An Algorithmic Proof of the Lopsided Lovász Local Lemma

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

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

Abstract

For any probability space and a collection of events with what we call a lopsided association graph, we give an algorithmic proof of the Lopsided Lovász Local Lemma, and its stronger known forms. The only algorithmic assumption is the availability of a resampling oracle for each event. The resampling oracles act as an abstraction layer that isolates the details of the proof from the details of the probability space. The number of resamplings performed by the algorithm is bounded by expressions similar to those obtained by Moser and Tardos, and others. The overall computational efficiency depends on the runtime of the resampling oracles. We develop efficient resampling oracles for the known uses of the Lopsided Lovász Local Lemma, unifying previous algorithmic applications and presenting new results for packings of Latin transversals, rainbow matchings and rainbow spanning trees.
# Contents

1 Introduction  
1.1 The MaximalSetResample algorithm  
1.2 Related work  
1.3 Our techniques  

2 Relationship between lopsidependency and resampling  
2.1 Example: monotone events on lattices  

3 Implementation in special cases  
3.1 Independent random variables  
3.2 Random permutations  
3.3 Perfect matchings  
3.4 Spanning trees  

4 Applications  
4.1 Rainbow matchings  
4.2 Rainbow spanning trees  
4.3 Latin transversals  

5 Analysis of the algorithm  
5.1 Notation and basic facts  
5.2 Analysis in Shearer’s setting  
5.3 Shearer’s criterion with slack  
5.4 Lovász Local Lemma as a special case of Shearer’s Lemma  
5.5 Lovász Local Lemma without a slack  
5.6 Cluster expansion  

A Comparison to the framework of Achlioptas and Iliopoulos
1 Introduction

The Lovász Local Lemma (LLL) is a powerful tool with numerous uses in combinatorics and theoretical computer science. It asserts the existence of a point in a probability space that simultaneously avoids specified undesired events if their probabilities and dependences are bounded by certain criteria. The classical formulation of the LLL \cite{11,32,12} is as follows.

Let $\Omega$ be a discrete probability space with probability measure $\mu$. Let $E_1, \ldots, E_n$ be “undesired” events in that space. Let $G$ be a graph with vertex set $[n] = \{1, \ldots, n\}$. We denote the edge set by $E(G)$, and the neighborhood of vertex $i$ by $\Gamma(i) = \{ j \neq i : (i, j) \in E(G) \}$. Also, let $\Gamma^+(i) = \Gamma(i) \cup \{i\}$ and $\Gamma^+(I) = \bigcup_{i \in I} \Gamma^+(i)$.

**Theorem 1.1** (General Lovász Local Lemma \cite{11,32}). Suppose that the events satisfy the following condition that controls their dependences

$$\Pr_{\mu}[E_i \mid \bigcap_{j \in J} E_j] = \Pr_{\mu}[E_i] \quad \forall i \in [n], J \subseteq [n] \setminus \Gamma^+(i) \quad \text{(Dep)}$$

and the following criterion that controls their probabilities

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

Then $\Pr_{\mu}[^{\bigcap}_{i=1}^{n} \overline{E_i}] \geq \prod_{i=1}^{n} (1 - x_i) > 0$.

When (Dep) holds, $G$ is called a *dependency graph*. Erdős and Spencer \cite{12} showed that Theorem 1.1 still holds when (Dep) is generalized to

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

In this case $G$ is called a “lopsided dependency graph” and the theorem is called the Lopsided Lovász Local Lemma (LLL). The LLLL has had several interesting uses in combinatorics and theoretical computer science, e.g., existence of Latin transversals \cite{12} and optimal thresholds for satisfiability \cite{13}.

**Algorithms.** Algorithms to efficiently find such a desired point in $\bigcap_{i=1}^{n} \overline{E_i}$ have been the subject of much research. After decades of effort in this direction, a beautiful result was discovered by Moser \cite{24} who proposed an extremely elegant algorithm to avoid all undesired events and proved that it is efficient for the canonical application of the LLL, the bounded-degree $k$-SAT problem. Subsequently, Moser and Tardos \cite{25} generalized the algorithm and its analysis to the general LLL with events on a space of independent random variables. There have been numerous extensions of this work, such as \cite{1,9,10,15,19,20,27}.

All of this work assumes that the probability space consists of independent random variables that can be individually manipulated, as Moser and Tardos did. Thus, these results cannot be viewed as algorithmic proofs of the LLL (Lopsided or not) in full generality. Some applications of the General LLL and typical applications of the Lopsided LLL involve more general probability spaces that do not consist of many independent variables. Common examples include random permutations, matchings and spanning trees (see \cite{21,22} for a comprehensive survey).

---

1 We note that random spanning trees form an example scenario for the General LLL (without lopsidependency) for which the independent-variable model does not suffice; see Section 3.4.
This past year, two efficient algorithms have been developed for some applications of the local lemma that go beyond the independent-variable model. Harris and Srinivasan [16] give an extension of the Moser-Tardos algorithm to the scenario of random permutations. Achlioptas and Iliopoulos [2] define an abstract framework that relates to the local lemma but, for better or for worse, abandons an explicit connection with the probabilistic LLL formulation. Thus, despite impressive progress there has been no algorithmic approach that could recover the general LLL in a variable-free setting. We discuss this further in Section 1.2 and Appendix A.

Our contributions. We propose an algorithmic framework that provides a constructive proof of the Lopsided Lovász Local Lemma in an abstract setting. To make this precise, let us introduce further notation.

An atomic event \( \omega \) in the probability space \( \Omega \) will be called a state. We write \( \omega \sim \mu \) to denote that a random state \( \omega \) is distributed according to \( \mu \), and \( \omega \sim \mu | E_i \) to denote that the distribution is \( \mu \) conditioned on \( E_i \).

Algorithmic assumptions. Our framework requires only two assumptions in order to give an algorithmic proof of the Lopsided Local Lemma.

- **Sampling from \( \mu \):** We are able to generate a random state \( \omega \sim \mu \).
- **Resampling oracles:** For each event \( E_i \), we have a (randomized) resampling oracle \( r_i : \Omega \to \Omega \) 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) If \( E_i \) occurs in a state \( \omega \) and \( E_j \) does not, for \( j \notin \Gamma^+(i) \), then \( E_j \) does not occur in \( r_i(\omega) \) either. (Resampling an undesired event that occurs cannot cause new non-neighbor events to occur.)

Let us emphasize the difference between conditioning on \( E_i \) versus conditioning on \( \overline{E_i} \), both in (Lop) and in the definition of resampling oracles. Executing \( r_i(\omega) \) removes conditioning on \( E_i \), whereas the behavior of \( r_i(\omega) \) is undefined if \( E_i \) does not occur in \( \omega \).

Main Result. Our main result is that, for any probability space, if algorithmic access is provided via resampling oracles, then the Lopsided Lovász Local Lemma can be proven algorithmically.

**Theorem (Informal).** For any probability space and any events with resampling oracles, if the Local Lemma criteria are satisfied, then our algorithm finds a point avoiding all events in oracle-polynomial time.

Lopsided Association. The only remaining question is the availability of the resampling oracles. Here, we draw a direct connection with the conditions of the local lemma. We prove that the existence of resampling oracles with respect to a graph \( G \) is equivalent to \( G \) being what we call a lopsided association graph:

\[
\Pr_\mu [E_i \land F] \geq \Pr_\mu [E_i] \cdot \Pr_\mu [F] \quad \forall i \in [n], \forall F \in \mathcal{F}_i \tag{LopA}
\]

where \( \mathcal{F}_i \) contains all events \( F \) whose indicator variable is a monotone non-decreasing function of the indicator variables of \( (E_j : j \notin \Gamma^+(i)) \).

**Theorem (Informal).** Resampling oracles exist for events with a graph \( G \) if and only if \( G \) is a lopsided association graph.

We refer to Section 2 for more details. Being a lopsided association graph is a slightly stronger property than being a lopsidependency graph, but not by much: every lopsidependency graph that we are aware of
in the literature is actually a lopsided association graph as well. In Section 4, we discuss several scenarios originally involving a lopsidependency graph that turns out to be a lopsided association graph. To see formally that \( \text{(LopA)} \) implies \( \text{(Lop)} \), simply take \( F = \bigcup_{j \in J} E_j \) for any \( J \subseteq [n] \setminus \Gamma^+(i) \).

We remark that the equivalence of resampling oracles and lopsided association is only existential — it does not guarantee that the resampling oracle is computationally efficient. However, in all scenarios that we have studied, efficient implementations of the resampling oracle arise naturally from existing work, or can be devised with minimal effort: In particular this is the case for independent variables, random permutations, perfect matchings, and spanning trees in complete graphs.

Thus, we recover the main result of Moser-Tardos [25] and give a conceptually simpler proof of the result of Harris-Srinivasan [16] (with a slightly weaker bound on the number of resamplings). As new application scenarios, we can handle algorithmically the LLLL applied to perfect matchings and spanning trees in complete graphs, as discussed in Section 3. Moreover, our results hold even for stronger forms of the local lemma, including those studied by Pegden [27] and Kolipaka-Szegedy [19]. We provide several applications in Section 4.

1.1 The MaximalSetResample algorithm

A striking aspect of the work of Moser and Tardos [25] is the simplicity and flexibility of their algorithm — in any iteration, any event \( E_i \) that occurs can be resampled. One reason that their analysis can accommodate this flexibility seems to lie in their use of the independent-variable model. If two events share no variables, the order in which they were resampled is irrelevant, as there is effectively no conditioning between those resamplings. This property is no longer true in the context of lopsidependency graphs — there are complicating dependences between different resamplings. Through much care and technical effort, Harris and Srinivasan [16] were able to show that the Moser-Tardos algorithm indeed works in lopsidependent scenarios with permutations.

We propose a mildly different algorithm that will allow us to handle with ease all lopsidependent scenarios that have resampling oracles. Our algorithm resamples, in turn, a maximal independent set of events that occur, chosen according to an arbitrary but fixed ordering. This additional structure on the sequence of resamplings seems to simplify the analysis in the lopsidependent setting. Pseudocode is shown in Algorithm 1. Very similar algorithms have been proposed before, particularly parallel algorithms [25, 19] although our interest lies not in the parallel aspects but rather with handling lopsidependency as cleanly as possible.

We also invite the reader to compare our algorithm with the GeneralizedResample algorithm of Kolipaka-Szegedy [19]. The two algorithms are similar in that they work at the level of abstract distributions and they repeatedly choose a maximal independent set \( J \) of bad events to “fix”. However, the way that the bad events are fixed is different: GeneralizedResample needs to sample from \( \mu_{[\bigcap_{j \in J} \Gamma^+(i) \cap E_j]} \), which is a complicated operation without a clear implementation. Thus our algorithm can be viewed as a modification to GeneralizedResample to make it efficient in all known scenarios.

Trivially, if MaximalSetResample terminates then it has found a state \( \omega \in \bigcap_{i=1}^n E_i \). Therefore, we only have to prove that MaximalSetResample terminates in polynomial time. We present the full analysis in Section 5. Our main result is the following.

**Theorem 1.2.** Suppose that the events \( E_1, \ldots, E_n \) satisfy \( \text{(GLL)} \) and that the algorithmic assumptions stated above hold. Then the expected number of calls to a resampling oracle by MaximalSetResample is \( O\left( \sum_{i=1}^n \frac{x_i}{1-x_i} \sum_{j=1}^n \log \frac{1}{1-x_j} \right) \).
Algorithm 1 MaximalSetResample uses resampling oracles to output a state $\omega \in \bigcap_{i=1}^{n} E_i$.

1: Initialize $\omega$ with a random state sampled from $\mu$;
2: $t := 0$;
3: repeat
4: $t := t + 1$;
5: $J_t := \emptyset$
6: while there is $i \notin \Gamma^+(J_t)$ such that $E_i$ occurs in $\omega$ do
7: Pick the smallest such $i$;
8: $J_t := J_t \cup \{i\}$;
9: $\omega := r_i(\omega)$; $\triangleright$ Resample $E_i$
10: end while
11: until $J_t = \emptyset$;
12: return $\omega$.

Bissacot et al. [5] devised an improvement of the (GLL) criterion called the \textit{cluster expansion} criterion, based on insights from statistical physics. The cluster expansion criterion has given improved results in several applications of the local lemma [6, 16, 26]. We also prove a stronger version of Theorem 1.2 which uses the cluster expansion criterion. Previous algorithmic work has used the cluster expansion criterion in the variable model [1, 27] and for permutations [16].

To state this result, let us denote the family of independent sets in the graph $G$ by $\text{Ind}$. Theorem 1.3. Assume the algorithmic assumptions above and let the events satisfy the following criterion:

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

Then MaximalSetResample resamples $O\left( \sum_{i=1}^{n} y_i \sum_{j=1}^{n} \ln(1 + y_j) \right)$ events in expectation.

