THE MOSER-TARDOS FRAMEWORK WITH PARTIAL RESAMPLING

DAVID G. HARRIS[1] AND ARAVIND SRINIVASAN[2]

Abstract. The resampling algorithm of Moser & Tardos is a powerful approach to develop constructive versions of the Lovász Local Lemma. We develop a *partial* resampling approach motivated by this methodology: when a bad event holds, we resample an appropriately-random *subset* of the variables that define this event, rather than the entire set as in Moser & Tardos. This is particularly useful when the bad events are determined by sums of random variables. This leads to several improved algorithmic applications in scheduling, graph transversals, packet routing etc. For instance, we improve the approximation ratio of a generalized D-dimensional scheduling problem studied by Azar & Epstein from $O(D)$ to $O(\log D / \log \log D)$, and settle a conjecture of Szabó & Tardos on graph transverals asymptotically.

1. Introduction

The Lovász Local Lemma (LLL) [7] is a fundamental probabilistic tool. The breakthrough of Moser & Tardos shows that a very natural *resampling* approach yields a constructive approach to the LLL [23]; this, along with a few subsequent investigations [9, 18], gives a fairly comprehensive suite of techniques to develop algorithmic versions of the LLL. The basic algorithm of [23] is as follows. Suppose we have "bad" events $E_1, E_2, \ldots, E_m$, each E_i being completely determined by a subset $\{j \in S_i : X_j\}$ of *independent* random variables $X_1, X_2, \ldots, X_\ell$. Then, assuming that the standard sufficient conditions of the LLL hold, the following resampling algorithm quickly converges to a setting of the X_j's that simultaneously avoids all the E_i:

- first sample all the X_j's (independently) from their respective distributions;
- **while** some bad event is true, pick one of these, say E_i, arbitrarily, and resample (independently) all the variables $\{j \in S_i : X_j\}$.

We develop a *partial resampling* approach motivated by this, which we simply call the Partial Resampling Algorithm (PRA); the idea is to carefully choose a distribution D_i over *subsets* of $\{j \in S_i : X_j\}$ for each i, and then, every time we need to resample, to first draw a subset from D_i, and then only resample the X_j's that are contained in this subset. This partial-resampling approach leads to algorithmic results for many applications that are not captured by the LLL.

Conference versions of this work. Preliminary versions of parts of this paper appeared in two papers by the authors: [10, 11].

In order to motivate our applications, we start with two classical problems: scheduling on unrelated parallel machines [21], and low-congestion routing [27]. In the former, we have n jobs and K machines (we interchange the standard use of the indices i and j here, and use K in place of the usual "m", in order to conform to the rest of our notation), and each job i needs to be scheduled on any element of a given subset X_i of the machines. If job i is scheduled on machine j, then j incurs a given load of $p_{i,j}$. The goal is to minimize the *makespan*, the maximum total load on any machine. The standard way to approach this is to introduce an auxiliary parameter T, and ask

[1]Department of Applied Mathematics, University of Maryland, College Park, MD 20742. Research supported in part by NSF Award CNS-1010789. Email: davidgharris29@hotmail.com.

[2]Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. Research supported in part by NSF Awards CCR-0208005, ITR CNS-0426683, CNS-0626636, and CNS-1010789. Email: srin@cs.umd.edu.

if we can schedule with makespan T [21, 31]. Letting $[k]$ denote the set $\{1, 2, \ldots, k\}$, a moment's reflection leads to the following integer-programming formulation:

$$\forall i \in [n],\ \sum_{j \in X_i} x_{i,j} = 1; \tag{1}$$

$$\forall j \in [K],\ \sum_i p_{i,j} x_{i,j} \leq T; \tag{2}$$

$$\forall (i,j),\ p_{i,j} > T \implies x_{i,j} = 0; \tag{3}$$

$$\forall (i,j),\ x_{i,j} \in \{0,1\}. \tag{4}$$

(Although (3) is redundant for the IP, it will be critical for the natural LP relaxation [21].) In low-congestion routing, we are given a collection of (source, destination) pairs $\{(s_i, t_i) :\ i \in [n]\}$ in a V-vertex, K-edge directed or undirected graph G with edge-set E; each edge $f \in E$ has a capacity c_f, and we are also given a collection $X_i = \{P_{i,j}\}$ of possible routing paths for each (s_i, t_i)-pair, with each such path indexed by i and an auxiliary index j. We aim to choose one path from X_i for each i, in order to minimize the *relative congestion*: the minimal T such that the maximum load on any edge f is at most $T \cdot c_f$. We get a similar IP formulation:

$$\text{minimize } T \text{ subject to } \left[\forall i,\ \sum_j x_{i,j} = 1;\ \forall f \in E,\ \sum_{(i,j):\ f \in P_{i,j}} x_{i,j} \leq T \cdot c_f;\ x_{i,j} \in \{0,1\}. \right]$$

