
$M' := M$;
while $A' \neq \emptyset$ **do**
  Pick $(u, v) \in A'$ arbitrarily;
  Pick $(x, y) \in M' \setminus A'$ uniformly at random, with (x, y) randomly ordered;
  With probability $1 - \frac{1}{2|M' \setminus A'|+1}$,
    Add $(u, y), (v, x)$ to M' and remove $(u, v), (x, y)$ from M';
  Remove (u, v) from A';
end while
return M'.

the edge (u, v) should appear with probability $1/(2|M' \setminus A'| + 1)$ since u has $2|M' \setminus A'| + 1$ choices to be matched with and v is 1 of them. Consequently, we keep the edge (u, v) with probability $1/(2|M' \setminus A'| + 1)$ and conditioned on this $M' \setminus A'$ is uniformly random by the inductive hypothesis. Conditioned on (u, v) not being part of the matching, we re-match (u, v) with another random edge $(x, y) \in M' \setminus A'$ where (x, y) is randomly ordered. In this case, u and v get matched to a uniformly random pair of vertices $x, y \in V(M' \setminus A')$, as they should be. The rest of the matching $M' \setminus A' \setminus \{(x, y)\}$ is uniformly random on $V(M' \setminus A' \setminus \{x, y\})$ by the inductive hypothesis.

Therefore, after each step $M' \setminus A'$ is uniformly random conditioned on containing A'. At the end, $A' = \emptyset$ and M' is uniformly random. ■

Lemma III.4. *The resampling oracle $r_A(M)$ applied to a perfect matching satisfying event E_A does not cause any new event E_B such that $B \notin \Gamma^+(A)$.*

Proof: Observe that all the new edges that the resampling oracle adds to M are incident to some vertex matched by A. So if an event E_B was not satisfied before the operation and it is satisfied afterwards, it must be the case that B contains some edge not present in A but sharing a vertex with A. Hence, $A \cup B$ is not a matching and $A \sim B$. ■

D. Spanning trees

Here, the probability space Ω is the set of all spanning trees in K_n. Let us consider events E_A for a set of edges A, where E_A occurs for $T \in \Omega$ iff $A \subseteq T$. Define $A \sim B$ unless A and B are vertex-disjoint. Lu et al. [23, Lemma 7] show that this in fact defines a *dependency* graph for spanning trees. It is worth nothing that this scenario illustrates an algorithmic use of the original LLL (not the Lopsided LLL) for which the independent-variable model does not suffice and one must design a non-trivial resampling oracle.

We now show how to implement the resampling oracle in this setting. We note that as a subroutine, we use an algorithm to generate a uniformly random spanning tree in a given graph G. This can be done efficiently by a random walk, for example.

Algorithm 5 Resampling oracle for spanning trees

Function $r_A(T)$:
Check that $A \subseteq T$, otherwise **return** T.
Let $W = V(A)$, the vertices covered by A.
Let $T_1 = \binom{V \setminus W}{2} \cap T$, the edges of T disjoint from W.
Let $F_1 = \binom{V \setminus W}{2} \setminus T$, the edges disjoint from W not present in T.
Let $G_2 = (K_n \setminus F_1)/T_1$ be a multigraph obtained by deleting F_1 and contracting T_1.
Generate a uniformly spanning tree T_2 in G_2.
return $T_1 \cup T_2$.

Lemma III.5. *If A is a fixed forest and T is a uniformly random spanning tree in K_n conditioned on $A \subseteq T$, then $r_A(T)$ produces a uniformly random spanning tree in K_n.*

Proof: First, observe that since T_2 is a spanning tree of $G_2 = (K_n \setminus F_1)/T_1$, it is also a spanning tree of K_n/T_1 where T_1 is a forest, and therefore $T_1 \cup T_2$ is a spanning tree of K_n. We need to prove that it is a uniformly random spanning tree.

First, we appeal to Lemma 6 of Lu et al. [23], which states that given a forest F in K_n with components of sizes $f_1, f_2, \ldots, f_m$, the number of spanning trees containing F is exactly

$$n^{n-2} \prod_{i=1}^{m} \frac{f_i}{n^{f_i-1}}.$$

Equivalently (since n^{n-2} is the total number of spanning trees), for a uniformly random spanning tree T, $\Pr[F \subseteq T] = \prod_{i=1}^{m} \frac{f_i}{n^{f_i-1}}$. This has the surprising consequence that for vertex-disjoint forests F_1, F_2, we have $\Pr[F_1 \cup F_2 \subseteq T] = \Pr[F_1 \subseteq T] \cdot \Pr[F_2 \subseteq T]$, i.e., the containment of F_1 and F_2 are independent events. (In a general graph, the appearances of different edges in a random spanning tree are negatively correlated, but here we are in a complete graph.)

Let $W = V(A)$ and let B be any forest on $V \setminus W$, i.e., vertex-disjoint from A. By the above, the appearance of B in a uniformly random spanning tree is independent of the appearance of A. Hence, if T is uniformly random, we have $\Pr[B \subseteq T \mid A \subseteq T] = \Pr[B \subseteq T]$. This implies that the distribution of $T \cap \binom{V \setminus W}{2}$ is exactly the same for a uniformly random spanning tree T as it is for one conditioned on $A \subseteq T$ (formally, by applying the inclusion-exclusion formula). Therefore, the forest $T_1 = T \cap \binom{V \setminus W}{2}$ is distributed as it should be in a random spanning tree restricted to $V \setminus W$.

The final step is that we extend T_1 to a spanning tree $T_1 \cup T_2$, where T_2 is a uniform spanning tree in $G_2 = (K_n \setminus F_1)/T_1$. Note that we G_2 is a multigraph, i.e., it is important that we preserve the multiplicity of edges after contraction. The spanning trees T_2 in $G_2 = (K_n \setminus F_1)/T_1$ are in a one-to-one correspondence with spanning trees in K_n conditioned on $T \cap \binom{V \setminus W}{2} = T_1$: This is because each such tree T_2 extends T_1 to a different spanning tree of K_n, and each spanning tree where $T \cap \binom{V \setminus W}{2} = T_1$ can be obtained in this way. Therefore, for a fixed T_1, $T_1 \cup T_2$ is a uniformly random spanning tree conditioned on $T \cap \binom{V \setminus W}{2} = T_1$. Finally, since the distribution of T_1 is equal to that of a uniformly random spanning tree restricted to $V \setminus W$, $T_1 \cup T_2$ is a uniformly random spanning tree. ∎

Lemma III.6. *The resampling oracle $r_A(T)$ applied to a spanning tree satisfying E_A does not cause any new event E_B such that $B \notin \Gamma^+(A)$.*

Proof: Note that the only edges that we modify are those incident to $W = V(A)$. Therefore, any new event E_B that the operation of r_A could cause must be such that B contains an edge incident to W and not contained in A. Such an edge shares exactly one vertex with some edge in A and hence $B \sim A$. ∎

E. Composition of resampling oracles for product spaces

Suppose we have a probability space $\Omega = \Omega_1 \times \Omega_2 \times \ldots \times \Omega_N$, where on each Ω_i we have resampling oracles r_{ij} for events $E_{ij}, j \in \mathcal{E}_i$, with respect to a causality graph G_i. Our goal is to show that there is a natural way to combine these resampling oracles in order to handle events on Ω that are obtained by taking intersections of the events E_{ij}. The following theorem formalizes this notion.

