Alternative proofs of the asymmetric Lovász local lemma and Shearer's lemma

Ioannis Giotis
Department of Mathematics, National and Kapodistrian University of Athens
igiotis@cs.upc.edu

Lefteris Kirousis
Department of Mathematics, National and Kapodistrian University of Athens
lkirousis@math.uoa.gr

John Livieratos
Department of Mathematics, National and Kapodistrian University of Athens
jlivier89@math.uoa.gr

Kostas I. Psaromiligkos
Department of Mathematics, University of Chicago
kostaspsa@math.uchicago.edu

Dimitrios M. Thilikos
Department of Mathematics, National and Kapodistrian University of Athens
and AlGCo project, CNRS, LIRMM, France
sedthilk@thilikos.info

Abstract

We provide new algorithmic proofs for two forms of the Lovász Local Lemma: the Asymmetric version and Shearer's Lemma. Our proofs directly compute an upper bound for the probability that the corresponding Moser-type algorithms last for at least n steps. These algorithms iteratively sample the probability space; when and if they halt, a correct sampling, i.e. one where all undesirable events are avoided, is obtained. Our computation shows that this probability is exponentially small in n. In contrast most extant proofs for the Lovász Local Lemma and its variants use counting arguments that give estimates of only the expectation that the algorithm lasts for at least n steps. For the asymmetric version, we use the results of Bender and Richmond on the multivariable Lagrange inversion. For Shearer's Lemma, we follow the work of Kolipaka and Szegedy, combined with Gelfand's formula for the spectral radius of a matrix.

1 Introduction

Suppose we have a number of "undesirable" events, defined on a common probability space, that we want to avoid. There are various *sufficient* conditions that guarantee that we can. One of the most useful and celebrated

In: A. Editor, B. Coeditor (eds.): Proceedings of the XYZ Workshop, Location, Country, DD-MMM-YYYY, published at http://ceur-ws.org

such conditions, is the Lovász Local Lemma (LLL), which first appeared in a paper by Erdős and Lovász [EL75] in 1975. The sufficient conditions of LLL and its variants require that the probabilities of the undesirable events be upper bounded in terms of parameters of a graph that expresses the dependencies between the events (for a survey, see [Sze13]). The proofs of the original results were non-algorithmic

Essentially, the first algorithmic proof of a special case of the *symmetric* LLL, where the sufficient condition requires a uniform bound for the probabilities of all events in terms of the number of dependencies of each event, was given by Moser [Mos09]. The first algorithmic proof of the *asymmetric* version, where the condition requires the existence of a number in $(0, 1)$ for each event, such that the probability of each event to occur is bounded by an expression of these numbers, was given by Moser and Tardos [MT10]. These algorithmic results were expressed in the *variable framework*, where each event is assumed to depend on a number of *independent random variables*. The algorithms iteratively sample the variables; when and if they stop, a sampling where all undesirable events are avoided is obtained. Such algorithms were shown to come to a halt using a counting argument that became known as the "entropic method" [Tao09].

In this paper, we first prove the asymmetric LLL by directly computing an upper bound for the probability that the Moser-type algorithms last for at least n steps. This probability is expressed by a *recurrence relation*, which we solve by employing the result of Bender and Richmond on the multivariable *Lagrange inversion formula* [BR98]. Recurrence solving was also utilized for the symmetric LLL by Giotis et al. in [GKPT15]. Previous proofs in the literature that were based on counting arguments gave an estimate of only the expectation of the algorithm lasting for at least n steps. Furthermore, we give an algorithmic proof of the most general form of LLL, that of Shearer's Lemma [She85]. This result gives a sufficient and *necessary* condition for avoiding all the undesirable events. Following closely the work of Kolipaka and Szegedy [KRS11], we again express the running time of a probabilistic algorithm by a recurrence relation, that we solve using *Gelfand's formula* for the *spectral radius* of a matrix (see [HJ90]).

The variable setting framework:

Let X_i, $i = 1, ..., l$ be mutually independent random variables on a common probability space, taking values in the finite sets D_i, $i = 1, ..., l$ respectively. Suppose also that $E_1, \ldots, E_m$ are "undesirable" events. The *scope* $\mathrm{sc}(E_j)$ of an event E_j is the minimal subset of variables such that one can determine whether E_j occurs or not knowing only their values. Events are assumed to be ordered according to their index. Our aim is to find an assignment of values such that none of the events occur.