Our class of problems. Given the above two examples, we are ready to define the class of problems that we will study. As above, we have n "categories" (finite sets) $X_1, X_2, \ldots, X_n$; we need to choose one element from each category, which is modeled by "assignment constraints" (1) on the underlying indicator variables $x_{i,j}$. In addition, we have K (undesirable) Boolean functions $B_1, B_2, \ldots, B_K$, each of which is an *increasing* function of the variables $x_{i,j}$; we aim to choose the $x_{i,j}$ in order to satisfy the assignment constraints, and such that all the B_k are falsified. It is easily seen that our two applications above, have the undesirable events B_k being *linear* threshold functions of the form "$\sum_{i,j} a_{k,i,j} x_{i,j} > b_k$"; we also allow explicitly-nonlinear B_k, some of which will be crucial in our packet-routing application. We develop a *partial resampling* approach to our basic problem in Section 2; Theorem 2.5 presents some general conditions under which our algorithm quickly computes a feasible solution $\{x_{i,j}\}$.

The probabilistic analysis of the Moser-Tardos and related algorithms is governed by *witness trees*. While these are easy to count when all bad-events are essentially the same (the "Symmetric LLL"), this can be complicated in the more general ("Asymmetric") case.

A key technical tool in our analysis is a new formula for counting the witness trees. This greatly simplifies the analysis of the Asymmetric LLL. It is critical to obtaining usable formulas for complicated applications of the Partial Resampling framework, but it is also very useful for analyzing the standard Moser-Tardos framework.

We will need the following relative of the standard Chernoff upper-tail bound:

Definition 1.1. ***(The Chernoff separation function)*** *For $0 < \mu \leq t$, letting $\delta = \delta(\mu, t) = t/\mu - 1 \geq 0$, we define*

$$\textit{Chernoff}(\mu, t) = \left(\frac{e^\delta}{(1+\delta)^{1+\delta}} \right)^\mu;$$

i.e., Chernoff(μ, t) is the Chernoff bound that a sum of $[0,1]$-bounded and independent random variables with mean μ will exceed t. If $t < \mu$ we define Chernoff$(\mu, t) = 1$.

Let us next motivate our result by describing three families of applications.

1.1. **The case of non-negative linear threshold functions.** The scheduling and routing applications had each B_k being a non-negative linear threshold function: our constraints (i.e., the complements of the B_k) were of the form

$$\forall k \in [K],\ \sum_{i,j} a_{k,i,j} x_{i,j} \leq b_k. \tag{5}$$

(The matrix A of coefficients $a_{k,i,j}$ here, has K rows indexed by k, and some N columns that are indexed by pairs (i,j).) Recall that all our problems will have the assignment constraints (1) as well. There are two broad types of approaches for such problems, both starting with the natural LP relaxation of the problem, wherein we allow each $x_{i,j}$ to lie in $[0,1]$. Suppose the LP relaxation has a solution $\{y_{i,j}\}$ such that for all k, "$\sum_{i,j} a_{k,i,j} y_{i,j} \leq b'_k$", where $b'_k < b_k$ for all k; by scaling, we will assume throughout that $a_{k,i,j} \in [0,1]$. The natural question is:

> "What conditions on the matrix A and vectors b' and b ensure that there is an integer solution that satisfies (1) and (5), which, furthermore, can be found efficiently?"

The first of the two major approaches to this is polyhedral. Letting D denote the maximum column sum of A, i.e., $D = \max_{i,j} \sum_k a_{k,i,j}$, the rounding theorem of [17] shows constructively that for all k,

$$b_k = b'_k + D \tag{6}$$

suffices. The reader is asked to verify that given a solution to the LP-relaxation of makespan minimization that satisfies (1), (2) and (3), bound (6) implies that we can find a schedule with makespan at most $2T$ efficiently. This 2-approximation is the currently best-known bound for this fundamental problem, and what we have seen here is known to be an alternative to the other polyhedral proofs of [21, 31].