Theorem III.7. *Let $\Omega_1, \ldots, \Omega_N$ be probability spaces, where for each Ω_i we have resampling oracles r_{ij} for events $E_{ij}, j \in \mathcal{E}_i$ with respect to a causality graph G_i. Let $\Omega = \Omega_1 \times \Omega_2 \times \ldots \Omega_N$ be the probability space with the respective product measure. For any set J of pairs $(i,j), j \in \mathcal{E}_i$ where each $i \in [N]$ appears at most once, define an event E_J on Ω to occur in a state $\omega = (\omega_1, \ldots, \omega_N)$ iff E_{ij} occurs in ω_i for each $(i,j) \in J$. Define a causality graph G on these events by $J \sim J'$ iff there exist pairs $(i,j) \in J, (i,j') \in J'$ such that $j \sim j'$ in G_i. Then there exist resampling oracles r_J for the events E_J with respect to G, which are obtained by successive calls to $r_{ij}, (i,j) \in J$.*

Proof: For $J = \{(i_1, j_1), (i_2, j_2), \ldots, (i_k, j_k)\}$, we define

$$r_J(\omega_1, \ldots, \omega_N) = (r_{i_1 j_1}(\omega_1), r_{i_2 j_2}(\omega_2), \ldots, r_{i_k j_k}(\omega_k))$$

where we interpret $r_{i_\ell j_\ell}$ as the identity if there is no pair in J whose first element is i_ℓ. We claim that these are resampling oracles with respect to G as defined in the theorem.

Let us denote by μ the default probability distribution on Ω, and by μ_i the respective probability distribution on Ω_i. For the first condition, suppose that $\omega \sim \mu|_{E_J}$. By the product structure of Ω, this is the same as having $\omega = (\omega_1, \ldots, \omega_N)$ where the components are independent and $\omega_\ell \sim \mu_\ell|_{E_{i_\ell j_\ell}}$ for each $(i_\ell, j_\ell) \in J$, and $\omega_\ell \sim \mu_\ell$ for indices not appearing in J. By the properties of the resampling oracles $r_{i_\ell j_\ell}$, we have $r_{i_\ell j_\ell}(\omega) \sim \mu_\ell$. Since the resampling oracles are applied independently, we have $r_J(\omega) = (r_{i_1 j_1}(\omega_1), r_{i_2 j_2}(\omega_2), \ldots, r_{i_k j_k}(\omega_k)) \sim \mu_1 \times \mu_2 \times \ldots \times \mu_N = \mu$.

For the second condition, note that if $\omega \notin E_{J'}$ and $r_J(\omega) \in E_{J'}$, it must be the case that there is $(i_\ell, j_\ell) \in J$ and $(i_\ell, j'_\ell) \in J'$ such that $\omega_\ell \notin E_{i_\ell j'_\ell}$ and $r_{i_\ell j_\ell}(\omega) \in E_{i_\ell j'_\ell}$. However, this is possible only if $j_\ell \sim j'_\ell$ in the causality graph G_{i_ℓ}. By the definition of G, this means that $J \sim J'$ as well. ∎

As a result, we can extend our resampling oracles to spaces like N-tuples of independently random permutations, independently random spanning trees, etc. We make use of this in our applications.

IV. Applications

Let us present a few applications of our framework. Our application with rainbow spanning trees is new, even in the existential sense. Our applications with Latin transversals and rainbow matchings are also new to the best of our knowledge, although they could also have been obtained using existing algorithms [18], [2].

A. Rainbow spanning trees

Given an edge-coloring of K_n, a spanning tree is called rainbow if each of its edges has a distinct color. The existence of a single rainbow spanning tree is completely resolved by the matroid intersection theorem: It can be decided efficiently whether a rainbow spanning tree exists for a given edge coloring, and it can be found efficiently if it exists. However, the existence of multiple edge-disjoint rainbow spanning trees is more challenging. An attractive conjecture of Brualdi and Hollingsworth [9] states that if n is even and K_n is properly edge-colored by $n-1$ colors, then the edges can be decomposed into $n/2$ rainbow spanning trees, each tree using each color exactly once. Until recently, it was only known that every such edge-coloring contains 2 edge-disjoint rainbow spanning trees [4]. In a recent development, it was proved that if every color is used at most $n/2$ times (which is true for any proper coloring) then there exist $\Omega(n/\log n)$ edge-disjoint rainbow spanning trees [10]. In fact this result seems to be algorithmically efficient, although this was not claimed by the authors. We prove that using our framework, we can find $\Omega(n)$ rainbow spanning trees under a slight strengthening of the coloring assumption.

Theorem IV.1. *Given an edge-coloring of K_n such that each color appears on at most $\frac{1}{32}(\frac{7}{8})^7 n$ edges, at least $\frac{1}{32}(\frac{7}{8})^7 n$ edge-disjoint rainbow spanning trees exist and can be found in $O(n^4)$ resampling oracle calls with high probability.*

This result relies on Theorem I.3, our algorithmic version of the cluster expansion local lemma. We note that if there is constant multiplicative slack in the assumption on color appearances, the number of resamplings improves to $O(n^2)$, using the result in Theorem V.10 with constant ϵ slack.

Our approach to prove this theorem is simply to sample $\frac{1}{32}(\frac{7}{8})^7 n$ independently random spanning trees and hope that they will be (a) pairwise edge-disjoint, and (b) rainbow. This unlikely proposition in fact happens to be true with positive probability, thanks to the local lemma and the independence properties in random spanning trees that we mentioned in Section III-D.