A dependency graph is now defined as follows: its vertex set is $\{1, ..., m\}$; two vertices i, j are connected with an edge if $\mathrm{sc}(E_i) \cap \mathrm{sc}(E_j) \neq \emptyset$. We denote the *neighborhood*, in the dependency graph, of an event E_j by N_j (by assumption $j \notin N_j$). We also assume that $N_j \neq \emptyset$, $j = 1, \ldots, m$. This is a natural assumption, since it can be easily verified that "isolated" events can be ignored.

2 Asymmetric Lovász Local Lemma

Our first objective is to provide an algorithmic proof of the following theorem:

Theorem 2.1 (Asymmetric Lovász Local Lemma)**.** *Suppose that there exist* $\chi_1, \chi_2, \ldots, \chi_m \in (0, 1)$*, such that*

$$\Pr(E_j) \leq \chi_j \prod_{i \in N_j} (1 - \chi_i),$$

for all $j \in \{1, \ldots, m\}$*. Then,* $\Pr[\overline{E}_1 \wedge \overline{E}_2 \wedge \cdots \wedge \overline{E}_m] > 0$.

To proceed with the proof of Theorem 2.1, consider M-Algorithm below. It successively produces random assignments, by resampling the variables in the scopes of occurring events, until it finds one under which no undesirable event occurs. When the variables in the scope of an occurring event E_j are resampled, the algorithm checks if E_j still occurs (lines 2 and 3 of the Resample routine) and, in case it doesn't, proceeds to check its neighborhood.

A *round* is the duration of any Resample call during an execution of M-Algorithm. A Resample call made from line 3 of the main algorithm is a *root call*, while one made from within another call is a *recursive call*.

Lemma 2.2. *Consider an arbitrary call of* Resample(E_j)*. Let* $\mathcal{X}_j$ *be the set of events that do not occur at the start of this call. Then, if and when this call terminates, all events in* $\mathcal{X}_j \cup \{E_j\}$ *do not occur.*

Algorithm 1 M-Algorithm.

1: Sample the variables X_i, $i = 1, ..., l$ and let α be the resulting assignment.
2: **while** there exists an event that occurs under the current assignment, let E_j be the least indexed such event and **do**
3: Resample(E_j)
4: **end while**
5: Output current assignment α.

Resample(E_j)

1: Resample the variables in sc(E_j).
2: **if** E_j occurs **then**
3: Resample(E_j)
4: **else**
5: **while** some event whose index is in N_j occurs under the current assignment, let E_k be the least indexed such event and **do**
6: Resample(E_k)
7: **end while**
8: **end if**

Proof. For the purposes of this proof, say that Resample(E_j) is the *main* call, suppose it terminates and that α is the produced assignment of values. Furthermore, suppose that $E_k \in \mathcal{X}_j \cup \{E_j\}$ and that E_k occurs under α.

Let $E_k \in \mathcal{X}_j$. Then, under the assignment at the beginning of the main call, E_k did not occur. Thus, it must be the case that at some point during this call, a resampling of some variables caused E_k to occur. Let Resample(E_s) be the last time E_k became occurring, and thus remained occurring until the end of the main call. During Resample(E_s), only the variables in sc(E_s) were resampled. Thus, E_k is in the neighborhood of E_s. But then, by line 5 of the Resample routine, Resample(E_s) couldn't have terminated and thus, neither could the main call. Contradiction.

Thus $E_k = E_j$. Since under the assignment at the beginning of the main call, E_j occurred, by lines 2 and 3 of the Resample routine, it must be the case that during some resampling of the variables in sc(E_j), E_j became non-occurring. The main call could not have ended after this resampling, since E_j occurs under the assignment α produced at the end of this call. Then, there exists some $r \in N_j$ such that Resample(E_r) is the subsequent Resample call. Thus $E_j \in \mathcal{X}_r$ and we obtain a contradiction as in the case where $E_k \in \mathcal{X}_j$ above. □

An immediate corollary of Lemma 2.2, is that the events of the root calls of Resample are pairwise distinct, therefore there can be *at most* m such root calls in any execution of M-Algorithm.

