A new notion of commutativity for the algorithmic Lovász Local Lemma

David G. Harris
University of Maryland, College Park
davidgharris29@gmail.com

Fotis Iliopoulos*
Institute for Advanced Study
fotios@ias.edu

Vladimir Kolmogorov†
Institute of Science and Technology Austria
vnk@ist.ac.at

February 16, 2021

Abstract

The Lovász Local Lemma (LLL) is a powerful tool in probabilistic combinatorics which can be used to establish the *existence* of objects that satisfy certain properties. The breakthrough paper of Moser and Tardos and follow-up works revealed that the LLL has intimate connections with a class of stochastic local search algorithms for finding such desirable objects. In particular, it can be seen as a sufficient condition for this type of algorithms to converge fast.

Besides conditions for convergence, there are other natural questions one may ask of these algorithms. For instance, "are they parallelizable?", "how many solutions can they output?", "what is the expected 'weight' of a solution?", etc. These questions and more have been answered for a class of LLL-inspired algorithms called commutative. In this paper we introduce a new, very natural and more general notion of commutativity (essentially matrix commutativity) which allows us to show a number of new refined properties of LLL-inspired local search algorithms with significantly simpler proofs.

1 Introduction

The Lovász Local Lemma (LLL) is a powerful tool in probabilistic combinatorics which can be used to establish the *existence* of objects that satisfy certain properties [7]. At a high level, it states that given a collection of bad events in a probability space μ, if each bad-event is not too likely and, further, is independent of most other bad events, then the probability of avoiding all of them is strictly positive.

In its simplest, "symmetric" form, it states that if each bad-event has probability at most p and is dependent with at most d others, where $epd \leq 1$, then with positive probability no bad-events become true. In particular, a configuration avoiding all the bad-events exists. Although the LLL applies to general probability spaces, most constructions in combinatorics use a simpler setting we refer to as the *variable version LLL*. Here, the probability space μ is a cartesian product with n independent variables, and each bad-event is determined by a subset of the variables. Two bad-events conflict if they depend on a common variable.

*This material is based upon work directly supported by the IAS Fund for Math and indirectly supported by the National Science Foundation Grant No. CCF-1900460. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. This work is also supported by the National Science Foundation Grant No. CCF-1815328.

†Supported by the European Research Council under the European Unions Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement no 616160

For example, consider a CNF formula with n variables in which each clause has k literals and each variable appears in at most L clauses. For each clause c we can define the bad event B_c that c is violated in a chosen assignment of the variables. For a uniformly random assignment of the variables, each bad-event has probability $p = 2^{-k}$ and affects at most $d \leq kL$ others. So when $L \leq \frac{2^k}{ek}$, the formula is satisfiable. Note that, crucially, this criterion does not depend on the value n.

A generalization known as the Lopsided LLL (LLLL) allows bad-events to be *positively* correlated with others; this is as good as independence for the purposes of the LLL. Some notable probability spaces satisfying the LLLL include the uniform distribution on permutations and the variable setting, where two bad-events B, B' are dependent only if they *disagree* on the value of a common variable.

In a seminal work, Moser and Tardos [22] presented a simple local search algorithm to make the variable-version LLLL constructive. This algorithm can be described as follows:

Algorithm 1 The Moser-Tardos resampling algorithm

1: Draw the state σ from distribution μ
2: **while** some bad-event B is true on σ **do**
3: Select, arbitrarily, a bad-event B true on σ
4: Resample, according to distribution μ, all variables in σ affecting B

Moser & Tardos showed that if the symmetric LLL criterion (or more general asymmetric criterion) is satisfied, then this algorithm quickly converges. Following this work, a large effort has been devoted to making different variants of the LLL constructive. This research has taken many directions, including further analysis of Algorithm 1 and its connection to different LLL criteria [4, 19, 20, 23].

One particular line of research has been to use this type of local search algorithm and variants for general probability spaces beyond the variable LLL. These include random permutations or random matchings of the complete graph [1, 2, 14, 18, 16], as well as settings which are not directly connected to the LLL itself [3, 5, 15]. At a high level of generality, we summarize this in the following framework. There is a discrete state space Ω, with a collection $\mathcal{F}$ of subsets (which we sometimes call *flaws*) of Ω. We also have some problem-specific randomized procedure called the *resampling oracle* $\mathfrak{R}_f$ for each flaw; it takes some random action to attempt to "fix" that flaw, resulting in a new state $\sigma' \leftarrow \mathfrak{R}_f(\sigma)$. With these ingredients, we define the general local search algorithm as follows:

Algorithm 2 The general local search algorithm

1: Draw the state σ from some distribution μ
2: **while** some flaw f holds on σ **do**
3: Select a flaw f of σ, according to some rule $\mathcal{S}$.
4: Update $\sigma \leftarrow \mathfrak{R}_f(\sigma)$.

Besides conditions for convergence to flawless objects, one may naturally ask further questions regarding properties of Algorithm 2. For instance, "is it parallelizable?", "how many solutions can it output?", "what is the expected 'weight' of a solution?", etc. These questions and more have been answered for the Moser-Tardos (MT) algorithm in a long series of papers [4, 6, 8, 9, 11, 13, 19, 22]. As a prominent example, the result of Haeupler, Saha and Srinivasan [9], as well as follow-up work of Harris and Srinivasan [12, 13], allows one to argue about the dynamics of the MT process, resulting in several new applications such as estimating the entropy of the output distribution, partially avoiding bad events and dealing with super-polynomially many bad events.

We note one important difference between Algorithm 1 and Algorithm 2: the choice of which flaw to resample, if multiple flaws are simultaneously true. The flaw selection rule $\mathcal{S}$ should s select a flaw $f \ni \sigma$

for the state σ at time t; it may depend on the prior states and may be randomized.

The original MT algorithm allows nearly complete freedom for $\mathcal{S}$. For general local search algorithms, this is much more constrained, in general; only a few relatively rigid rules are known to converge, such as selecting the flaw of least index [16]. However, in [21], Kolmogorov identified a property of resampling oracles that allows a free choice for resampling rule called *commutativity*. This free choice for $\mathcal{S}$, while seemingly a minor detail, turns out to play a key role in extending the additional algorithmic properties of the MT algorithm to the general setting of Algorithm 2. For instance, it leads to parallel algorithms [21] and to bounds on the output distribution [17].

Our main contribution is to introduce a notion of commutativity, essentially matrix commutativity, that is both more general and simpler than the definition in [21]. This will provide a streamlined and high-level explanation for a variety of results and bounds involving distributional properties of Algorithm 2. Most of these results had already been shown, in slightly weaker forms, in a variety of prior works such as [21, 17, 12]. However, the proofs were computationally heavy and narrowly targeted to certain probability spaces, with numerous technical side conditions and restrictions. It was not clear how these results could be combined or how they related to each other.

Before we provide formal definitions, let us give some intuition. For each flaw f, consider an $|\Omega| \times |\Omega|$ transition matrix A_f. Each row of A_f describes the probability distribution obtained by resampling f at a given state σ. We call the algorithm *T-commutative* if the transition matrices commute for any pair of flaws which are "independent" (in the LLL sense). We show a number of results for T-commutative algorithms:

1. We obtain bounds on the distribution of the state at the termination of Algorithm 2. In many cases, this distribution approximates the *LLL-distribution*, i.e., the distribution induced by conditioning on avoiding all bad events. The bounds are similar those shown in [17] for general commutative resampling oracles. However, they are more general and avoid a number of technical conditions. Furthermore, the proofs are much simpler.

2. For some probability spaces, stronger and specialized distributional bounds are available, which go beyond the "generic" LLL bounds. Some examples were shown for the permutation setting [12] and the variable-version LLLL [12]. Previously, these bounds had been shown with ad-hoc arguments specialized to each probability space. Our construction recovers most of these results automatically.

3. We develop a generic parallel implementation of Algorithm 2. This extends results on parallel algorithms of [21, 11]. Again, the results are more general and have simpler proofs.

4. In many settings, flaws are formed from smaller "atomic" events [11] – for example, in the permutation setting, events of the form $\pi x = y$. We show that, if the atomic events satisfy the generalized commutativity definition, then so do the larger "composed" events. Due to some technical restrictions, this natural property did not seem to hold for the original commutativity definition of [21].

1.1 Overview of our approach

Although it will require significant definitions and technical development to state our results formally, let us try to provide a high-level summary here. As a starting point, consider Algorithm 1. One of the main techniques introduced by Moser & Tardos [22] to analyze this algorithm was a construction referred to as a *witness tree*. For each resampling of a bad-event B at a given time, they generate a corresponding witness tree which records an "explanation" of why B was true at that time. More properly, this witness tree provides a history of all the prior resampling which affected the variables involved in B.

The main technical lemma governing the behavior of the MT algorithm is the "Witness Tree Lemma," which states that the probability that a given witness tree is produced is at most the product of the probabilities of the corresponding events. The bound on the algorithm runtime, as well as parallel algorithms and

distributional properties, then follows by taking a union bound over witness trees. This infinite sum can be bounded using techniques similar to the analysis of Galton-Watson processes.

Versions of this Witness Tree Lemma have been shown for some variants of the MT algorithm [10, 15] Iliopoulos [17] further showed that it held for general spaces which satisfy the commutativity property; this, in turn, leads to the nice algorithmic properties such as parallel algorithms.

Our main technical innovation is to generalize the Witness Tree Lemma, in two distinct ways. First, instead of keeping track of a *scalar* product of probabilities in a witness tree, we instead consider a *matrix product.* We bound the probability of a given witness tree (or, more properly, a slight generalization known as the witness DAG) in terms of the products of the transition matrices of the corresponding flaws. Commutativity can thus be rephrased and simplified in terms of *matrix commutativity* for the transition matrices.

Second, we change the criterion for when to add nodes to the witness tree. In the construction of Moser & Tardos, the general rule is to add a node for flaw f if the tree already included a node corresponding to some later-resampled flaw g which is dependent with f. In our construction, we only add the new node f if it can increase some transition probabilities corresponding to the tree. This is a strictly more restrictive criterion, and leads to more "compressed" or concise explanations of the resamplings. This in turn leads to improved convergence bounds as well as simpler, unified proofs.

At the end, we obtain the scalar form of the Witness Tree Lemma by projecting everything to a one-dimensional space. For this, we take advantage of some methods of [3] for viewing the evolution of Algorithm 2 in terms of spectral bounds.

1.2 Outline of the paper

In Section 2, we provide basic definitions for resampling oracles and for analyzing the trajectories in them. In particular, in Section 2.1, we provide our new matrix-based definition for commutativity. In addition to being more general than the previous definitions, it also is easier to work with algebraically.

In Section 3, we define the witness DAG following [8]. We show that the probability of producing a given witness DAG is bounded in terms of the products of the transition matrices of the flaws it contains.

In Section 4, we show how to project this matrix bound to get useful probabilistic bounds on the behavior of Algorithm 2. We also relate it to standard criteria such as the symmetric or asymmetric LLL.

In Section 5, we show that the new commutativity definition leads to bounds on the distribution at the termination of Algorithm 2.

In Section 6, we show that, if there is slack in the LLL conditions, then resampling process is likely to have low depth. As a consequence, parallel algorithms can be used to implement Algorithm 2.

In Section 7, we consider a construction for building resampling oracles out of smaller "atomic" events.

2 Background and Basic Definitions

Throughout the paper we consider implementations of Algorithm 2. For each flaw f, state $\sigma \in f$, and state $\sigma' \in \Omega$, we define $A_f[\sigma, \sigma']$ to be the probability that applying the resampling oracle $\mathfrak{R}_f$ to σ yields state σ', i.e.

$$A_f[\sigma, \sigma'] = \Pr(\mathfrak{R}_f(\sigma, r) = \sigma')$$

For $\sigma \notin f$, we define $A_f[\sigma, \sigma'] = 0$. We sometimes write $\sigma \xrightarrow{f} \sigma'$ to denote that the algorithm resamples flaw f at σ and moves to σ'.

We define a *trajectory* T to be a finite or countably infinite sequence of flaws $(f_1, f_2, \dots,)$, and $\mathrm{length}(T)$ is its *length* (possibly $\mathrm{length}(T) = \infty$). For an execution of Algorithm 2, we define the *actual trajectory* $\hat{T}$ to be $(f_1, f_2, \dots,)$ where f_i is the flaw selected for resampling at time i. Note that $\mathrm{length}(\hat{T})$ is the

number of iterations executed by the search algorithm; when the algorithm does not terminate, we have $\text{length}(\hat{T}) = \infty$.

The key to analyzing Algorithm 2 is to keep track of the possible ways in which resampling certain flaws f can cause other flaws g. For our purposes, we use an *undirected* notion of dependence. Formally, we suppose that we have a symmetric relation $\sim$ on Ω, with the property that $f \sim f$ for all f and for every distinct pair of flaws $f \not\sim g$, we are guaranteed that resampling flaw f cannot introduce g or vice-versa, i.e. $\mathfrak{R}_f$ never maps a state $\Omega - g$ into g and likewise $\mathfrak{R}_g$ never maps a state from $\Omega - f$ into f.

We remark that some previous analyses of local search algorithms [2] have used a directed notion of causality or have further conditions of when $f \sim f$ for a flaw f. These can sometimes give more precise bounds, but they are not directly compatible with our definitions and framework.

For an arbitrary event $E \subseteq \Omega$, we define e_E to be the indicator vector for E, i.e. $e_E[\sigma] = 1$ if $\sigma \in E$ and $e_E\sigma = 0$ otherwise. For a state $\sigma \in \Omega$, we write e_σ as shorthand for $e_{\{\sigma\}}$, i.e. the basis vector which has a 1 in position σ and zero elsewhere. Note that, with this notation, $e_\sigma^\top A_f$ is the vector representing the probability distribution obtained by resampling flaw f at state σ.

For flaw f, we define $\overline{\Gamma}(f)$ to be the set of flaws g with $f \sim g$, and we also define $\Gamma(f) = \overline{\Gamma}(f) \setminus \{f\}$. We say that a set $I \subseteq \mathcal{F}$ is *stable* if $f \not\sim g$ for all distinct pairs $f, g \in I$.

For vectors u, v we write $u \preceq v$ if $u[i] \leq v[i]$ for all entries i.

Regenerating oracles. The original Moser-Tardos algorithm, and extensions to other probability spaces, can be viewed in terms of *regenerating oracles* [16], i.e. each resampling action $\mathfrak{R}_f$ should convert the distribution of μ conditioned on f into the unconditional distribution μ. We provide more detail later in Section 4, but, we can summarize this crisply with our matrix notation: the resampling oracle $\mathfrak{R}$ is regenerating if μ is a left-eigenvector of each matrix A_f, with associated eigenvalue $\mu(f)$, i.e.

$$\forall f \qquad \mu^\top A_f = \mu(f) \cdot \mu^\top \tag{1}$$

2.1 The new commutativity definition

The original definition of commutativity given by Kolmogorov [21] required that for every $f \nsim g \in \mathcal{F}$, there is an injective mapping from state transitions of the form $\sigma_1 \xrightarrow{f} \sigma_2 \xrightarrow{g} \sigma_3$ to state transitions of the form $\sigma_1 \xrightarrow{g} \sigma_2' \xrightarrow{f} \sigma_3$, so that $A_f[\sigma_1, \sigma_2]A_g[\sigma_2, \sigma_3] = A_g[\sigma_1, \sigma_2']A_f[\sigma_2', \sigma_3]$.