MaximalSetResample also gives an algorithmic proof of the local lemma under the optimal criterion, known as Shearer’s criterion [30], if there is $\epsilon$ slack. We remark that this result is used as a black box to derive the results above. We defer a detailed discussion to Section 5.

1.2 Related work

There has been a flurry of activity in the area of the algorithmic local lemma in recent years. Let us review the developments that are most relevant to our work.

After decades of effort with various partial successes in implementing the local lemma algorithmically, a major advance was achieved by Moser and Tardos [24, 25] who gave an algorithmic form of the local lemma using the (GLL) criterion, assuming the independent variable model. Our Theorem 1.2 generalizes that result to all scenarios of the lopsided local lemma with resampling oracles, using a slightly different algorithm and a slightly larger number of resamplings. Pegden [27] generalized the Moser-Tardos result to use the cluster expansion (CLL) criterion, again in the independent variable model. Our Theorem 1.3 extends Theorem 1.2 in the same way that [27] extends [25]. Harris-Srinivasan [16] showed that the Moser-Tardos result (including Pegden’s extension) still holds for probability spaces involving permutations. This was a highly technical result. As a consequence of our results, we can also handle random permutations as a special case, using a less involved analysis.
Kolipaka-Szegedy [19] generalized the Moser-Tardos result to Shearer’s criterion, which is known to be the optimal criterion that guarantees in general the possibility of avoiding all bad events. The work of [19] assumes the independent variable model. Our analysis extends this to all scenarios of the lopsided local lemma with resampling oracles, but in a somewhat weaker sense because we require slack in Shearer’s criterion. In fact, we argue that the assumption of slack is inherent in Shearer’s criterion since it defines an open set; nevertheless, quantitatively our result gives larger running times than [19]. They also design the GeneralizedResample algorithm, which proves Shearer’s lemma in a variable-free setting. We want to point out that this algorithm involves sampling from complicated conditional distributions and does not seem to be efficiently implementable, even in simple scenarios. Nevertheless, our analysis in Section 5 incorporates several ideas from [19].

Giotis et al. [14] show that a variant of Moser’s algorithm gives an algorithmic proof of the symmetric Lovász Local Lemma [11], assuming the independent variable model. While this result is relatively limited when compared to the results above, it is intriguing in terms of techniques that are somewhat related to ours, as we discuss below. Our theorems are substantially more general in that we do not assume the variable model, and we recover the (asymmetric) Lopsided LLL without any slack. The main conceptual advance from [14] to our work is the definition of the abstract resampling oracles, a carefully modified algorithm, and a more general analysis that works even under Shearer’s criterion.

The recent work of Achlioptas and Iliopoulos [2] provides a general variable-free framework for algorithmic forms of the local lemma; it is perhaps the closest to ours in terms of versatility of applications. However, [2] does not claim a direct translation from any particular form of the Lovász local lemma to an algorithm. We defer a detailed comparison with this work to Appendix A.

1.3 Our techniques

Several exciting techniques go into the analysis of the algorithmic results mentioned above: these can be roughly categorized as the entropy method [24, 2], witness trees/sequences [25, 16, 19] and forward-looking combinatorial analysis [14] (using generating functions). Out of these, the method of witness trees is perhaps the most elegant one, as it gives an extremely clean analysis in the setting of independent random variables [25]. This method was also extended to the setting of random permutations [16]. However, this was already a considerable technical effort and it is not clear how it would generalize to a more abstract setting. The entropy method has been used successfully by [2] in the versatile setting of flaw-correcting action graphs. Nevertheless we do not see how to relate it explicitly to the Lovász Local Lemma, especially its stronger forms like Shearer’s Lemma. A very brief discussion of Shearer’s Lemma is provided in Section 5.

The combinatorial analysis of [14] is intriguing, since instead of a backward-looking argument like that of witness trees, it analyzes a forward-looking structure, the tree of resampled events as they cause one another. This seems more natural and more amenable to analysis in an abstract setting like the one we propose here. However, in [14] the analysis is limited to the symmetric LLL on a space of independent random variables.

Our approach can be roughly described as forward-looking analysis with a careful modification of the Moser-Tardos algorithm, formulated in the abstract resampling framework. Our main conceptual contribution is the simple definition of the resampling oracles, and a small modification of the Moser-Tardos algorithm (described in Section 1.1) so that the resamplings can be readily incorporated into the forward-looking analysis. Moreover, this analysis can accommodate various criteria that control the probabilities of the undesired events, including Shearer’s criterion, which plays a crucial role below. 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, 14].
Our main technical contribution is a novel analysis that overcomes this drawback, and seems interesting in its own right. We prove that, counter-intuitively, the local lemma criteria always have slack. For example, in the familiar case of Theorem 1.1, we show that there is slack even when the LLL criteria are tight, i.e., \( p_i = x_i \prod_{j \in \Gamma(i)} (1 - x_j) \) for all \( i \). More precisely, we can replace the probabilities by \( p'_i = p_i \left( 1 + (2 \sum x_j)^{-1} \right) \) and the conclusion of the local lemma still applies to the probabilities \( p'_i \). The proof crucially uses Shearer’s criterion and we do not think that it can be proven with more elementary tools \([11, 32]\). The intuition here is that Shearer’s criterion defines an open set, so it can never be tight. Hence the assumption of a small slack can be made “for free”.

Remarkably, the slack that we prove for the (GLL) criterion is equal to the bound Moser and Tardos proved on the number of resamplings of their algorithm. This suggests an intriguing connection between algorithmic convergence rate and distance to infeasibility. We feel this deserves further investigation.

2 Relationship between lopsidependency and resampling

The algorithms in this paper make no reference to the lopsidependency condition \((\text{Lop})\) and instead assume the existence of the resampling oracle. There is a close relationship between these two assumptions: the existence of a resampling oracle is equivalent to a slight strengthening of the lopsidependency condition that we state below. This stronger condition is satisfied in all uses of the lopsided local lemma that we are aware of. We denote by \( E_i[\omega] \) the \{0, 1\}-valued function indicating whether \( E_i \) occurs at a state \( \omega \in \Omega \).

**Lemma 2.1.** Consider a fixed \( i \in [n] \) and assume \( \Pr_{\mu_i}[E_i] > 0 \). The following statements are equivalent.

- There exists a resampling oracle \( r_i \) satisfying the conditions (R1) and (R2) (ignoring issues of computational efficiency).
- \( \Pr_{\mu}[E_i \land F] \geq \Pr_{\mu}[E_i] \Pr_{\mu}[F] \) for any event \( F \) such that \( F[\omega] \) is a monotone non-decreasing function of \( (E_j[\omega] : j \notin \Gamma^+(i)) \).

In particular, these statements imply that the lopsidependency condition \((\text{Lop})\) holds.

**Proof.** First, let us assume that the resampling oracle exists. Consider the coupled states \((\omega, \omega')\) where \( \omega \sim \mu_i|E_i \) 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 \( \omega \) then it does not occur at \( \omega' \) either. This establishes that

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

which implies \( \Pr_{\mu}[F \land E_i] \geq \Pr_{\mu}[F] \Pr_{\mu}[E_i] \). In particular this implies \((\text{Lop})\), 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 \( \{E_{ij} : j \notin \Gamma^+(i)\} \) 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, or bipartite flow) 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 [23]): 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 \( \Gamma(A) = \{ w \in W : \exists u \in A; (u, w) \in E \} \), 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 \( Pr_{\mu}[F \land E_i] \geq Pr_{\mu}[F] \land Pr_{\mu}[E_i] \) for every event \( F \) monotone in \( (E_j : j \notin \Gamma^+(i)) \). This is equivalent to \( Pr_{\mu}[F \land E_i] = Pr_{\mu}[E_i] \geq Pr_{\mu}[F \land E_i] \geq Pr_{\mu}[E_i] \). Assuming \( Pr[E_i] > 0 \), we can rewrite this as \( Pr_{\mu}[F | E_i] \leq Pr_{\mu}[F] \). 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 \( Pr_{\mu}[F' | E_i] = \sum_{u \in A} p_u \) and \( Pr_{\mu}[F'] = \sum_{w \in \Gamma(A)} p_w \). This verifies the sufficient condition for the existence of the resampling oracle.

### 2.1 Example: monotone events on lattices

Let us present an example of a setting where Lemma 2.1 implies the existence of a non-trivial resampling oracle, even though the lopsided association graph is empty (inspired by [21]). The probability space here is \( \Omega = \{0,1\}^M \). Let \( \mu : \{0,1\}^M \rightarrow [0,1] \) be a probability distribution over \( \{0,1\}^M \), \( \sum_{x \in \{0,1\}^M} \mu(x) = 1 \), that is log-supermodular, meaning that

\[
\mu(x \lor y) \mu(x \land y) \geq \mu(x) \mu(y) \quad \forall x, y \in \{0,1\}^M.
\]

In particular, any product distribution is log-supermodular. Consider monotone increasing events \( E_i \), i.e. such that \( x' \geq x \in E_i \Rightarrow x' \in E_i \). (The partial ordering \( x' \geq x \) here is defined by \( x'_i \geq x_i \) for all \( i \in M \).) Note that any monotone increasing function of such events is again monotone increasing. It follows from the FKG inequality that the condition of Lemma 2.1 is satisfied for such events with an empty lopsided association graph. Therefore, a resampling oracle 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 [18, Theorem 6].

**Theorem 2.2.** Let \( \mu_1 \) and \( \mu_2 \) be probability measures on \( \{0,1\}^M \) satisfying

\[
\mu_1(x \lor y) \mu_2(x \land y) \geq \mu_1(x) \mu_2(y) \quad \forall x, y \in \{0,1\}^M.
\]
Then there exists a probability distribution \( \nu : \{0,1\}^M \times \{0,1\}^M \rightarrow \mathbb{R} \) satisfying

\[
\begin{align*}
\mu_1(x) &= \sum_y \nu(x,y) \\
\mu_2(y) &= \sum_x \nu(x,y) \\
\nu(x,y) &= 0 \text{ unless } x \geq y.
\end{align*}
\]

Algorithm 2 Resampling oracle for a monotone increasing event \( E \). Let \( \nu \) be the function guaranteed by Theorem 2.2 when \( \mu_1(x) = \mu(x) \mathbb{1}_{x \in E} + \mu(x) \) 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 \( y \).

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

3 Implementation in special cases

The resampling oracles can be implemented in all LLLL scenarios of which we are aware. In the following, we present the implementation in the four main settings where the LLLL applies: for independent random variables (which was the setting of [25]), for random permutations (handled by [16]), for perfect matchings in complete graphs (some of whose applications are made algorithmic by [2]), and for spanning trees in complete graphs (which is a new scenario that we can handle).

3.1 Independent random variables

This is the most common setting, considered originally by Moser and Tardos [25]. Here, \( \Omega \) is a product space, with independent random variables \( \{X_a : a \in \mathcal{U}\} \). The probability measure \( \mu \) 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 [25]. Sampling from \( \mu \) means generating a fresh set of random variables independently. The resampling oracle \( r_i \) takes a state \( \omega \) 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 \( \mu \) conditioned on \( E_i \) has all variables outside of \( A_i \) independently random. Hence, resampling the variables of \( A_i \) produces the distribution \( \mu \). Clearly, resampling \( \{X_a : a \in A_i\} \) does not affect any events that do not intersect \( A_i \).

3.2 Random permutations

The probability space \( \Omega \) is the space of \( N \) permutations \( \pi_1, \pi_2, \ldots, \pi_N \), where \( \pi_k \) acts on a set of elements \( [n_k] \). The measure \( \mu \) here is uniform over the cartesian product of the \( N \) permutations (i.e. the permutations are independent and uniformly random). Bad events are assumed here to be “simple” in the following sense: Each bad event is defined by a set of triples \( T(E_i) = \{(k_1, x_1, y_1), \ldots, (k_{t(i)}, x_{t(i)}, y_{t(i)})\} \). The event \( E_i \) occurs if \( \pi_{k_j}(x_j) = y_j \) for each \( 1 \leq j \leq t(i) \). Let \( \nu_{bl}(E_i) = \{x : \exists y, (k, x, y) \in T(E_i)\} \).
denote the variables of πj relevant to event Ej. Let us define a relation i ∼ i′ to hold iff there are triples (k, x, y) ∈ T(Ei), (k, x′, y′) ∈ T(Ei′) such that x = x′ or y = y′; i.e., the two events entail the same value in either the range or domain of some permutation. This relation defines a lopsidedependency graph. It is known that the lopsided LLL holds in this setting.