Consider now *rooted forests*, i.e. forests of trees such that each tree has a special node designated as its root, whose vertices are labeled by events E_j, $j \in \{1, \ldots, m\}$. We will use such forests to depict the executions of M-Algorithm.

Definition 2.3. A labeled rooted forest $\mathcal{F}$ is called *feasible* if:

1. the labels of its roots are *pairwise distinct*,
2. the labels of any two siblings (i.e. vertices with a common parent) are distinct and
3. an internal vertex labeled by E_j has either one child labeled again by E_j or at most $|N_j|$ children, with labels whose indices are in N_j.

The number of nodes of a feasible forest $\mathcal{F}$ is denoted by $|\mathcal{F}|$.

The nodes of such a labeled forest are ordered as follows: children of the same node are ordered as their labels are; nodes in the same tree are ordered by preorder (respecting the ordering between siblings) and finally if the label on the root of a tree T_1 precedes the label of the root of T_2, all nodes of T_1 precede all nodes of T_2.

Given an execution of M-Algorithm that lasts for *at least* n rounds, we construct, in a unique way, a feasible forest with n nodes, by creating one node for each Resample call and labeling it with its argument, where the root calls correspond to the roots of the trees and a recursive call made from line 3 or 6 of a Resample(E_j) call gives rise to a child of the corresponding node of this Resample(E_j) call. We say that a feasible forest $\mathcal{F}$

constructed this way is the *n-witness* forest of M-Algorithm's execution and we define $W_{\mathcal{F}}$ to be the event M-Algorithm executes with n-witness forest $\mathcal{F}$.

Define P_n to be the probability that M-Algorithm lasts for *at least* n rounds. It is easy to see that:

$$P_n = \Pr\left[\bigcup_{\mathcal{F}:|\mathcal{F}|=n} W_{\mathcal{F}}\right] = \sum_{\mathcal{F}:|\mathcal{F}|=n} \Pr\left[W_{\mathcal{F}}\right], \tag{1}$$

where the last equality holds because the events $W_{\mathcal{F}}$ are disjoint.

It is easy to see that M-Algorithm introduces various dependencies to the probabilistic calculations. For example, suppose that the i-th node of a witness forest $\mathcal{F}$ is labeled by E_j and its children have labels with indices in N_j. Then, under the assignment produced at the end of the i-th round of this execution, E_j does not occur.

To avoid such dependencies, we introduce ValAlg below.

Algorithm 2 ValAlg.

Input: Feasible forest $\mathcal{F}$ with labels $E_{j_1}, \ldots, E_{j_n}$.

1: Sample the variables X_i, $i = 1, ..., l$.
2: **for** s=1,. . . ,n **do**
3: **if** E_{j_s} does not occur under the current assignment **then**
4: **return** `failure` and exit.
5: **else**
6: Resample the variables in sc(E_{j_s})
7: **end if**
8: **end for**
9: **return** `success`.

A round of ValAlg is the duration of any **for** loop executed at lines 2-8. If it manages to go through its input without coming upon a non-occurring event at any given round, it returns `success`.Thus, the success or failure of ValAlg is not conditioned on whether it produces an assignment such that no event occurs.

The following result concerns the distribution of the random assignments at any round of ValAlg.

Lemma 2.4 (Randomness Lemma)**.** *At the beginning of any given round of* ValAlg*, the distribution of the current assignment of values to the variables X_i, $i = 1, ..., l$, given that* ValAlg *has not failed, is as if all variables have been sampled anew.*

Proof. This follows from the fact that at each round, the variables for which their values have been exposed, are immediately resampled. □

Now, given a feasible forest $\mathcal{F}$ with n nodes, we say that $\mathcal{F}$ is validated by ValAlg if the latter returns `success` on input $\mathcal{F}$. The event of this happening is denoted by $V_{\mathcal{F}}$. We also set:

$$\hat{P}_n = \sum_{\mathcal{F}:\ |\mathcal{F}|=n} \Pr[V_{\mathcal{F}}]. \tag{2}$$

Lemma 2.5. *For any feasible forest $\mathcal{F}$, the event $W_{\mathcal{F}}$ implies the event $V_{\mathcal{F}}$, therefore $P_n \leq \hat{P}_n$.*

Proof. Indeed, if the random choices made by an execution of M-Algorithm that produces as witness forest $\mathcal{F}$ are made by ValAlg on input $\mathcal{F}$, then clearly ValAlg will return `success`. □

By Lemma 2.5, it suffices to prove that $\hat{P}_n$ is exponentially small to n.