The second approach to our problem is randomized rounding [27]: given an LP-solution $\{y_{i,j}\}$, choose exactly one j independently for each i, with the probability of choosing j in category i equaling $y_{i,j}$. The standard "Chernoff bound followed by a union bound over all K rows" approach [27] shows that Chernoff$(b'_k, b_k) \leq 1/(2K)$ suffices, for our goal to be achieved with probability at least $1/2$. That is, there is some constant $c_0 > 0$ such that

$$b_k \geq c_0 \cdot \frac{\log K}{\log(2\log K/b'_k)} \text{ if } b'_k \leq \log K;\ b_k \geq b'_k + c_0 \cdot \sqrt{b'_k \cdot \log K} \text{ if } b'_k > \log K \tag{7}$$

suffices. In particular, the low-congestion routing problem can be approximated to within $O(\log K/\log\log K)$ in the worst case, where K denotes the number of edges.

Let us compare these known bounds (6) and (7). The former is good when all the b'_k are "large" (say, much bigger than, or comparable to, D – as in the 2-approximation above for scheduling); the latter is better when D is too large, but unfortunately does not exploit the sparsity inherent in D – also note that $K \geq D$ always since the entries $a_{k,i,j}$ of A lie in $[0,1]$. A natural question is whether we can interpolate between these two: especially consider the case (of which we will see an example shortly) where, say, all the values b'_k are $\Theta(1)$. Here, (6) gives an $O(D)$-approximation, and (7) yields an $O(\log K/\log\log K)$-approximation. Can we do better? We answer this in the affirmative in Theorem 5.2 – we are able to essentially replace K by D in (7), by showing constructively that for *any* desired constant $C_1 > 0$, there exists a constant $C_0 > 0$ such that

$$b_k \geq C_0 \cdot \frac{\log D}{\log(2\log D/b'_k)} \text{ if } b'_k \leq \log D;\ b_k \geq b'_k + b'_k D^{-C_1} + C_0 \cdot \sqrt{b'_k \cdot \log D} \text{ if } b'_k > \log D \tag{8}$$

suffices.

Application to multi-dimensional scheduling. Consider the following D-dimensional generalization of scheduling to minimize makespan, studied by Azar & Epstein [4]. Their $(D+1)$-approximation here also holds for the following generalization, and again follows quickly from (6). Here, when job

i gets assigned to machine j, there are D dimensions to the load on j (say runtime, energy, heat consumption, etc.): in dimension ℓ, this assignment leads to a load of $p_{i,j,\ell}$ on j (instead of values such as $p_{i,j}$ in [21]), where the numbers $p_{i,j,\ell}$ are given. Analogously to (1), (2) and (3), we ask here: *given a vector $(T_1, T_2, \ldots, T_D)$, is there an assignment that has a maksepan of at most T_ℓ in each dimension ℓ?* The framework of [4] and (6) gives a $(D+1)$-approximation[1] while our bound (8) yields an $O(\log D / \log \log D)$-approximation; since $D \ll K$ typically in this application, this is also a significant improvement over the $O(\log K / \log \log K)$-approximation that follows from (7).

Comparison with other known bounds. As described above, our bound (8) improves over the two major approaches here. However, two related results deserve mention. First, a bound similar to (8) is shown in [19, 10], but with D^*, the maximum number of nonzeroes in any column of A, playing the role of D. Note that $D^* \geq D$ always, and that $D^* \gg D$ is possible. Moreover, the bound of [19] primarily works when all the b'_k are within an $O(1)$ factor of each other, and rapidly degrades when these values can be disparate; the bound of [10] is nonconstructive.

1.2. **Transversals with omitted subgraphs.** Given a partition of the vertices of an undirected graph $G = (V, E)$ into blocks (or *classes*), a *transversal* is a subset of the vertices, one chosen from each block. An *independent transversal*, or independent system of representatives, is a transversal that is also an independent set in G. The study of independent transversals was initiated by Bollobás, Erdős & Szemerédi [5], and has received a considerable amount of attention (see, e.g., [1, 2, 3, 14, 15, 16, 22, 32, 33]). Furthermore, such transversals serve as building blocks for other graph-theoretic parameters such as the linear arboricity and the strong chromatic number [2, 3]. We improve (algorithmically) a variety of known sufficient conditions for the existence of good transversals, in Section 4. In particular, Szabó & Tardos present a conjecture on how large the blocks should be, to guarantee the existence of transverals that avoid K_s [32]; we show that this conjecture is true asymptotically for large s. We also study weighted transversals, as considered by Aharoni, Berger & Ziv [1], and show that near-optimal (low- or high-) weight transversals exist, and can be found efficiently. In particular, we improve the quantitative bounds of [10] and show that "large-weight" (existentially-optimal) independent transversals exist, once the smallest block-size becomes reasonably large.