Algorithm 3 Resampling oracle for permutations

1: Function ri(π1, π2, . . . , πN):
2: for k = 1 to N do
3: Xk := vblk(Ei), i.e., the variables in πk affecting event Ei;
4: πk := Shuffle(πk, Xk);
5: end for
6: return
7: Function Shuffle(π, X):
8: Fix an arbitrary order X = (x1, x2, . . . , xt);
9: for i = t down to 1 do
10: Swap π(xi) with π(z) for z uniformly random among [nk] \ {x1, . . . , xi−1};
11: end for
12: return π;

Harris and Srinivasan [16] showed how under the LLL criteria, a collection of permutations avoiding all bad events can be found algorithmically. We implement the resampling oracle based on their algorithm (see Algorithm 3.2). To prove the correctness of this resampling oracle within our framework, we need the following lemma.

Lemma 3.1. Suppose that a permutation π has some arbitrary fixed assignment on the variables in X, π|X = φ, and it is uniformly random among all permutations satisfying π|X = φ. Then the output of Shuffle(π, X) is a uniformly random permutation.

The procedure is known as the Fisher-Yates shuffle for generating uniformly random permutations (and was used in [16] as well). 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. Our procedure 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 = {x1, . . . , xt}. By induction, after performing the swap for xi, the permutation is uniform among all permutations with a fixed assignment of {x1, . . . , xi−1} (consistent with φ). This holds because, before the swap, the permutation was by induction uniform conditioned on the assignment of {x1, . . . , xi} being consistent with φ, and we choose a uniformly random swap for xi among the available choices. This makes every permutation consistent with φ on {x1, . . . , xi−1} equally likely after this swap.

Hence we can verify the first assumption for our resampling oracle.

Lemma 3.2. If i ∈ [n] and π1, . . . , πN are sampled independently and uniformly conditioned on the event Ei, then after calling the resampling oracle, ri(π1, . . . , πN) are independent and uniformly random.
Proof. Let $X_k = vbl_k(E_i)$. The event $E_i$ is defined by a specific assignment to the variable set $X_k$ in $\pi_k$. Lemma 3.1 proves that each $\pi_k$ is uniformly random conditioned on this event, then after calling the resampling oracle, $\pi_k$ is uniformly random (and this is done independently for each permutation). \qed

The second condition is that resampling of satisfied events does not affect non-neighbor events. This is true because of the following lemma.

**Lemma 3.3.** The resampling oracle $r_i(\pi_1, \ldots, \pi_N)$ applied to permutations satisfying $E_i$ does not cause any new event outside of $N^+(I)$.

**Proof.** Suppose $E_j$ changed its status during a call to $r_i(\pi_1, \pi_2, \ldots, \pi_N)$. This means that something changed among its relevant variables $vbl_k(E_j)$. This could happen in two ways:

1. either a variable $z \in vbl_k(E_j)$ was swapped because $z \in X_k = vbl_k(E_i)$; then clearly $j \in \Gamma^+(i)$.
2. or, a variable in $vbl_k(E_j)$, although outside of $X_k$, received a new value by a swap with some variable in $X_k = vbl_k(E_i)$. Note that in the Shuffle procedure, every time a variable $z$ outside of $X_k$ changes its value, it is by a swap with a fresh variable of $X_k$, 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. \qed

### 3.3 Perfect matchings

Here, the probability space $\Omega$ is the set of all perfect matchings in $K_{2n}$, with the uniform measure. This is a setting considered by \cite{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, $\Pr[M \in N^+] > 0$ only if $A$ is a (partial) matching. Let us define $A \sim B$ iff $A \cup B$ is not a matching. It was proved in \cite{21} that this defines a lopsidedependency graph.

Our goal is to implement a resampling oracle in this setting. We describe such an operation in Algorithm 3.3.

**Algorithm 4** Resampling oracle for perfect matchings

1. Function $r_A(M)$:
2. Check that $A \subseteq M$, otherwise return $M$.
3. $A' := A$;
4. $M' := M$;
5. While $A' \neq \emptyset$ do
6. Pick $(u, v) \in A'$ arbitrarily;
7. Pick $(x, y) \in M' \setminus A'$ uniformly at random, with $(x, y)$ randomly ordered;
8. With probability $1 - \frac{1}{2|M' \setminus A' | + 1}$, add $(u, y), (v, x)$ to $M'$ and remove $(x, y)$ from $M'$;
9. Remove $(u, v)$ from $A'$;
10. End while
11. Return $M'$.

**Lemma 3.4.** 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 $V(M' \setminus A') \cup \{u, v\}$, 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. \qed

**Lemma 3.5.** 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 $A \sim B$. \qed

### 3.4 Spanning trees

Here, the probability space $\Omega$ 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. [21] 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 General 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 $G$ (which can be done efficiently for example by a random walk).

**Algorithm 5** Resampling oracle for spanning trees

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

**Lemma 3.6.** 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 $K_n$. 

12
Finally, since the distribution of distributed as it should be in a random spanning tree restricted to $T$ in a complete graph.

Therefore, for a fixed $T$, 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 $B \sim A$. 

\end{proof}


\section{Applications}

Let us present a few applications of our framework. We believe the algorithmic aspect is new in all of them, and some are new even in the existential sense.

\subsection{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 \cite{33}. However, finding rainbow matchings algorithmically is more difficult. Achlioptas

\end{proof}

\begin{lemma}
The resampling oracle $r_A(T)$ applied to a spanning tree satisfying $E_A$ does not cause any new event $E_B$ such that $B \notin \Gamma_+^{\prime}(A)$. \end{lemma}

\begin{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 $B \sim A$. 

\end{proof}

\subsection{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 \cite{33}. However, finding rainbow matchings algorithmically is more difficult. Achlioptas

\end{proof}

\begin{lemma}
The resampling oracle $r_A(T)$ applied to a spanning tree satisfying $E_A$ does not cause any new event $E_B$ such that $B \notin \Gamma_+^{\prime}(A)$. \end{lemma}

\begin{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 $B \sim A$. 

\end{proof}

\subsection{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 \cite{33}. However, finding rainbow matchings algorithmically is more difficult. Achlioptas

\end{proof}

\subsection{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 \cite{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 $\gamma n$ edges, $\gamma < \frac{1}{2e} \approx 0.184$. Our first result is that we can do this for $\gamma = 0.21$. The improvement comes from the application of the “cluster expansion” form of the local lemma, which is still efficient in our framework.

**Theorem 4.1.** Given an edge-coloring of $K_{2n}$ where each color appears on at most $0.21n$ edges, a rainbow perfect matching exists and can be found in $O(n)$ resampling oracle calls with high probability.

In fact, we can find many disjoint rainbow matchings — up to a linear number, if we replace 0.21 above by a smaller constant.

**Theorem 4.2.** Given an edge-coloring of $K_{2n}$ where each color appears on at most $n/21$ edges, at least $n/21$ edge-disjoint rainbow perfect matchings exist and can be found efficiently.

We postpone the proof to Section 3.3 since it follows from our result for Latin transversals there.

**Proof of Theorem 4.1.** 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 3.3, 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 4.3.** For any edge-coloring of $K_{2n}$ such that every color appears at most $q = 0.21n$ times, the lopside association graph above satisfies the assumptions of Theorem 5.33 with constant slack for $p = \frac{1}{(2n-1)(2n-3)}$ and $y = 3p$ 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 $Q_v$ 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 $Q_v$ is at most $(q - 1)(2n - 1)$.

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

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

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 $Q_v$. (The event $E_{ef}$ itself is also contained in these cliques.) Therefore,

$$\sum_{I \subseteq \Gamma^+(E_{ef}), I \in \text{Ind}} y^I = \prod_{v \in e \cup f} (1 + \sum_{E_{e'f'} \in Q_v} y_{e'f'}) \leq (1 + (q - 1)(2n - 1)y)^4.$$
We assume \( q \leq \gamma n \) (\( \gamma < 0.5 \)) and \( y = \beta p = \frac{\beta}{(2n-1)(2n-3)} \). We get \( \sum_{I \subseteq \Gamma^+ (E_{ef})} y_I^f \leq (1 + \frac{1}{2} \gamma \beta)^4 \). The reader can check that with \( \beta = 3 \) and \( \gamma = 0.21 \), we have \( \frac{\beta}{(1 + \frac{1}{2} \gamma \beta)^4} > 1.003 \). Therefore,

\[
\frac{y}{\sum_{I \subseteq \Gamma^+ (E_{ef})} y_I^f} \geq \frac{\beta p}{(1 + \frac{1}{2} \gamma \beta)^4} > 1.003p
\]

which is the assumption of Theorem 5.33 with a constant \( \epsilon > 0 \) slack.

By Theorem 5.33, MaximalSetResample with the resampling oracle for matchings and the dependency graph defined above will find a rainbow perfect matching in time \( O(\sum_{E_{ef}} \log(1 + y_{ef})) = O(\sum_{E_{ef}} y_{ef}) \) 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 running time is \( O(n) \) with high probability. This proves Theorem 4.1.

### 4.2 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. Similar to rainbow matchings, rainbow spanning trees have been subject to extensive research in combinatorics. In contrast to rainbow matchings, 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 [7] 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 [3]. 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 [8]. In fact this result seems to be algorithmically efficient, although this was not claimed by the authors. We prove that using our local lemma framework, we can find \( \Omega(n) \) rainbow spanning trees under a slight strengthening of the coloring assumption. We believe that this result cannot be derived from prior work, and is in fact new even in the existential sense.

**Theorem 4.4.** Given an edge-coloring of \( K_n \) such that each color appears on at most \( n/82 \) edges, at least \( n/82 \) edge-disjoint rainbow spanning trees exist and can be found in \( O(n^2) \) resampling oracle calls with high probability.

Our approach to prove this theorem is simply to sample \( n/82 \) independently random spanning trees and hope that they will be (a) pairwise edge-disjoint, (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 3.4.

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

- \( E_{ef}^i \): For each \( i \in [t] \) and two edges \( e \neq f \) in \( K_n \) of the same color, \( E_{ef}^i \) occurs if \( \{e, f\} \subset T_i \);
- \( E_{e,j}^{ij} \): For each \( i \neq j \in [t] \) and an edge \( e \in K_n \), \( E_{e,j}^{ij} \) 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 [21], the probability of a bad event of the first type is $\Pr[E_{ef}^i] = 3/n^2$ 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 $\Pr[E_{eij}^i] = (2/n)^2 = 4/n^2$, 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 3.4 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 $E_{ef}^i$, we define $r_{ef}^i$ as an application of the resampling oracle $r_{(e,f)}$ to the tree $T_i$. For an event $E_{eij}^i$, we define $r_{eij}^i$ 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 3.6 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. Again, we are somewhat conservative for the sake of simplicity. The graph contains the following kinds of edges:

- $E_{ef}^i \sim E_{ef'}^i$, whenever $e \cup f$ intersects $e' \cup f'$;
- $E_{ef}^i, E_{ef'}^j \sim E_{eij}^i$, whenever $e'$ intersects $e \cup f$;
- $E_{eij}^i \sim E_{eij'}^i, E_{eij'}^j$ 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 5.33). Let us assume that each event of type $E_{ef}^i$ consists of: (1) events $E_{ef'}^i$, 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 $E_{eij}^i$ where $e'$ intersects $e \cup f$; again, 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 $E_{eij}^i$. The neighborhood of $E_{eij}^i$ consists of: (1) events $E_{ef'}^i$ and $E_{ef'}^j$, 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 $E_{eij}^i, E_{eij}^j$ 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 $y_{ef}^i = y_{eij}^i = y$. The cluster expansion criteria will be satisfied if we set the parameters so that

$$p \leq \frac{y}{(1 + (n-1)(t-1)y)^4(1 + (n-1)(q-1)y)^4} \leq \frac{y}{\sum_{I \subseteq T \cup \{E_i\} : E_i \in \text{Ind} y} I^f}.$$

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 $\beta = 3$
and $\gamma = 1/82$, we get $\frac{\beta}{(1 + 4\gamma \beta)^8} > 1.006$. Therefore,

$$1.006 p \leq \frac{\beta p}{(1 + 4\gamma \beta)^8} \leq \frac{y}{(1 + (n-1)(t-1)y)^4(1 + (n-1)(q-1)y)^4}$$

which verifies the assumption of Theorem 5.33 in fact with a constant slack. Theorem 5.33 implies that MaximalSetResample terminates after $O(\sum y_{ef}^i + \sum y_{ij}^i)$ resampling oracles with high probability. The total number of events here is $O(tq n^2) = O(n^4)$ and for each event the respective variable is $y = O(1/n^2)$. Therefore, the number of resampling oracles is $O(n^2)$.

\[\square\]

### 4.3 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 \[\square\]. 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 \[\square\]; this paper introduced the “cluster expansion” strengthening of the local lemma. These results were made algorithmically efficient by the work of Harris and Srinivasan \[\square\]. 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 \[\square\]: If $n = 2^k$ and each color appears in $A$ at most $\varepsilon n^2$ times ($\varepsilon = 10^{-10^{10}}$ 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 4.5.** For any $n \times n$ matrix $A$ where each color appears at most $n/21$ times, there at least exist $n/21$ disjoint Latin transversals which can be found in $O(n^2)$ resampling oracle calls w.h.p.