For notational convenience, suppose that the neighborhood of an event E_j is $N_j := \{j_1, \ldots, j_{k_j}\}$, $j = 1, \ldots, m$ (recall that N_j is a set of indices of events). Now, let $\mathbf{n} = (n_1, \ldots, n_{2m})$, $n_1, \ldots, n_{2m} \geq 0$ be such that $\sum_{i=1}^{2m} n_i = n$ and $\mathbf{n} - (1)_j := (n_1, \ldots, n_j - 1, \ldots, n_{2m})$. Define, for $j = 1, \ldots, m$, the *multivariate generating functions*:

$$Q_j(\mathbf{t}) = \sum_{\mathbf{n}:n_j \geq 1} Q_{\mathbf{n},j}\mathbf{t}^{\mathbf{n}} \text{ and } R_j(\mathbf{t}) = \sum_{\mathbf{n}:n_{m+j} \geq 1} R_{\mathbf{n},j}\mathbf{t}^{\mathbf{n}}, \tag{3}$$

where $\mathbf{t} = (t_1, \ldots, t_{2m})$, $\mathbf{t}^{\mathbf{n}} := t_1^{n_1} \cdots t_{2m}^{n_{2m}}$ and:

$$Q_{\mathbf{n},j} = \Pr[E_j]\Big(Q_{\mathbf{n}-(1)_j,j} + R_{\mathbf{n}-(1)_j,j}\Big), \tag{4}$$

$$R_{\mathbf{n},j} = \Pr[E_j] \cdot \sum_{\mathbf{n}^1+\cdots+\mathbf{n}^{k_j}=\mathbf{n}-(1)_{m+j}} \Big(Q_{\mathbf{n}^1,j_1} + R_{\mathbf{n}^1,j_1}\Big) \cdots \Big(Q_{\mathbf{n}^{k_j},j_{k_j}} + R_{\mathbf{n}^{k_j},j_{k_j}}\Big) \tag{5}$$

and where $Q_{\mathbf{n},j} = 0$ (resp. $R_{\mathbf{n},j} = 0$) when $n_j = 0$ (resp. $n_{m+j} = 0$) and there exists an $i \neq j$ (resp. $i \neq m+j$) such that $n_i \geq 1$ and $Q_{\mathbf{0},j} = R_{\mathbf{0},j} = 1$, where $\mathbf{0}$ is a sequence of $2m$ zeroes.

Since a *rooted tree* is trivially a rooted forest, Definition 2.3 applies accordingly. Also, V_T is the event the validation algorithm succeeds on input the feasible tree T.

Now, consider the coefficients of the generating functions in (3). It is not difficult to see that, if T_1, T_2 are two feasible trees with $n = \sum_{i=1}^{2m} n_i$ nodes each, whose roots are labeled with E_j and where, T_1's root has a unique child labeled by E_j, whereas T_2's root has children labeled by indices in N_j, then:

$$Pr[V_{T_1}] \leq Q_{\mathbf{n},j} \text{ and } Pr[V_{T_2}] \leq R_{\mathbf{n},j}.$$

Also, observe that:

$$\hat{P}_n \leq \sum_{\mathbf{n}} \sum_{\mathbf{n}^1+\ldots+\mathbf{n}^m=\mathbf{n}} \Big(Q_{\mathbf{n}^1,1} + R_{\mathbf{n}^1,1}\Big) \cdots \Big(Q_{\mathbf{n}^m,m} + R_{\mathbf{n}^m,m}\Big).$$

Our aim is to show that both $Q_{\mathbf{n},j}$ and $R_{\mathbf{n},j}$ are exponentially small to n, for any given sequence of $\mathbf{n}$. Thus, by ignoring polynomial factors, the same will hold for $\hat{P}_n$.