1.3. **Packet routing with low latency.** A well-known packet-routing problem is as follows. We are given an undirected graph G with N packets, in which we need to route each packet i from vertex s_i to vertex t_i along a *given simple path P_i*. The constraints are that each edge can carry only one packet at a time, and each edge traversal takes unit time for a packet; edges are allowed to queue packets. The goal is to conduct feasible routings along the paths P_i, in order to minimize the *makespan* T, the relative of the scheduling notion above that refers to the time by which all packets are delivered. Two natural lower-bounds on T are the *congestion* C (the maximum number of the P_i that contain any given edge of G) and the *dilation* D (the length of the longest P_i); thus, $(C+D)/2$ is a universal lower-bound, and there exist families of instances with $T \geq (1 + \Omega(1)) \cdot (C + D)$ [28]. A seminal result of [20] is that $T \leq O(C + D)$ for all input instances, using constant-sized queues at the edges; the big-Oh notation hides a rather large constant. Building on further improvements [29, 26], our work [10] developed a nonconstructive $7.26(C + D)$ and a constructive $8.84(C + D)$ bound; we improve these further to a constructive $5.70(C + D)$ here.

Informal discussion of the Partial Resampling Algorithm. To understand the intuition behind our Partial Resampling Algorithm, consider the situation in which we have bad events of the form $Z_1 + \cdots + Z_v \geq \mu + t$, where the expected value of $Z_1 + \cdots + Z_v$ is μ. There are two basic ways to set this up for the standard LLL. The most straightforward way would be to construct a

[1] As usual in this setting, an "approximation" here is an algorithm that either proves that the answer to this question is negative (by showing that the LP is infeasible), or presents the desired approximation simultaneously for each T_ℓ.

single bad-event for $Z_1 + \cdots + Z_v \geq \mu + t$. In this case, the single event would depend on v variables, which might be very large. Alternatively, one could form $\binom{v}{\mu+t}$ separate bad-events, corresponding to every possible set of $\mu + t$ variables. Each of these bad-events individually would have a very low probability, and the overall dependency would also be low. The problem with this approach is that the collective probability of the bad-events has become very large. In effect, one is approximating the probability of the bad-event $Z_1 + \cdots + Z_v \geq \mu + t$ by a union-bound over all $\binom{v}{\mu+t}$ subsets. When t is small, this union bound is very inaccurate.

In fact, both of these approaches are over-counting the dependence of the bad-event. In a sense, a variable Z_i is causing the bad-event only if it is "the straw that breaks the camel's back," that is, if it is the key variable which brings the sum $Z_1 + \cdots + Z_v$ over the threshold $\mu + t$. Really, only about t of the variables are "guilty" of causing the bad-event. The first μ variables were expected to happen anyway; after reaching a total $\mu + t$ variables, any remaining variables are redundant. Any individual variable Z_i only has a small chance of being a guilty variable.

In effect, there are many variables which have have some effect on the bad event, but this effect is typically very small. The standard LLL, which is based on a binary classification of whether a variable affects a bad-event, cannot see this. We will give a new criterion which quantifies how likely a variable is to be responsible for the bad-event. This will in turn greatly lower the dependency between bad events, while still keeping an essentially accurate bound on the probability. Further specific comparisons with the standard LLL are made in Sections 4 and 5.1.

In general, the Partial Resampling Algorithm tends to work well when there are common configurations, which are not actually forbidden, but are nonetheless "bad" in the sense that they are leading to a forbidden configuration. So, in the case of a sum of random variables, if a large group of these variables is simultaneously one, then this is bad but, by itself, still legal. We will see other examples of more complicated types of bad-but-legal configurations.

Organization of the paper. The partial resampling algorithm is discussed in detail in Section 2; a related, but incomparable, approach is presented in Section 3. Applications are discussed in the following three sections: transversals with omitted subgraphs, improved integrality gaps for column-sparse packing problems, and packet routing in Sections 4, 5.1, and 6 respectively. The appendix contains numerical results and a useful lemma.

2. The Partial Resampling Algorithm

2.1. **Notation.** We begin by discussing some basic definitions that will apply throughout.