We note that this implies Theorem 4.2 as a special case: For an edge-coloring of $K_{2n}$ where no color appears more than $n/21$ 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 $n/21$ times, by Theorem 4.5 we can find $n/21$ Latin transversals; these correspond to rainbow matchings in $K_{2n}$.

Our approach to proving Theorem 4.5 is similar to the proof of Theorem 4.4: sample $n/21$ independently random permutations and hope that they will be (a) disjoint, (b) Latin. For similar reasons to Theorem 4.4 the local lemma works out and our framework makes this algorithmic.

**Proof.** Let $t = n/21$ and let $\pi_1, \ldots, \pi_t$ be independently random permutations on $[n]$. We consider the following two types of bad events:

- $E_{ef}^i$: 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 $E_{ef}^i$ occurs if $\pi_i(u) = v$ and $\pi_i(x) = y$;

- $E_{ij}^i$: For each $i \neq j \in [t]$ and $e = (u, v) \in [n] \times [n]$, the event $E_{ij}^i$ 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 $\Pr[E_{ef}^i] = \frac{1}{n(n-1)}$ and the probability for the second type is $\Pr[E_{ij}^i] = \frac{1}{n^2}$. Thus the probability of each bad event is at most $p = \frac{1}{n(n-1)}$. 

17
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 3.2, the resampling oracle for permutations is consistent with the following lopsided dependency graph graph.

- \( E_{i,j}^i \sim E_{i,j}^i \) whenever there is some intersection between the edges \( e, f \) and \( e', f' \);
- \( E_{i,j}^i, E_{i,j}^j \sim E_{i,j}^i \) whenever there is some intersection between \( e' \) and \( e, f \);
- \( E_{i,j}^i \sim E_{i,j}^i, E_{i,j}^j \) whenever \( e' \) intersects \( e \).

By Lemma 3.3, 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 4.4, so we skim over some details. The neighborhood of each event \( E_{i,j}^i \) consist of 8 cliques: 4 cliques of events of type \( E_{i,j}^i \) and 4 cliques of events of type \( E_{i,j}^j \), 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 \( E_{i,j}^i \) also consists of 8 cliques: 4 cliques of events \( E_{i,j}^i \) or \( E_{i,j}^j \), 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 \( E_{i,j}' \) or \( E_{i,j}'' \), 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 4.4

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

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 \( \frac{\beta}{(1 + \beta n)^4} \geq 1 \), which is achieved by \( \beta = 2.91 \) and \( \gamma = 1/21 \) (with a constant slack). Therefore, Theorem 5.33 implies that MaximalSetResample will terminate within \( O(\sum y_{i,j} + \sum y_{i,j}') = O(n^2) \) resampling oracles with high probability.

## 5 Analysis of the algorithm

We begin with a framework of stable set sequences similar to that of Kolipaka and Szegedy [19], while also borrowing some formalism from Scott and Sokal [29]. We remark that much of the machinery of [19] is useful for us, even though the way we apply it is different. While [19] follows the framework of witness trees/sequences growing backward in time (which goes back to Moser and Tardos [25]), we analyze similar sequences growing forward in time. There are certain similarities but also differences in how these two concepts lead to proofs of algorithmic efficiency. Ultimately, our analysis that proves Theorems 1.2 and 1.3 involves numerous additional ideas, particularly in Sections 5.4/5.6.

### 5.1 Notation and basic facts

**Definition 5.1.** One execution of the outer repeat loop in MaximalSetResample is called an iteration. For a sequence of non-empty sets \( I = (I_1, \ldots, I_t) \), we say that the algorithm follows \( I \) if \( I_n \) is the set resampled
in iteration $s$ for $1 \leq s < t$, and $I_t$ is a set of the first $m$ events resampled in iteration $t$ for some $m \geq 1$ (a subset of the maximal independent set constructed in iteration $t$).

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

**Definition 5.2.** $\mathcal{I} = (I_1, I_2, \ldots, I_t)$ is called a stable set sequence if $I_1, \ldots, I_t \in \text{Ind}(G)$ and $I_{s+1} \subseteq \Gamma^+(I_s)$ for each $1 \leq s < t$. We call the sequence $\mathcal{I}$ proper if each independent 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 consider an empty sequence also a stable set sequence, of length 0.

**Lemma 5.3.** 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 $r_i$ executed on a satisfied event $E_i$ can only cause new events in the neighborhood $\Gamma^+(i)$ (and this neighborhood is not explored again until the following iteration). Since $J_s$ is a maximal independent set of satisfied events, all the events satisfied in the following iteration are neighbors of some event in $J_s$, i.e., $J_{s+1} \subseteq \Gamma^+(J_s)$. 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 $\mathcal{I} = (I_1, \ldots, I_t)$, $p_{\mathcal{I}} = \prod_{s=1}^t p^{I_s}$. As with the Moser-Tardos analysis [25], a coupling argument is convenient to analyze the algorithm’s sequence of resamplings.

**Lemma 5.4.** 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 $\omega$ satisfies $E_i$; if not, we terminate. Otherwise, we apply the resampling oracle $r_i$ and replace $\omega$ 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 $\omega$ after each resampling oracle is distributed according to $\mu$: 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 $r_i(\omega)$ and by coupling the distribution in the next iteration is 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 $p_{\mathcal{I}}$. 

\[ \square \]
We emphasize that we do not claim that the distribution of the current state $\omega \in \Omega$ is $\mu$ after each resampling oracle performed by the MaximalSetResample algorithm. This would mean that the algorithm is not making any progress in its search for a state avoiding all events. It is only the $\mathcal{I}$-checking process that has this property.

**Definition 5.5.** Let $\text{Stab}$ denote the set of all stable set sequences and $\text{Prop}$ the set of proper stable set sequences. For $\mathcal{I} = (I_1, \ldots, I_t) \in \text{Prop}$, let us call $\sigma(\mathcal{I}) = \sum_{s=1}^{t} |I_s|$ the total size of the sequence.

**Lemma 5.6.** The probability that MaximalSetResample runs for at least $\ell$ iterations is upper-bounded by $\sum_{\mathcal{I}=(I_1,\ldots,I_\ell) \in \text{Prop}} p_{\mathcal{I}}$. The probability that MaximalSetResample resamples at least $s$ events is upper-bounded by $\sum_{\mathcal{I} \in \text{Prop} : \sigma(\mathcal{I})=s} p_{\mathcal{I}}$.

**Proof.** If the algorithm runs for at least $\ell$ iterations, it means that it follows some proper sequence $\mathcal{I} = (I_1, I_2, \ldots, I_\ell)$. By Lemma 5.4, the probability that the algorithm follows a particular stable set sequence $\mathcal{I}$ is at most $p_{\mathcal{I}}$. By the union bound, the probability that the algorithm runs for at least $\ell$ iterations is at most $\sum_{\mathcal{I}=(I_1,\ldots,I_\ell) \in \text{Prop}} p_{\mathcal{I}}$.

Similarly, 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 $\sum_{\mathcal{I} \in \text{Prop} : \sigma(\mathcal{I})=s} p_{\mathcal{I}}$. \hfill $\Box$

We note that these bounds could be larger than 1 and thus vacuous. The events that “the algorithm follows $\mathcal{I} = (I_1, \ldots, I_\ell)$” are disjoint for different sequences of fixed total size $\sigma(\mathcal{I})$, while they could overlap for a fixed length $\ell$ (because we can take $I_\ell$ to be different prefixes of the sequence of events resampled in iteration $t$). In any case, the upper bound of $p_{\mathcal{I}}$ on each of the events could be quite loose.

**Lemma 5.7.** The expected number of events resampled by MaximalSetResample is at most $\sum_{\mathcal{I} \in \text{Prop}} p_{\mathcal{I}}$.

**Proof.** By a standard argument,

$$
E[\text{number of resampled events}] = \sum_{s=1}^{\infty} \Pr[\text{at least } s \text{ events are resampled}].
$$

By Lemma 5.6, this is upper-bounded by $\sum_{s=0}^{\infty} \sum_{\mathcal{I} \in \text{Prop} : \sigma(\mathcal{I})=s} p_{\mathcal{I}} = \sum_{\mathcal{I} \in \text{Prop}} p_{\mathcal{I}}$. \hfill $\Box$

### 5.2 Analysis in Shearer’s setting

In this section we discuss a strong version of the local lemma due to Shearer [30]. For each $S \subseteq [n]$, define the following variants of the multivariate independence polynomial.

$$
q_S = q_S(p) = \sum_{I \in \text{Ind} : S \subseteq I} (-1)^{|I \setminus S|} p^I.
$$

(2)

Note that $q_S = 0$ for $S \notin \text{Ind}$.

**Lemma 5.8** (Shearer [30]). Let $G$ be a lopsidependency graph\(^2\) for the events $E_1, \ldots, E_n$. Let $p_i = \Pr_{\mu}[E_i] \in (0, 1)$. If $q_S \geq 0$ for all $S \subseteq [n]$ then $\Pr[\bigcap_{i=1}^{n} E_i] \geq q_\emptyset$.

\(^2\) Shearer’s work actually only discusses the scenario of dependency graphs. The fact that Shearer’s argument continues to hold for lopsidependency graphs was noted by Scott and Sokal [29].
This implies Theorem 1.1 as we will see in Section 5.4 and in fact gives the tight criterion under which all events can be avoided for a given dependency graph $G$.

Kolipaka and Szegedy showed that the stable set sequences of the previous section can be related to Shearer’s $q_S$ polynomials. We need the following identity, which we prove for completeness.

**Lemma 5.9** (Kolipaka-Szegedy [19, Lemma 15]). For any $I \in \text{Ind}$

$$q_I = p^I \sum_{S \subseteq \Gamma^+(I)} q_S.$$  

**Proof.** By definition, the right-hand-side can be expanded to

$$p^I \sum_{S \subseteq \Gamma^+(I)} q_S = p^I \sum_{S \subseteq \Gamma^+(I)} \sum_{J \in \text{Ind: } S \subseteq J} (-1)^{|J \setminus S|} p^J = p^I \sum_{J \in \text{Ind}} p^J \sum_{S \subseteq J \cap \Gamma^+(I)} (-1)^{|J \setminus S|}.$$  

Observe that if $J \cap \Gamma^+(I) \neq \emptyset$ then $\sum_{S \subseteq J \cap \Gamma^+(I)} (-1)^{|J \setminus S|} = 0$; otherwise we obtain $(-1)^{|J|}$. Therefore, we can write

$$p^I \sum_{S \subseteq \Gamma^+(I)} q_S = p^I \sum_{J \in \text{Ind: } J \cap \Gamma^+(I) = \emptyset} (-1)^{|J|} p^J.$$  

Since each independent set containing $I$ can be written as $I \cup J$ where $J$ is independent and disjoint from $\Gamma^+(I)$, the last expression enumerates exactly the independent sets containing $I$, and can be written as $\sum_{J \in \text{Ind: } J \subseteq \ell} (-1)^{|J \setminus I|} p^J$ which is the definition of $q_I$.  

This implies the following bound, which appears implicitly in [19] as Eq. (2).

**Lemma 5.10.** If $q_S \geq 0$ for all $S \subseteq [n]$ and $q_0 > 0$, then

$$\sum_{I=(I_1=J, \ldots, I_\ell) \in \text{Stab}} p_I \leq \frac{q_J}{q_0} \quad \forall J \in \text{Ind}, \forall \ell \geq 1.$$  

**Proof.** We proceed by induction: for $\ell = 1$, there is only one such stable set sequence $I = (J)$. By Lemma 5.9 we have $q_J = p^J \sum_{S \subseteq \Gamma^+(J)} q_S \geq p^J q_0$. (Recall that $q_S \geq 0$ for all $S \subseteq [n]$.) Hence, $p(J) = p^J \leq q_J/q_0$.

The inductive step:

$$\sum_{I=(I_1=J, I_2, \ldots, I_\ell) \in \text{Stab}} p_I = p^J \sum_{J' \in \text{Ind: } J' \subseteq \Gamma^+(J)} \sum_{I=(I_2=J', \ldots, I_\ell) \in \text{Stab}} p_I.$$  

By the inductive hypothesis, $\sum_{I=(I_1=J, I_2, \ldots, I_\ell) \in \text{Stab}} p_I \leq \frac{q_J}{q_0}$. Also, recall that $q_{J'} = 0$ if $J' \notin \text{Ind}$. Therefore,

$$\sum_{I=(I_1=J, I_2, \ldots, I_\ell) \in \text{Stab}} p_I \leq p^J \sum_{J' \subseteq \Gamma^+(J)} \frac{q_{J'}}{q_0} \frac{q_J}{q_0}$$  

using Lemma 5.9 to obtain the last inequality.  

Another useful fact is the following.