This definition is cumbersome to use, as well as lacking important symmetry and invariance properties. As one of the major contributions of this paper, we introduce a more natural notion of algorithmic commutativity that is also more general than the notion of [21].

Definition 2.1 (Transition matrix commutativity)**.** *We say that the resampling oracle is* transition matrix commutative *(abbreviated* T-commutative*) with respect to dependence relation $\sim$ on $\mathcal{F}$ if $A_f A_g = A_g A_f$, for every $f, g \in \mathcal{F}$ such that $f \nsim g$.*

Observation 2.2. *If the resampling oracle is commutative in the sense of [21], then it is T-commutative.*

Proof. Consider $f \not\sim g$ and states σ, σ'. By symmetry, we need to show that $A_f A_g[\sigma, \sigma'] \leq A_g A_f[\sigma, \sigma']$. Since $f \not\sim g$, we can see that both the LHS and RHS are zero unless $\sigma \in f \cap g$.

Let V denote the set of states σ'' with $A_f[\sigma, \sigma'']A_g[\sigma'', \sigma'] > 0$. By the definition of [21], there is an injective function $F : V \to \Omega$ such that $A_f[\sigma, \sigma'']A_g[\sigma'', \sigma'] = A_g[\sigma, F(\sigma'')]A_f[F(\sigma''), \sigma']$. Therefore, we have

$$(A_f A_g)[\sigma, \sigma'] = \sum_{\sigma'' \in V} A_f[\sigma, \sigma'']A_g[\sigma'', \sigma'] = \sum_{\sigma'' \in V} A_g[\sigma, F(\sigma'')]A_f[F(\sigma''), \sigma']$$

Since function F is injective, each term of the form $A_g[\sigma, \tau]A_f[\tau, \sigma']$ is counted at most once in this sum with $\tau = F(\sigma'')$. So $(A_f A_g)[\sigma, \sigma'] \leq \sum_{\tau \in f} A_g[\sigma, \tau]A_f[\tau, \sigma'] = (A_g A_f)[\sigma, \sigma']$. □

We remark that dependency information can in fact be recovered from knowledge of the transition matrices themselves:

Observation 2.3. *The relation defined by $f \sim g$ iff $f = g$ or $A_f A_g \neq A_g A_f$ gives a T-commutative dependency relation for the resampling oracle.*

Proof. Consider a pair of distinct flaws f, g such that flaw f causes flaw g. There must be a transition $\sigma \xrightarrow{f} \tau$ such that $\sigma \notin g$ and $\tau \in g$. Since matrices A_f, A_g have non-negative entries, this implies that $A_f A_g[\sigma, \tau] > 0$, while $A_g A_f[\sigma, \tau] = 0$. Thus, $A_f A_g \neq A_g A_f$. □

When this definition applies, we define A_I to be the matrix $\prod_{f \in I} A_f$ for a stable set I; note that this product is well-defined (without specifying ordering of I) since the matrices A_f all commute.

For the remainder of this paper, we assume that the resampling oracle $\mathfrak{R}$ is T-commutative unless explicitly stated otherwise.

3 Witness DAGs and matrix bounds

In this section we study *witness DAGs*, a key graph structure developed in [8] for analyzing the evolution of commutative resampling oracles. At a high level, the role of a witness DAG is to give an "explanation" of why a certain (not necessarily bad) event E appeared during the execution of the algorithm. That is, if we want to bound the probability that E will ever appear during the algorithm execution, we simply add up the probabilities of appearance of all the witness DAGs that explain it.

Formally, consider a directed acyclic graph G, where each vertex $v \in G$ has a label $L(v)$ from the set $\mathcal{F}$. We say that G is a *witness DAG* (abbreviated *wdag*) if it satisfies the property that for all pairs of vertices $v, w \in G$, there is an edge between v and w (in either direction) if and only if $L(v) \sim L(w)$. For a wdag G with sink nodes $v_1, \ldots, v_k$, note that $L(v_1), \ldots, L(v_k)$ are all distinct and $\{L(v_1), \ldots, L(v_k)\}$ is a stable set which we denote by $\mathrm{sink}(G)$. We say that a flaw f is *unrelated* to a wdag G if there is no node $v \in G$ with $L(v) \sim f$.

There is a key connection between wdags and the transition matrices, which we define as follows. For any wdag H, we define an associated $|\Omega| \times |\Omega|$ matrix A_H inductively as follows. If $H = \emptyset$, then A_H is the identity matrix on Ω. Otherwise, we choose an arbitrary source node v of H and set $A_H = A_{L(v)} A_{H-v}$.

Proposition 3.1. *The definition of A_H does not depend on the chosen source node v. Furthermore, there is an enumeration of the nodes of H as $v_1, \ldots, v_t$ such that $A_H = \prod_{i=1}^{t} A_{L(v_i)}$.*

Proof. We show this by induction on $|H|$. When $|H| = 0$ this is vacuously true.

For the second property, we have $A_H = A_v A_{H-v}$ for a source node v. By induction hypothesis, there is an ordering $v_2, \ldots, v_t$ of $H - v$ such that $A_{H-v} = A_{L(v_2)} \ldots A_{L(v_t)}$. Setting $v_1 = v$, we have $A_H = A_{L(v_1)} \ldots A_{L(v_t)}$.

For the first property, suppose H has two source nodes v_1, v_2. We need to show that we get the same value by recursing on v_1 or v_2, i.e

$$A_{L(v_1)} A_{H-v_1} = A_{L(v_2)} A_{H-v_2} \tag{2}$$

Now apply the induction hypothesis to $H - v_1$ and $H - v_2$, noting that v_2 is a source node of $H - v_1$ and v_1 is a source node of $H - v_2$. We get $A_{H-v_1} = A_{L(v_2)} A_{H-v_1-v_2}, A_{H-v_2} = A_{L(v_1)} A_{H-v_1-v_2}$. Thus, in order to show Eq. (2), it suffices to show that $A_{L(v_1)} A_{L(v_2)} = A_{L(v_2)} A_{L(v_1)}$. Since v_1, v_2 are both source nodes, we have $L(v_1) \not\sim L(v_2)$. Thus, this follows from T-commutativity. □

As a warm-up, in Section 3.1 we show how to use wdags to show fast convergence for commutative algorithms. The main point here is to demonstrate how the new commutativity definition helps with the crucial task of bounding the probability of appearance of a given wdag. In Section 3.2 we prove a more general bound that will be useful to capture properties of commutative algorithms beyond fast convergence as well as more sophisticated convergence bounds.

3.1 Warm-up: Proving bounds for fast convergence

As in the original proof of Moser and Tardos [22], in order to bound the expected number of steps of a commutative algorithm, we will estimate the expected number of times each flaw $f \in F$ is resampled. Towards that end, we first describe a simple procedure for constructing wdags from a trajectory of the algorithm. Since we will later show a more general result, we omit a few technical proofs for clarity.

Consider an execution of Algorithm 2. For each time t, we will generate a corresponding wdag G_t which provides the history of the t^{th} resampling. This wdag is built backward in time for $s = t, \dots, 1$. Initially, at $s = t$, it has a single node labeled by f_t. For $s = t-1, \dots, 1$, if the wdag G_t constructed so far has a node with label g where $g \sim f_s$, then we add a vertex labeled f_s; otherwise, we do not modify G_t.

We say that a wdag H *appears* if $H \cong G_t$ for any value t. It is not hard to see that the number of times the algorithm resamples a given flaw f is equal to the number of appearing wdags whose (unique) sink is labeled by f. Thus, to calculate the expected number running time of Algorithm 2, we need to sum the probabilities of appearance of the wdags.

One of the main ingredients in the original proof of Moser and Tardos is that the probability of appearance of each wdag G is at most $\prod_{v \in G} \mu(L(v))$, i.e., the product of probabilities of the bad events that label its vertices. Their proof crucially used the properties of the variable setting and, thus, it does not extend to other probability spaces. Our key message is that T-commutativity allows us to bound the probability of a certain wdag appearing by considering a product of transition matrices over the label of its vertices.

The following key result ties together wdags with the algorithm dynamics. (Recall that μ denotes the initial state distribution.)

Lemma 3.2. *For a given wdag H, the probability that H appears (in the sense of Section 3.1) is at most $\mu^\top A_H \vec{1}$.*

Proof. By a limiting argument, it suffices to show that if run Algorithm 2 for any finite number of steps t starting with state σ, then the probability that H appears is at most $e_\sigma^\top A_H \vec{1}$. We prove this claim by induction on t. If H is the empty wdag, or $t = 0$, or σ is flawless, the claim can be easily seen to be hold vacuously.

So suppose that $t \geq 1$ and $\mathcal{S}$ selects a flaw g to resample in σ, and define $\mathcal{E}_H$ to be the event that H appears when running the search algorithm $\mathcal{A}$. By conditioning on the random seed used by the flaw choice strategy $\mathcal{S}$ (if any), we may assume that the search strategy $\mathcal{S}$ is deterministic.

We can now view the evolution of $\mathcal{A}$ as a two-part process: we first resample g, reaching state σ' with probability $A_g[\sigma, \sigma']$. We then execute a new search algorithm $\mathcal{A}'$ starting at state σ', wherein the flaw selection rule $\mathcal{S}'$ on history $(\sigma', \sigma_2, \dots, \sigma_t)$ is the same as the choice of $\mathcal{S}$ on history $(\sigma, \sigma', \sigma_2, \dots, \sigma_t)$. Let us denote by G'_s the wdags produced for this new search algorithm $\mathcal{A}'$.

Suppose that H appears, so that $G_s \cong H$ for some value $s \leq t$. In this case, one of the two conditions must hold: (i) H has a unique source node v labeled g and $G'_{s-1} \cong H - v$; or (ii) g is unrelated to H and $G'_{s-1} \cong H$.

In the first case, then in order for event $\mathcal{E}_H$ to occur on the original search algorithm $\mathcal{A}$, we must also have $\mathcal{E}_{H-v}$ hold on $\mathcal{A}'$ within $t-1$ timesteps. By induction hypothesis, this has probability at most $e_{\sigma'}^\top A_H \vec{1}$ for a fixed σ'. Summing over σ' gives a total probability of

$$\sum_{\sigma'} A_g[\sigma, \sigma'] e_{\sigma'}^\top A_{H-v} \vec{1} = e_\sigma^\top A_g A_{H-v} \vec{1} = e_\sigma^\top A_H \vec{1}$$

as required.

Otherwise, if g is unrelated to H, then in order for event $\mathcal{E}_H$ to occur for $\mathcal{A}$, we must also have $\mathcal{E}_H$ occur for $\mathcal{A}'$ within $t-1$ timesteps. By induction hypothesis, this has probability at most $e_{\sigma'}^\top A_H \vec{1}$ for a fixed σ'. Summing over σ' gives a total probability of

$$\sum_{\sigma'} A_g[\sigma, \sigma'] e_{\sigma'}^\top A_H \vec{1} = e_\sigma^\top A_g A_H \vec{1}.$$

Since A_g commutes with A_H, this is at most $e_\sigma^\top A_H A_g \vec{1}$. Since A_g is stochastic, this in turn is at most $e_\sigma^\top A_H \vec{1}$, which completes the induction.

To conclude the proof of the lemma, notice that if we start the search with state τ, then the probability that H appears in T_t is at most $e_\tau^\top A_H \vec{1}$. Integrating over τ, gives $\sum_\tau \mu[\tau] e_\tau^\top A_H \vec{1} = \mu^\top A_H \vec{1}$. □

This can be used to show a generalization of the key Witness Tree Lemma of Moser and Tardos:

Corollary 3.3. *Suppose the resampling oracle is regenerating. Then, for a given wdag H, the probability that H appears is at most $\prod_{v \in H} \mu(L(v))$.*

Proof. Let $f_1, \ldots, f_t$ be the labels of the vertices in H, ordered from source nodes to sink nodes. We can write $A_H = A_{f_1} \cdots A_{f_t}$. Since μ is a left-eigenvector of every transition matrix (see Eq. (1)), we have

$$\mu^\top A_H \vec{1} = \mu^\top A_{f_1} \cdots A_{f_t} \vec{1} = \mu(f_1) \cdots \mu(f_t) \mu^\top \vec{1} = \mu(f_1) \cdots \mu(f_t)$$ □

As we have already argued, this directly gives a bound on the expected number of steps of the algorithm.

Corollary 3.4. *Suppose the resampling oracle is regenerating. Then, the expected number of steps of the algorithm is at most*

$$\sum_{f \in F} \sum_{\substack{\text{wdags } H \text{ with} \\ \text{sink}(H) = \{f\}}} \prod_{v \in H} \mu(L(v))$$

We emphasize that we are not aware of any direct proof of Corollary 3.3; it seems necessary to first show the matrix bound of Lemma 3.2, and then project down to scalars. Given Corollary 3.4, standard counting arguments based on Galton-Watson branching processes lead to the LLL convergence conditions.

3.2 General matrix bounds

We now turn to a more general construction of wdags to explain flaw resamplings, which will be used in all the formal proofs later in the paper. We say that f is *dominated by* H if $A_f A_H \vec{1} \preceq A_H \vec{1}$; recall that this means that $e_\sigma^\top A_f A_H \vec{1} \leq e_\sigma^\top A_H \vec{1}$ for all states σ.

To get tight bounds for distributional properties or parallel algorithms, we need to "locally" explain the history of a given resampling. In order to do so, we use a more refined way to generate wdags. Consider a trajectory T. For each finite $t \leq \text{length}(T)$, we can generate a corresponding wdag $G_t^T = \text{GenWitness}(Q, T, t)$ which provides the history of a resampling at time t. Again, the main idea is to build the tree backward in time for $s = t, \ldots, 1$; if the current wdag G_t^T does not dominate flaw f_s, we should add a vertex labeled f_s. For a variety of technical reasons in our analysis, we may also want to add a node labeled f_s, even if f_s is dominated; this explains the role of Q (more details will be provided later). Note that choice of Q does not affect Algorithm 2 itself, only the analysis.

Algorithm 3 Forming witness $G_t^T = \text{GenWitness}(Q, T, t)$

Initialize G_t^T to contain a single vertex labeled f_t
for $s = t-1, \ldots, 1$ **do**
  if $f_s \in Q(G_t^T)$ or f_s is not dominated by G_t^T **then**
    Add to G_t^T a node v_s labeled f_s, with an edge from v_s to each v_j such that $L(v_j) \sim f_s$

We write $G_{[s,t]}^T$ to be the wdag G_t^T just after iteration s, so that $G_{[s,t]}^T$ is derived from $G_{[s+1,t]}^T$ by adding (or not) a vertex labeled f_s. In this case we have $G_t^T = G_{[1,t]}^T$ and $G_{[t,t]}^T$ is a singleton node labeled f_t. We also write for convenience that G_t^T is the empty graph if $t = 0$ or $t > \text{length}(T)$.