We have n categories, which we identify with the set $[n] = \{1, \ldots, n\}$. Each category has a set of possible assignments, which may be countably infinite or finite, and which we identify with (a subset of) the integers. We specify a probability distribution p_i on each category i, with $\sum_j p_{i,j} = 1$, where j ranges over the set of valid assignment to category i. Henceforth we will not be explicit about the set of possible assignments, so we write simply $\sum_j p_{i,j} = 1$.

We refer to any ordered pair $\langle i, j \rangle$ where j is an assignment of category i, as an *element*; we sometimes refer to an element as (i, j) as well. We let X denote the set of all elements. We suppose $|X| = N$ and we sometimes identify X with the set $[N]$. Given any vector $\vec{\lambda} = (\vec{\lambda}_{i,j})$ indexed by elements $\langle i, j \rangle$, we define, for any set $Y \subseteq X$,

$$\vec{\lambda}^Y = \prod_{\langle i,j \rangle \in Y} \vec{\lambda}_{i,j}. \tag{9}$$

Events as set-families, and increasing bad events. Suppose we are given some K *increasing* bad events $B_1, B_2, \ldots, B_K$, such that each B_k is an *upward-closed* collection of subsets of X. Note that in the preceding sentence – and in a few places later – we identify each event such as B_k with a family of subsets of X in the obvious manner: i.e., if $\mathcal{F}_k$ is this family, then B_k is true iff there is some $G \in \mathcal{F}_k$ such that all members of G (each of which is an "element" in our terminology of a few

lines above) have been chosen. Equivalently, since each B_k is upward-closed, we can identify each bad event B_k with its *atomic bad events* $A_{k,1}, \ldots, A_{k,m_k}$: B_k holds iff there is some j such that all members of $A_{k,j}$ have been chosen, and each $A_{k,j}$ is minimal inclusion-wise. That is, viewing each B_k as a family of subsets, we have

$$B_k = \{Y \subseteq X \mid A_{k,1} \subseteq Y \vee \cdots \vee A_{k,m_k} \subseteq Y\}. \tag{10}$$

We are not making any sparsity assumptions about the bad events B_k, for example that they depend only on a (small) subset of the categories.

2.2. **Fractional hitting-sets.** In order to use our algorithm, we will need to specify an additional parameter, for each bad event B_k. We must specify a *fractional hitting-set* B'_k, which essentially tells us how to resample the variables in that bad event.

Definition 2.1. *Let $B \subseteq 2^X$ be a given **increasing** bad event on the ground set X. (As usual, 2^X denotes the family of subsets of X.) Suppose $C : 2^X \to [0,1]$ is some weight function on the subsets of X. We say that C is a* fractional hitting set for B *if, viewing B as a family of subsets of X as in (10), we have for all $A \in B$ that*

$$\sum_{Y \subseteq A} C(Y) \geq 1. \tag{11}$$

Remark regarding Definition 2.1. It clearly suffices to show that (11) holds for the minimal atomic bad events A of B. We may assume, without loss of generality, that if Y is a subset of X which contains two members of the same category, that is, it contains $\langle i, j\rangle$ and $\langle i, j'\rangle$, then $C(Y) = 0$.

We will fix a fractional hitting-set for each bad event B_k, and so we write B'_k for the fractional hitting-set associated with B_k. One possible choice is the bad-event itself, which is easily verified to be a valid fractional hitting-set:

Definition 2.2. *Let $B \subseteq X$ be a given bad event, with atomic bad events $A_1, \ldots, A_m$. We define the* trivial hitting-set *for B by*

$$C(Y) = \begin{cases} 1 & \text{if } Y = A_t \text{ for some } t = 1, \ldots, m \\ 0 & \text{otherwise} \end{cases}$$

If the trivial hitting-set is used, then our analysis essentially reduces to the ordinary Moser-Tardos algorithm (but we will still show improvements in that case). We will discuss later how to construct such fractional hitting-sets, but for now we suppose that B'_k is specified.

2.3. **Partial Resampling Algorithm and the Main Theorem.** We present our partial resampling algorithm, and main theorem related to it.

Consider the following relative of the Moser-Tardos algorithm. We refer to this as the *Partial Resampling Algorithm* or abbreviate as *PRA*.