**Lemma 5.11.**

$$\sum_{J \in \text{Ind}} q_J = \sum_{S \subseteq [n]} q_S = 1.$$  

21
Theorem 5.14. Let \( I \) be any (inclusion-wise) maximal independent set. By definition, we have \( q_I = p^I \). Also, by Lemma 5.9

\[
1 = \frac{q_I}{p^I} = \sum_{S \subseteq \Gamma^+(I)} q_S.
\]

Note that \( \Gamma^+(I) = [n] \), otherwise we could extend \( I \) to a larger independent set. Also, \( q_S = 0 \) for \( S \notin \text{Ind} \), which proves the lemma.

Corollary 5.12. If \( q_S \geq 0 \) for all \( S \subseteq [n] \) and \( q_0 > 0 \),

\[
\sum_{I \in \text{Prop}} p_I \leq \frac{1}{q_0}.
\]

Proof. The expression \( \sum_{I=(I_1,...,I_t) \in \text{Stab}} p_I \) can be viewed as a sum over all proper sequences \( I \in \text{Prop} \) of length at most \( \ell \) (by padding by empty sets up to length \( \ell \)). Clearly, this is a non-decreasing quantity in \( \ell \). By Lemma 5.10, there is a limit \( \sum_{I=(I_1,...,I_t) \in \text{Prop}} p_I = \lim_{\ell \to \infty} \sum_{I=(I_1,...,I_t) \in \text{Stab}} p_I \leq \frac{q_J}{q_0} \). By summing over \( J \in \text{Ind} \) and applying Lemma 5.11 we obtain

\[
\sum_{I \in \text{Prop}} p_I \leq \frac{1}{q_0} \sum_{J \in \text{Ind}} q_J = \frac{1}{q_0}.
\]

Summary at this point. By Lemma 5.7 and Corollary 5.12, MaximalSetResample can produce a state in \( \bigcap_{i=1}^n E_i \) after at most \( 1/q_0 \) resamplings, in expectation. However, this should not be viewed as a statement of efficiency. Shearer’s Lemma proves that \( \Pr[\bigcap_{i=1}^n E_i] \geq q_0 \) so, in expectation, \( 1/q_0 \) independent samples from \( \mu \) would also suffice to find a state in \( \bigcap_{i=1}^n E_i \).

To summarize, it seems that we have assembled a fair amount of technical machinery, only to obtain a trivial result. However, the result can be substantially improved if we assume that Shearer’s criterion holds with some slack. This idea of exploiting slack has appeared in previous work, e.g., [25, 9, 15, 19]. The next section discusses how slack affects our analysis.

5.3 Shearer’s criterion with slack

Let us assume that Shearer’s criterion holds with some slack in the following natural sense.

Definition 5.13. We say that \( p_1, \ldots, p_n \) satisfy Shearer’s criterion with a bound of \( q_0' \) at a slack of \( \epsilon \), if \( p_I \leq (1-\epsilon)p_I' \) and the \( p_i' \) values satisfy Shearer’s criterion. (That is, \( q_S' = \sum_{I \subseteq \text{Ind}: S \subseteq I} (-1)^{|S|} p^I \geq 0 \) for all \( S \subseteq [n] \) and \( q_0' > 0 \).)

Theorem 5.14. Recall that \( p_i = \Pr_{\mu}[E_i] \). If the \( p_i \) satisfy Shearer’s criterion with a bound of \( q_0' \) at a slack of \( \epsilon \), then the probability that MaximalSetResample resamples more than \( s = \frac{1}{\epsilon}(\ln(1/q_0') + t) \) events is at most \( e^{-t} \).

Proof. By Lemma 5.6 the probability that MaximalSetResample resamples more than \( s \) events is at most \( \sum_{I \in \text{Prop}: \sigma(I) = [s]} p_I \). By the slack assumption, we have

\[
\Pr[\text{resample more than } s \text{ events}] \leq \sum_{I \in \text{Prop}: \sigma(I) = [s]} p_I \leq (1-\epsilon)^s \sum_{I \in \text{Prop}: \sigma(I) = [s]} p_I'.
\]
since we have \( p_i \leq (1 - \epsilon)p'_i \) for each event appearing in a sequence \( I \). The hypothesis is that the probabilities \( p'_i \) satisfy Shearer’s criterion. Consequently, Corollary 5.12 implies that \( \sum_{I \in \text{Prop}} p'_I \leq 1/q'_0 \). Thus, for \( s = \frac{1}{\epsilon}(\ln(1/q'_0) + t) \) we obtain

\[
\Pr[\text{resample more than } s \text{ events}] \leq (1 - \epsilon)^s/q'_0 \leq e^{-\ln(1/q'_0)+t}/q'_0 = e^{-t}.
\]

In other words, the probability that MaximalSetResample requires more than \( 1/\epsilon \ln(1/q'_0) \) resamplings decays exponentially fast; in particular the expected number of resampled events is \( O(1/\epsilon \ln(1/q'_0)) \). Typically, \( 1/q'_0 \) is a quantity that grows as a simple exponential of \( n \) and hence we can consider this an efficient running time bound. In particular, as we show next, the setting of the original Lovász Local Lemma is one example where we get an efficient algorithm.

### 5.4 Lovász Local Lemma as a special case of Shearer’s Lemma

Shearer’s Lemma is a strengthening of the general local lemma: if \( p_1, \ldots, p_n \) satisfy (GLL) then they must also satisfy Shearer’s criterion. Nevertheless, there does not seem to be a direct proof of this fact in the literature. Shearer [30] indirectly proves this fact by showing that, when his criterion is violated, it is possible that \( \Pr[\bigcap_{i=1}^n E_i] = 0 \). Scott and Sokal prove this fact using analytic properties of the partition function [29, Corollary 5.3].

We give a direct, elementary proof of this fact. These arguments will be crucial in subsequent sections to show that there is slack even when the (GLL) criteria are tight.

**Lemma 5.15** (GLL implies Shearer). Suppose that the (GLL) criterion holds. Then

\[
q_S = \sum_{I \in \text{Ind}: S \subseteq I} (-1)^{|I\setminus S|} p^I \geq 0 \quad \forall S \subseteq [n]
\]

and \( q_0 \geq \prod_{i=1}^n (1 - x_i) \).

To prove this lemma, we need some additional notation and facts.

**Definition 5.16.** Given probabilities \( p_1, \ldots, p_n \) and a lopsidedependency graph \( G \), define

\[
\tilde{q}_S = \tilde{q}_S(p) = \sum_{I \in \text{Ind}: I \subseteq S} (-1)^{|I|} p^I.
\]

**Claim 5.17** (Eq. (3.5) of Scott-Sokal [29]). For any \( a \in S \), we have

\[
\tilde{q}_S = \tilde{q}_S \setminus \{a\} - p_a \tilde{q}_{S \setminus \Gamma^+(a)}.
\]

**Proof.** Every independent set \( I \subseteq S \) either contains \( a \) or does not. In addition, if \( a \in I \) then \( I \) is independent iff \( I \setminus \{a\} \) is an independent subset of \( S \setminus \Gamma^+(a) \).

**Claim 5.18** (Eq. (2.52) of Scott-Sokal [29]). For every \( S \subseteq [n] \),

\[
\tilde{q}_S = \sum_{Y \subseteq [n] \setminus S} q_Y.
\]
Proof. The proof is similar to the proof of Lemma 5.9. By definition of $q_Y$,
\[ \sum_{Y \subseteq [n] \setminus S} q_Y = \sum_{Y \subseteq [n]} \sum_{I \in \text{Ind}: Y \subseteq I} (-1)^{|I \setminus Y|} p^I = \sum_{I \in \text{Ind}} p^I \sum_{Y \subseteq I \subseteq S} (-1)^{|I \setminus Y|}. \]
If $I \setminus S \neq \emptyset$ then the last alternating sum is zero. Therefore, the sum simplifies to
\[ \sum_{I \in \text{Ind}: I \subseteq S} (-1)^{|I|} p^I = \bar{q}_S, \]
as required. \qed

Claim 5.19 (Eq. (2.46) of Scott-Sokal [29]). For $I \in \text{Ind}$,
\[ q_I = p_I \bar{q}_{[n]\setminus \Gamma^+(I)}. \]
Proof. Directly from Lemma 5.9 and Claim 5.18.
\[ q_I = p_I \sum_{S \subseteq \Gamma^+(I)} q_S = p_I \bar{q}_{[n]\setminus \Gamma^+(I)}. \] \qed

Claim 5.20 (Simultaneous positivity of $q_S$ and $\bar{q}_S$).
\[ q_I \geq 0 \forall I \in \text{Ind} \implies \bar{q}_S \geq q_\emptyset \forall S \subseteq [n] \]
\[ \bar{q}_S \geq 0 \forall S \subseteq [n] \implies q_I \geq p_I \bar{q}_{[n]} \forall I \in \text{Ind}. \]
Proof. (3) follows from Claim 5.18 (since $q_Y = 0$ for $Y \notin \text{Ind}$). To see (4), first note that $q_I \geq 0$ for all $I \in \text{Ind}$, by Claim 5.19. Consequently, by Claim 5.18, $\bar{q}_{[n]} = \min_S \bar{q}_S$. Clearly $p_I = \min_I p_I$ too. It follows from Claim 5.19 again that $q_I = p_I \bar{q}_{[n]\setminus \Gamma^+(I)} \geq p_I \bar{q}_{[n]}$. \qed

Claim 5.21. If $p_a \leq x_a \prod_{j \in \Gamma(a)} (1 - x_j)$, then for each $S$ containing $a$, we have
\[ \bar{q}_S \geq (1 - x_a) \bar{q}_{S \setminus \{a\}}. \]
Proof. We proceed by induction on $|S|$. The base case, $S = \emptyset$, is trivial: there is no $a \in S$ to choose. Consider $S \neq \emptyset$ and an element $a \in S$. By Claim 5.17, we have $\bar{q}_S = \bar{q}_{S \setminus \{a\}} - p_a \bar{q}_{S \setminus \Gamma^+(a)}$. By the inductive hypothesis applied iteratively to the elements of $(S \setminus \{a\}) \setminus (S \setminus \Gamma^+(a)) = \Gamma(a) \cap S$, we have
\[ \bar{q}_{S \setminus \{a\}} \geq \bar{q}_{S \setminus \Gamma^+(a)} \prod_{i \in \Gamma(a) \cap S} (1 - x_i). \]
Therefore, we can write
\[ \bar{q}_S = \bar{q}_{S \setminus \{a\}} - p_a \bar{q}_{S \setminus \Gamma^+(a)} \geq \bar{q}_{S \setminus \{a\}} \left(1 - \frac{p_a}{\prod_{i \in \Gamma(a) \cap S} (1 - x_i)}\right). \]
By the claim’s hypothesis, $p_a \leq x_a \prod_{i \in \Gamma(a)} (1 - x_i) \leq x_a \prod_{i \in \Gamma(a) \cap S} (1 - x_i)$, so we conclude that $\bar{q}_S \geq (1 - x_a) \bar{q}_{S \setminus \{a\}}$. \qed

Since $\bar{q}_\emptyset = 1$, the last claim implies $\bar{q}_S \geq \prod_{i \in S} (1 - x_i)$ for all $S \subseteq [n]$. Implication (4) of Claim 5.20 then shows that $q_S \geq 0$ for all $S$. By Claim 5.19, $q_\emptyset = \bar{q}_{[n]} \geq \prod_{i=1}^n (1 - x_i)$, completing the proof of Lemma 5.15.

Our analysis of Shearer’s criterion with slack in Section 5.3 together with the fact that (GLL) implies Shearer’s criterion, immediately yields the following algorithmic result.
**Theorem 5.22.** Suppose there are \( x_1, \ldots, x_n, \epsilon \in (0, 1) \) such that \( \Pr_{\mu}[E_i] \leq (1 - \epsilon)x_i \prod_{j \in \Gamma(i)}(1 - x_j) \). 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.** Let \( \Pr_{\mu}[E_i] = p_i = (1 - \epsilon)p_i' \). Then \( p'_1, \ldots, p'_n \) satisfy (GLL). Let \( q'_S \) be Shearer’s parameter defined in (2), using \( p' \) instead of \( p \). Lemma 5.15 implies that \( q'_S \geq 0 \) for all \( S \subseteq [n] \) and \( q'_0 \geq \prod_{i=1}^{n}(1 - x_i) \).

Using the terminology of Definition 5.13, \( p'_1, \ldots, p'_n \) satisfy Shearer’s criterion with a bound of \( \prod_{i=1}^{n}(1 - x_i) \) at a slack of \( \epsilon \). By Theorem 5.14 the probability that MaximalSetResample resamples more than \( \frac{1}{\epsilon}(\ln(1/q'_0) + t) \) events is at most \( e^{-t} \). Since \( s \geq \frac{1}{\epsilon}(\ln(1/q'_0) + t) \), the proof is complete. \( \square \)

5.5 Lovász Local Lemma without a slack

Here we show that our algorithm is in fact efficient even when the (GLL) criterion is tight. This might be surprising in light of Corollary 5.12, which gives an exponential bound of \( 5^n \) when (GLL) criterion is tight. The reason that we can prove a stronger bound is that Shearer’s criterion is never tight: this criterion defines an open set, so for any probabilities \( p_1, \ldots, p_n \) satisfying that criterion, we can find slightly lower probabilities such that the criterion is still satisfied. This allows us to produce a tiny slack that makes the analysis much more efficient. The way we produce the slack is formalized in the following lemma.