Proof: We apply our algorithm in the setting of t independent and uniformly random spanning trees $T_1, \dots, T_t \subset K_n$, with the following two types of bad events:

- E^i_{ef}: For each $i \in [t]$ and two edges $e \neq f$ in K_n of the same color, E^i_{ef} occurs if $\{e, f\} \subset T_i$;
- E^{ij}_e: For each $i \neq j \in [t]$ and an edge e in K_n, E^{ij}_e occurs if $e \in T_i \cap T_j$.

Clearly, if no bad event occurs then the t trees are rainbow and pairwise edge-disjoint.

By Lemma 6 in [23], the probability of a bad event of the first type is $\Pr[E^i_{ef}] = 3/n^2$ if $|e \cup f| = 3$ and $4/n^2$ if $|e \cup f| = 4$. The probability of a bad event of the second type is $\Pr[E^{ij}_e] = (2/n)^2 = 4/n^2$, since each of the two trees contains e independently with probability $2/n$. Hence, the probability of each bad event is upper-bounded by $p = 4/n^2$.

In Section III-D we constructed a resampling oracle r_A for a single spanning tree. Here we extend this to the setting of multiple spanning trees as follows: For an event E^i_{ef}, we define r^i_{ef} as an application of the resampling oracle $r_{\{e,f\}}$ to the tree T_i. For an event E^{ij}_e, we define r^{ij}_e as an application of the resampling oracle $r_{\{e\}}$ independently to the trees T_i and T_j. It is easy to check using Lemma III.5 that for independent uniformly random spanning trees conditioned on either type of event, the respective resampling oracle generates independent uniformly random spanning trees.

Let us define the following dependency graph; we are somewhat conservative for the sake of simplicity. The graph contains the following kinds of edges:

- $E^i_{ef} \sim E^i_{e'f'}$ whenever $e \cup f$ intersects $e' \cup f'$;
- $E^i_{ef}, E^j_{ef} \sim E^{ij}_{e'}$ whenever e' intersects $e \cup f$;
- $E^{ij}_e \sim E^{ij'}_{e'}, E^{i'j}_{e'}$ whenever e' intersects e.

We claim that the resampling oracle for any bad event can cause new bad events only in its neighborhood. This follows from the fact that the resampling oracle affect only the trees relevant to the event (in the superscript), and the only edges modified are those incident to those relevant to the event (in the subscript).

Let us now verify the cluster expansion criteria (so we can apply Theorem I.3). Let us assume that each color appears on at most q edges, and we generate t random spanning trees. We claim that the neighborhood of each bad event can be partitioned into 4 cliques of size $(n-1)(t-1)$ and 4 cliques of size $(n-1)(q-1)$.

First, let us consider an event of type E^i_{ef}. The neighborhood of E^i_{ef} consists of: (1) events $E^i_{e'f'}$ where e' or f' shares a vertex with $e \cup f$; these events form 4 cliques, one for each vertex of $e \cup f$, and the size of each clique is at most $(n-1)(q-1)$, since the number of incident edges to a vertex is $n-1$, and the number of other edges of the same color is at most $q-1$. (2) events $E^{ij}_{e'}$ where e' intersects $e \cup f$; these events form 4 cliques, one for each vertex of $e \cup f$, and each clique has size at most $(n-1)(t-1)$, since its events can be identified with the $(n-1)$ edges incident to a fixed vertex and the remaining $t-1$ trees.

Second, let us consider an event of type E^{ij}_e. The neighborhood of E^{ij}_e consists of: (1) events $E^i_{e'f'}$ and $E^j_{e'f'}$ where e intersects $e' \cup f'$; these events form 4 cliques, one for each vertex of e and either i or j in the superscript, and the size of each clique is at most $(n-1)(q-1)$ by an argument as above. (2) events $E^{i'j}_{e'}, E^{ij'}_{e'}$ where e' intersects e; these events form 4 cliques, one for each vertex of e and either $i'j$ or ij' in the superscript. The size of each clique is at most $(n-1)(t-1)$, since the events can be identified with the $(n-1)$ edges incident to a vertex and the remaining $t-1$ trees.

Considering the symmetry of the dependency graph, we set the variables for all events equal to $y^i_{ef} = y^{ij}_e = y$. The cluster expansion criteria will be satisfied if we set the parameters so that

$$p \leq \frac{y}{(1+(n-1)(t-1)y)^4(1+(n-1)(q-1)y)^4} \leq \frac{y}{\sum_{I \subseteq \Gamma^+(E), I \in \mathsf{Ind}} y^I}.$$

The second inequality holds due to the structure of the neighborhood of each event that we described above. We set $y = \beta p = 4\beta/n^2$ and assume $t \leq \gamma n, q \leq \gamma n$. The reader can verify that with the settings $\beta = (\frac{8}{7})^8$ and $\gamma = \frac{1}{32}(\frac{7}{8})^7$, we get $\frac{\beta}{(1+4\gamma\beta)^8} = 1$. Therefore,

$$p \leq \frac{\beta p}{(1+4\gamma\beta)^8} \leq \frac{y}{(1+(n-1)(t-1)y)^4(1+(n-1)(q-1)y)^4}$$

which verifies the assumption of Theorem I.3. Theorem I.3 implies that MaximalSetResample terminates after $O((\sum y^i_{ef} + \sum y^{ij}_e)^2)$ resampling oracle calls with high probability. The total number of events here is $O(tqn^2) = O(n^4)$ and for each event the respective variable is $y = O(1/n^2)$. Therefore, the expected number of resampling oracle calls is $O(n^4)$. ∎

B. Rainbow matchings

Given an edge-coloring of K_{2n}, a perfect matching is called rainbow if each of its edges has a distinct color. This can be viewed as a non-bipartite version of the problem of Latin transversals. It is known that given any *proper* $(2n-1)$-edge-coloring of K_{2n} (where each color forms a perfect matching), there exists a rainbow perfect matching [33]. However, finding rainbow matchings algorithmically is more difficult. Achlioptas and Iliopoulos [2] showed how to find a rainbow matching in K_{2n} efficiently when each color appears on at most γn edges, $\gamma < \frac{1}{2e} \simeq 0.184$. Our first result is that we can do this for $\gamma = \frac{27}{128} \simeq 0.211$. The improvement comes from the application of the "cluster expansion" form of the local lemma, which is still efficient in our framework. (We note that an updated version of the Achlioptas-Iliopoulos framework [3] also contains this result.)

Theorem IV.2. *Given an edge-coloring of K_{2n} where each color appears on at most $\frac{27}{128}n$ edges, a rainbow perfect matching exists and can be found in $O(n^2)$ resampling oracle calls with high probability.*

In fact, we can find many disjoint rainbow matchings — up to a linear number, if we replace $\frac{27}{128}$ above by a smaller constant.