There are three possible choices for Q that we will use. The default rule Q_0, which is used for most convergence and distributional results, is defined by setting $f \in Q_0(G)$ if and only if G has a source node labeled f. When analyzing parallel algorithms, we will use an alternative rule Q_1 defined by setting $f \in Q_1(H)$ if and only H contains a source node labeled $g \sim f$. Finally, for some distributional bounds, we use the trivial rule Q_2 defined by $Q_2(G) = \mathcal{F}$, i.e. we always add a node at every step. **For the remainder of this paper, we assume that $Q = Q_0$ or $Q = Q_1$ unless specifically stated otherwise.**

We say that a wdag H *appears* if $H \cong \text{GenWitness}(Q, \hat{T}, t)$ for any value t. For a given rule Q, we denote by $\mathfrak{H}_Q$ the set of all non-empty wdags G which can be produced as $G = \text{GenWitness}(Q, T, t)$ for any search strategy $\mathcal{S}$ and corresponding trajectory T during the evolution of Algorithm 2. Note the following important characterization:

Proposition 3.5. *Any wdag in $\mathfrak{H}_Q$ has a single sink node.*

Proof. Consider forming $G_t^T = \text{GenWitness}(Q, T, t)$. Suppose that at step s of Algorithm 3, the flaw f_s is unrelated to $G_{[s+1,t]}^T$. So A_{f_s} commutes with $A_{L(v)}$ for every $v \in G_{[s+1,t]}^T$; by Proposition 3.1, this implies that $A_{f,s}$ commutes with $A_{G_{[s+1,t]}^T}$, and so we have

$$e_\sigma^\top A_{f_s} A_{G_{[s+1,t]}^T} \vec{1} = e_\sigma^\top A_{G_{[s+1,t]}^T} A_{f_s} \vec{1}$$

Since matrix A_{f_s} is stochastic, this is at most $e_\sigma^\top A_{G_{[s+1,t]}^T} \vec{1}$. So f_s is dominated by $G_{[s+1,t]}^T$. Also, since f_s is unrelated to $G_{[s+1,t]}^T$, the latter does not have a source node labeled $g \sim f_s$. Hence, for either rule Q_0 or rule Q_1, we would not add a new vertex v_s to $G_{[s,t]}^T$.

Thus, whenever we add a vertex v_s to G_t^T, it has an edge to an already-existing node of G_t^T. In particular, G_t^T never gets an additional sink node (aside from the node corresponding to f_t). □

In light of Proposition 3.5, we define $\mathfrak{H}_Q(f)$ to be the set of wdags $H \in \mathfrak{H}_Q$ with $\text{sink}(H) = \{f\}$, and note that $\mathfrak{H} = \bigcup_f \mathfrak{H}_Q(f)$. If Q is understood, we write simply $\mathfrak{H}$ and $\mathfrak{H}(f)$. We again show the following key result:

Lemma 3.6. *For a given wdag H, the probability that H appears (in the sense of Section 3.2) is at most $\mu^\top A_H \vec{1}$.*

We need a few preliminary results to prove Lemma 3.6.

Proposition 3.7. *Consider some wdag G. If GenWitness$(Q, T, t) = G$ for a trajectory $T = (f_1, f_2, \ldots,)$ with $t \geq 1$, then the wdag $G' =$ GenWitness$(Q, T', t-1)$ for the shifted trajectory $T' = S(T)$ is uniquely determined according to the following rule:*

- *If G contains a unique source node v labeled f_1, then $G' = G_t^T - v$*

- *Otherwise, $G' = G$ and f_1 is dominated by G_t^T*

Proof. If $t = 1$, then G_t^T consists of a single node labeled f_1 and $G_{t-1}^{T'}$ is the empty wdag, and this satisfies the first condition. So let us suppose that $t > 1$, in which case Algorithm 3 obtains G_t^T by possibly adding a node v_1 labeled f_1 to $G' = G_{t-1}^{T'}$. If Algorithm 3 adds node v_1 to G', then f_1 is the label of a source node v of G_t^T, and $G' = G_t^T - v$. If Algorithm 3 does not add such node, then $G_t^T = G'$. Since $Q = Q_0$ or $Q = Q_1$, we know that G' does not have a source node labeled f_1, and also f_1 must be dominated by G'. Since $G' = G_t^T$, these imply that f_1 is dominated by $G_t^T = G$ as well. □

Proposition 3.8. *Let H be a wdag, and $t_{\max}$ be a non-negative integer. If we run Algorithm 2 starting with state σ, then* $\Pr\Big(\bigcup_{t=1}^{t_{\max}} G_t^{\hat{T}} \cong H\Big) \leq e_\sigma^\top A_H \vec{1}$.

Proof. Define $\mathcal{E}_{H,t_{\max}}$ to be the event that $G_t^{\hat{T}} \cong H$ for some $t \leq t_{\max}$ during the execution of the search algorithm $\mathcal{A}$. By conditioning on the random seed used by the flaw choice strategy $\mathcal{S}$ (if any), we may assume that the search strategy $\mathcal{S}$ is deterministic. We prove the claim by induction on $t_{\max}$.

If H is the empty wdag, the RHS is one and the statement is vacuous. So, suppose that H is non-empty. Now if $t_{\max} = 0$ or σ is flawless, then $\mathcal{E}_{H,t_{\max}}$ is impossible and again this is vacuous. So let us suppose that $t_{\max} \geq 1$, and that $\mathcal{S}$ selects a flaw g to resample in σ. We can now view the evolution of $\mathcal{A}$ as a two-part process: we first resample g, reaching state σ' with probability $A_g[\sigma, \sigma']$. We then execute a new search algorithm $\mathcal{A}'$ starting at state σ', wherein the flaw selection rule $\mathcal{S}'$ on history $(\sigma', \sigma_2, \ldots, \sigma_r)$ is the same as the choice of $\mathcal{S}$ on history $(\sigma, \sigma', \sigma_2, \ldots, \sigma_r)$.

Suppose now that $\mathcal{E}_{H,t_{\max}}$ holds for $\mathcal{A}$, i.e. $G_t^{\hat{T}} \cong H$ for some $t \leq t_{\max}$. Note that the actual trajectory $\hat{T}'$ for $\mathcal{A}'$ is given by $\hat{T}' = S(\hat{T})$. Thus, by Proposition 3.7, one of the two conditions must hold: (i) either H has a unique source node labeled v and $G_{t-1}^{\hat{T}'} \cong H - v$; or (ii) H has no such node and $G_{t-1}^{\hat{T}'} \cong H$ and g is dominated by H.

In the first case, there must also hold $\mathcal{E}_{H-v,t_{\max}-1}$ for $\mathcal{A}'$. By induction hypothesis, this has probability at most $e_{\sigma'}^\top A_H \vec{1}$ conditional on a fixed σ'. Summing over σ', we get a total probability of

$$\sum_{\sigma'} A_g[\sigma, \sigma'] e_{\sigma'}^\top A_{H-v} \vec{1} = e_\sigma^\top A_g A_{H-v} \vec{1} = e_\sigma^\top A_H \vec{1}$$

In the second case, there must also hold $\mathcal{E}_{H,t_{\max}-1}$ for $\mathcal{A}'$. By induction hypothesis, this has probability at most $e_{\sigma'}^\top A_H \vec{1}$ conditional on a fixed σ'. Summing over σ', we get a total probability of

$$\sum_{\sigma'} A_g[\sigma, \sigma'] e_{\sigma'}^\top A_H \vec{1} = e_\sigma^\top A_g A_H \vec{1}$$

Since g is dominated by H, this is at most $e_\sigma^\top A_H \vec{1}$, again completing the induction. □

Proof of Lemma 3.6. Suppose that we start the search at state τ. In order for H to appear, we must have $G_t^{\hat{T}} \cong H$ for some integer t. By countable additivity of the probability measure, we have

$$\Pr(H \text{ appears}) = \Pr\Big(\bigcup_{t=0}^{\infty} G_t^{\hat{T}} \cong H\Big) = \lim_{t_{\max} \to \infty} \Pr\Big(\bigcup_{t=0}^{t_{\max}} G_t^{\hat{T}} \cong H\Big)$$

By Proposition 3.8, each term in this limit is at most $e_\tau^\top A_H \vec{1}$, so the limit is also at most $e_\tau^\top A_H \vec{1}$. Integrating over τ, we get a total probability of $\sum_\tau \mu[\tau] e_\tau^\top A_H \vec{1} = \mu^\top A_H \vec{1}$. □

We now summarize how Lemma 3.6 governs the behavior of Algorithm 2.

Proposition 3.9. *For a trajectory T and values $0 \leq t' < t \leq \text{length}(T)$ we have $G_t^T \neq G_{t'}^T$*

Proof. We show this by induction on t'. When $t' = 0$, this is clear since $G_{t'}^T$ is empty and G_t^T is not. For the induction step, suppose $t' > 0$ and $G_{t'}^T = G_t^T$. Let $T' = S(T)$. By Proposition 3.7, both $G_{t-1}^{T'}$ and $G_{t'-1}^{T'}$ are updated in the same manner depending on the flaw f_1. Thus, $G_{t-1}^{T'} = G_{t'-1}^{T'}$. But this contradicts the induction hypothesis. □

Proposition 3.10. *The expected number of steps taken in Algorithm 2 is at most $\sum_{H \in \mathfrak{H}_Q} \mu^\top A_H \vec{1}$. In particular, if this sum converges, then Algorithm 2 terminates with probability one.*

Proof. Suppose we run Algorithm 2 resulting in trajectory $\hat{T}$. For each finite value $t \leq \text{length}(\hat{T})$, consider the wdag $H_t = G_t^{\hat{T}}$. This wdag H_t clearly appears, and by Proposition 3.9, all such wdags H_t are distinct. As a result, we have $\text{length}(\hat{T}) \leq \sum_{H \in \mathfrak{H}} [[H \text{ appears}]]$. Taking the expectation of both sides and applying Lemma 3.6 gives

$$\mathbf{E}[\text{length}(\hat{T})] \leq \sum_{H \in \mathfrak{H}_Q} \Pr(H \text{ appears}) \leq \sum_{H \in \mathfrak{H}_Q} \mu^\top A_H \vec{1}$$ □

As we show in Appendix A, under some natural conditions the T-commutativity property is necessary in order to obtain Lemma 3.6.

4 Estimating weights of wdags

The statement of Lemma 3.6 in terms of matrix products is very general and powerful, but difficult for calculations. In order to use it for calculating algorithm runtime (as in Proposition 3.10), or other algorithm properties, we need to bound the sums of the form

$$\sum_{H \in \mathfrak{H}_Q} \mu^\top A_H \vec{1}$$

There are two, quite distinct, issues that arise in this calculation. First, for a given fixed wdag H, we need to estimate $\mu^\top A_H \vec{1}$; second, we need to bound the sum of these quantities over $H \in \mathfrak{H}_Q$. The second issue is at the heart of the probabilistic and algorithmic conditions for the LLL. As discussed by Moser & Tardos [22], it can be viewed in terms of the evolution of certain Galton-Watson branching processes.

The first issue is not as familiar, since most previous analyses of local search algorithms have focused on scalar-valued weights. As discussed in [3], these prior estimates can be viewed in terms of spectral bounds on the matrices A_f. (We emphasize that this approach, in terms of the flaw charges γ_f is only an expedient to bounding the matrix products $\mu^\top A_H \vec{1}$; for some applications, it is possible to take advantage of higher-dimensional information to get more detailed bounds [3].) In this method, we define a quantity called the *charge* γ_f for each flaw f as follows.[1]

$$\gamma_f = \max_{\tau \in \Omega} \sum_{\sigma \in f} \frac{\mu(\sigma)}{\mu(\tau)} A_f[\sigma, \tau] \tag{3}$$

The following result of [18] illustrates the connection between this measure and the Lopsided Lovász Local Lemma (LLLL):

[1] The work [3] provides a more general definition of charge and distortion, where the "benchmark" probability distribution can be different from the initial probability distribution μ. This can be useful in showing convergence of Algorithm 2 for non-commutative resampling oracles. However, this more general definition does not seem to give useful bounds for distributional properties and parallel algorithms in the context of commutative resampling oracles. Hence, we adopt the simpler definitions here.

Theorem 4.1 ([18]). *Given a family of flaws $\mathcal{F}$ and measure μ over Ω, then for each set $S \subseteq \mathcal{F} - \Gamma(f)$ we have $\mu\Big(f \mid \bigcap_{g\in S} \overline{g}\Big) \leq \gamma_f$, where the γ_f are the charges of the algorithm as defined in* (3).

Moreover, as shown in [2], the charge γ_f captures the compatibility between the actions of the algorithm for resampling flaw f and the measure μ. To see this, define the *distortion* associated with f as

$$d_f := \max_{\tau\in\Omega} \frac{\sum_{\sigma\in f} \frac{\mu(\sigma)A[\sigma,\tau]}{\mu(f)}}{\mu(\tau)} = \max_{\tau\in\Omega} \frac{\mu^\top A_f e_\tau}{\mu(\tau)\mu(f)} \geq 1, \tag{4}$$

i.e., the maximum possible inflation of a state probability (relative to its probability under μ) incurred by (i) sampling a state $\sigma \in f$ according to μ; and then (ii) resampling flaw f at σ. Now observe from (3) that

$$\gamma_f = \max_{\tau\in\Omega} \frac{1}{\mu(\tau)} \sum_{\sigma\in f} \mu(\sigma)A_f[\sigma,\tau] = d_f \cdot \mu(f). \tag{5}$$

A resampling oracle $\mathfrak{R}$ with $d_f = 1$ for all f, is called a *regenerating oracle* [16], as it perfectly removes the conditional of the resampled flaw. Such regenerating oracles can be used capture applications of the more standard versions of the LLLL. (This is equivalent to satisfying Eq. (1).)

For a wdag H, let us define the scalar values

$$w(H) = \prod_{v\in H} \gamma_{L(v)}$$

We get the following estimate for $\mu^\top A_H \vec{1}$ in terms of $w(H)$:

Theorem 4.2. *For any event $E \subseteq \Omega$ we have $\mu^\top A_H e_E \leq \mu(E) \cdot w(H)$. In particular, with $E = \Omega$, we have $\mu^\top A_H \vec{1} \leq w(H)$.*

Proof. From definition of γ_f, it can be observed that $\mu^\top A_f \preceq \gamma_f \mu^\top$ for any f. In particular, $\mu^\top A_f \cdot \theta \leq \gamma_f \theta$ for any vector θ. Now, by Proposition 3.1, we can write $A_H = A_{f_1} \dots A_{f_t}$ where $f_1, \dots, f_t$ are the labels of the nodes of H. We thus have:

$$\mu^\top A_H e_E = \mu^\top A_{f_1} \dots A_{f_t} e_E \leq \mu^\top \gamma_{f_1} A_{f_2} \dots A_{f_t} \leq \dots \leq \gamma_{f_1} \dots \gamma_{f_t} \mu^\top e_E = w(H)\mu(E) \qquad \square$$

Corollary 4.3. *For a regenerating oracle, a wdag H appears with probability at most $w(H) = \prod_{v\in H} \mu(L(v))$.*

In light of Theorem 4.2, we define for any flaw f the key quantity

$$\Phi_Q(f) = \sum_{H\in\mathfrak{H}_Q(f)} w(H).$$

We write $\Phi(f)$ alone if Q is clear from context. With these notations, we have the following crisp corollaries of our previous estimates:

Corollary 4.4. *1. Any given wdag H appears with probability at most $w(H)$.*

2. *The expected number of resamplings of any flaw f is at most $\Phi_Q(f)$.*
3. *The expected runtime of Algorithm 2 is at most $\sum_f \Phi_Q(f)$.*
4. *If $\Phi_Q(f) < \infty$ for all f, then Algorithm 2 terminates with probability one.*

The main way to bound $\Phi_Q(f)$ is to inductively bound sums $\sum_{H \in \mathfrak{W}} w(H)$, where $\mathfrak{W}$ is defined as the collection of *all* possible wdags, not just those which could be produced as $\hat{G}^T$. We define $\mathfrak{W}(I)$ to be the collection of all wdags H with $\mathrm{sink}(H) = I$. The sum over $\mathfrak{W}$ is tractable because of the fundamental observation that if $G \in \mathfrak{W}(I)$ has sink nodes $v_1, \ldots, v_t$, then $G' = G - v_1 - \cdots - v_t$ is a smaller wdag in $\mathfrak{W}(J)$ for $J \subseteq \bigcup_{f \in I} \overline{\Gamma}(f)$. Shearer's criterion for the LLL [24] essentially boils down to using this recursion to show that $\sum_{H \in \mathfrak{W}} w(H) < \infty$. For some probability spaces, such as the variable LLLL, we may have additional structural restrictions on the wdags.