As an intuitive point for $\mathbf{n} = (n_1, \ldots, n_{2m})$, note that, in a feasible tree T, n_j corresponds to the numbers of nodes u of T that have a unique child so that both u and its child are labeled with Ej, $j = \{1, \ldots, 2m\}$. On the other hand, n_{m+j} corresponds to the number of nodes labeled by E_j whose children are labeled by indices in N_j, $j = 1, \ldots, m$.

By multiplying both sides of (4) and (5) by $\mathbf{t}^n$ and adding all over suitable $\mathbf{n}$, we get the system of equations $(\mathbf{Q}, \mathbf{R})$:

$$\begin{aligned} Q_j(\mathbf{t}) =& t_j f_j((\mathbf{Q}, \mathbf{R})), \\ R_j(\mathbf{t}) =& t_{m+j} f_{m+j}((\mathbf{Q}, \mathbf{R})), \end{aligned} \tag{6}$$

where, for $\mathbf{x} = (x_1, \ldots, x_{2m})$ and $j = 1 \ldots, m$:

$$f_j(\mathbf{x}) = \chi_j \left(\prod_{i \in N_j} (1 - \chi_i) \right) (x_j + x_{m+j} + 2), \tag{7}$$

$$f_{m+j}(\mathbf{x}) = \chi_j \prod_{i \in N_j} (1 - \chi_i)(x_i + x_{m+i} + 2). \tag{8}$$

To solve the system, we will directly use the result of Bender and Richmond in [BR98] (Theorem 2). Let $g := pr_s^{2m}$ be the $2m$-ary projection on the s-th coordinate and let $\mathcal{B}$ be the set of trees $B = (V(B), E(B))$ whose vertex set is $\{0, 1, \ldots, 2m\}$ and with edges directed towards 0. By [BR98], we get:

$$[\mathbf{t}^{\mathbf{n}}] g((\mathbf{Q}, \mathbf{R})(\mathbf{t})) = \frac{1}{\prod_{j=1}^{2m} n_j} \sum_{B \in \mathcal{B}} [\mathbf{x}^{\mathbf{n-1}}] \frac{\partial(g, f_1^{n_1}, \ldots, f_{2m}^{n_{2m}})}{\partial B}, \tag{9}$$

where the term for a tree $B \in \mathcal{B}$ is defined as:

$$[\mathbf{x}^{\mathbf{n-1}}] \prod_{r \in V(B)} \left\{ \left(\prod_{(i,r) \in E(B)} \frac{\partial}{\partial x_i} \right) f_r^{n_r}(\mathbf{x}) \right\}, \tag{10}$$

where $r \in \{0, \ldots, 2m\}$ and $f_0^{n_0} := g$.

We consider a tree $B \in \mathcal{B}$ such that (10) is not equal to 0. Thus, $(i, 0) \neq E(B)$, for all $i \neq s$. On the other hand, $(s, 0) \in E(B)$, lest vertex 0 is isolated, and each vertex has out-degree *exactly* one, lest a cycle is formed

or connectivity is broken. From vertex 0, we get $\frac{\partial pr_s^{2m}(\mathbf{x})}{\partial x_s} = 1$. Since we are are interested only in factors of (10) that are exponential in n, we can ignore the derivatives (except the one for vertex 0), as they introduce only polynomial (in n) factors to the product. Thus, we have that (10) is equal to the coefficient of $\mathbf{x^{n-1}}$ in:

$$\prod_{j=1}^{m}\left\{\left(\chi_j^{n_j}\prod_{i\in N_j}(1-\chi_i)^{n_j}\right)(x_j+x_{m+j}+2)^{n_j}\left(\chi_j^{n_{m+j}}\prod_{i\in N_j}(1-\chi_i)^{n_{m+j}}(x_i+x_{m+i}+2)^{n_{m+j}}\right)\right\}. \tag{11}$$

We will say that the first part of (11) is the one with the factors whose exponents are n_j and the second, those whose exponents are n_{m+j}, $j = 1, \ldots, n$.