Theorem IV.3. *Given an edge-coloring of K_{2n} where each color appears on at most $\frac{7^7}{8^8}n$ edges, at least $\frac{7^7}{8^8}n$ edge-disjoint rainbow perfect matchings exist and can be found in $O(n^4)$ resampling oracle calls whp.*

We postpone the proof to Section IV-C, since it follows from our result for Latin transversals there.

Proof of Theorem IV.2: We apply our algorithm in the setting of uniformly random perfect matchings $M \subset K_{2n}$, with the following bad events (identical to the setup in [2]): For every pair of edges e, f of the same color, E_{ef} occurs if $\{e, f\} \subset M$. If no bad event E_{ef} occurs then M is a rainbow matching. We also define the following dependency graph: $E_{ef} \sim E_{e'f'}$ unless e, f, e', f' are four disjoint edges. Note that this is more conservative than the dependency graph we considered in Section III-C, where two events are only connected if they do not form a matching together. The more conservative definition will simplify our analysis. In any case, our resampling oracle is consistent with this lopsidependency

graph in the sense that resampling E_{ef} can only cause new events $E_{e'f'}$ such that $E_{ef} \sim E_{e'f'}$. We show that this setup satisfies the criteria of the cluster expansion lemma.

Lemma IV.4. *For any edge-coloring of K_{2n} such that every color appears at most $q = \frac{27}{128}n$ times, the lopsided association graph above satisfies the assumptions of Theorem I.3 with $p = \frac{1}{(2n-1)(2n-3)}$ and $y = (\frac{4}{3})^4 p$ for each event.*

Proof: Consider the neighborhood of a bad event $\Gamma(E_{ef})$. It contains all events $E_{e'f'}$ such that there is some intersection among the edges e, f, e', f'. Such events can be partitioned into 4 cliques: for each vertex $v \in e \cup f$, let $\mathcal{Q}_v$ denote all the events $E_{e'f'}$ such that $v \in e'$ and f' has the same color as e'. The number of edges e' incident to v is $2n-1$, and for each of them, the number of other edges of the same color is by assumption at most $q-1$. Therefore, the size of $\mathcal{Q}_v$ is at most $(q-1)(2n-1)$.

In the following, we use the short-hand notation $y^I = \prod_{i \in I} y_i$. Consider the assumptions of the cluster expansion lemma: for each event E_{ef}, we should have

$$p_{ef} = \Pr[E_{ef}] \leq \frac{y_{ef}}{\sum_{I \subseteq \Gamma^+(E_{ef}), I \in \mathsf{Ind}} y^I}.$$

We have $p_{ef} = \Pr[\{e, f\} \subset M] = \frac{1}{(2n-1)(2n-3)}$. Let us denote this probability simply by p. By symmetry, we set all the variables y_{ef} to the same value, $y_{ef} = y = \beta p$ for some $\beta > 1$. Note that an independent subset of $\Gamma^+(E_{ef})$ can contain at most 1 event from each clique $\mathcal{Q}_v$. (The event E_{ef} itself is also contained in these cliques.) Therefore,

$$\sum_{I \subseteq \Gamma^+(E_{ef}), I \in \mathsf{Ind}} y^I = \prod_{v \in e \cup f} (1 + \sum_{E_{e'f'} \in \mathcal{Q}_v} y_{e'f'}) \leq (1 + (q-1)(2n-1)y)^4.$$

We assume $q \leq \gamma n$ ($\gamma < 0.5$) and $y = \beta p = \frac{\beta}{(2n-1)(2n-3)}$. We get $\sum_{I \subseteq \Gamma^+(E_{ef}), I \in \mathsf{Ind}} y^I \leq (1 + \frac{1}{2}\gamma\beta)^4$. The reader can check that with $\beta = (\frac{4}{3})^4$ and $\gamma = \frac{27}{128} = \frac{1}{2}(\frac{3}{4})^3$, we have $\frac{\beta}{(1+\frac{1}{2}\gamma\beta)^4} = 1$. Therefore,

$$\frac{y}{\sum_{I \subseteq \Gamma^+(E_{ef}), I \in \mathsf{Ind}} y^I} \geq \frac{\beta p}{(1 + \frac{1}{2}\gamma\beta)^4} \geq p$$

which is the assumption of Theorem I.3. ∎

By Theorem I.3, MaximalSetResample with the resampling oracle for matchings and the dependency graph defined above will find a rainbow perfect matching in time $O(\sum_{E_{ef}} y_{ef} \sum_{E_{ef}} \log(1 + y_{ef})) = O((\sum_{E_{ef}} y_{ef})^2)$ with high probability. The number of bad events E_{ef} is $O(n^3)$, because each color class has $O(n)$ edges so the number of edge pairs of equal color is $O(n^3)$. We have $y_{ef} = O(1/n^2)$, and hence the total number of resamplings is $O(n^2)$ with high probability. This proves Theorem IV.2. ∎

C. Latin transversals

A Latin transversal in an $n \times n$ matrix A is a permutation $\pi \in S_n$ such that the entries $A_{i,\pi(i)}$ ("colors") are distinct for $i = 1, 2, \ldots, n$. In other words, it is a set of distinct entries, exactly one in each row and one in each column. It is easy to see that this is equivalent to a bipartite version of the rainbow matching problem: A_{ij} is the color of the edge (i, j) and we are looking for a perfect bipartite matching where no color appears twice. It is a classical application of the Lovász local lemma that if no color appears more than $\frac{1}{4e}n$ times in A then there exists a Latin transversal [14]. An improvement of this result is that if no color appears more than $\frac{27}{256}n$ times in A then a Latin transversal exists [7]; this paper introduced the "cluster expansion" strengthening of the local lemma. (Note that $\frac{27}{256} = \frac{3^3}{4^4}$.) These results were made algorithmically efficient by the work of Harris and Srinivasan [18]. We note that our framework provides an alternative way to make these results algorithmic, using the resampling oracle for permutations.

Beyond finding one Latin transversal, one can ask whether there exist multiple disjoint Latin transversals. A remarkable existential result was proved by Alon, Spencer and Tetali [6]: If $n = 2^k$ and each color appears in A at most ϵn times ($\epsilon = 10^{-10^{10}}$ in their proof), then A can be partitioned into n disjoint Latin transversals. Here, we show how to find a linear number of Latin transversals algorithmically.