Some related useful quantities are $\Psi(I) = \sum_{H \in \mathfrak{W}(I)} w(H)$ and $\overline{\Psi}(I) = \sum_{J \subseteq I} \Psi(J)$. For a flaw f, we write $\Psi(f)$ as shorthand for $\Psi(\{f\})$. Note that $\Phi_Q(f) \leq \Psi(f)$ for any Q. A useful and standard formula (see e.g., [16, Claim 59]) is that for any stable set I we have $\Psi(I) \leq \prod_{f \in I} \Psi(f)$ and $\overline{\Psi}(I) \leq \prod_{f \in I}(1 + \Psi(f))$.

We summarize a few bounds on these quantities, based on versions of LLL criteria, as follows:

Proposition 4.5. *1. (Symmetric criterion) Suppose that $\gamma_f \leq p$ and $|\overline{\Gamma}(f)| \leq d$ for parameters p, d with $epd \leq 1$. Then $\Psi(f) \leq e\gamma_f \leq ep$ for all f.*

2. (Neighborhood bound) Suppose that every f has $\sum_{g \in \overline{\Gamma}(f)} \gamma_g \leq 1/4$. Then $\Psi(f) \leq 4\gamma_f$ for all f.

3. (Asymmetric criterion) Suppose there is some function $x : \mathcal{F} \to [0, 1)$ with the property that

$$\forall f \qquad \gamma_f \leq x(f) \prod_{g \in \Gamma(f)} (1 - x(g)).$$

Then $\Psi(f) \leq \frac{x(f)}{1-x(f)}$ for all f.

4. (Cluster-expansion criterion) Suppose there is some function $\eta : \mathcal{F} \to [0, \infty)$ with the property that

$$\forall f \qquad \eta(f) \geq \gamma_f \cdot \sum_{\substack{I \subseteq \overline{\Gamma}(f) \\ I \text{ stable}}} \prod_{g \in I} \eta(g)$$

Then $\Psi(f) \leq \eta(f)$ for all f.

5. (Clique-bound criterion) Suppose that the dependency graph is covered by a collection $\mathcal{V}$ of cliques, i.e. $f \sim g$ iff there exists $v \in \mathcal{V}$ with $f, g \in v$, and suppose there is some function $\zeta : \mathcal{V} \to [0, \infty)$ with the property that

$$\forall v \in \mathcal{V} \qquad \zeta(v) \geq 1 + \sum_{f \in v} \gamma_f \prod_{u \in \mathcal{V} : f \in u} \zeta(u)$$

Then $\Psi(f) \leq \prod_{u \in \mathcal{V} : f \in u} \zeta(u)$ for all f.

Proof. For completness, we briefly sketch the proofs. For the cluster-expansion criterion, we use an induction on wdag depth to show that the total weight of all wdags $H \in \mathfrak{W}(I)$ is at most $\prod_{f \in I} \eta(f)$.

For the clique-bound criterion, apply the cluster-expansion criterion using function $\eta(f) = \gamma_f \prod_{v \in \mathcal{V} : f \in v} \zeta(v)$.

For the asymmetric criterion, apply the cluster-expansion criterion using function $\eta(f) = \frac{x(f)}{1-x(f)}$

For the neighborhood bound criterion, apply the asymmetric criterion using $x(f) = 2\gamma_f$ for all f.

For the symmetric criterion, apply the cluster-expansion criterion using function $\eta(f) = e\gamma_f$. □

To emphasize the connection between various LLL-type bounds, our analysis of wdags, and the behavior of Algorithm 2, we record the following results:

Proposition 4.6. *Let R denote the expected runtime of Algorithm 2. Under the conditions of Proposition 4.5, we have the following bounds on R:*

1. *If the symmetric criterion holds, then $R \leq e \sum_f \gamma_f \leq O(|\mathcal{F}|/d)$.*
2. *If the neighborhood-bound criterion holds, then $R \leq 4 \sum_f \gamma_f \leq O(|\mathcal{F}|)$.*
3. *If the asymmetric criterion holds, then $R \leq \sum_f \frac{x(f)}{1-x(f)}$*
4. *If the cluster-expansion criterion holds, then $R \leq \sum_f \eta(f)$.*
5. *If the clique-bound criterion holds, then $R \leq \sum_{v \in \mathcal{V}} \zeta(v)$.*

5 Distributional properties

The most important consequence of commutativity is that it leads to good bounds on the distribution of the output of Algorithm 2. Heuristically, these states should be similar in distribution to the initial distribution μ. We also obtain bounds on the intermediate states of Algorithm 2; as discussed in [13], these can be useful for algorithmic applications with exponentially many flaw, as implementing a step of Algorithm 2 requires finding searching the state σ to find a flaw which is currently true on σ, if any.

Consider an event E, which is an arbitrary subset of Ω, and let us define $P(E)$ to be the probability that E occurs at any time during the evolution of Algorihm 2. We will show an upper bound on $P(E)$; this immediately also bounds the probability that E holds on the terminal state of Algorithm 2. To analyze this, we consider adding a new flaw f^E, with an arbitrary resampling rule (e.g. to do nothing). We also modify the flaw-selection strategy $\mathcal{S}$ to always select to resample E, if available. For this expanded set of flaws $\mathcal{F}^E$, we define $f^E \sim g$ for all existing flaws $g \in \mathcal{F}$; as a consequence of this trivial definition of $\sim$ for f^E, the new resampling oracle we obtain remains T-commutative.

Let us define $\mathfrak{H}^E$ to be the set of wdags which are produced as GenWitness(Q_0, T, t) where event E holds at time t but not at times $0, \dots, t-1$. To avoid confusion, all other quantities $\mathfrak{H}, w(H), \gamma_f, \Psi(I), \mathfrak{W}(I)$ etc. should be interpreted in terms of the *original* flaw set $\mathcal{F}$.

The following is our fundamental observation for distributional bounds:

Proposition 5.1. $P(E) \leq \sum_{H \in \mathfrak{H}^E} \mu^\top A_H \vec{1}$.

Proof. Suppose Algorithm 2 (with respect to the original search strategy $\mathcal{S}$) first reaches a state in E at some time $t+1$, with corresponding trajectory $\hat{T} = (f_1, \dots, f_t)$. Note that Algorithm 2 with search strategy $\mathcal{S}$ agrees with Algorithm 2 with the new search strategy $\mathcal{S}'$ at previous times $0, \dots, t$. The resulting wdag $H = \text{GenWitness}(Q_0, T, t+1)$ is in $\mathfrak{H}^E$. Thus, whenever E is true in the execution of Algorithm 2 on search strategy $\mathcal{S}$, there is $H \in \mathfrak{H}^E$ which appears for search strategy $\mathcal{S}'$. By Lemma 3.6, for any fixed such H this has probability at most $\mu^\top A_H \vec{1}$ □

To obtain a more legible bound, we need a few additional defintiions. For event E and state $\sigma \in E$, define $\tilde{\Gamma}(E, \sigma)$ to be the set of flaws $f \in \mathcal{F}$ which can cause E to occur via state σ, i.e. f maps some state $\sigma' \notin E$ to $\sigma \in E$. We also define $\tilde{\Gamma}(E) = \bigcup_{\sigma \in E} \tilde{\Gamma}(E, \sigma)$, i.e. the set of flaws which can cause E. We say that a set $I \subseteq \tilde{\Gamma}(E)$ of flaws is *orderable* for E if there is an enumeration $I = \{g_1, \dots, g_r\}$ such that

$$\forall i = 1, \dots, r \qquad A_{g_i} A_{g_{i+1}} \dots A_{g_r} e_E \not\preceq A_{g_{i+1}} \dots A_{g_r} e_E \tag{6}$$

We denote by $\mathfrak{I}(E)$ the collection of stable sets which are orderable for E.

With these notations, we get the following crisper bound:

Theorem 5.2. $P(E) \le \sum_{I \in \mathfrak{I}(E)} \sum_{H \in \mathfrak{W}(I)} \mu^\top A_H e_E$.

Proof. Consider $H \in \mathfrak{H}^E$ with sink node v labeled f^E which is generated as $H = \text{GenWitness}(Q_0, T, t)$ for some trajectory T of the search strategy $\mathcal{S}'$. For $i = 1, \dots, t$ let $H_i = G^T_{[i,t]} - v$, and so that $H_1 = G - v$.

We claim by induction on s that $\text{sink}(H_s)$ is orderable to E for all $s \le t$. The base case is $s = t$; this holds since $\text{sink}(H_s) = \emptyset$. The induction step holds immediately if $H_{s-1} = H_s$, so suppose that H_{s-1} has a new sink node labeled f_s. Thus f_s is unrelated to H_s. Letting $G' = G^T_{[s,t]} = H_s \cup \{v\}$, we then have for any state σ:

$$e_\sigma^\top A_{f_s} A_{G'} \vec{1} = e_\sigma^\top A_{H_s} A_{f^E} \vec{1} = e_\sigma^\top A_{H_s} A_{f_s} A_{f^E} \vec{1} = e_\sigma^\top A_{H_s} A_{f_s} e_E$$

We first claim that $f_s \in \tilde{\Gamma}(E)$. For, if not, then f_s only maps states already in E to E, and so $e_\tau^\top A_{f_s} e_E \le e_\tau^\top e_E$ for any state τ. Thus, we would have $e_\sigma^\top A_{H_s} A_{f_s} e_E \le e_\sigma^\top A_{H_s} e_E$, and so $e_\sigma^\top A_{f_s} A_{G'} \vec{1} \le e_\sigma^\top A_{G'} \vec{1}$. In particular, f_s would be dominated by G'. Also, $f_s \ne L(u)$ for any source node v of $G^T_{[s,t]}$. So, by rule Q_0, we would not add node labeled f_s to $G^T_{[s,t]}$. Hence $H_{s-1} = H_s$, contradicting our assumption that we add a new sink node to H_{s-1}.

Next, let $I = \text{sink}(H_s)$; by induction hypothesis, I is orderable to E. Enumerate $I = \{g_1, \dots, g_r\}$ to satisfy Eq. (6). We next claim that $\text{sink}(H_{s-1}) = I \cup \{f_s\}$ remains orderable for E. For, if not, then by considering the ordering $\{f_s, g_1, \dots, g_r\}$ for $I \cup \{f_s\}$, then for all states σ it would hold that

$$e_\sigma^\top A_{f_s} A_{g_1} \dots A_{g_r} e_E \le e_\sigma^\top A_{g_1} \dots A_{g_r} e_E \tag{7}$$

Letting V denote the sink nodes of H_s, we have $A_{G'} = A_{H_s - V} A_V A_{f^E}$. Then, for any state σ, we have

$$e_\sigma^\top A_{f_s} A_{G'} \vec{1} = e_\sigma^\top A_{f_s} A_{H_s - V} A_V A_{f^E} \vec{1} = e_\sigma^\top A_{f_s} A_{g_1} \dots A_{g_r} A_{H_s - V} e_E$$

By Eq. (7), this is at most $e_\sigma^\top A_{g_1} \dots A_{g_r} A_{H_s - v} e_E = e_\sigma^\top A_{G'} \vec{1}$. So again f_s is dominated by $G' = G^T_{[s,t]}$, and we would not add node labeled f_s to $G^T_{[s,t]}$. This concludes the induction.

Thus, the wdag $H_1 = G - v$ is in $\mathfrak{W}(I)$ where $I = \text{sink}(H_1)$ is orderable to E. To get the upper bound on $P(E)$, we take a union bound over possible choices for such H_1; by Proposition 5.1, we have

$$P(E) \le \sum_{I \in \mathfrak{I}(E)} \sum_{H \in \mathfrak{W}(I)} \mu^\top A_H A_{f^E} \vec{1}$$

Since f^E only maps states in E, this is at most $\sum_{I \in \mathfrak{I}(E)} \sum_{H \in \mathfrak{W}(I)} \mu^\top A_H e_E$. □

Using our scalar bounds from Section 4, this gives two immediate corollaries:

Corollary 5.3. $P(E) \le \mu(E) \sum_{I \in \mathfrak{I}(E)} \Psi(I)$.

Corollary 5.4. $P(E) \le \mu(E) \overline{\Psi}(\tilde{\Gamma}(E))$.

We note that Iliopoulos [17] had previously shown a bound similar to Corollary 5.4, but it had three additional technical restrictions: (i) it only worked for commutative resampling oracles, not T-commutative resampling oracles; (ii) it additionally required the construction of a commutative resampling oracle for the event E itself; and (iii) if the resampling oracle is not regenerating, it gives a strictly worse bound.

The following result shows how to apply Theorem 5.2 and Corollary 5.4 with common LLL criteria. The proofs are immediate from bounds on Ψ shown in Proposition 4.5.

Proposition 5.5. *Under four criteria of Proposition 4.5, we have the following estimates for P:*

1. *If the symmetric criterion holds, then $P(E) \le \mu(E) \cdot e^{e|\tilde{\Gamma}(E)|p}$.*

2. *If the neighborhood-bound criterion holds, then* $P(E) \leq \mu(E) \cdot e^{4\sum_{f \in \tilde{\Gamma}(E)} \gamma_f}$.

3. *If function* x *satisfies the asymmetric criterion, then* $P(E) \leq \mu(E) \cdot \prod_{f \in \tilde{\Gamma}(E)} \frac{1}{1-x(f)}$.

4. *If function* η *satisfies the cluster-expansion criterion, then* $P(E) \leq \mu(E) \cdot \sum_{I \in \mathfrak{I}(E)} \prod_{g \in I} \eta(g)$.

For some probability spaces, Theorem 5.2 and Corollary 5.3 yield tighter bounds than standard LLL distributional estimates. For example, we can recover a result of [12] for the permutation setting, where the underlying probability space Ω is the uniform distribution on permutations on n letters, and each flaw has the form $g_1 \cap \cdots \cap g_r$, where each g_i is an atomic event of the form $\pi x_i = y_i$, and where the dependency graph is given by $f \sim g$ if $f \cap g = \emptyset$. We then get the following distributional result:

Theorem 5.6 ([12])**.** *In the permutation LLL setting, consider an event* $E = g_1 \cap \cdots \cap g_r$ *where each* g_i *is an atomic event. We have*

$$P(E) \leq \frac{(n-r)!}{n!} \prod_{i=1}^{r} \Big(1 + \sum_{f \in \mathcal{F}: f \sim g_i} \Psi(f)\Big)$$

The work [12] showed this using an ad-hoc analysis based on a variant of witness trees; as we discuss in Appendix B, this bound is recovered automatically from Corollary 5.3.

5.1 Alternate distributional properties for injective oracles

A number of resampling oracles have an additional useful property that we refer to as *injectivity*.[2] We say that $\mathfrak{R}$ is *injective* if for every flaw f and state σ there is at most one state σ' with $A_f[\sigma', \sigma] > 0$. In this case, there is a different type of distributional bounds available which can be stronger than Theorem 5.3 for "complex" events (i.e., events which are composed from simpler events).