We now group the factors of each part of (11) separately, according to the i's. We have already argued each vertex i has out-degree 1. Note also that the j's such that $i \in N_j$ are exactly the $j \in N_i$. Thus, the exponent of the term $x_i + x_{m+i} + 2$ in the first part of (11) is n_i and in the second, $\sum_{j\in N_i} n_{m+j}$. Taking all this together, we get that the product of (11) is equal to:

$$\prod_{i=1}^{m}\left\{\left(\chi_i^{n_i}(1-\chi_i)^{\sum_{j\in N_i} n_j}(x_i+x_{m+i}+2)^{n_i}\right)\left(\chi_i^{n_{m+i}}(1-\chi_i)^{\sum_{j\in N_i} n_{m+j}}(x_i+x_{m+i}+2)^{\sum_{j\in N_i} n_{m+j}}\right)\right\}. \tag{12}$$

Using the binomial theorem and by ignoring polynomial factors, we get that the coefficient of $\mathbf{x^{n-1}}$ in (12) is:

$$\prod_{i=1}^{m}\left(\chi_i^{n_i}(1-\chi_i)^{\sum_{j\in N_i} n_j}\binom{n_i}{n_i}\right)\left((1-\chi_i)^{n_{m+i}}\chi_i^{n_{m+i}}(1-\chi_i)^{\sum_{j\in N_i} n_{m+j}-n_{m+i}}\binom{\sum_{j\in N_i} n_{m+j}}{n_{m+i}}\right). \tag{13}$$

By expanding $(\chi_i + 1 - \chi_i)^{\sum_{j\in N_i} n_{m+j}}$, we get that (13) is at most:

$$\prod_{i=1}^{m}\left(\chi_i^{n_i}(1-\chi_i)^{\sum_{j\in N_i} n_j}\right)\left((1-\chi_i)^{n_{m+i}}\right) < \prod_{i=1}^{m}(1-\chi_i)^{\sum_{j\in N_i} nj}(1-\chi_i)^{n_{m+i}}. \tag{14}$$

By letting $\chi := \min_{i=1,\ldots,m}\{\chi_i\}$, we obtain that (14) is bounded from above by $\prod_{i=1}^{2m}(1-\chi_i) = (1-\chi_i)^{\sum_{i=1}^{2m}}$. Thus, $[\mathbf{t^n}]g((\mathbf{Q},\mathbf{R})(\bar{t})) \leq (1-\chi)^n$, which is exponentially small to n. Thus, the proof is now complete.

3 Shearer's Lemma

We now turn our attention to Shearer's Lemma. For a graph G with vertex set $V := \{1, \ldots, m\}$, an *independent set* I is a subset of V with no edges between its vertices ($\emptyset$ is trivially such a set). Let $I(G) = \{I_0, I_1, \ldots, I_s\}$ denote the set of independent sets of G and suppose that its elements are ordered according to their indices, where $I_0 = \emptyset$. For any $I \in I(G)$, let $N(I)$ be the set of vertices of I, together with their neighboring ones in G. Following [KRS11], we sat that I *covers* J if $J \subseteq N(I)$.

Theorem 3.1 (Shearer's Lemma)**.** *Given a graph $G = (V, E)$ on m vertices and a vector $\bar{p} = (p_1, \ldots, p_m) \in (0,1)^m$, the following are* equivalent*:*

1. *For all $I \in I(G)$, $q_I(G, \bar{p}) = \sum_{J\in I(G):\ I\subseteq J}(-1)^{|J\setminus I|}\prod_{j\in J} p_j > 0$,*
2. *For every (finite) sequence of events $E_1, \ldots, E_m$, if their dependency graph is G and if $Pr[E_j] = p_j$, $j = 1, \ldots, m$, then $Pr[\overline{E}_1 \wedge \cdots \wedge \overline{E}_m] > 0$.*

We will not be concerned with the proof of ($2 \Rightarrow 1$). It is the direction that gives us the necessity of condition 1 (see [She85]).

The algorithm we will use is the GENERALIZED RESAMPLE algorithm, designed by Kolipaka and Szegedy in[KRS11]. Abusing the notation, we will sometimes say that an independent set $I \in I(G)$ contains events (instead of indices of events). Also, for any $I = \{j_1, ..., j_k\} \in I(G)$, $k \geq 0$, let vbl(I) denote the set of variables contained in the scopes of the events in I.