Theorem IV.5. *For any $n \times n$ matrix A where each color appears at most $\frac{7^7}{8^8}n$ times, there at least exist $\frac{7^7}{8^8}n$ disjoint Latin transversals which can be found in $O(n^4)$ resampling oracle calls w.h.p.*

We note that again, if there is constant multiplicative slack in the assumption on color appearances, the number of resamplings improves to $O(n^2)$. This also implies Theorem IV.3 as a special case: For an edge-coloring of K_{2n} where no color appears more than $\frac{7^7}{8^8}n$ times, let us label the vertices arbitrarily $(u_1, \ldots, u_n, v_1, \ldots, v_n)$ construct a matrix A where

A_{ij} is the color of the edge (u_i, v_j). If no color appears more than $\frac{7^7}{8^8}n$ times, by Theorem IV.5 we can find $\frac{7^7}{8^8}n$ Latin transversals; these correspond to rainbow matchings in K_{2n}.

Our approach to proving Theorem IV.5 is similar to the proof of Theorem IV.1: sample $\frac{7^7}{8^8}n$ independently random permutations and hope that they will be (a) disjoint, and (b) Latin. For similar reasons to Theorem IV.1, the local lemma works out and our framework makes this algorithmic.

Proof: Let $t = \frac{7^7}{8^8}n$ and let $\pi_1, \ldots, \pi_t$ be independently random permutations on $[n]$. We consider the following two types of bad events:

- E^i_{ef}: For each $i \in [t]$ and $e = (u, v), f = (x, y) \in [n] \times [n]$ such that $u \neq v, x \neq y, A_{uv} = A_{xy}$, the event E^i_{ef} occurs if $\pi_i(u) = v$ and $\pi_i(x) = y$;
- E^{ij}_e: For each $i \neq j \in [t]$ and $e = (u, v) \in [n] \times [n]$, the event E^{ij}_e occurs if $\pi_i(u) = \pi_j(u) = v$.

Clearly, if none of these events occurs then the permutations $\pi_1, \ldots, \pi_t$ correspond to pairwise disjoint Latin transversals. The probability of a bad event of the first type is $\Pr[E^i_{ef}] = \frac{1}{n(n-1)}$ and the probability for the second type is $\Pr[E^{ij}_e] = \frac{1}{n^2}$. Thus the probability of each bad event is at most $p = \frac{1}{n(n-1)}$.

It will be convenient to think of the pairs $e = (x, y) \in [n] \times [n]$ as edges in a bipartite complete graph. As we proved in Section III-B, the resampling oracle for permutations is consistent with the following lopsidependency graph graph.

- $E^i_{ef} \sim E^i_{e'f'}$ whenever there is some intersection between the edges e, f and e', f';
- $E^i_{ef}, E^j_{ef} \sim E^{ij}_{e'}$ whenever there is some intersection between e' and e, f;
- $E^{ij}_e \sim E^{ij'}_{e'}, E^{i'j}_{e'}$ whenever e' intersects e.

By Lemma III.2, the resampling oracle for a given event never causes a new event except in its neighborhood.

Let us now verify the cluster expansion criteria. The counting here is quite similar to the proof of Theorem IV.1, so we skim over some details. The neighborhood of each event E^i_{ef} consist of 8 cliques: 4 cliques of events of type $E^i_{e'f'}$ and 4 cliques of events of type E^{ij}_e, corresponding in each case to the 4 vertices of $e \cup f$. In the first case, each clique has at most $n(q-1)$ events, determined by selecting an incident edge and another edge of the same color. In the second case, each clique has at most $n(t-1)$ events, determined by selecting an incident edge and another permutation.

The neighborhood of each event E^{ij}_e also consists of 8 cliques: 4 cliques of events $E^i_{e'f'}$ or $E^j_{e'f'}$, corresponding to the choice of either i or j in the superscript, and one of the two vertices of e. The size of each clique is at most $n(q-1)$, determined by choosing an incident edge and another edge of the same color. Then, we have 4 cliques of events $E^{ij'}_{e'}$ or $E^{i'j}_{e'}$, determined by switching either i' or j' in the superscript, and choosing one of the vertices of e. The size of each clique is at most $n(t-1)$, determined by choosing an incident edge and a new permutation in the superscript.

As a consequence, the cluster expansion criterion here is almost exactly the same as in the case of Theorem IV.1:

$$p \leq \frac{y}{(1 + n(t-1)y)^4(1 + n(q-1)y)^4}.$$

We have $p = \frac{1}{n(n-1)}$ here and we set $y = \beta p$. For $t, q \leq \gamma n$, it's enough to satisfy $\frac{\beta}{(1+\beta\gamma)^8} \geq 1$, which is achieved by $\beta = (\frac{8}{7})^8$ and $\gamma = \frac{7^7}{8^8}$. Therefore, Theorem I.3 implies that MaximalSetResample will terminate within $O((\sum y^i_{ef} + \sum y^{ij}_e)^2) = O(n^4)$ resampling oracle calls with high probability. ∎

V. Analysis of the algorithm

In this section, we present a simple self-contained analysis of the MaximalSetResample algorithm in the setting of the Lovász Local Lemma with some slack in the (GLL) conditions. We use a framework of stable set sequences similar to that of Kolipaka and Szegedy [21], which builds on Shearer's work [31]. While the ultimate purpose of this machinery is the handle the Lovász Lemma Lemma in Shearer's optimal form, and to prove Theorem I.2 and Theorem I.3 without any assumption of slack, we defer this to the full version of the paper [19].

We remark that much of the Kolipaka-Szegedy machinery is useful for us, even though the way we apply it is different. Whereas Kolipaka-Szegedy uses witness trees/sequences growing *backwards in time* (as Moser and Tardos did), we analyze similar sequences growing *forward in time*. There are both similarities and differences in how these two viewpoints lead to proofs of algorithmic efficiency.

A. Stable set sequences and the coupling argument

Definition V.1. *One execution of the outer repeat loop in MaximalSetResample is called an* iteration. *For a fixed sequence of non-empty sets $\mathcal{I} = (I_1, \ldots, I_t)$, we say that the algorithm* follows *$\mathcal{I}$ if I_s is the set resampled in iteration s for $1 \leq s < t$, and I_t is a prefix of the sequence of events resampled in iteration t. (That is, I_t contains the first m events resampled in iteration t, for some $m \geq 1$.)*

Recall that $\mathsf{Ind} = \mathsf{Ind}(G)$ denotes the independent sets (including the empty set) in the graph G under consideration.