Most known resampling oracles, including virtually all of the commutative ones, are injective. For example, it holds for the variable LLLL, for the uniform distribution of permutations, the uniform distribution on matchings of K_n, and the uniform distribution on hamiltonian cycles of K_n [11]. **Throughout Section 5.1, we assume** $\mathfrak{R}$ **is injective.**

For a wdag H, we say that a wdag G is a *prefix* of H if G is a subgraph of H and for each directed edge $(u, v) \in H$, where $v \in G$, we also have $u \in G$. If $H \ncong G$ we say it is a *strict prefix*. For a state $E \subset \Omega$, we define $\mathfrak{A}(E)$ to be the collection of all pairs (H, σ) such that $\sigma \in E$ and no strict prefix H' of H has $e_E^\top A_{H-H'} e_\sigma > 0$. We have the following characterization of $\mathfrak{A}(E)$:

Proposition 5.7. *If* $(H, \sigma) \in \mathfrak{A}(E)$ *then* $\mathrm{sink}(H) \subseteq \tilde{\Gamma}(E, \sigma)$.

Proof. Let v be a sink node of H. Then, consider prefix $H' = H - v$ and note that $A_{H-H'} = A_f$. By definition of E-minimality, there is exactly one state τ that can map to σ to f. This state τ is not in E. Hence, we have $f \in \tilde{\Gamma}(E, \sigma)$. □

With this definition, we will show the following bound; note that, due Proposition 5.7, this bound is at least as strong as Corollary 5.4.

Theorem 5.8. $P(E) \leq \sum_{(H,\sigma) \in \mathfrak{A}(E)} \mu^\top A_H e_\sigma$

To show the Theorem 5.8, consider a trajectory T ending in a state ρ. We define $\mathcal{E}_{H,\sigma}$ to be the event that there is some time $t > 0$ such that H is a prefix of $G = \mathrm{GenWitness}(Q_2, \hat{T}, t)$ and $A_{G-H}[\sigma, \rho] > 0$.

[2] In previous papers [1, 3, 21] this property was referred to as *atomicity*. We use the terminology "injectivity" to distinguish it from our later discussion of "atomic" events.

Proposition 5.9. *Let $T = (g, f_2, f_3, \dots,)$ be a trajectory and let $T' = S(T)$. If event $\mathcal{E}_{H,\sigma}$ holds for T, then one of the following two conditions must hold: (i) H has a source node labeled g and event $\mathcal{E}_{H-v,\sigma}$ holds for $\hat{T}'$; or (ii) H has no source node labeled g, and g is unrelated to H, and event $\mathcal{E}_{H,\sigma'}$ holds for $\hat{T}'$ where state σ' satisfies $A_g[\sigma, \sigma'] > 0$.*

Proof. Suppose that H is a prefix of some $G = \text{GenWitness}(Q_2, T, t)$. Since Q_2 always add nodes, G has a unique source node v labeled g. Also, the wdag $G' = \text{GenWitness}(Q_2, T', t-1)$ satisfies $G' = G - v$.

If H has a source node u labeled g as well, then wdag $H' = H - u$ is a prefix of $G' = G - v$. Also, we have $A_{G'-H'}[\sigma, \rho] = A_{G-H}[\sigma, \rho] > 0$. Thus, the event $\mathcal{E}_{H',\sigma}$ occurs in the trajectory T'.

On the other hand, suppose H has no such source node. We claim that g must be unrelated to H. For, if H contained some node u with $L(u) \sim g$, then this would also correspond to some node u of G. There is a directed path in G from v to u, so by definition of prefix v would also be in H.

So the wdag $H' = H$ is a prefix of $G' = G - v$. Now let σ' denote the unique state with $A_{G'-H}[\sigma', \rho] > 0$, if any such state exists. By definition, the event $\mathcal{E}_{H,\sigma'}$ would hold for trajectory T'. In addition, we also would have $0 < e_\sigma^\top A_{G-H} e_\rho = e_\sigma^\top A_g A_{G'-H} e_\rho \geq e_\sigma^\top A_g e_{\sigma'}$; thus, $A_g[\sigma, \sigma'] > 0$. □

We have the key estimate:

Lemma 5.10. *If Algorithm 2 starts at state τ, then event $\mathcal{E}_{H,\sigma}$ occurs with probability at most $e_\tau^\top A_H e_\sigma$.*

Proof. We will show by induction on $t_{\max}$ that the probability that $\mathcal{E}_{H,\sigma}$ occurs within $t_{\max}$ steps is at most $e_\tau^\top A_H e_\sigma$. Taking the limit as $t_{\max} \to \infty$ will then give the claimed result. The base cases where either $t_{\max} = 0$ or when τ is flawless are clear. For the induction step, suppose that we have fixed a deterministic resampling rule $\mathcal{S}$ which selects flaw g in τ, and suppose that we resample state τ to τ', Let $\mathcal{A}$ be the original search algorithm starting at τ and let $\mathcal{A}'$ be the new search algorithm starting at τ'.

Suppose that H has a source node v labeled g. By Proposition 5.9, the event $\mathcal{E}_{H-v,\sigma}$ occurs in $\mathcal{A}'$. Summing over τ', we thus get the bound

$$\Pr(\mathcal{E}_{H,\sigma}) \leq \sum_{\tau'} A_g[\tau, \tau'] \Pr(\mathcal{E}_{H-v,\sigma} \text{ holds for } \mathcal{A}' \text{ starting at } \tau')$$

By induction hypothesis, this is at most

$$\sum_{\tau'} A_g[\tau, \tau'] e_{\tau'}^\top A_{H-v} e_\sigma = e_\tau^\top A_g A_{H-v} e_\sigma = e_\tau^\top A_H e_\sigma$$

as desired.

In the second case, suppose that H does not contain a source node labeled g. By Proposition 5.9, then g is unrelated to H. Also, $\mathcal{E}_{H,\sigma'}$ must hold for $\mathcal{A}'$ for some state σ' with $A_g[\sigma, \sigma'] > 0$. We therefore integrate over τ' and take a union bound over all possible states σ' to get:

$$\Pr(\mathcal{E}_{H,\sigma} \text{ holds on } \mathcal{A}) \leq \sum_{\sigma': A_g[\sigma,\sigma'] > 0} \Pr(\mathcal{E}_{H,\sigma'} \text{ holds on } \mathcal{A}' \text{ starting at state } \tau')$$

From the induction hypothesis and the fact that matrices A_g and A_H commute, this implies that

$$\Pr(\mathcal{E}_{H,\sigma} \text{ holds in } \mathcal{A}) \leq \sum_{\sigma': A_g[\sigma,\sigma'] > 0} A_g[\tau, \tau'] e_{\tau'}^\top A_H e_{\sigma'} = \sum_{\sigma': A_g[\sigma,\sigma'] > 0} e_\tau^\top A_g A_H e_{\sigma'} = e_\tau^\top A_H \sum_{\sigma': A_g[\sigma,\sigma'] > 0} A_g e_{\sigma'}$$

Let us define the vector $x = e_\tau^\top A_H$, and we can write this as $\sum_i x[i] \sum_{\sigma' : A_g[\sigma,\sigma'] > 0} A_g[i,\sigma']$. The term i only contributes here if $A_g[i,\sigma'] > 0$ and $A_g[\sigma,\sigma'] > 0$. Since $\mathfrak{R}$ is injective, this occurs only for $i = \sigma$. So we have:

$$\sum_i x[i] \sum_{\sigma' : A_g[\sigma,\sigma'] > 0} A_g[i,\sigma'] = \sum_{\sigma'} x[\sigma] A_g[\sigma,\sigma']$$

which, by stochasticity, is precisely $x[\sigma]$. So overall, we have shown $e_\tau^\top A_H \sum_{\sigma' \in V} A_g e_{\sigma'} = e_\tau^\top A_H e_\sigma$, which implies that $\Pr(\mathcal{E}_{H,\sigma} \text{ holds on } \mathcal{A}) \leq e_\tau^\top A_H e_\sigma$ This completes the induction. □

We now prove Theorem 5.8:

Proof of Theorem 5.8. Consider running the Algorithm 2 until it reaches a state in E or a flawless state. Suppose it reaches a state $\rho \in E$ at some time t. Then event $\mathcal{E}_{\rho,G}$ has occured where $G = \text{GenWitness}(Q_2, \hat{T}, t)$. Accordingly, we may select pair (H,σ) such that (i) $\sigma \in E$; (ii) event $\mathcal{E}_{H,\sigma}$ holds; and (iii) H has minimal size subject to the first two conditions.

We claim that $(H,\sigma) \in \mathfrak{A}(E)$. For, if not, there would be some H' which is a strict prefix of H and $\sigma' \in E$ with $e_{\sigma'}^\top A_{H-H'} e_\sigma > 0$. In this case, we would have $e_{\sigma'} A_{G-H'} e_\rho = e_{\sigma'}^\top A_{H-H'} A_{G-H} e_\rho \geq e_{\sigma'}^\top A_{H-H'} e_\sigma \cdot e_\sigma^\top A_{G-H} e_\rho > 0$. Also, H' is a prefix of H which is a prefix of G. So event $\mathcal{E}_{H',\sigma'}$ also occurs, contradicting minimality of H.

Thus, a necessary condition for reaching E is that $\mathcal{E}_{H,\sigma}$ holds for some $(H,\sigma) \in \mathfrak{A}(E)$. A union bound over $\mathfrak{A}(E)$ gives $P(E) \leq \sum_{(H,\sigma) \in \mathfrak{A}(E)} \Pr(\mathcal{E}_{H,\sigma})$; by Lemma 5.10, each summand is at most $\mu^\top A_H e_\sigma$. □

As usual, we can use our scalar weights and Theorem 4.2 to get some simplified bound:

Corollary 5.11. *We have the bounds*

$$P(E) \leq \sum_{(H,\sigma) \in \mathfrak{A}(E)} \mu(\sigma) w(H) \leq \sum_{\sigma \in E} \mu(\sigma) \overline{\Psi}(\tilde{\Gamma}(E,\sigma)) \leq \mu(E) \cdot \max_{\sigma \in E} \overline{\Psi}(\tilde{\Gamma}(E,\sigma))$$

We also get a bound for disjunctive events:

Corollary 5.12. *Let $\mathcal{C}$ be a collection of events in Ω and let $E = \bigcup_{C \in \mathcal{C}} C$. Then*

$$P(E) \leq \mu(E) \cdot \max_{C \in \mathcal{C}} \overline{\Psi}(\tilde{\Gamma}(C))$$

Proof. For a state $\sigma \in E$, there must be event $C_\sigma \in \mathcal{C}$ holding on σ. Consider some $g \in \tilde{\Gamma}(E,\sigma)$; there must be a state $\tau \notin E$ which gets mapped via g to σ. In particular, event C_σ is false on τ. So $g \in \tilde{\Gamma}(C_\sigma)$. This implies that $\overline{\Psi}(\tilde{\Gamma}(E,\sigma)) \leq \overline{\Psi}(\tilde{\Gamma}(C_\sigma))$ and so $\overline{\Psi}(\tilde{\Gamma}(E,\sigma)) \leq \max_{C \in \mathcal{C}} \overline{\Psi}(\tilde{\Gamma}(C))$. The result now follows from Corollary 5.11. □

We remark that a slightly weaker version of Corollary 5.12 had been shown in [12] for the variable LLL, based on arguments specifically tailored to that space.

6 Parallel algorithms

Moser & Tardos [22] described a simple parallel version of their resampling algorithm, which can be summarized as follows:

Algorithm 4 Parallel Moser-Tardos algorithm

1: Draw state X from distribution μ
2: **while** some bad-event is true on X **do**
3: Select some arbitrary MIS I of bad-events true on X
4: Resample, in parallel, all variables involved in events in I

A variety of parallel resampling algorithms have also been developed for other probability spaces [14, 10]. One main benefit of the commutativity property is that it enables much more general parallel implementations of Algorithm 2. As a starting point, [21] discussed a generic framework for parallelization which we summarize as follows:

Algorithm 5 Generic parallel resampling framework

1: Initialize the state σ
2: **while** some flaw holds on σ **do**
3: Set $V \neq \emptyset$ to be the set of flaws currently holding on σ
4: **while** $V \neq \emptyset$ **do**
5: Select, arbitrarily, a flaw $f \in V$.
6: Update $\sigma \leftarrow \mathfrak{R}_\sigma(\sigma)$.
7: Remove from V all flaws g such (i) $\sigma \notin g$; or (ii) $f \sim g$

Each iteration of the main loop (lines 2 – 7) is called a *round*. We emphasize this is a *sequential* algorithm, which can be viewed as a version of Algorithm 2 with an unusual flaw-selection choice. Most known parallel local search algorithms, including Algorithm 4, fall into this framework. Harris [11] further showed a general method for simulating each round in parallel, for resampling oracles which satisfy a property called *obliviousness* (see Section 7 for a formal definition).

One of the main results of [21] is that, when the resampling oracle is commutative, the total number of rounds in Algorithm 5 is polylogarithmic with high probability. Thus, an RNC implementation of each round gives an overall RNC search algorithm. We will now show that the same bound holds for T-commutative resampling oracles, via bounding the weights of certain classes of wdags.

We define V_k to be the set of flaws V in round k, and we define I_k to be the set of flaws which are actually resampled at round k (i.e. a flaw f selected at some iteration of line 5). Note that I_k is a stable set. Let $b_k = \sum_{i<k} |I_i|$ be the total number of resamplings made before round k; thus $b_1 = 0$, and when "serialize" Algorithm 5 and view it as an instance of Algorithm 2, the resamplings in round k of Algorithm 5 correspond to the resamplings at iterations $b_k + 1, \dots, b_{k+1}$ of Algorithm 2.

Proposition 6.1. *For all $f \in V_k$ there exists $g \in I_{k-1}$ with $f \sim g$.*

Proof. First, suppose that $f \notin V_{k-1}$. In this case, the only way f could become true at round k would be that some $g \sim f$ was resampled at round $k-1$, i.e. $g \in I_{k-1}$. Otherwise, suppose that $f \in V_{k-1}$. Then either it was removed from V_{k-1} due to resampling of some $g \sim f$, or f became false during round $k-1$. In the latter case, note that in order to later become true at the beginning of round k, there must be some other $g' \in I_{k-1}$ resampled after g with $g' \sim f$.

We now show the claim by induction on k. For the base case $k = 1$, we can easily see that if f is resampled at round 1 then the wdag with a singleton node labeled f appears.

For the induction step, suppose that $V_k \neq \emptyset$. So there is some $f \in V_k$. By our above claim, there must be some $g \in I_{k-1}$ with $g \sim f$. Now by induction hypothesis there is some wdag H with sink node labeled g and depth $k-1$ which appears. If we form a new dag H' by adding a sink node labeled f, we get a wdag with depth k and sink node labeled f which appears. □

Proposition 6.2. *Consider running Algorithm 5 obtaining trajectory $\hat{T}$. Then, for each t in the range $b_k + 1, \ldots, b_k$ the wdag $G_t^{\hat{T}} = \text{GenWitness}(Q_1, \hat{T}, t)$ has depth precisely k.*

Proof. For each $j = 1, \ldots, k$, let us define the corresponding wdag $H_j = G^{\hat{T}}_{[b_1+1,t]}$. We show by backwards induction on j the following properties hold:

- The depth of H_j is precisely $k - j + 1$
- The nodes $v \in H_j$ with depth $k - j + 1$ correspond to resamplings in round j

The base case $j = k$ is clear, since then H_j consists of a singleton node corresponding to the resampling at time t in round k.