**Definition 5.23.** For any lopsidedependency graph \( G \), the Shearer region is:

\[
S = \{ p = (p_1, \ldots, p_n) \in (0, 1)^n : \forall S \subseteq [n], \, \bar{q}_S(p) > 0 \} .
\]

Observe that this is indeed exactly the region where Shearer’s Lemma applies. By Claim 5.20 if \( q_0 > 0 \) and \( q_S \geq 0 \, \forall S \subseteq [n] \), then \( \bar{q}_S > 0 \) for all \( S \subseteq [n] \). Converely, if \( \bar{q}_S > 0 \) for all \( S \subseteq [n] \), then \( q_I \geq p_n \bar{q}_n \geq q_0 > 0 \) for all \( I \subseteq \text{Ind} \). This makes it clear that the Shearer region is an open set. For any \( (p_1, \ldots, p_n) \in S \), Shearer’s lemma implies that \( \Pr[\bigcap_{i \in S} \overline{E_i}] \geq \bar{q}_S(p_1, \ldots, p_n) \) (by considering Claim 5.19) and applying Shearer’s lemma to the induced subgraph on \( S \).

**Lemma 5.24.** Let \( p_1, \ldots, p_n \in (0, 1) \) satisfy (GLL). Let \( \epsilon = 1/(2 \sum_{i=1}^{n} \frac{x_i}{1-x_i}) \) and \( p'_i = (1 + \epsilon)p_i \). Then \( (p'_1, \ldots, p'_n) \) is in the Shearer region, and

\[
q'_0 = \sum_{I \in \text{Ind}} (-1)^{|I|} \prod_{i \in I} p'_i = \frac{1}{2} \prod_{i=1}^{n}(1 - x_i).
\]

In particular, this lemma implies that increasing the probabilities by a factor of \( 1 + 1/(2 \sum_{i=1}^{n} \frac{x_i}{1-x_i}) \) does not damage the conclusion of the Lovász local lemma; all events can be avoided, and in fact Shearer’s lower bound \( q_0 \) deteriorates at most by a factor of 2. The expression \( \sum_{i=1}^{n} \frac{x_i}{1-x_i} \) looks familiar — this is the running time of the Moser-Tardos algorithm [25]. What we prove here is that probabilities satisfying the LLL criteria always have a slack proportional to the inverse of this quantity.

Before we get to the proof of the lemma, let us consider the partial derivatives of \( \bar{q}_S \). Observe that these coefficients are multilinear polynomials in \( p_1, \ldots, p_n \):

\[
\bar{q}_S(p_1, \ldots, p_n) = \sum_{I \subseteq S : I \subseteq \text{Ind}} (-1)^{|I|} \prod_{i \in I} p_i .
\]
Claim 5.25. For any \( i \in S \),
\[
\frac{\partial \tilde{q}_S}{\partial p_i} = -\tilde{q}_S \setminus \Gamma^+(i)
\]
and for any \( j \in S \setminus \Gamma^+(i) \),
\[
\frac{\partial^2 \tilde{q}_S}{\partial p_i \partial p_j} = \tilde{q}_S \setminus \Gamma^+(i) \setminus \Gamma^+(j).
\]
For other choices of \( i, j \), the partial derivatives are 0. In particular, for any point in the Shearer region, \( \frac{\partial \tilde{q}_S}{\partial p_i} \leq 0 \) and \( \frac{\partial^2 \tilde{q}_S}{\partial p_i \partial p_j} \geq 0 \) (\( \tilde{q}_S(p_1, \ldots, p_n) \) is “continuous supermodular”).

**Proof.** For any \( i \in S \), we have \( \tilde{q}_S = \tilde{q}_S \setminus \{i\} - p_i \tilde{q}_S \setminus \Gamma^+(i) \) by Claim 5.17. The polynomials \( \tilde{q}_S \setminus \{i\} \) and \( \tilde{q}_S \setminus \Gamma^+(i) \) do not depend on \( p_i \) and hence \( \frac{\partial \tilde{q}_S}{\partial p_i} \) is equal to \(-\tilde{q}_S \setminus \Gamma^+(i)\). Repeating this argument one more time for \( j \in S \setminus \Gamma^+(i) \), we get \( \frac{\partial^2 \tilde{q}_S}{\partial p_i \partial p_j} = -\tilde{q}_S \setminus \Gamma^+(i) \setminus \{j\} + p_j \tilde{q}_S \setminus \Gamma^+(i) \setminus \Gamma^+(j) \). Again, \( \tilde{q}_S \setminus \Gamma^+(i) \setminus \{j\} \) and \( \tilde{q}_S \setminus \Gamma^+(i) \setminus \Gamma^+(j) \) do not depend on \( p_j \) and hence \( \frac{\partial^2 \tilde{q}_S}{\partial p_i \partial p_j} = \tilde{q}_S \setminus \Gamma^+(i) \setminus \Gamma^+(j) \).

Clearly, we have \( \frac{\partial \tilde{q}_S}{\partial p_i} = 0 \) unless \( i \in S \), and \( \frac{\partial^2 \tilde{q}_S}{\partial p_i \partial p_j} = 0 \) unless \( i \in S \) and \( j \in S \setminus \Gamma^+(i) \). Since all the coefficients \( \tilde{q}_S \) are positive in the Shearer region, we have \( \frac{\partial \tilde{q}_S}{\partial p_i} \leq 0 \) and \( \frac{\partial^2 \tilde{q}_S}{\partial p_i \partial p_j} \geq 0 \) for all \( i, j \). \( \square \)

Now we can prove Lemma 5.24.

**Proof.** Consider the line segment from \((p_1, \ldots, p_n)\) to \((p'_1, \ldots, p'_n)\) where \( p'_i = (1 + \epsilon) p_i \). Note that \( p'_i \leq (1 + \frac{1 - \epsilon}{x_i}) p_i = \frac{1}{x_i} p_i < 1 \) by the LLL criterion. Let us define
\[
\tilde{Q}_S(\lambda) = \tilde{q}_S((1 + \lambda) p_1, \ldots, (1 + \lambda) p_n).
\]
As long as \( \tilde{Q}_S(\lambda) > 0 \) for all \( S \subseteq [n] \), we are in the Shearer region and by Claim 5.25 we have
\[
\frac{d^2 \tilde{Q}_S}{d\lambda^2} = \sum_{i,j=1}^n \frac{\partial^2 \tilde{q}_S}{\partial p_i \partial p_j} p_i p_j \geq 0
\]
i.e., \( \tilde{Q}_S(\lambda) \) is a convex function.

By Claim 5.25 and Claim 5.21, for any \( i \in S \) we have
\[
\frac{\partial \tilde{q}_S}{\partial p_i} = -\tilde{q}_S \setminus \Gamma^+(i) (p_1, \ldots, p_n) \geq -\tilde{q}_S (p_1, \ldots, p_n) \cdot \prod_{i \in S \setminus \Gamma^+(i)} \frac{1}{1 - x_i}.
\]
Therefore,
\[
\frac{d \tilde{Q}_S}{d\lambda} \bigg|_{\lambda=0} = \sum_{i=1}^n p_i \frac{\partial \tilde{q}_S}{\partial p_i} \geq -\sum_{i=1}^n p_i \cdot \tilde{q}_S (p_1, \ldots, p_n) \cdot \prod_{i \in S \setminus \Gamma^+(i)} \frac{1}{1 - x_i}.
\]
Since \( p_i \leq x_i \prod_{j \in \Gamma(i)} (1 - x_j) \leq x_i \prod_{j \in S \setminus \Gamma(i)} (1 - x_j) \), this implies
\[
\frac{d \tilde{Q}_S}{d\lambda} \bigg|_{\lambda=0} \geq -\sum_{i=1}^n \frac{x_i}{1 - x_i} \cdot \tilde{q}_S (p_1, \ldots, p_n) = -\sum_{i=1}^n \frac{x_i}{1 - x_i} \cdot \tilde{Q}_S(0).
\]

26
As we argued the function $\tilde{Q}_S(\lambda)$ is convex as long as $\tilde{Q}_S(\lambda) > 0$ for all $S \subseteq [n]$. Let us suppose that this is the case in some interval $\lambda \in [0, \lambda^*]$. (We know certainly that this is true for $\lambda^* = 0$, by Lemma 5.15.) Then we have

$$Q_S(\lambda^*) - Q_S(0) = \lambda^* \frac{dQ_S}{d\lambda} \bigg|_{\lambda=0} \geq -\lambda^* \sum_{i=1}^{n} \frac{x_i}{1-x_i} \cdot Q_S(0) = -\frac{\lambda^*}{2\epsilon} Q_S(0)$$

using our choice of $\epsilon = 1/(2 \sum_{i=1}^{n} \frac{x_i}{1-x_i})$. This proves in fact that $Q_S(\lambda) > 0$ for all $S \subseteq [n]$ and $\lambda \in [0, \epsilon]$: Suppose not and take $\lambda^* \in [0, \epsilon]$ to be the supremum of $\lambda \geq 0$ such that $Q_S(\lambda) > 0$ for all $S \subseteq [n]$. By the arguments above, $Q_S(\lambda^*) \geq (1 - \frac{\lambda^*}{2\epsilon})Q_S(0) \geq \frac{1}{2}Q_S(0) > 0$. However, for any $\lambda > \lambda^*$ we have $Q_S(\lambda) \leq 0$ which contradicts the continuity of $Q_S$.

Therefore, we can conclude that $Q_S(\epsilon) \geq \frac{1}{2}Q_S(0)$. In particular,

$$q_0((1 + \epsilon)p_1, \ldots, (1 + \epsilon)p_n) = \tilde{Q}_{[n]}(\epsilon) \geq \frac{1}{2} \tilde{Q}_{[n]}(0) \geq \frac{1}{2} \prod_{i=1}^{n} (1 - x_i)$$

where we used Claim 5.18 and Lemma 5.13 in the last step. \qed

This implies our algorithmic LLL result.

**Theorem 5.26.** If there are values $x_i \in (0, 1)$ such that $\Pr_{\mu}[E_i] \leq x_i \prod_{j \in \Gamma(i)} (1 - x_j)$, then the probability that MaximalSetResample resamples more than $s = 4 \sum_{i=1}^{n} \frac{x_i}{1-x_i} (\sum_{j=1}^{n} \ln \frac{1}{1-x_j} + 1 + t)$ events is at most $e^{-t}$.

**Proof.** Directly from Theorem 5.14 applied to the probabilities $p'_i = (1 + \epsilon)p_i$, $\epsilon = \frac{1}{4 \sum_{i=1}^{n} \frac{x_i}{1-x_i}}$: By Lemma 5.24 the $p'_i$'s satisfy Shearer’s criterion with $q'_0 \geq \frac{1}{2} \prod_{i=1}^{n} \frac{1}{1-x_i}$. We have $p_i \leq (1 - \epsilon)p'_i$ (for $\epsilon \leq 1/2$, which we can assume w.l.o.g.) and so the $p_i$'s satisfy Shearer’s criterion with a slack of $\epsilon$. By Theorem 5.14 the probability that we resample more than $\frac{1}{\epsilon} (\ln \frac{1}{q'_0} + t) \leq 4 \sum_{i=1}^{n} \frac{x_i}{1-x_i} (\sum_{j=1}^{n} \ln \frac{1}{1-x_j} + 1 + t)$ events is at most $e^{-t}$. \qed

5.6 Cluster expansion

Although Shearer’s Lemma is the “optimal” form of the local lemma, Shearer’s criterion is rather unwieldy. Theorem 5.27 states a more convenient strengthening of the general local lemma criterion recently derived by Bissacot et al. [5]. An algorithmic form in the variable model was proven by Pegden [27]. In this section, we prove Theorem 1.3, which makes this result algorithmic in the general framework of resampling oracles.

**Theorem 5.27 (Bissacot et al. [5]).** Suppose that the events $E_1, \ldots, E_n$ have a dependency graph satisfying the criterion (CLL). Then $\Pr_{\mu}[\prod_{i=1}^{n} E_i] > 0$.

To see that this strengthens the general local lemma, one may verify that (GLL) implies (CLL): if $p_I \leq x_i \prod_{j \in \Gamma(i)} (1 - x_j)$, we can take $y_i = \frac{x_i}{1-x_i}$ then use the simple bound $\sum_{I \subseteq \Gamma^+(i), I \in \text{Ind}} y^I \leq \sum_{I \subseteq \Gamma^+(i)} y^I = \prod_{j \in \Gamma^+(i)} (1 + y_j)$. On the other hand, Shearer’s lemma strengthens Theorem 5.27: we will show next that (CLL) implies that Shearer’s criterion holds. This explicit connection is apparently new. (The proof in [5] is analytic and seemingly unrelated to Shearer’s.)