Definition V.2. *$\mathcal{I} = (I_1, I_2, \ldots, I_t)$ is called a* stable set sequence *if $I_1, \ldots, I_t \in \mathsf{Ind}(G)$ and $I_{s+1} \subseteq \Gamma^+(I_s)$ for each $1 \leq s < t$. We call the sequence $\mathcal{I}$* proper *if each set I_s is nonempty.*

Note that if $I_s = \emptyset$ for some s, then $I_t = \emptyset$ for all $t > s$. Therefore, the nonempty sets always form a prefix of the stable set sequence. Formally, we also consider an empty sequence as a stable set sequence of length 0.

Lemma V.3. *If MaximalSetResample follows a sequence $\mathcal{J} = (J_1, \ldots, J_t)$, then $\mathcal{J}$ is a stable set sequence.*

Proof: By construction, the set J_s chosen in each iteration is independent in G. For each $i \in J_s$, we execute the resampling oracle r_i. Recall that r_i can only cause new events in the neighborhood $\Gamma^+(i)$, and this neighborhood will not be explored again until the following iteration. The iteration terminates when no event outside $\Gamma^+(J_s)$ occurs. This shows that $J_{s+1} \subseteq \Gamma^+(J_s)$. In the last iteration, this also holds for a subset of the resampled events. ∎

We use the following notation: For $i \in [n]$, $p_i = \Pr_\mu[E_i]$. For $S \subseteq [n]$, $p^S = \prod_{i \in S} p_i$. For a stable set sequence $\mathcal{I} = (I_1, \ldots, I_t)$, $p_{\mathcal{I}} = \prod_{s=1}^t p^{I_s}$. We relate stable set sequences to executions of the algorithm by the following coupling argument. Although our use of stable set sequences is inspired by Kolipaka-Szegedy [21], their coupling argument is different as theirs is backward-looking in nature and ours is forward-looking.

Lemma V.4. *For any proper stable set sequence $\mathcal{I} = (I_1, I_2, \ldots, I_t)$, the probability that the MaximalSetResample algorithm follows $\mathcal{I}$ is at most $p_{\mathcal{I}}$.*

Proof: Given $\mathcal{I} = (I_1, I_2, \ldots, I_t)$, let us consider the following "$\mathcal{I}$-checking" random process. We start with a random state $\omega \sim \mu$. In iteration s, we process the events of I_s in the ascending order of their indices. For each $i \in I_s$, we check whether ω satisfies E_i; if not, we terminate. Otherwise, we apply the resampling oracle r_i and replace ω by $r_i(\omega)$. We continue for $s = 1, 2, \ldots, t$. We say that the $\mathcal{I}$-checking process succeeds if every event is satisfied when checked and the process runs until the end.

By induction, the state ω after each resampling oracle call is distributed according to μ: Assuming this was true in the previous step and conditioned on E_i satisfied, we have $\omega \sim \mu|_{E_i}$. By assumption, the resampling oracle r_i removes this conditioning and produces again a random state $r_i(\omega) \sim \mu$. Therefore, whenever we check event E_i, it is satisfied with probability $\Pr_\mu[E_i]$ (conditioned on the past). By a telescoping product of conditional probabilities, the probability that the $\mathcal{I}$-checking process succeeds is exactly $\prod_{s=1}^t \prod_{i \in I_s} \Pr_\mu[E_i] = \prod_{s=1}^t p^{I_s} = p_{\mathcal{I}}$.

To conclude, we argue that the probability that MaximalSetResample follows the sequence $\mathcal{I}$ is at most the probability that the $\mathcal{I}$-checking process succeeds. To see this, suppose that we couple MaximalSetResample and the $\mathcal{I}$-checking process, so they use the same source of randomness. In each iteration, if MaximalSetResample includes i in J_t, it means that E_i is satisfied. Both procedures apply the resampling oracle $r_i(\omega)$, so the states in the next iteration are the same. Therefore, the event that MaximalSetResample follows the sequence $\mathcal{I}$ is contained in the event that the $\mathcal{I}$-checking process succeeds, which happens with probability $p_{\mathcal{I}}$. ∎

It is important to understand that the MaximalSetResample algorithm's state ω does *not* satisfy $\omega \sim \mu$ in every iteration; otherwise, that would mean that the algorithm is no better than repeated independent sampling from μ. Rather, it is the $\mathcal{I}$-checking process whose state ω always satisfies $\omega \sim \mu$. Let us also remark that Lemma V.4 does not use any properties of the sets J_t chosen by the MaximalSetResample algorithm (e.g. maximality).

Definition V.5. *Let* Stab *denote the set of all stable set sequences and* Prop *the set of proper stable set sequences. For $\mathcal{I} = (I_1, \ldots, I_t) \in \mathsf{Prop}$, let us call $\sigma(\mathcal{I}) = \sum_{s=1}^t |I_s|$ the total size of the sequence.*

Lemma V.6. *The probability that MaximalSetResample resamples at least s events is at most $\sum_{\mathcal{I} \in \mathsf{Prop}: \sigma(\mathcal{I}) = s} p_{\mathcal{I}}$.*

Proof: If the algorithm resamples at least s events, it means that it follows some proper sequence $\mathcal{I}$ of total size $\sigma(\mathcal{I}) = s$. By the union bound, the probability of resampling at least s events is upper-bounded by $\sum_{\mathcal{I} \in \mathsf{Prop}: \sigma(\mathcal{I}) = s} p_{\mathcal{I}}$. ∎

As remarked in Section I, the original proof of the LLL (Theorem I.1) trivially generalizes to the scenario of directed dependency graphs [5]. The analysis of this subsection also trivially generalizes. First, one changes the definition of $\Gamma(i)$ to denote the vertices j such that there is a directed arc from i to j. Next, since $\mathsf{Ind}(G)$ is not defined for directed graphs, we simply omit the condition $I_1, \ldots, I_t \in \mathsf{Ind}(G)$, from Definition V.2.

B. Analysis in the LLL setting with slack

Let us prove the following (crude) bound on the sum over all stable set sequences. We note that this first bound is typically exponentially large.

Lemma V.7. *Provided that the p_i satisfy the* (GLL) *criterion, $p_i \le x_i \prod_{j\in\Gamma(i)}(1-x_j)$, we have*

$$\sum_{\mathcal{I}\in\mathsf{Prop}} p_{\mathcal{I}} \;\le\; \prod_{i=1}^{n} \frac{1}{1-x_i}.$$

The proof will use the following trivial fact.