For the induction step, observe that we form H_j from H_{j+1} by adding nodes corresponding to resamplings in I_j. By induction hypothesis, H_{j+1} has depth $k - j + 1$. Since I_j is a stable set, we have $\mathrm{depth}(H_j) \leq 1 + \mathrm{depth}(H_{j+1})$ and furthermore the nodes at maximal depth correspond to resamplings in I_j. By induction hypothesis, this implies that $\mathrm{depth}(H_{j-1}) \leq k - j + 1$ and that nodes $v \in H_j$ with depth $k - j + 1$ correspond to resamplings in round j. So we just need to show that there is at least one such node.

Consider any node v of H_{j+1} with depth $k - (j + 1) + 1$ and $L(v) = g$; by induction hypothesis this corresponds to a resampling in round $j + 1$. By Proposition 6.1, we have $g \sim f_s$ for some time s in round j. Let $H' = G^{\hat{T}}_{[s+1,t]}$. If v is no longer a source node in $G^{\hat{T}}_{[s+1,t]}$, then the node w with an edge to v would be a node of H_j of depth $k - j + 1$, as desired.

Otherwise, suppose that v is such a source node. Since $g = L(v) \sim f_s$, our definition of Q_1 ensures that Algorithm 3 will add a node labeled f_s as a source node, which will have an edge to v. So f_s has depth $k - j + 1$ in H_j.

This completes the induction. The stated bound then holds since $G_t^{\hat{T}} = G^{\hat{T}}_{[b_j+1,t]}$ for $j = 1$. □

Proposition 6.3. *For any $f \in \mathcal{F}$ and index $k \geq 1$, we have $\Pr(f \in V_k) \leq \sum_{\substack{H \in \mathfrak{H}_{Q_1}(f) \\ \mathrm{depth}(H)=k}} \mu^\top A_H \vec{1}$..*

Proof. As we have discussed, Algorithm 5 can be viewed as an instantiation of Algorithm 2 with a flaw selection rule $\mathcal{S}$. For a fixed f, let us define a new flaw selection rule $\mathcal{S}_f$ as follows: it agrees with $\mathcal{S}$ up to round k; it then selects f to resample at round k if it is true. The behavior of Algorithm 2 for $\mathcal{S}$ and $\mathcal{S}_f$ is identical up through the first b_{k-1} resamplings. Furthermore, we have $f \in V_k$ for Algorithm 5 if and only if Algorithm 2 selects f for resampling at iteration $b_k + 1$.

Consider the resulting wdag $\mathrm{GenWitness}(Q_1, \hat{T}, b_k + 1)$; by Proposition 6.2 it has depth k. Furthermore, it has a sink node labeled f. Finally, since it is produced from a trajectory corresponding to a flaw selection rule $\mathcal{S}$, it is in $\mathfrak{H}_{Q_1}$. Thus, if $f \in V_k$, then there is some $H \in \mathfrak{H}_{Q_1}(f)$ with $\mathrm{depth}(H) = k$ which appears. To bound the probability of $f \in V_k$, we take a union bound over all such H and apply Lemma 3.6 □

Corollary 6.4. *1.* $\sum_k \mathbf{E}[|V_k|] \leq \sum_f \Phi_{Q_1}(f)$

2. For any integer $t \geq 1$, the probability that Algorithm 5 runs for more than $2t$ rounds is at most $\sum_{H \in \mathfrak{H}_{Q_1} : \mathrm{depth}(H) \geq t} w(H)/t$.

Proof. By Theorem 4.2 and Proposition 6.3, we have $\mathbf{E}[|V_k|] \leq \sum_{H \in \mathfrak{H}_{Q_1}, \mathrm{depth}(H)=k} w(H)$ for each k. Using this bound, we first compute

$$\sum_k \mathbf{E}[|V_k|] \leq \sum_k \sum_{\substack{H \in \mathfrak{H}_{Q_1} \\ \mathrm{depth}(H)=k}} w(H) = \sum_{H \in \mathfrak{H}_{Q_1}} w(H) = \sum_f \Phi_{Q_1}(f)$$

For the second claim, let us define the random variable $Y = \sum_{k \geq t} |V_k|$. We then have:

$$\mathbf{E}[Y] = \sum_{k \geq t} \mathbf{E}[|V_k|] \leq \sum_{\substack{H \in \mathfrak{H}_{Q_1} \\ \mathrm{depth}(H) = k}} w(H)$$

Now, if Algorithm 5 reaches iteration $2t$, then necessarily $V_k \neq \emptyset$ for $k = t, \ldots, 2t$, and so $Y \geq t$. By Markov's inequality applied to Y, we thus get

$$\Pr(\text{Alg reaches round } 2t+1) \leq \Pr(Y \geq t) \leq \mathbf{E}[Y]/t \leq \sum_{\substack{H \in \mathfrak{H}_{Q_1} \\ \mathrm{depth}(H) \geq t}} w(H)/t \qquad \square$$

The usual strategy to bound the sum over wdags H with $\mathrm{depth}(H) \geq t$ in Corollary 6.4 is to use an "inflated" weight function defined as

$$w_\epsilon(H) = w(H)(1+\epsilon)^{|H|} = \prod_{v \in H} \Big((1+\epsilon)\gamma_{L(v)} \Big)$$

and corresponding sum

$$W_\epsilon = \sum_{H \in \mathfrak{H}_{Q_1}} w_\epsilon(H),$$

for some $\epsilon > 0$. This gives the following results:

Proposition 6.5. *With probability at least $1 - \delta$, Algorithm 5 terminates in $O(\frac{\log(1/\delta + \epsilon W_\epsilon)}{\epsilon})$ rounds and has $\sum_k |V_k| \leq O(W_\epsilon/\delta)$. Furthermore, if the resampling oracle is regenerating and satisfies the computational requirements given in [11] for input length n, then with probability $1 - 1/\operatorname{poly}(n)$ the algorithm of [11] terminates in $O(\frac{\log^4(n + \epsilon W_\epsilon)}{\epsilon})$ time on an EREW PRAM.*

Proof. We show only the first result; the second depends on numerous definitions and results of [11].

For the bound on $\sum_k |V_k|$, we simply use Corollary 6.4(1) and Markov's inequality. For the bound on the number of rounds, we calculate

$$\sum_{\substack{H \in \mathfrak{H}_{Q_1} \\ \mathrm{depth}(H) \geq t}} w(H) = \sum_{\substack{H \in \mathfrak{H}_{Q_1} \\ \mathrm{depth}(H) \geq t}} w_\epsilon(H)(1+\epsilon)^{-|H|} \leq (1+\epsilon)^{-t} \sum_{H \in \mathfrak{H}_{Q_1}} w_\epsilon(H) = (1+\epsilon)^{-t} W_\epsilon$$

By Corollary 6.4(2), we thus need $(1+\epsilon)^{-t} W_\epsilon / t \leq \delta$ to ensure termination by round $2t$ with probability at least δ. Straightforward analysis shows that this holds for $t = O(\frac{\log(1/\delta + \epsilon W_\epsilon)}{\epsilon})$. $\square$

Bounding W_ϵ is very similar to bounding $\sum_H w(H) = W_0$, except with a small "slack" in the charges. More specifically, we need to satisfy Proposition 4.5 except with the charges γ_f replaced with inflated values $(1+\epsilon)\gamma_f$. For example, using standard estimates (see [8, 21, 3]) we can get the following simplified bounds:

Proposition 6.6.

1. *Suppose that the resampling oracle is regenerating and that the vector of probabilities $p(1+\epsilon)$ still satisfies the LLLL criterion. Then $W_{\epsilon/2} \leq O(m/\epsilon)$. In particular, Algorithm 5 terminates after $O(\frac{\log(m/\delta)}{\epsilon})$ rounds with probability $1 - \delta$.*
2. *Suppose that $\gamma_f \leq p$ and $|\overline{\Gamma}(f)| \leq d$ such that $epd(1+\epsilon) \leq 1$. Then $W_{\epsilon/2} \leq O(m/\epsilon)$. Algorithm 5 terminates after $O(\frac{\log(m/\delta)}{\epsilon})$ rounds with probability at least $1 - \delta$.*

7 Compositional properties for resampling oracles

In many applications, the flaws and their resampling oracles are built out of a collection of simpler, "atomic" events. For example, in the permutation LLL setting, these would be events of the form $\pi x = y$. In [11], Harris described a generic construction for this type of composition when the atomic events satisfy an additional property referred to as *obliviousness*. Let us now review this construction, and how it works with T-commutativity.

Consider a set $\mathcal{A}$ of events, along with a resampling oracle $\mathfrak{R}$ and a dependency relation $\sim$. The set $\mathcal{A}$ should be thought of as "pre-flaws", that is, it has all the *structural* algebraic properties of a resampling oracle, but does not necessarily satisfy any convergence condition such as the LLLL. If we run the local search algorithm with this set of flaws $\mathcal{A}$, the algorithm will likely not converge.

It is allowed, but not required, to have $f \sim f$ for a pre-flaw f. This will make a significant difference in determining the dependency relation for the composed flaws. Recall that in our framework a flaw f always should have $f \sim f$.

For the compositional construction, it is necessary to define explicitly how the resampling oracle $\mathfrak{R}_f$ uses the random seed. We suppose that $\sigma' \leftarrow \mathfrak{R}_f(\sigma)$ is generated by first drawing a random seed r from some probability space R_f, and then setting $\sigma' = F(\sigma, r)$ for some *deterministic* function F. For brevity, we write this as $\sigma' = r\sigma$. With this notation, we can state the definition:

Definition 7.1 (Oblivious resampling oracle [11])**.** *The resampling oracle $\mathfrak{R}$ is called* oblivious *[11] if for every pair $f, g \in \mathcal{A}$ with $f \not\sim g$ and for each $r \in R_f$, one of the following two properties holds:*

- *For all $\sigma \in f \cap g$ we have $r\sigma \in g$*
- *For all $\sigma \in f \cap g$ we have $r\sigma \notin g$*

Let us now suppose that this condition holds. For each $f \in \mathcal{A}$ and $g_1, \dots, g_s \in \mathcal{A}$ with $g_i \not\sim f$, we define $R_{f;g_1,\dots,g_s}$ to be the set of values $r \in R_f$ such that $r\sigma \in g_1 \cap \dots \cap g_t$. With some abuse of notation, we also use $R_{f;g_1,\dots,g_s}$ to refer to the probability distribution of drawing r from R_f, conditioned on having r in the set $R_{f;g_1,\dots,g_s}$. Note that in light of Definition 7.1 this is well-defined irrespective of σ.

For a stable set $E \subseteq \mathcal{A}$, we define $\langle E \rangle$ to be the intersection of the events in E, i.e., $\langle E \rangle = \bigcap_{f \in E} f$. From $\mathcal{A}$, one can construct an enlarged set of events

$$\overline{\mathcal{A}} = \{\langle E \rangle \mid E \text{ a stable subset of } \mathcal{A}\}$$

We define the relation $\sim$ on $\overline{\mathcal{A}}$ by setting $\langle E \rangle \sim \langle E' \rangle$ iff either (i) $E = E'$ or (ii) there exist $f \in E, f' \in E'$ with $f \sim f'$. We also define a corresponding resampling oracle $\mathfrak{R}$ on $\overline{\mathcal{A}}$ which will satisfy all its required structural properties. The intent is to choose the flaw set $\mathcal{F}$ to be some arbitrary subset of $\overline{\mathcal{A}}$.; as before, $\overline{\mathcal{A}}$ does not necessarily satisfy any LLLL convergence criterion.

To determine $\mathfrak{R}$, consider some $g = \langle E \rangle$ for a stable set E, with some arbitrary enumeration $E = \{f_1, \dots, f_t\}$. We define R_g to be the probability distribution on tuples $r = (r_1, \dots, r_t)$ wherein each r_i is drawn independently from $R_{f_i;f_{i+1},\dots,f_s}$, and we set $r\sigma = r_t \dots r_1\sigma$.

Theorem 7.2 ([11])**.** *Suppose that $\mathfrak{R}$ is an oblivious resampling oracle for $\mathcal{A}$, which is not necessarily T-commutative. Then:*

- *$\mathfrak{R}$ is an oblivious resampling oracle for $\overline{\mathcal{A}}$.*
- *The relation $\sim$ is a dependency relation for $\overline{\mathcal{A}}$.*
- *If the resampling oracle $\mathfrak{R}$ on $\mathcal{A}$ is regenerating, then the resampling oracle on $\overline{\mathcal{A}}$ is also regenerating.*

- *If the resampling oracle $\mathfrak{R}$ on $\mathcal{A}$ is injective, then the resampling oracle on $\overline{\mathcal{A}}$ is also injective.*

It would seem reasonable that if $\mathcal{A}$ is commutative in the sense of Kolmogorov, then $\overline{\mathcal{A}}$ would be as well. Unfortunately, we do not know how to show such a result. We can show, however, that if $\mathcal{A}$ is T-commutative, then $\overline{\mathcal{A}}$ is as well, plus inheriting further nice properties. This is a good illustration of how the new definition of commutativity is easier to work with, beyond its advantage of greater generality.

Proposition 7.3. *Suppose that $\mathcal{A}$ is oblivious but not necessarily T-commutative. For a given flaw $g = \langle E \rangle$, let us suppose that, in order to define $\mathfrak{R}_g$, we have enumerated the stable-set E as $E = \{f_1, \ldots, f_t\}$. Then $A_g = c_g A_{f_1} \ldots A_{f_t}$ where scalar c_g is given by*

$$c = \prod_{i=1}^{t} \frac{1}{\Pr_{r_i \sim R_{f_i}}(r_i \in R_{f_i}; f_{i+1}, \ldots, f_t)}$$

Proof. By definition of $\mathfrak{R}_g$, we have $A_g[\sigma, \sigma'] = \Pr(r_t \ldots r_1 \sigma = \sigma')$, where each r_i is drawn independently from $R_{f_i; f_{i+1}, \ldots, f_t}$. Let us define $S_i = R_{f_i; f_{i+1}, \ldots, f_t}$ and $\sigma_i = r_i \ldots r_1 \sigma$ for $i = 0, \ldots, t$ (where $\sigma_0 = \sigma$). By enumerating over possible values for $\sigma_1, \ldots, \sigma_t$, we get:

$$A_g[\sigma, \sigma'] = \sum_{\substack{\sigma_1, \ldots, \sigma_t \\ \sigma_t = \sigma'}} \prod_{i=1}^{t} \Pr_{r_i \sim S_i}(r_i \sigma_{i-1} = \sigma_i) \tag{8}$$

Now, suppose that $\sigma_i \notin f_j$ for some $j > i$. In this case, the term $\Pr_{r_i \sim S_i}(r_i \sigma_{i-1} = \sigma_i)$ in Eq. (8) must be zero, since $r_i \in S_i \subseteq R_{f_i; f_j}$. So we may restrict the sum to terms with $\sigma_i \in f_{i+1} \cap \cdots \cap f_t$ for all $i = 0, \ldots, t$. For each such term, we have

$$\Pr_{r_i \sim S_i}(r_i \sigma_{i-1} = \sigma_i) = \frac{\Pr_{r_i \sim R_i}(r_i \sigma_{i-1} = \sigma_i \wedge r_i \in S_i)}{\Pr_{r_i \sim R_i}(r_i \in S_i)} = \frac{\Pr_{r_i \sim R_i}(r_i \sigma_{i-1} = \sigma_i)}{\Pr_{r_i \sim R_i}(r_i \in S_i)} = \frac{A_{f_i}[\sigma_{i-1}, \sigma]}{\Pr_{r_i \sim R_i}(r_i \in S_i)}$$