- Select one element from each category $i = 1, \ldots, n$. The probability of selecting $\langle i, j\rangle$ is $p_{i,j}$.
- Repeat the following, as long as there is some k such that the current assignment makes the bad event B_k true:
  - Select, arbitrarily, some atomic bad event $A \in B_k$ that is currently true. We refer to this set A as the *violated set*.
  - Select exactly one subset $Y \subseteq A$. The probability of selecting a given Y is given by
  $$\mathbf{P}(\text{select } Y) = \frac{B'_k(Y)}{\sum_{Y' \subseteq A} B'_k(Y')};$$
  we refer to Y as the *resampled set*.
  - Resample all the categories in Y independently, using the vector p.

This algorithm is similar to, and inspired by, the Moser-Tardos algorithm. The main difference is that in [23], if there is a bad event that is currently true, we would resample *all* the variables which it depends upon. Here, we only resample a (carefully-chosen, random) subset of these variables.

We will need to keep track of the dependency graph corresponding to our fractional hitting set. This is more complicated than the usual Moser-Tardos setting, because we will need to distinguish two ways that subsets of elements Y, Y' could affect each other: they could share a variable, *or* they could both be potential resampling targets for some bad-event. In the usual Moser-Tardos analysis, we only need to keep track of the first type of dependency. The following symmetric relation $\approx$ (and its two supporting relations $\sim$ and $\bowtie$) will account for this:

Definition 2.3. ***(Symmetric relations $\sim$, $\bowtie_k$, and $\approx$)*** *Let $Y, Y' \subseteq X$.*

We say $Y \sim Y'$ iff there exists a triple (i, j, j') such that $\langle i, j\rangle \in Y$ and $\langle i, j'\rangle \in Y'$: i.e., iff Y and Y' overlap in a category. We also write $i \sim Y$ (or $Y \sim i$) to mean that Y involves category i (i.e., $\langle i, j\rangle \in Y$ for some j).

For each k, we say $Y \bowtie_k Y'$ iff $Y \not\sim Y'$ and there is some atomic event $A'' \in B_k$ with $Y, Y' \subseteq A''$.

Relation $\approx$ is defined between pairs (Y, k): We define $(Y, k) \approx (Y', k')$ iff $Y \sim Y'$ or $Y \bowtie_k Y'$. Note that, by definition, it is impossible for both to occur simultaneously.

Remark. At this point, we can clarify a subtle point in the framework of the Resampling Algorithm. We have defined multiple bad-events $B_1, \ldots, B_K$, each of which can in turn contain multiple atomic bad-events. This two-layer structure seems redundant. It would seem more natural to keep track of atomic bad-events only. Equivalently, there would be only one bad-event ($K = 1$), including multiple atomic bad-events. Indeed, in many of our applications, such as the tranvsersals with omitted subgraphs, we will use this approach. When using the trivial hitting-set, then a single global bad-event is always equivalent to multiple bad-events.

However, atomic bad-events which are contained in the same bad-event may have complicated and non-linear interactions due to $\bowtie$. This is particularly problematic when we have $B'_k(Y) \neq B'_{k'}(Y)$, and both of these are non-zero. For some constructions, such as column-sparse packing and packet-routing, these interactions are difficult to control. In such cases, it is useful to define multiple bad-events. We will need to analyze the $\bowtie$ interactions within a single bad-event, but we will not have to worry about interactions across the separate bad-events. This allows us to compute a "badness" measure for a single bad-event, depending only on local parameters, and then add this up across the entire space.

We now also define:

Definition 2.4. ***(Values $G^k_{i,j}$, G^k_i, and G^k that depend on a vector $\vec{\lambda}$)*** *Suppose we are given an assignment of non-negative real numbers $\vec{\lambda}_{i,j}$ to each element $\langle i, j\rangle$. For each bad event B_k, recalling the notation (9), we define*

$$G^k_{i,j}(\vec{\lambda}) = \sum_{Y \ni \langle i,j\rangle} B'_k(Y)\vec{\lambda}^Y$$

along with "summation notations"

$$G^k_i(\vec{\lambda}) = \sum_j G^k_{i,j}(\vec{\lambda}) \text{ and } G^k(\vec{\lambda}) = \sum_{i,j} G^k_{i,j}(\vec{\lambda}).$$

We will often omit the dependence on $\vec{\lambda}$ if it is clear from context. Roughly speaking, $G^k_{i,j}$ is the probability that variable i takes on value j, and causes bad-event B_k to occur, and is selected for resampling.

Our main theorem is as follows. In part (a), it assumes that the vector p has been given. In parts (b) and (c), it assumes the existence of a suitable vector $\vec{\lambda}$, from which p is explicitly derived in the statement of the theorem.