A *round* of GENRESAMPLE is the duration of each repetition of lines 2 and 3. Let $\mathbb{P}_n$ be the probability GENRESAMPLE executes for *at least* n rounds. Our aim again, is to show that there is a $d \in (0,1)$ such that $\mathbb{P}_n \leq d^n$.

Consider an execution of GENRESAMPLE that lasts for at least n rounds and let $I_{i_1}, \ldots I_{i_n}$ be the independent sets selected at line 2.

Algorithm 3 GenResample.

Sample the variables X_i, $i = 1, ..., l$ and let α be the resulting assignment.
while there exists an event that occurs under the current assignment, let I_i be the least indexed *maximal* independent set that contains only occurring events and **do**
    Resample every variable in vbl(I_i) and let α be the current assignment.
end while
Output current assignment α.

Lemma 3.2. I_{i_t} covers $I_{i_{t+1}}$, *for all* $t \in \{1, \ldots, n-1\}$.

Proof. It suffices to prove that if the index of an event is in $I_{i_{t+1}}$, then it is also in $N(I_{i_t})$. Suppose E_j is in $I_{i_{t+1}}$. Then, under the assignment produced at the end of round t, E_j occurred. To obtain a contradiction, suppose that $E_j \notin N(I_{i_t})$. E_j does not occur at the beginning of round t, lest I_{i_t} is not maximal. But then, since it is not dependent on any event in I_{i_t}, it cannot occur at the beginning of round $t+1$. Thus, $E_j \notin I_{i_{t+1}}$. Contradiction. □

Now, as in the previous section, we will depict an execution of GenResample by a graph structure. In place of forests, we will now use *directed paths*, that have one vertex of in-degree 0 (source), one of out-degree 0 (sink), and all edges directed from the source to the sink. The vertices of such a path are labeled by independent sets from $I(G)$ and we order them from the source to the sink.

Definition 3.3. A directed labeled path $\mathcal{P}$ with $|\mathcal{P}| = n$ vertices, whose labels are $I_{i_1}, \ldots, I_{i_n}$, is *feasible* if I_{i_t} covers $I_{i_{t+1}}$, for all $t \in \{1, \ldots, n\}$ (in the terminology of [KRS11], $(I_{i_1}, \ldots, I_{i_k})$ is a *stable set sequence*).

A feasible path with n nodes, whose labels correspond to the independent sets selected by GenResample in an execution that lasted for at least n rounds, is called the n-witness path of GenResample's execution.

Again, we want to avoid the dependencies that GenResample produces during its execution. We will thus use a validation algorithm, in the spirit of ValAlg of section 2. The rounds of GenVal are defined as usual.

Algorithm 4 GenVal.

Input: Feasible path $\mathcal{P}$ with labels $I_{i_1}, \ldots, I_{i_n}$.
Sample the variables X_i, $i = 1, ..., l$.
for t=1,. . . ,n **do**
    if there exists an event E_j, with $j \in I_{i_t}$, that does not occur under the current assignment **then**
        return `failure` and exit.
    else
        Resample the variables in vbl(I_{i_t})
    end if
end for
return `success`.

Also, Lemma 2.4 holds for GenVal too. Define $V_{\mathcal{P}}$ to be the event that GenVal returns success, on input $\mathcal{P}$ and suppose we have real numbers $Q_{n,i}$ such that $Pr[V_{\mathcal{P}}] \leq Q_{n,i}$, where P is a path with n-nodes, whose source is labeled by $I_i \in I(G)$ and where $Q_{1,I} = \prod_{j \in I} p_j$, for all $I \in I(G)$.