Substituting into Eq. (8), we get:

$$\begin{aligned} A_g[\sigma, \sigma'] &= \sum_{\substack{\sigma_1, \ldots, \sigma_t \\ \sigma_t = \sigma'}} \frac{A_{f_1}[\sigma_0, \sigma_1] \ldots, A_{f_t}[\sigma_{t-1}, \sigma_t]}{\prod_{i=1}^{t} \Pr_{r_i \sim R_i}(r_i \in S_i)} = \frac{\sum_{\sigma_t = \sigma'} A_{f_1}[\sigma_0, \sigma_1] \ldots, A_{f_t}[\sigma_{t-1}, \sigma_t]}{\prod_{i=1}^{t} \Pr_{r_i \sim R_i}(r_i \in S_i)} \\ &= (c A_{f_1} \ldots A_{f_t})[\sigma, \sigma'] \end{aligned}$$

□

Proposition 7.4. *If $\mathcal{A}$ is T-commutative and oblivious, then the transition matrix A_g for a flaw $g = \langle E \rangle$ does not depend on the enumeration of E.*

Proof. Let $E = \{f_1, \ldots, f_t\}$. By Proposition 7.3, we have $A_g = c A'_g$, where $A'_g = A_{f_1} \ldots A_{f_t}$. Since the matrices A_{f_i} all commute, A'_g does not depend on the enumeration of E. Furthermore, since matrix A_g is stochastic, the constant c can be uniquely determined from A'_g, i.e. choose an arbitrary state $\sigma \in g$ and let $c = \frac{1}{\sum_{\sigma'} A'_g[\sigma, \sigma']}$. □

Theorem 7.5. *If the resampling oracle is T-commutative and oblivious on $\mathcal{A}$, then it is also T-commutative on $\overline{\mathcal{A}}$.*

Proof. Let $g = \langle E \rangle$ and $g' = \langle E' \rangle$ for stable sets E, E' such that $g \not\sim g'$. So $f \not\sim f'$ for all $f \in E$ and $f' \in E'$. By Proposition 7.3 we have

$$A_g A_{g'} = c_g c_{g'} \Big(\prod_{f \in E} A_f \prod_{f' \in E'} A_{f'} \Big), \qquad A_{g'} A_g = c_{g'} c_g \Big(\prod_{f' \in E'} A_{f'} \prod_{f \in E} A_f \Big)$$

for scalar constants $c_g, c_{g'}$.

All these matrices $A_f, A_{f'}$ commute with each other, so both quantities are equal. □

A Necessity of T-commutativity for Lemma 3.6

Consider a set of events $\mathcal{B}^*$ with a dependency relation $\sim$. We say that $\mathcal{B}^*$ is *complete* if for each $\sigma \in \Omega$ there exists a flaw $h_\sigma = \{\sigma\} \in \mathcal{B}^*$, and with $h_\sigma \sim g$ for all $g \in \mathcal{B}^*$. Note that this definition is satisfied if $\mathcal{B}^*$ is generated by atomic events corresponding to permutations, perfect matchings of hypergraphs, or spanning trees.

We show now that if T-commutativity fails in a complete set of events, even for a single pair of flaws, then some wdags may appear with probability arbitrarily higher than their weight.

Theorem A.1. *Let $\mathcal{B}^*$ be a complete set of regenerating oracles which contains a pair $f, g \in \mathcal{B}^*$ with $f \nsim g$ and $A_f A_g \neq A_g A_f$. Then for any $C > 0$ there exists a set of flaws $\mathcal{B} \subseteq \mathcal{B}^*$ with $|\mathcal{B}| = 3$, wdag H with a single sink and a flaw resampling strategy $\mathcal{S}$ such that the probability that H appears in the execution of the algorithm is at least $C \cdot w(H) = C \cdot \prod_{v \in H} \mu(L(v))$.*

Proof. Consider states σ, τ with $A_f A_g[\sigma, \tau] \neq A_g A_f[\sigma, \tau]$. Denote $x = A_f A_g e_\tau$ and $y = A_g A_f e_\tau$, then $x[\sigma] \neq y[\sigma]$. Assume w.l.o.g. that $x[\sigma] < y[\sigma]$. Note that $\mu^\top A_f A_g = \mu^\top A_g A_f = \gamma_f \gamma_g \cdot \mu^\top$ since the oracles are regenerating, and therefore $\mu^\top x = \mu^\top y = \gamma_f \gamma_g \cdot \mu[\tau] = \gamma_f \gamma_g \gamma_h$.

Consider the following strategy $\mathcal{S}$ given a current state σ_1: (i) if $\sigma_1 \neq \sigma$ then prioritize flaws f, g, h at steps 1,2,3 respectively; (ii) if $\sigma_1 = \sigma$ then prioritize flaws g, f, h at steps 1,2,3 respectively. We say that the run *succeeds* if the sequence of addessed flaws is (f, g, h) in the first case and (g, f, h) in the second case. Clearly, the probability of success equals $e_{\sigma_1}^\top A_f A_g e_\tau = e_{\sigma_1}^\top x$ in the first case and $e_{\sigma_1}^\top A_g A_f e_\tau = e_{\sigma_1}^\top y$ in the second case. If σ_1 is distributed according to μ then the probability of success is

$$\begin{aligned} p &= \mu[\sigma] \cdot e_\sigma^\top y + \sum_{\sigma_1 \in \Omega - \{\sigma\}} \mu[\sigma_1] \cdot e_{\sigma_1}^\top x = \mu[\sigma] \cdot (e_\sigma^\top y - e_\sigma^\top x) + \sum_{\sigma_1 \in \Omega} \mu[\sigma_1] \cdot e_{\sigma_1}^\top x \\ &= \mu^\top x + \mu[\sigma] \cdot (y[\sigma] - x[\sigma]) > \gamma_f \gamma_g \gamma_h \end{aligned}$$

Furthermore, if the run succeeds then the last state is distributed according to μ (since step 3 resamples h at state τ, and the oracles are regenerating).

Now consider the trajectory T which repeats the sequence f, g, h for n times, and the corresponding wdag $H = G_{3n}^T$ which has a single sink node labeled h. Let $\mathcal{S}^n$ be the strategy $\mathcal{S}$ repeated cyclically. From the previous paragraph, the probabality that the run starting with some distribution μ produces H is given by $c_\mu \cdot p^{n-1}$, where c_μ depends only on the initial distribution. Note that $w(H) = (\gamma_f \gamma_g \gamma_h)^n$. Choosing n sufficiently large now gives the claim. □

B Distributional bound for permutation LLLL: Proof of Theorem 5.6

Consider the setting where Ω is the set of permutations on n letters and $\mathcal{A}$ is the set of atomic events $\pi x = y$, which we also write $\langle x, y \rangle$. The resampling oracle here, for such an event, is to update the state $\pi \leftarrow (y\ z)\pi$, where z is uniformly drawn $[n]$.

Throughout this section, let us fix event $E = \langle C \rangle$, for a stable set C. Consider a stable set $I \subseteq \mathcal{A}$. We can form a bipartite graph $G_{I,E}$ as follows: the left vertices correspond to C (we call these C-nodes), and the right vertices correspond to I (we call these I-nodes). There is an edge from (x, y) to (x', y') if $x = x'$ or $y = y'$. (In this case, for brevity, we write $(x, y) \sim (x', y')$.) Since I and C are stable, the graph $G_{I,E}$ has degree at most two. (Each (x', y') on the left may have an edge to some node (x', y) and to an a node of the form (x, y'), but no other nodes). So, $G_{I,E}$ decomposes into paths and cycles.

We define the *active conditions* $\mathrm{Active}(I)$ as follows. First, for each I-node (x', y'), we have $(x', y') \in \mathrm{Active}(I)$. Next, consider some maximal path starting and ending at C-nodes (which we call a *C-path*). The

path can be written (in one of its two orientations) as $(x_1, y_1), (x_1, y_2), (x_2, y_2), \ldots, (x_k, y_{k-1}), (x_k, y_k)$. In this case, we also have an active condition (x_k, y_1) in $\mathrm{Active}(I)$. (It is possible that $k = 1$ here, in which case (x_1, y_1) is an isolated C-node of $G_{I,E}$.) We say that a permutation π *satisfies* I if $\pi x = y$ for all $(x, y) \in \mathrm{Active}(I)$. We also write $a(I) = |\mathrm{Active}(I)|$.

The explanation for active conditions comes from the following observation:

Proposition B.1. *Consider a state $\pi \in \Omega$. If π does not satisfy I, then $e_\pi^\top A_I e_E = 0$. Otherwise, we have*

$$e_\pi^\top A_I e_E = \frac{(n - |C|)!}{n^{|I|}(n - a(I))!}$$

Proof. We show this by induction on $|I|$. In the base case $I = \emptyset$, A_I is the identity, and note that $\mathrm{Active}(I)$ is simply the set C. In this case, $e_\pi^\top A_I e_E$ is simply the indicator function that $\pi \in E$, which holds iff $\pi x = y$ for all $(x, y) \in \mathrm{Active}(I)$ iff $\pi x = y$ for all $(x, y) \in C$.

For the induction step, let us first show that $e_\pi^\top A_I e_E = 0$ if π does not satisfy I. First, suppose that $\pi x \neq y$ for some $f = \langle x, y \rangle \in I$. In this case, we can write $A_I = A_f A_{I-f}$. Since $e_\pi^\top A_f = 0$, we clearly have $e_\pi^\top A_I e_E = 0$.

Next, suppose that $\pi x_1 \neq y_k$ where $(x_1, y_1), (x_1, y_2), (x_2, y_2), \ldots, (x_k, y_{k-1}), (x_k, y_k)$ is a path starting at C-node (x_1, y_1) and ending at (x_k, y_k). If $k = 1$, then none of the flaws $f \in I$ are neighbors of event (x_1, y_1), and in particular if (x_1, y_1) is false on π then it is also false after resampling them all. So in this case, again if $\pi x_1 \neq y_k$ then $e_\pi^\top A_I e_E = 0$.

So let us assume that $k > 1$, and hence we know (x_1, y_2) is an I-node. Thus, as discussed above, we must have $\pi x_1 = y_2$. We can write $A_I = A_f A_{I-f} e_E$ where $f = \langle x_1, y_2 \rangle$, and so

$$e_\pi^\top A_I e_E = \sum_{\pi'} A_f[\pi, \pi'] e_{\pi'}^\top A_{I-f} e_E.$$

Consider some possible state π' here which can be obtained from π by resampling f. By induction hypothesis, the summand is zero unless π' satisfes $I - f$. Removing f from I changes the active conditions: now (x_1, y_1) becomes an isolated node in $G_{I-f,E}$, and there is a new maximal path starting at (x_2, y_2) which gives rise to an active condition (x_k, y_2).

We know that $\pi' = (y_1\ z_1)\pi$ for some value z_1. We also know that $\pi x_1 = y_2, \pi' x_1 = y_1$. This means that we must have $z_1 = y_2$ and hence $\pi' = (y_1\ y_2)\pi$. Also, π' must satisfy the active conditions $\pi' x_k = y_2$. Hence $\pi x_k = (y_1\ y_2)\pi' x_k = (y_1\ y_2) y_2 = y_1$ as desired.

So we have shown that the desired bound holds if π does not satisfy I. Suppose now that π satisfies I. There are a number of possible cases here:

1. Suppose as before that $G_{I,E}$ has a maximal path $(x_1, y_1), (x_1, y_2), (x_2, y_2), \ldots, (x_k, y_{k-1}), (x_k, y_k)$ starting and ending at C-nodes and $k > 1$. In this case, by the above argument, letting $f = \langle x_1, y_2 \rangle$, we have again:

   $$e_\pi^\top A_I e_E = \sum_{\pi'} A_f[\pi, \pi'] e_{\pi'}^\top A_{I-f} e_E.$$

   Since π satisfies $\pi x_k = y_1$ and $\pi x_1 = y_2$, there is precisely one possible term π' here, corresponding to $\pi' = (y_1\ y_2)\pi$, which satisfies $\mathrm{Active}(I - f)$. So by induction hypothesis we have

   $$e_\pi^\top A_I e_E = A_f[\pi, \pi'] \frac{(n - |C|)!}{n^{|I-f|}(n - a(I - f))!}$$

   Here, $a(I - f) = a(I)$ and $A_f[\pi, \pi'] = 1/n$. So this is $\frac{(n-|C|)!}{n^{|I|}(n-a(I))!}$, as claimed.

2. Suppose that $G_{I,E}$ contains a cycle, or contains a maximal path which begins at an I-node and terminates in a C-node. There are many different sub-cases here, including dependency on the whether the x-coordinates or y-coordinates change first. All these cases are completely analogous; for simplicity of exposition, we will consider just one of these cases.

   Suppose that $G_{I,E}$ contains a maximal path of the form $(x_1, y_2), (x_2, y_2), \ldots, (x_{k-1}, y_k), (x_k, y_k)$ where (x_1, y_2) is an I-node and (x_k, y_k) is a C-node Letting $f = \langle x_1, y_2 \rangle \in I$, we have in this case:

$$e_\pi^\top A_I e_E = \sum_{\pi'} A_f[\pi, \pi'] e_{\pi'}^\top A_{I-f} e_E.$$

   Consider $\pi' = (y_2\ z_2)\pi$. By induction hypothesis, π' must satisfy $\mathrm{Active}(I - f)$. Removing f destroys the active condition (x_1, y_2) but adds a new active condition (x_k, y_2) corresponding to the maximal path $(x_2, y_2), \ldots, (x_k, y_k)$. This is the only condition that could possibly be affected in π'. There is exactly one value z_2 which satisfies this condition. as $\pi' x_k = y_2$ iff $(y_2\ z_2)\pi x_k = y_2$ iff $\pi x_k = z_2$. As $a(I - f) = a(I)$ again, the calculation is precisely analogous to the previous case.

3. Suppose that $G_{I,E}$ has a maximal path $(x_1, y_1), (x_1, y_2), \ldots, (x_{k-1}, y_k), (x_k, y_k), (x_k, y_k)$ where (x_1, y_1) and (x_k, y_k) are I-nodes. Letting $f = \langle x_1, y_1 \rangle \in I$, we have again:

$$e_\pi^\top A_I e_E = \sum_{\pi'} A_f[\pi, \pi'] e_{\pi'}^\top A_{I-f} e_E.$$

   Removing this f destroys an active condition (x_1, y_1) and adds no new ones. Furthermore, in order for $\pi' = (y_1\ z_1)\pi$ to satisfy an active condition $(x, y) \in \mathrm{Active}(I - f)$, we must have $z_1 \neq y$ (as currently $\pi x = y$. Thus, there are precisely $n - a(I - f)$ choices for z_1.