27
In the following, we assume that the dependency graph is connected. First, we prove the following strengthening of Lemma 5.10 (which was stated in [19], with a sketch of a proof). Recall that Stab denotes the set of all stable set sequences (possibly including empty sets). Let us denote by Stab_\ell the set of stable set sequences (\(I_1, \ldots, I_\ell\)) of length \(\ell\), and by Stab_\ell(J) the subset of Stab_\ell such that the first set in the sequence is \(J\). We recall that Stab can be identified with all proper sequences of length \(\ell\) (by padding with empty sets). Stab_0 contains exactly one sequence, the empty one \(I = ()\) (for which we have \(p(I) = 1\)).

**Lemma 5.28.** Suppose that the dependency graph \(G\) is connected. Then the following are equivalent:

- \(\forall J \in \text{Ind}; q_J > 0;\)
- \(\forall J \in \text{Ind}, \lim_{\ell \to \infty} \sum_{I \in \text{Stab}_\ell(J)} p_I\) exists and is finite.

Moreover, in this case

\[
\lim_{\ell \to \infty} \sum_{I \in \text{Stab}_\ell(J)} p_I = \frac{q_J}{q_{\emptyset}}.
\]

**Proof.** Let us define the following linear operator on \(\mathbb{R}^{\text{Ind}}\):

\[
(Mx)_I = p_I \sum_{J \subseteq \Gamma^+(I), J \in \text{Ind}} x_J.
\]

We observe that \(M\) as a matrix in \(\mathbb{R}^{\text{Ind \times Ind}}\) is nonnegative and irreducible: For any pair \(I, I' \in \text{Ind}\), there is a stable set sequence \((I_1 = I, I_2, \ldots, I_\ell = I')\). For example, we choose greedily \(I_{s+1}\) maximal in \(\Gamma^+(I_s)\), starting from vertices that have neighbors outside of \(\Gamma^+(I_s)\), as long as this is possible. This way we ensure that \(\Gamma^+(I_{s+1})\) contains \(\Gamma^+(I_s)\) and at least one additional vertex, as long as \(\Gamma^+(I_s) \neq [n]\). Eventually, we reach a set \(I_{\ell-1}\) such that \(\Gamma^+(I_{\ell-1}) = [n]\). At this point, we can set \(I_\ell = I'\) and we are done.

By the Perron-Frobenius Theorem, \(M\) has a unique eigenvector with all coordinates positive, and its eigenvalue is simple and maximal in absolute value.

We claim that \((q_I)_{I \in \text{Ind}}\) is an eigenvector of eigenvalue 1. This follows from Lemma 5.9 (recall that \(q_S = 0\) for \(S \notin \text{Ind}\))

\[
(Mq)_I = p_I \sum_{J \subseteq \Gamma^+(I), J \in \text{Ind}} q_J = p_I \sum_{S \subseteq \Gamma^+(I)} q_S = q_I.
\]

Now consider

\[
\Sigma_I = \lim_{\ell \to \infty} \sum_{I \in \text{Stab}_\ell(I)} p_I.
\]

Note that the sequence is non-decreasing: each sequence in Stab_\ell(I) can be extended by an empty set to a sequence in Stab_{\ell+1}(I). We can expand the sum in the usual way:

\[
\sum_{I \in \text{Stab}_\ell(I)} p_I = p_I \sum_{J \subseteq \Gamma^+(I), J \in \text{Ind}} \sum_{I \in \text{Stab}_{\ell-1}(J)} p_I.
\]

Therefore, if the limit as \(\ell \to \infty\) exists for all \(I \in \text{Ind}\), then

\[
\Sigma_I = \lim_{\ell \to \infty} \sum_{I \in \text{Stab}_\ell(I)} p_I = \lim_{\ell \to \infty} p_I \sum_{J \subseteq \Gamma^+(I), J \in \text{Ind}} \sum_{I \in \text{Stab}_{\ell-1}(J)} p_I = p_I \sum_{J \subseteq \Gamma^+(I), J \in \text{Ind}} \Sigma_J.
\]

In other words, \((\Sigma_I)_{I \in \text{Ind}}\) is an eigenvector of \(M\) of eigenvalue 1 as well.
If \( q_I > 0 \) for all \( I \in \text{Ind} \), then \( \Sigma_I \) is finite for each \( I \in \text{Ind} \) by Lemma 5.10, and both are positive eigenvectors with eigenvalue 1. Conversely, if \( \Sigma_I \) is finite for all \( I \in \text{Ind} \) then it forms an eigenvector of eigenvalue 1, and so does \( (q_I)_{I \in \text{Ind}} \). Therefore, in both cases the Perron-Frobenius theorem implies that the two vectors are scalar multiples of each other, both have all coordinates positive and \( \Sigma_J = \alpha q_J \) for some \( \alpha > 0 \). Since \( \Sigma_\emptyset = 1 \), it must be the case that \( \Sigma_J = q_J / q_\emptyset \) for all \( J \in \text{Ind} \).

On the other hand, \( y^J = \prod_{j \in J} y_j \) is an upper bound on this quantity.

**Lemma 5.29.** If \( p_i \leq y_i / \sum_{I \subseteq \Gamma^+(i)} p_I \) for all \( i \in [n] \), then for any fixed \( J \in \text{Ind} \) and any \( \ell \geq 1 \),

\[
\sum_{I \in \text{Stab}_\ell(J)} p_I \leq y^J.
\]

**Proof.** We proceed by induction on \( \ell \). For \( \ell = 1 \), we have \( \sum_{I \in \text{Stab}(J)} p_I = p(J) \leq y^J \) by assumption.

For the inductive step, consider \( \sum_{I \in \text{Stab}_{\ell+1}(J)} p_I \). By unrolling the first step in the sequence and applying the inductive hypothesis,

\[
\sum_{I \in \text{Stab}_{\ell+1}(J)} p_I = \sum_{L \subseteq \Gamma^+(J), L \in \text{Ind}} \sum_{I \in \text{Stab}_\ell(L)} p_I \leq \sum_{L \subseteq \Gamma^+(J), L \in \text{Ind}} y^L.
\]

We can upper-bound \( p^J \) using our assumption:

\[
p^J = \prod_{j \in J} p_j \leq \prod_{j \in J} y_j \prod_{I \subseteq \Gamma^+(j)} p_I \leq \sum_{I \subseteq \Gamma^+(J), I \in \text{Ind}} y^I
\]

because each independent subset of \( \Gamma^+(J) \) can be decomposed into \( I = \bigcup_{j \in J} I_j \) where \( I_j \) is an independent subset of \( \Gamma^+(j) \), \( j \in J \). This implies that \( \sum_{I \in \text{Stab}_{\ell+1}(J)} p_I \leq y^J \), finishing the inductive step.

**Lemma 5.30.** If \( G \) is connected and \( p_i \leq y_i / \sum_{I \subseteq \Gamma^+(i)} p_I \) for all \( i \in [n] \), then for each \( J \in \text{Ind} \), \( q_J > 0 \),

\[
y^J \geq \frac{q_J}{q_\emptyset},
\]

and for every \( S \subseteq [n] \),

\[
\bar{q}_S \leq \bar{y}_{[n]} \sum_{J \in \text{Ind} : J \subseteq [n] \setminus S} y^J.
\]

**Proof.** If \( p_i \leq y_i / \sum_{I \subseteq \Gamma^+(i)} p_I \) then by Lemma 5.29 \( \lim_{\ell \to \infty} \sum_{I \in \text{Stab}_\ell(J)} p_I \leq y^J \) for all \( J \in \text{Ind} \) (a bounded non-decreasing sequence always converges). By Lemma 5.28 \( q_J > 0 \) for all \( J \in \text{Ind} \) and \( \lim_{\ell \to \infty} \sum_{I \in \text{Stab}_\ell(J)} p_I = q_J / q_\emptyset \). Therefore, \( q_J / q_\emptyset \leq y^J \).

From here, we get by Claim 5.13 \( \bar{q}_S = \sum_{Y \subseteq [n] \setminus S} q_Y = \sum_{J \in \text{Ind} : J \subseteq [n] \setminus S} q_J \), using the fact that \( q_Y = 0 \) for \( Y \notin \text{Ind} \). Finally, \( q_J \leq q_\emptyset y^J \) using the first inequality that we proved.

In particular, recalling that \( \sum_{J \in \text{Ind}} q_J = 1 \), this lemma implies that \( \sum_{J \in \text{Ind}} y^J \geq 1 / q_\emptyset \). Hence, Shearer’s lower bound on the probability of avoiding all events in this case is \( q_\emptyset \geq 1 / \sum_{J \in \text{Ind}} y^J \). A simplified (weaker) bound is obtained by summing over all sets \( J \), which gives \( q_\emptyset \geq 1 / \sum_{J \subseteq [n]} y^J = 1 / \prod_{i=1}^n (1 + y_i) \). In the following, we want to prove that a similar lower bound holds even after increasing the probabilities slightly (similar to Lemma 5.24). First, let us prove the following inequality.
Lemma 5.31. If \( p_i \leq y_i / \sum_{I \subseteq \Gamma^+(i), I \in \text{Ind}} y_j \), then for each \( S \subseteq T \subseteq [n] \),
\[
\tilde{q}_S \leq \tilde{q}_T \sum_{J \in \text{Ind}, J \subseteq T \setminus S} y^J.
\]

Proof. Let us assume first that \( G[T] \) is connected. We simply apply the second result of Lemma 5.30 to the subgraph induced by \( T \). Note that the criteria \( p_i \leq y_i / \sum_{I \subseteq \Gamma^+(i)} y^I \) are still satisfied when restricted to an induced subgraph; the denominator can only get smaller. Also, observe that the definition of \( \tilde{q}_S = \sum_{I \in \text{Ind}, J \subseteq S} (-1)^{|I|} y^I \) does not change when we work with an induced subgraph of \( G \). In contrast, the definition of \( q_S \) involves supersets of \( S \), so we would need to be more careful when working with \( q_S \). However, the second result of Lemma 5.30 involves only \( \tilde{q}_S \) and a summation over independent subsets. Hence when applied to \( G[T] \), we can re-write it as \( \tilde{q}_S \leq \tilde{q}_T \sum_{J \in \text{Ind}, J \subseteq T \setminus S} y^J \).

Finally let us remove the assumption that \( G[T] \) is connected. For \( G[T] \) disconnected, suppose that the vertex sets of the respective components are \( T_1, \ldots, T_k \). Then the coefficient \( \tilde{q}_S \) decomposes as follows:
\[
\tilde{q}_S = \sum_{I \in \text{Ind}, J \subseteq S} p^I = \prod_{j=1}^{k} \sum_{I_j \in \text{Ind}, J \subseteq J \cap T_j} p^{I_j} = \prod_{j=1}^{k} \tilde{q}_{S \cap T_j}.
\]

We get the same expression for \( \tilde{q}_T \). Similarly, the expression \( \tilde{q}_T \) decomposes as
\[
\tilde{q}_T = \sum_{J \in \text{Ind}, J \subseteq T \setminus S} y^J = \prod_{j=1}^{k} \sum_{J_j \in \text{Ind}, J \subseteq T_j} y^{J_j}.
\]

For each \( j = 1, \ldots, k \), we have \( \tilde{q}_{S \cap T_j} \leq \tilde{q}_{T_j} \sum_{J_j \in \text{Ind}, J \subseteq T_j} y^{J_j} \) since \( G[T_j] \) is connected. Taking a product over \( j = 1, \ldots, k \), we recover the statement for \( G[T] \).

Next we prove our main result concerning the cluster expansion criterion: It always has a slack, similarly to the general LLL (compare with Lemma 5.24).