Theorem 2.5. ***(Main Theorem)*** *In each of the following three cases, the PRA converges to a feasible configuration avoiding all bad events with probability one.*

(a) Suppose there exists $\mu : 2^X \times [K] \to [0, \infty)$ which satisfies, for all $Y \subseteq X$ and all $k \in [K]$,

$$\mu(Y,k) \geq p^Y B'_k(Y)\Big(\prod_{\substack{Y' \sim Y \\ k}} (1 + \mu(Y',k'))\Big)\Big(1 + \sum_{Y'' \bowtie_k Y} \mu(Y'',k)\Big)$$

Then, the expected number of resamplings of any category i is at most $\sum_{(Y,k):\ Y \sim i} \mu(Y,k)$.

For the next two cases, we assume we are given an assignment of non-negative real numbers $\vec{\lambda}_{i,j}$ to each element $\langle i,j \rangle$, and suppose that we run the PRA with $p_{i,j} = \vec{\lambda}_{i,j} / \sum_{j'} \vec{\lambda}_{i,j'}$ for all i,j. We will use Definition 2.4 and $\vec{\lambda}_i \doteq \sum_j \vec{\lambda}_{i,j}$ in these two cases.

(b) Suppose that for all k, $G^k(\vec{\lambda}) < 1$; suppose further that

$$\forall i,\ \vec{\lambda}_i \geq 1 + \sum_k \frac{G^k_i(\vec{\lambda})}{1 - G^k(\vec{\lambda})}.$$

Then the expected number of resamplings of a category i is at most $\vec{\lambda}_i$.

(c) Suppose the $\bowtie_k$ relations are null: that is, whenever there is an atomic event A with $Y, Y' \subseteq A$ and $B'_k(Y) > 0, B'_k(Y') > 0$ then $Y \sim Y'$. Suppose further that $\forall i,\ \vec{\lambda}_i \geq 1 + \sum_k G^k_i(\vec{\lambda})$. Then the expected number of resamplings of a category i is at most $\vec{\lambda}_i$.

2.4. **Proof ingredient: Witness Trees.** A key component of our proofs will be the notion of *witness trees* similar to [23]; we develop this notion now. As in the proof of [23], we define an execution log of this algorithm to be a listing of all pairs (Y,k) that were encountered during the run; it is crucial to note that we do *not* list the violated sets A themselves. Given a log, we define the *witness tree* which provides a justification for any given resampling in the execution log. Not listing the violated sets themselves, is one of our critical ideas, and helps prune the space of possible witness trees substantially.

We form the witness tree in a manner similar to [23], driven by the relation $\approx$, but there is a key difference. Each event Y, B_k is only allowed to have a *single* child due to the relation $\bowtie_k$. As we will see, a single $\bowtie$-child provides all the information needed. A much simpler proof for the critical Witness Tree Lemma could be obtained if we allow for multiple $\bowtie_k$-children, but later on this gives formulas that are more complicated and slightly weaker.

Here is how we build a witness tree for an event of interest E (for example, the final resampling). The event E goes at the root of the tree. This, and all nodes of the tree, will be labeled by the corresponding pair (Y,k) – we will sometimes also refer to these labels as (Y, B_k). Stepping backward in time, suppose the current event being processed is labeled (Y,k). Suppose that either there is a node (Y',k') in the current tree with $Y' \sim Y$, *or* there is an node (Y',k) and $Y \bowtie_k Y'$ *and the node* (Y',k) *does not currently have a child* (Y'',k) *with* $Y'' \bowtie_k Y$. In this case, we find the node v of lowest level (i.e. highest depth) which satisfies either of these conditions, and make the new node labeled (Y,k) a child of the node v. If there are no nodes satisfying either condition, we skip (Y,k). Continue this process going backward in time to complete the construction of the witness tree.

Because this will come up repeatedly in our discussion, given a node $v = (Y,k)$, we refer to a child (Y',k) with $Y \bowtie_k Y'$ as a $\bowtie$-*child.* If a node v has a $\bowtie$-child v' we say it v *is saturated by* v' otherwise we say v is *unsaturated*. Evidently in this construction each node may have one or zero $\bowtie$-children.

As in [25], note that all nodes in any level of the witness tree must form an independent set under $\sim$; this will be useful in our analysis.

We next introduce our main Lemma 2.6, which parallels the analysis of [23], connecting the witness trees to the execution logs. However, as pointed out after the proof of the lemma, its proof is much more involved than in [23], since we cannot conduct a direct coupling as in [23].