   By induction hypothesis, for each such choice of π', the value of $e_{\pi'}^\top A_{I-f} e_E$ is $\frac{(n-|C|)!}{n^{|I-f|}(n-a(I-f))!} = \frac{(n-|C|)!}{n^{|I|-1}(n-a(I)+1)!}$. Summing over the $n - a(I) + 1$ choices for z_1, we get a total probability of

$$(n - a(I) + 1) \cdot \frac{1}{n} \cdot \frac{(n - |C|)!}{n^{|I|-1}(n - a(I) + 1)!} = \frac{(n - |C|)!}{n^{|I|}(n - a(I))!}$$ □

Proposition B.2. *If $f \in I$, then $e_\pi^\top A_f A_I e_E = \frac{1}{n} e_\pi^\top A_I e_E$ for any state π.*

Proof. Let $f = \langle x, y \rangle$, and consider state π. If $\pi x \neq y$, then both LHS and RHS are equal to zero (since (x, y) is an active condition of I). Suppose that π satisfies $\pi x = y$, and we resample π to $\pi' = (y\ z)\pi$. We can write

$$e_\pi^\top A_f A_I e_E = \sum_{\pi'} A_f[\pi, \pi'] e_{\pi'}^\top A_I e_E.$$

By Proposition B.1, the summand is non-zero only if π' satisfies $\mathrm{Active}(I)$, and in particular satisfies $\pi' x = y$. This occurs only if $z = y$ in which case $\pi' = \pi$ with summand $\frac{1}{n} e_\pi^\top A_I e_E$. □

Proposition B.3. *Consider a stable set $I = \{\langle F_1 \rangle, \ldots, \langle F_k \rangle\}$ of $\overline{\mathcal{A}}$ and let $J = F_1 \cup \cdots \cup F_k$. For any state π, we have*

$$e_\pi^\top A_I e_E = n^{|J|} \prod_{i=1}^{k} \frac{(n - |F_i|)!}{n!} e_\pi^\top A_J e_E$$

Proof. We show this by induction on k. Let us define $f_i = \langle F_i \rangle$ for each i. Case $k = 0$ holds vacuously. For the induction step, let $I' = \{\langle F_1 \rangle, \ldots, \langle F_{k-1} \rangle\}$ and $J' = F_1 \cup \cdots \cup F_{k-1}$. By induction hypothesis, we can write

$$e_\pi^\top A_I e_E = e_\pi^\top A_{f_k} A_{I'} e_E = e_\pi^\top n^{|J'|} A_{f_k} \prod_{i=1}^{k-1} \frac{(n - |F_i|)!}{n!} A_{J'} e_E$$

Let us write $F_k = \{g_1, \ldots, g_t\}$. By Proposition 7.3, we have $A_{f_k} = cA_{g_1} \ldots A_{g_t} = cA_{F_k}$, where the scalar constant c is given by

$$c = \prod_{i=1}^{t} \frac{1}{\Pr_{r \in R_{g_i}}(r \in R_{g_i}; g_{i+1}, \ldots, g_t)} = \frac{n^t (n-t)!}{n!}$$

Thus, we have

$$e_\pi^\top A_I e_E = e_\pi^\top n^{|J'|} A_{J'} \prod_{i=1}^{k-1} \frac{(n - |F_i|)!}{n!} \cdot \frac{n^t (n-t)!}{n!} A_{F_k} e_E$$

We have $J = J' \cup F_k$, and $A_{J'} A_{F_k} = A_J \prod_{g \in F_k \cap J'} A_g$ and $t = |F_{k+1}|$, so we can write this as:

$$e_\pi^\top A_I e_E = n^{t+|J'|} \prod_{i=1}^{k} \frac{(n - |F_i|)!}{n!} e_\pi^\top \Big(\prod_{g \in F_k \cap J} A_g \Big) A_J e_E \tag{9}$$

By Proposition B.2, we have $e_\pi^\top A_g A_J e_E = \frac{1}{n} A_J e_E$ for each $g \in F_k \cap J'$. Hence, we have

$$e_\pi^\top A_J \prod_{g \in F_k \cap J} A_g e_E = n^{-|F_k \cap J'|} e_\pi^\top A_J e_E,$$

and substituting back into Eq. (9), we have:

$$e_\pi^\top A_I e_E = n^{t+|J'|-|F_k \cap J'|} \prod_{i=1}^{k} \frac{(n - |F_i|)!}{n!} e_\pi^\top A_J$$

To finish the induction, observe that $t + |J'| - |F_k \cap J'| = |F_k| + |J'| - |F_k \cap J'| = |F_k \cup J'| = |J|$. □

Proposition B.4. *Consider a stable set $I \subseteq \mathcal{A}$ and some $f \in \mathcal{A}$ with $f \not\sim I$. Let $I' = I \cup \{f\}$. Then exactly one of the two conditions holds: (i) $a(I') = a(I)$ and there is a C-path with ane endpoint $g \sim f$; or (ii) $a(I') = a(I) + 1$ and $\mathrm{Active}(I) \subseteq \mathrm{Active}(I')$.*

Proof. Suppose that $f = \langle x, y \rangle$ and let $I' = I \cup \{f\}$. Every active condition corresponding to a I-node in $G_{I,E}$ is preserved in $G_{I',E}$, plus there will be one new active condition corresponding to (x, y).

The only way that $G_{I',E}$ could gain an additional active condition, beyond this one, is if (x, y) participates in a C-path. In this case, $G_{I,E}$ would contain C-paths with endpoints (x'', y''), (x, y') and (x', y), (x''', y''') respectively. Then the two active conditions (x, y'') and (x''', y) are removed in $\mathrm{Active}(I)$, replaced by two new active conditions (x, y) and (x''', y'') in $\mathrm{Active}(I')$. So $a(I') = a(I)$.

The only way that $G_{I',E}$ could lose an active condition would be if $G_{I,E}$ has a C-path with an endpoint $g \sim f$. We have already discussed what occurs if there are two such paths. If there is just one, then this is the only additional active condition lost in I', and so $a(I') = a(I)$.

In other cases, we have $\mathrm{Active}(I') = \mathrm{Active}(I) \cup \{(x, y)\}$ and the result holds. □

Proposition B.5. *Let $I = \{\langle F_1 \rangle, \ldots, \langle F_k \rangle\}$ be a stable set in $\overline{\mathcal{A}}$ and let $f = \langle F_{k+1} \rangle \in \overline{\mathcal{A}}$ where $f \not\sim I$. Consider the stable sets $J = F_1 \cup \cdots \cup F_k$ and $J' = J \cup F_{k+1}$ of $\mathcal{A}$. If $a(J') = a(J) + |J' - J|$, then f is dominated by I in $\overline{\mathcal{A}}$.*

Proof. Consider stable set $I' = I \cup \{f\} = \{\langle F_1\rangle, \ldots, \langle F_{k+1}\rangle\}$. We want to show that $e_\pi^\top A_{I'} e_E \le e_\pi^\top A_I e_E$ for any state π. By Proposition B.3, this is equivalent to showing that

$$n^{|J'|} \prod_{i=1}^{k+1} \frac{(n-|F_i|)!}{n!} e_\pi^\top A_{J'} e_E \le n^{|J|} \prod_{i=1}^{k} \frac{(n-|F_i|)!}{n!} e_\pi^\top A_J e_E$$

If we define $t = |J' - J|$, and divide common terms, this is equivalent to showing that:

$$\frac{e_\pi^\top A_{J'} e_E}{e_\pi^\top A_J e_E} \le \frac{n!}{n^t (n-|F_{k+1}|)!} \tag{10}$$

By Proposition B.4, the only way to have $a(J') = a(J) + t$ is to have $\mathrm{Active}(J) \subseteq \mathrm{Active}(J')$. In this case, by Proposition B.1, we have $e_\pi^\top A_{J'} e_E = 0 = e_\pi^\top A_J e_E$ if π does not satisfy J. If π does satisfy J, then by Proposition B.4 we have $e_\pi^\top A_J e_E = \frac{(n-|C|)!}{n^{|J|}(n-a(J))!}$ and also

$$e_\pi^\top A_{J'} e_E \le \frac{(n-|C|)!}{n^{|J'|}(n-a(J'))!} = \frac{(n-|C|)!}{n^{|J|+t}(n-a(J)-t)!}$$

Overall, we have

$$\frac{e_\pi^\top A_{J'} e_E}{e_\pi^\top A_J e_E} \le \frac{\frac{(n-|C|)!}{n^{|J|+t}(n-a(J)-t)!}}{\frac{(n-|C|)!}{n^{|J|}(n-a(J))!}} = \frac{(n-a(J))!}{n^t (n-a(J)-t)!} \le \frac{n!}{n^t (n-t)!} \le \frac{n!}{n^t (n-|F_{k+1}|)!}$$

which satisfies Eq. (10). □

Proposition B.6. *For any stable set $I \in \mathfrak{I}(E)$, there is an injective function $\phi_I : I \to C$ which satisfies the property that $g \sim \phi_I(g)$ for all $g \in I$.*

Proof. By definition, I can be ordered as $I = \{\langle F_1\rangle, \ldots, \langle F_k\rangle\}$ such that each $\langle F_i\rangle$ is not dominated by $\{F_1, \ldots, F_{i-1}\}$. Let us define $J_i = F_1 \cup \cdots \cup F_i$ for each i. By Proposition B.5, we must have $a(J_i) \ne a(J_{i-1}) + |F_i|$ for each value i, as otherwise F_i would be dominated by $\{\langle F_1\rangle, \ldots, \langle F_{i-1}\rangle\}$.

By Proposition B.4, there must be some $g_i \in F_i - J_{i-1}$ with $a(J_i \cup \{g_i\}) = a(J_i)$ and so the graph $G_{J_i,E}$ has a C-path P ending at a node $f \sim g_i$. This path P corresponds to some a C-path P', which is a subsequence of P, in the graph $G_{\{g_1,\ldots,g_{i-1}\},E}$, and so also have $a(\{g_1, \ldots, g_{i-1}\}) = a(\{g_1, \ldots, g_i\})$. Since $a(\emptyset) = |C|$, this implies that $a(\{g_1, \ldots, g_i\}) = |C|$ for all i.

Let $H = \{g_1, \ldots, g_k\}$. Now, all the paths in graph $G_{H,E}$ must be C-paths. For any such C-path of the form $h_1, g_{i_1}, h_2, g_{i_2}, \ldots, g_{i_s}, h_{s+1}$, we can select $\phi_I(g_{i_1}) = h_1, \ldots, \phi_I(g_{i_s}) = h_{h+1}$. □

We can now show Theorem 5.6. Let $C = \{g_1, \ldots, g_t\}$. First, we have $\mu(E) = \frac{(n-t)!}{n!}$. Next, consider enumerating a set $I \in \mathfrak{I}(E)$. By Proposition B.6, we can choose, for each $g \in C$, to either include some corresponding preimage $f = \phi_I^{-1}(g)$ in I, or not include it. Let us write $I_g = \emptyset$ if there is no such f, or $I_g = \{f\}$ for such f. Thus $I = \bigcup_{g \in C} I_g$. Overall, this shows that

$$\sum_{I \in \mathfrak{I}(E)} \Psi(I) \le \sum_{I_{g_1}, \ldots, I_{g_t}} \Psi(I_{g_1} \cup \cdots \cup I_{g_t}) \le \sum_{I_{g_1}, \ldots, I_{g_t}} \Psi(I_{g_1}) \cdots \Psi(I_{g_t})$$

where the last inequality follows from log-subadditivity of Ψ. This can be written as $\prod_{i=1}^{t} \sum_{I_{g_i}} \Psi(I_{g_i})$. The case of $I_{g_i} = \emptyset$ contributes 1, and the case of $I_{g_i} = \{f\}$ contributes $\Psi(f)$.

References

[1] Dimitris Achlioptas and Fotis Iliopoulos. Random walks that find perfect objects and the Lovász local lemma. *Journal of the ACM*, 63(3):Article #22, 2016.

[2] Dimitris Achlioptas, Fotis Iliopoulos, and Vladimir Kolmogorov. A local lemma for focused stochastic algorithms. *SIAM Journal on Computing*, 48(5):1583–1602, 2019.

[3] Dimitris Achlioptas, Fotis Iliopoulos, and Alistair Sinclair. Beyond the Lovász local lemma: Point to set correlations and their algorithmic applications. In *Proc. 60th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 725–744, 2019.

[4] Karthekeyan Chandrasekaran, Navin Goyal, and Bernhard Haeupler. Deterministic algorithms for the Lovász local lemma. *SIAM Journal on Computing*, 42(6):2132–2155, 2013.

[5] Antares Chen, David G. Harris, and Aravind Srinivasan. Partial resampling to approximate covering integer programs. *Random Structures & Algorithms*, pages 69–93, 2021.

[6] Kai-Min Chung, Seth Pettie, and Hsin-Hao Su. Distributed algorithms for the Lovász local lemma and graph coloring. *Distributed Computing*, 30(4):261–280, 2017.

[7] Paul Erdős and László Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. In *Infinite and finite sets (Colloq., Keszthely, 1973; dedicated to P. Erdős on his 60th birthday), Vol. II*, pages 609–627. Colloq. Math. Soc. János Bolyai, Vol. 10. 1975.

[8] Bernhard Haeupler and David G. Harris. Parallel algorithms and concentration bounds for the Lovász local lemma via witness DAGs. *ACM Transactions on Algorithms (TALG)*, 13(4):Article #25, 2017.

[9] Bernhard Haeupler, Barna Saha, and Aravind Srinivasan. New constructive aspects of the Lovász local lemma. *Journal of the ACM*, 58(6):Article #28, 2011.

[10] David G. Harris. Lopsidependency in the Moser-Tardos framework: Beyond the lopsided Lovász local lemma. *ACM Transactions on Algorithms*, 13(1):Article #17, 2016.

[11] David G. Harris. Oblivious resampling oracles and parallel algorithms for the Lopsided Lovász Local Lemma. In *Proc. 30th annual ACM-SIAM Symposium on Discrete Algorithm (SODA)*, pages 841–860, 2019.

[12] David G. Harris. New bounds for the Moser-Tardos distribution. *Random Structures & Algorithms*, 57(1):97–131, 2020.

[13] David G. Harris and Aravind Srinivasan. Algorithmic and enumerative aspects of the Moser-Tardos distribution. *ACM Transactions on Algorithms*, 13(3):Article #33, 2017.

[14] David G. Harris and Aravind Srinivasan. A constructive Lovász Local Lemma for permutations. *Theory of Computing*, 13(1):Article #17, 2017.

[15] David G. Harris and Aravind Srinivasan. The Moser–Tardos framework with partial resampling. *Journal of the ACM*, 66(5):Article #36, 2019.

[16] Nicholas J. A. Harvey and Jan Vondrák. An algorithmic proof of the Lovász local lemma via resampling oracles. *SIAM Journal on Computing*, 49(2):394–428, 2020.

[17] Fotis Iliopoulos. Commutative algorithms approximate the LLL-distribution. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, 2018.

[18] Fotis Iliopoulos and Alistair Sinclair. Efficiently list-edge coloring multigraphs asymptotically optimally. In *Proc. 14th annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2319–2336, 2020.

[19] Kashyap Kolipaka and Mario Szegedy. Moser and Tardos meet Lovász. In *Proc. 43rd annual ACM Symposium on Theory of Computing (STOC)*, pages 235–244, 2011.

[20] Kashyap Kolipaka, Mario Szegedy, and Yixin Xu. A sharper local lemma with improved applications. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 603–614. 2012.

[21] Vladimir Kolmogorov. Commutativity in the algorithmic Lovász local lemma. *SIAM Journal on Computing*, 47(6):2029–2056, 2018.

[22] Robin A. Moser and Gábor Tardos. A constructive proof of the general Lovász local lemma. *Journal of the ACM*, 57(2):Article #11, 2010.

[23] Wesley Pegden. An extension of the Moser-Tardos algorithmic local lemma. *SIAM Journal on Discrete Mathematics*, 28(2):911–917, 2014.

[24] J.B. Shearer. On a problem of Spencer. *Combinatorica*, 5(3):241–245, 1985.