Lemma 5.32. Let \( p_1, \ldots, p_n \in (0, 1) \) satisfy the cluster expansion criterion, \( p_i \leq y_i / \sum_{I \subseteq \Gamma^+(i)} y^I \) for some \( y_i > 0 \). Let \( \epsilon = 1/(2 \sum_{i=1}^{n} y_i) \) and \( p'_i = (1 + \epsilon) p_i \). Then \( (p'_1, \ldots, p'_n) \) is in the Shearer region, and
\[
\tilde{q}'_\emptyset = \sum_{I \in \text{Ind}} (-1)^{|I|} \prod_{i \in I} p'_i \geq \frac{1}{2 \sum_{I \in \text{Ind}} y^I}.
\]

Proof. Consider the line segment from \( (p_1, \ldots, p_n) \) to \( (p'_1, \ldots, p'_n) \) where \( p'_i = (1 + \epsilon) p_i \). Note that \( p'_i \leq p_i (1 + \frac{1}{2} y_i) < p_i \sum_{I \subseteq \Gamma^+(i), I \in \text{Ind}} y^I \leq 1 \) by assumption. Let us define
\[
\tilde{Q}_T(\lambda) = \tilde{q}_T ((1 + \lambda) p_1, \ldots, (1 + \lambda) p_n).
\]

As long as \( \tilde{Q}_T(\lambda) > 0 \) for all \( T \subseteq [n] \), we are in the Shearer region and by Claim 5.25 we have
\[
\frac{d^2 \tilde{Q}_T}{d\lambda^2} = \sum_{i,j=1}^{n} \frac{\partial^2 \tilde{q}_T}{\partial p_i \partial p_j} p_ip_j \geq 0
\]
i.e., \( \tilde{Q}_T(\lambda) \) is a convex function.
By Claim \[5.25\] and Lemma \[5.31\] for any \( i \in T \) we have
\[
\frac{\partial \bar{q}_T}{\partial p_i} = -\bar{q}_{T \backslash \Gamma^+(i)}(p_1, \ldots, p_n) \geq -\bar{q}_T(p_1, \ldots, p_n) \cdot \sum_{I \subseteq \Gamma^+(i), I \in \text{Ind}} y^I.
\]
Therefore,
\[
\frac{d\bar{Q}_T}{d\lambda} \bigg|_{\lambda = 0} = \sum_{i=1}^{n} p_i \frac{\partial \bar{q}_T}{\partial p_i} \geq -\sum_{i=1}^{n} p_i \cdot \bar{q}_T(p_1, \ldots, p_n) \cdot \sum_{I \subseteq \Gamma^+(i), I \in \text{Ind}} y^I.
\]
Since \( p_i \leq y_i / \sum_{I \subseteq \Gamma^+(i), I \in \text{Ind}} y^I \), this implies
\[
\frac{d\bar{Q}_T}{d\lambda} \bigg|_{\lambda = 0} \geq -\sum_{i=1}^{n} y_i \cdot \bar{q}_T(p_1, \ldots, p_n) = -\sum_{i=1}^{n} y_i \cdot \bar{Q}_T(0).
\]

As we argued the function \( \bar{Q}_T(\lambda) \) is convex as long as \( \bar{Q}_T(\lambda) > 0 \) for all \( T \subseteq [n] \). Let us suppose that this is the case in some interval \( \lambda \in [0, \lambda^*] \). (We know certainly that this is true for \( \lambda^* = 0 \), by Lemma \[5.31\] with \( S = \emptyset \)). Then we have
\[
\bar{Q}_T(\lambda^*) - \bar{Q}_T(0) \geq \lambda^* \frac{d\bar{Q}_T}{d\lambda} \bigg|_{\lambda = 0} \geq -\lambda^* \sum_{i=1}^{n} y_i \cdot \bar{Q}_T(0) = -\frac{\lambda^*}{2 \epsilon} \bar{Q}_T(0)
\]
using our choice of \( \epsilon = 1/(2 \sum_{i=1}^{n} y_i) \). This proves in fact that \( \bar{Q}_T(\lambda) > 0 \) for all \( T \subseteq [n] \) and \( \lambda \in [0, \epsilon] \): Suppose not and take \( \lambda^* \in [0, \epsilon] \) to be the supremum of \( \lambda \geq 0 \) such that \( \bar{Q}_T(\lambda) > 0 \) for all \( T \subseteq [n] \). By the arguments above, \( \bar{Q}_T(\lambda^*) \geq (1 - \frac{\lambda^*}{2 \epsilon}) \bar{Q}_T(0) \geq \frac{1}{2} \bar{Q}_T(0) > 0 \). However, for any \( \lambda > \lambda^* \) we have \( \bar{Q}_T(\lambda) \leq 0 \) which contradicts the continuity of \( \bar{Q}_T \).

Therefore, we can conclude that \( \bar{Q}_T(\epsilon) \geq \frac{1}{2} \bar{Q}_T(0) \). In particular,
\[
q_0((1 + \epsilon)p_1, \ldots, (1 + \epsilon)p_n) = \bar{Q}_{[\epsilon]}(\epsilon) \geq \frac{1}{2} \bar{Q}_{[\epsilon]}(0) = \frac{1}{2 \sum_{I \in \text{Ind}} y^I}
\]
where we used Claim \[5.18\] and Lemma \[5.31\] in the last step. \( \Box \)

This implies our algorithmic cluster expansion result. The following theorem subsumes Theorem \[1.3\] and adds a statement under the assumption of slack.

**Theorem 5.33.** Suppose that the events satisfy the \((\text{Cll})\) criterion. Then, with probability at least \( 1 - e^{-t} \), \( \text{MaximalSetResample} \) resamples no more than
\[
4 \left( \sum_{i=1}^{n} y_i \right) \left( \sum_{j=1}^{n} \ln(1 + y_j) + 1 + t \right) \text{ events.}
\]
If \((\text{Cll})\) is satisfied with an \( \epsilon \) slack, i.e., \( p_i \leq (1 - \epsilon)y_i / \sum_{I \subseteq \Gamma^+(i), I \in \text{Ind}} y^I \), then with probability at least \( 1 - e^{-t} \), \( \text{MaximalSetResample} \) resamples no more than
\[
\frac{1}{\epsilon} \left( \sum_{j=1}^{n} \ln(1 + y_j) + 1 + t \right) \text{ events.}
\]
Proof. Consider first the statement under the assumption of an $\epsilon$ slack. Let $p_i = (1 - \epsilon)p'_i$. By Lemma 5.31 and the discussion thereafter, the $p'_i$ satisfy Shearer’s criterion with $q'_0 \geq 1/\prod_{i=1}^n (1 + y_i)$. By Theorem 5.14, the probability that MaximalSetResample resamples more than $\frac{1}{\epsilon}(\ln \frac{1}{q'_0} + t) = \frac{1}{\epsilon} \left( \sum_{i=1}^n \ln (1 + y_i) + t \right)$ events is at most $e^{-t}$.

For the statement without slack, we apply Theorem 5.14 to the probabilities $p'_i = (1 + 2\epsilon)p_i$, $\epsilon = 1/(4 \sum_{i=1}^n y_i)$: By Lemma 5.32, the $p'_i$ satisfy Shearer’s criterion with $q'_0 \geq 1/(2 \sum_{I \in \text{Ind} y^I} y^I) \geq 1/(2 \prod_{i=1}^n (1 + y_i))$. We have $p_i \leq (1 - \epsilon)p'_i$ (for $\epsilon \leq 1/2$, which we can assume w.l.o.g.) and so the $p_i$ satisfy Shearer’s criterion with a slack of $\epsilon$. By Theorem 5.14, the probability that we resample more than $\frac{1}{\epsilon}(\ln \frac{1}{q'_0} + t) \leq 4(\sum_{i=1}^n y_i)(\sum_{j=1}^n \ln (1 + y_j) + 1 + t)$ events is at most $e^{-t}$.\]

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] S. Akbari and A. Alipour. Multicolored trees in complete graphs. J. Graph Theory, 54:3:221–232, 2007.

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

[5] 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.

[6] 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.

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

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

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

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

[11] Paul Erdős 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.

[12] Paul Erdős and Joel Spencer. The Lopsided Lovász Local Lemma and Latin transversals. Discrete Applied Mathematics, 30:151–154, 1991.
[13] H. Gebauer, T. Szabó, and G. Tardos. The local lemma is tight for SAT. In Proceedings of SODA, 2011.

[14] 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.

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

[16] 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.

[17] Nicholas J. A. Harvey. A note on the discrepancy of matrices with bounded row and column sums. Discrete Mathematics, 338:517–521, 2015.

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

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

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

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

[22] Austin Mohr. Applications of the lopsided Lovász local lemma regarding hypergraphs. PhD thesis, 2013.

[23] Michael Molloy and Bruce Reed. Graph Coloring and the Probabilistic Method. Springer, 2002.

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

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

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

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

[28] A. Schrijver. Combinatorial Optimization: Polyhedra and Efficiency. Springer, 2004.

[29] Alex Scott and Alan Sokal. The Repulsive Lattice Gas, the Independent-Set Polynomial, and the Lovász Local Lemma, 2004.

[30] James B. Shearer. On a problem of Spencer. Combinatorica, 5(3), 1985.
[31] Joel Spencer. Ramsey’s theorem: a new lower bound. *Journal of Combinatorial Theory (A)*, 18:108–115, 1975.

[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.
A Comparison to the framework of Achlioptas and Iliopoulos

Achlioptas and Iliopoulos [2] propose an algorithmic framework that abandons the probabilistic setup of the Lovász local lemma, and yet can derive efficient algorithms for finding “flawless objects” akin to those guaranteed by the local lemma. Their work [2] does not formally claim to make any particular form of the local lemma algorithmic but a conceptual connection is nevertheless present in their paper.

Since their formal setup of is somewhat lengthy to define, we refer the reader to their publication [2] for the necessary definitions. Briefly, their framework is defined on a space of possible states \( \Omega \), similar to our space \( \Omega \) but without any underlying probability distribution. It requires the existence of a certain digraph \( D_f \) of “actions” for each flaw \( f \) (corresponding to fixing a bad event \( E_f \)). The digraph \( D_f \) should have at least \( A_f \) outgoing arcs for each state satisfying a flaw \( f \), and at most one incoming arc for each state in \( \Omega \) (the “atomicity” property). Given these digraphs \( D_f \), a causality relationship is defined on the set of flaws, so that \( f \rightarrow g \) iff some arc in \( D_f \) leads from a state without flaw \( g \) to a state with flaw \( g \). We also have \( f \rightarrow f \) if some arc in \( D_f \) leads from a state with flaw \( f \) back to \( f \). The neighborhood \( \Gamma(f) \) is defined to contain all the flaws \( g \) such that \( f \rightarrow g \) (in contrast to our framework, the notion of being a neighbor is not necessarily symmetric here). The main result of [2] is that if this setup is amenable in the sense that

\[
\sum_{g \in \Gamma(f)} \frac{1}{A_g} < \frac{1}{e} \tag{5}
\]

then there is a randomized algorithm (a random walk) that finds a flawless object efficiently.

To relate this to the Lovász local lemma, it is instructive to consider the following “tight case”: Each state with flaw \( f \) has exactly \( A_f \) outgoing arcs in \( D_f \), and each state \( \omega \in \Omega \) has exactly one incoming arc. In this case, it is natural to consider a uniform distribution on \( \Omega \), let \( E_f \) denote the event of satisfying flaw \( f \), and define \( \Pr[E_f] = |f|/|\Omega| \). A counting argument reveals that \( \Pr[E_f] = 1/A_f \). Therefore, the assumption (5) can be interpreted as

\[
\sum_{g \in \Gamma(f)} \Pr[E_g] < \frac{1}{e} \tag{6}
\]

which is the “asymmetric” LLL criterion of Spencer [31] [23, pp. 221] (with an improved constant). This criterion is of intermediate strength between the symmetric and general LLL criteria, and hence cannot recover all results proven using the general LLL. For example, the result of [17] requires the full power of the general LLL.

In fact, in this case \( D_f \) can be viewed as a representation of a resampling oracle: \( r_f(\omega) \) would simply output the endpoint of a random arc originating from \( \omega \). It is easy to see that given a random state \( \omega \) conditioned in \( E_f \), \( r_f(\omega) \) outputs a uniformly random state in \( \Omega \). Also, \( r_f \) does not cause any new flaw outside of the neighborhood \( \Gamma(f) \). Therefore, our framework applies in this setting (not only under the assumption (6) but also under the general LLL assumption and its stronger forms up to Shearer’s criterion).

What goes in favor of the framework of [2] in our opinion is that:

- There is no need to sample from a special probability distribution \( \mu \); any starting point suffices.
- The framework is not restricted to the setting of the Lovász Local Lemma (and indeed some applications in [2] are outside of its scope).

At the same time, this versatility “defers more work to the user” — the framework requires the construction of a certain digraph of actions but there is no indication in [2] how the user should design the action digraph for a given application, and whether its existence is related to the Lovász Local Lemma. We believe that our framework has several advantages:

35
• We provide a direct connection between any application of the LLL with lopsided association, and an algorithmic implementation. In particular, we explain when a resampling oracle exists, by relating it to the lopsided association conditions.
• Our algorithm is efficient under stronger forms of the LLL that provide improved quantitative results in applications.
• The probabilistic nature of resampling oracles makes it convenient to leverage known probabilistic techniques in the implementation. For example, our resampling oracle for spanning trees uses several properties of random spanning trees, and a known algorithm for generating them. We were not able to handle this setting with the action graph framework described in [2].

To summarize, we feel that the probabilistic setup of the local lemma is very convenient in scenarios that require correcting flaws and avoiding undesired events. It is natural to design algorithms for problems of this nature with a probabilistic viewpoint, much like it is natural to tackle their existential variants via the probabilistic method.