Thus, it holds that the numbers $Q_{n,i}$ satisfy:

$$Q_{n,i} = \prod_{j \in I_i} p_j \sum_{I_i \text{ covers } J} Q_{n-1,J}. \tag{15}$$

Following again the terminology of [KRS11], we define the *stable set matrix* M, as an $s \times s$ matrix, whose element in the i-th row and j-th column is $\prod_{j \in I} p_j$ if I covers J and 0 otherwise. Furthermore, let $q_n = (Q_{n,1}, \ldots, Q_{n,s})$. Easily, (15) is equivalent to:

$$q_n = M q_{n-1}, \text{ thus } q_n = M^{n-1} q_1. \tag{16}$$

Note now that:

$$\mathbb{P}_n \leq \sum_{i=1}^{s} Q_{n,i} = \|q_n\|_1,$$

where $\|\cdot\|_1$ is the 1-*norm* defined on $\mathbb{R}^s$.

It is known that any vector norm, and thus 1-norm too, yields a norm for square matrices called the *induced norm* [HJ90] as follows:

$$\|M\|_1 := \sup_{x \neq 0} \frac{\|Mx\|_1}{\|x\|_1} \geq \frac{\|Mq_1\|_1}{\|q_1\|_1}. \tag{17}$$

By (16) and (17), we have that:

$$\|q_n\|_1 = \|M^{n-1}q_1\|_1 \leq \|M^{n-1}\|_1 \cdot \|q_1\|_1. \tag{18}$$

Since $\|q_1\|_1$ is a constant, it suffices to show that $\|M^{n-1}\|_1$ is exponentially small in n. Let $\rho(M)$ be the *spectral radius* of M (see [KRS11]). By Gelfand's formula (see again [HJ90]) used for the induced matrix norm $\|\cdot\|_1$, we have that:

$$\rho(M) = \lim_{n\to\infty} \|M^n\|^{1/n}.$$

Furthermore, in [KRS11] (Theorem 14), it is proved that the condition of Shearer's Lemma, i.e. Theorem 3.1 in our work, is *equivalent* to $\rho(M) < 1$. Thus, by selecting an $\epsilon > 0$ such that $\rho(M) + \epsilon < 1$, we have that there exists a constant (depending only on ϵ, M) such that $\|M^{n-1}\|_1 \leq (\rho(M)+\epsilon)^{n-1}$, which, together with (18), gives us that the bound for $\mathbb{P}_n$ is exponentially small in n.

Acknowledgements

L. Kirousis' research was partially supported by the Special Account for Research Grants of the National and Kapodistrian University of Athens. K. I. Psaromiligkos' research was carried out while he was an undergraduate student at the Department of Mathematics of the National and Kapodistrian University of Athens. D. Thilikos' research was supported by the project "ESIGMA" (ANR-17-CE40-0028).

References

[BR98] Edward A Bender and L Bruce Richmond. A multivariate Lagrange inversion formula for asymptotic calculations. *The Electronic Journal of Combinatorics*, 5(1):33, 1998.

[EL75] Paul Erdős and László Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. *Infinite and Finite Sets*, 10:609-627, 1975.

[GKPT15] Ioannis Giotis, Lefteris Kirousis, Kostas I. Psaromiligkos, and Dimitrios M. Thilikos. On the algorithmic Lovász local lemma and acyclic edge coloring. In *Proceedings of the twelfth workshop on analytic algorithmics and combinatorics*. Society for Industrial and Applied Mathematics, 2015. http://epubs.siam.org/doi/pdf/10.1137/1.9781611973761.2.

[HJ90] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge University Press, 1990.

[KRS11] Kashyap Kolipaka, Babu Rao, and Mario Szegedy. Moser and Tardos meet Lovász. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 235-244. ACM, 2011.

[Mos09] Robin A. Moser. A constructive proof of the Lovász local lemma. In *Proceedings of the 41st annual ACM Symposium on Theory of Computing*, pages 343-350, 2009.

[MT10] Robin A. Moser and Gábor Tardos. A constructive proof of the general Lovász local lemma. *Journal of the ACM (JACM)*, 57(2):11, 2010.

[She85] James B. Shearer. On a problem of Spencer. *Combinatorica*, 5(3):241-245, 1985.

[Sze13] M. Szegedy. The Lovász local lemma - a survey. In Springer, editor, *Computer Science - Theory and Applications*, pages 11-23, 2013.

[Tao09] Terence Tao. Moser's entropy compression argument. 2009. Available: `https://terrytao.wordpress.com/2009/08/05/mosers-entropy-compression-argument/`.