Fact V.8. *For any finite set S and any real values α_i, we have $\prod_{i\in S}(1+\alpha_i) = \sum_{S'\subseteq S}\prod_{i\in S'}\alpha_i$.*

Proof: It will be convenient to work with sequences of fixed length, where we pad by empty sets if necessary. Note that by definition this does not change the value of $p_{\mathcal{I}}$: e.g., $p_{(I_1,I_2)} = p_{(I_1,I_2,\emptyset,\ldots,\emptyset)}$. Recall that Stab denotes the set of all stable set sequences. We show the following statement by induction on ℓ: For any fixed independent set J,

$$\sum_{\substack{\mathcal{I}=(I_1,\ldots,I_\ell)\in\mathsf{Stab}\\ I_1=J}} p_{\mathcal{I}} \;\le\; \prod_{j\in J}\frac{x_j}{1-x_j}. \tag{2}$$

This is true for $\ell = 1$, since $p_{(J)} = p^J \le \prod_{j\in J} x_j$ by (GLL).

Let us consider the expression for $\ell+1$.

$$\begin{aligned}
\sum_{\substack{\mathcal{I}'=(I_0,I_1,\ldots,I_\ell)\in\mathsf{Stab}\\ I_0=J}} p_{\mathcal{I}'} \;&=\; p^J \sum_{\substack{\mathcal{I}=(I_1,\ldots,I_\ell)\in\mathsf{Stab}\\ I_1\subseteq\Gamma^+(J)}} p_{\mathcal{I}}\\
&\le\; p^J \sum_{I_1\subseteq\Gamma^+(J)}\prod_{i\in I_1}\frac{x_i}{1-x_i} \qquad \text{(by induction)}\\
&\le\; p^J \prod_{i\in\Gamma^+(J)}\left(1+\frac{x_i}{1-x_i}\right) \qquad \text{(by Fact V.8 with } \alpha_i = \tfrac{x_i}{1-x_i})\\
&=\; \Big(\prod_{i'\in J} p_{i'}\Big)\cdot\prod_{i\in\Gamma^+(J)}\frac{1}{1-x_i}\\
&\le\; \Big(\prod_{i'\in J} x_{i'}\prod_{j\in\Gamma(i')}(1-x_j)\Big)\cdot\prod_{i\in\Gamma^+(J)}\frac{1}{1-x_i} \qquad \text{(by (GLL))}\\
&\le\; \prod_{i\in J}\frac{x_i}{1-x_i}
\end{aligned}$$

because each factor $\frac{1}{1-x_i}$ for $i\in\Gamma^+(J)\setminus J$ is covered at least once by $(1-x_j)$ where $j\in\Gamma(i')$ for some $i'\in J$. This proves (2).

Finally, adding up over all sets $J\subseteq[n]$, we use Fact V.8 again to obtain

$$\sum_{\mathcal{I}=(I_1,\ldots,I_\ell)\in\mathsf{Stab}} p_{\mathcal{I}} \;\le\; \sum_{J\subseteq[n]}\prod_{j\in J}\frac{x_j}{1-x_j} \;=\; \prod_{i=1}^{n}\left(1+\frac{x_i}{1-x_i}\right) \;=\; \prod_{i=1}^{n}\frac{1}{1-x_i}.$$

Recall that the sequences on the left-hand side may contain empty sets. Rewriting this in terms of proper sequences, we have

$$\sum_{k=1}^{\ell}\sum_{\mathcal{I}=(I_1,\ldots,I_k)\in\mathsf{Prop}} p_{\mathcal{I}} \;\le\; \prod_{i=1}^{n}\frac{1}{1-x_i}.$$

Since this is true for every ℓ, and the left-hand-side is non-increasing in ℓ, the sequence as $\ell\to\infty$ has a limit and the bound still holds in the limit. ∎

As a corollary, we can show that our algorithm is efficient if (GLL) is satisfied with a slack.

Theorem V.9. *If* (GLL) *is satisfied with an ϵ slack, i.e.*

$$\Pr_{\mu}[E_i] \;\le\; (1-\epsilon)\cdot x_i\prod_{j\in\Gamma(i)}(1-x_j) \qquad \forall i\in[n],$$

then the probability that MaximalSetResample resamples more than $s = \frac{1}{\epsilon}(\sum_{i=1}^{n}\ln\frac{1}{1-x_i}+t)$ events is at most e^{-t}.

Proof: By Lemma V.6, the probability that MaximalSetResample resamples at least s events is upper bounded by $\sum_{\mathcal{I}\in\mathsf{Prop}:\sigma(\mathcal{I})=\lceil s\rceil} p_{\mathcal{I}}$. By the slack assumption, we have $p_i \le (1-\epsilon)p'_i$ and $p_{\mathcal{I}} \le (1-\epsilon)^{\sigma(\mathcal{I})}p'_{\mathcal{I}}$, where $p'_i = x_i\prod_{j\in\Gamma(i)}(1-x_j)$. Using Lemma V.7, we obtain

$$\sum_{\mathcal{I}\in\mathsf{Prop}:\sigma(\mathcal{I})=\lceil s\rceil} p_{\mathcal{I}} \;\le\; (1-\epsilon)^s \sum_{\mathcal{I}\in\mathsf{Prop}} p'_{\mathcal{I}} \;\le\; e^{-\epsilon s}\prod_{i=1}^{n}\frac{1}{1-x_i}.$$

For $s = \frac{1}{\epsilon}(\sum_{i=1}^{n}\ln\frac{1}{1-x_i} + t)$, we obtain

$$\sum_{\mathcal{I}\in\mathsf{Prop}:\sigma(\mathcal{I})=\lceil s\rceil} p_{\mathcal{I}} \;\le\; e^{-\epsilon s}\prod_{i=1}^{n}\frac{1}{1-x_i} \;\le\; e^{-t}.$$

Therefore, the probability of resampling more than s events is at most e^{-t}. ∎

The analysis of this subsection also trivially generalizes to the scenario of directed dependency graphs. No changes are necessary, other than the revised definition of $\Gamma(i)$ and stable set sequences, as discussed in Section V-A. Lemma V.7 yields an existential proof of Theorem I.1 for the scenario of directed dependency graphs, even if (GLL) holds without slack. Theorem V.9 yields an efficient algorithm for the scenario of directed dependency graphs if (GLL) holds with some slack.

C. Algorithms with more general criteria

A very minor modification of this proof yields the same result for the *cluster expansion* variant of the local lemma, which replaces the (GLL) criterion by (CLL). Recall that stable set sequences consist of independent sets in G, a fact which was ignored in the proof above. Incorporating that fact into the inductive proof of Lemma V.7 shows that, assuming (CLL) holds, we have

$$\sum_{\mathcal{I}\in\mathsf{Prop}} p_{\mathcal{I}} \;\le\; \sum_{I\in\mathsf{Ind}} y^I \;\le\; \prod_{i=1}^{n}(1+y_i).$$

As in Theorem V.9, assuming that (CLL) holds with an ϵ slack implies an efficient running time bound, which can be formulated as follows.

Theorem V.10. *If* (CLL) *is satisfied with an ϵ slack, i.e.,*

$$\Pr[E_i] \;\le\; (1-\epsilon)\cdot\frac{y_i}{\sum_{I\subseteq\Gamma^+(i),I\in\mathsf{Ind}} y^I},$$

then the probability that MaximalSetResample resamples more than $\frac{1}{\epsilon}\left(\sum_{j=1}^{n}\ln(1+y_j)+t\right)$ events is at most e^{-t}.

Finally, it is possible to show that a similar result holds even with respect to Shearer's criterion. Harnessing this result, we prove that the assumption of slack in the LLL / cluster expansion criteria is actually not necessary and can be eliminated. This leads to Theorem I.2 and Theorem I.3. We defer the details to a full version of the paper [19].

ACKNOWLEDGEMENTS

We thank Mohit Singh for discussions at the early stage of this work. We thank David Harris for suggesting the results of Section III-E.

REFERENCES

[1] Dimitris Achlioptas and Themis Gouleakis. Algorithmic improvements of the Lovász local lemma via cluster expansion. In *Proceedings of FSTTCS*, 2012.

[2] Dimitris Achlioptas and Fotis Iliopoulos. Random walks that find perfect objects and the Lovász local lemma. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 494–503, 2014.

[3] Dimitris Achlioptas and Fotis Iliopoulos. Random walks that find perfect objects and the Lovász local lemma. *CoRR*, abs/1406.0242v3, 2015.

[4] Saieed Akbari and Alireza Alipour. Multicolored trees in complete graphs. *J. Graph Theory*, 54:3:221–232, 2007.

[5] Noga Alon and Joel Spencer. *The Probabilistic Method.* Wiley, 2000.

[6] Noga Alon, Joel Spencer, and Prasad Tetali. Covering with latin transversals. *Discrete Applied Mathematics*, 57:1:1–10, 1995.

[7] R. Bissacot, R. Fernández, A. Procacci, and B. Scoppola. An improvement of the Lovász local lemma via cluster expansion. *Combin. Probab. Comput.*, 20:709–719, 2011.

[8] Julia Böttcher, Yoshiharu Kohayakawa, and Aldo Procacci. Properly coloured copies and rainbow copies of large graphs with small maximum degree. *Random Structures and Algorithms*, 40(4), 2012.

[9] Richard A. Brualdi and Susan Hollingsworth. Multicolored trees in complete graphs. *J. Combin. Theory Ser. B*, 68, 1996.

[10] James M. Carraher, Stephen G. Hartke, and Paul Horn. Edge-disjoint rainbow spanning trees in complete graphs, 2013.

[11] Karthekeyan Chandrasekaran, Navin Goyal, and Bernhard Haeupler. Deterministic algorithms for the Lovász local lemma. *SIAM Journal on Computing*, 42(6), 2013.

[12] Kai-Min Chung, Seth Pettie, and Hsin-Hao Su. Distributed algorithms for the Lovász local lemma and graph coloring. In *Proceedings of PODC*, 2014.

[13] Paul Erdös and László Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. In A. Hajnal et al., editor, *Infinite and finite sets*, volume 10 of *Colloquia Mathematica Societatis János Bolyai*, pages 609–628. North-Holland, Amsterdam, 1975.

[14] Paul Erdös and Joel Spencer. The Lopsided Lovász Local Lemma and Latin transversals. *Discrete Applied Mathematics*, 30:151–154, 1991.

[15] Heidi Gebauer, Tibor Szabó, and Gábor Tardos. The local lemma is tight for SAT. In *Proceedings of SODA*, 2011.

[16] Ioannis Giotis, Lefteris Kirousis, Kostas I. Psaromiligkos, and Dimitrios M. Thilikos. On the algorithmic Lovász local lemma and acyclic edge coloring. In *Proceedings of ANALCO*, 2015.

[17] Bernhard Haeupler, Barna Saha, and Aravind Srinivasan. New constructive aspects of the Lovász local lemma. *Journal of the ACM*, 58(6), 2011.

[18] David G. Harris and Aravind Srinivasan. A constructive algorithm for the Lovász local lemma on permutations. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 907–925, 2014.

[19] Nicholas J. A. Harvey and Jan Vondrák. An algorithmic proof of the Lovász local lemma via resampling oracles. *CoRR*, abs/1504.02044, 2015.

[20] Richard Holley. Remarks on the FKG inequalities. *Communications in Mathematical Physics*, 36:227–231, 1974.

[21] Kashyap Kolipaka and Mario Szegedy. Moser and Tardos meet Lovász. In *Proceedings of STOC*, 2011.

[22] Kashyap Kolipaka, Mario Szegedy, and Yixin Xu. A sharper local lemma with improved applications. In *Proceedings of APPROX/RANDOM*, 2012.

[23] Lincoln Lu, Austin Mohr, and László Székely. Quest for negative dependency graphs. *Recent Advances in Harmonic Analysis and Applications*, 25:243–258, 2013.

[24] Austin Mohr. *Applications of the lopsided Lovász local lemma regarding hypergraphs*. PhD thesis, University of South Carolina, 2013.

[25] Robin Moser. *Exact Algorithms for Constraint Satisfaction Problems*. PhD thesis, ETH Zürich, 2012.

[26] Robin A. Moser. A constructive proof of the Lovász local lemma. In *Proceedings of STOC*, 2009.

[27] Robin A. Moser and Gábor Tardos. A constructive proof of the general Lovász Local Lemma. *Journal of the ACM*, 57(2), 2010.

[28] Sokol Ndreca, Aldo Procacci, and Benedetto Scoppola. Improved bounds on coloring of graphs. *European Journal of Combinatorics*, 33(4), 2012.

[29] Wesley Pegden. An extension of the Moser-Tardos algorithmic local lemma. *SIAM J. Discrete Math*, 28:911–917, 2014.

[30] Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer, 2004.

[31] James B. Shearer. On a problem of Spencer. *Combinatorica*, 5(3), 1985.

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

[33] David E. Woolbright and Hung-Lin Fu. On the existence of rainbows in 1-factorizations of K_{2n}. *Journal of Combinatorial Designs*, 6:1:1–20, 1998.