Lemma 2.6 (Witness Tree Lemma)**.** *Suppose we construct witness trees as described. Let τ be any tree, with nodes labeled by $(Y_1, k_1), (Y_2, k_2) \ldots, (Y_t, k_t)$. Then the probability that the witness tree τ is produced by the execution log, is at most*

$$\mathbf{P}(\tau \text{ occurs as witness tree}) \leq \prod_{s=1}^{t} p^{Y_s} B'_{k_s}(Y_s).$$

Remark. Recall the notation (9) in parsing the value "p^{Y_s}" above.

Proof. We will construct a necessary conditions for the tree τ to appear. These conditions all have the following form: they contain some conditions on the past history of the PRA; as soon as these conditions are triggered, we demand that the random variable which is about to be sampled takes on a specific value. We refer to the preconditions of each of these checks as *triggers* and the demanded event to be the *target*. The target will be either that some category i we are about to resample takes on a specific value j; or that when we detect an atomic bad event $A \in B_k$, we select some specified $Y \subseteq A$ to resample. For the tree τ to appear, it must be the case that each trigger is detected exactly once in the execution log and no two triggers occur simultaneously. In this case, Lemma B.1 from Appendix B can be used to show that the probability of this event is at most the product of the individual target probabilities, namely $p_{i,j}$ and $B'_k(Y)$ respectively.

Recall that we do an initial sampling of all categories (which we can consider the zeroth resampling), followed by a sequence of resamplings. Now, consider a node s of τ labeled by (Y, k). Suppose $\langle i, j \rangle \in Y$. Because any node – labeled (Y', k'), say – in which i is sampled would satisfy $(Y, k) \sim (Y', k')$, it must be that any earlier resamplings of the category i must occur at a lower level of the witness tree. So if we let r denote the number of times that category i has appeared in lower levels of τ, then we are demanding that the r^{th} resampling of category i selects j. Such a condition has the form we mentioned earlier; the trigger is that we have come to resample i for the r^{th} time, and the target is that we select $\langle i, j \rangle$. The probability of this is $p_{i,j}$.

We next consider the probability of selecting set Y. Consider all the nodes $(Y_1, k_1), \ldots, (Y_l, k_l) \approx (Y, k)$ of the witness tree which are at a lower level than Y. To simplify the discussion, we will suppose that Y_i are distinct; if not, we would simply have to count events with their appropriate multiplicities.

We divide these nodes into four mutually exclusive categories:

(1) $Y_i \sim Y$
(2) $k_i = k$ and $Y_i \bowtie_k Y$ and (Y_i, k) is unsaturated
(3) $k_i = k$ and $Y_i \bowtie_k Y$ and (Y_i, k) is saturated by Y'_i and (Y_i, k) is the $\bowtie_k$-child of (Y, k).
(4) $k_i = k$ and $Y_i \bowtie_k Y$ and (Y_i, k) is saturated by Y'_i and (Y_i, k) is not the $\bowtie_k$-child of (Y, k).

Define a *potential time* for (Y, k) as some time in the execution of the PRA after we have selected each node of type (1), (2), or (3), and which satisfies that condition that, for each node of type (4), either Y'_i has not been selected or Y_i and Y'_i have been selected. Note that the potential times are not necessarily contiguous in time — these conditions can flip multiple times between satisfied and unsatisfied.

We begin by claiming that Y must have been selected during a potential time. For, suppose that there is a node Y_i of type (1), but Y occured before Y_i. Then, forming the witness tree in the backward manner, by the time Y is added to the witness tree, Y_i is already present. So Y would be placed below Y_i, contradicting the way we have defined the enumeration. A similar argument is seen to apply to nodes of type (2) and (3). For a node of type (4), we suppose for contradiction

that we encounter Y after Y_i' but before Y_i. In this case, going backward in time, when Y is added to the tree, node Y_i is present but lacks its $\bowtie_k$-child. So Y would eligible to be placed beneath Y_i.

We now claim that the first bad-event A for which Y is eligible during a potential time, that we must select Y. For, suppose not. In order for τ to appear we must eventually select Y; let Y'' be the last such set selected before Y, while Y was eligible during a potential time. Again to simplify the notation we suppose $Y'' \neq Y_1, \ldots, Y_l$.

We claim that Y'' must be placed below Y in the witness tree. This is obvious if $Y'' \sim Y$ so suppose $Y'' \bowtie_k Y$. First suppose that (Y, k) has no $\bowtie_k$ child. In this case, Y'' would be eligible to placed as a $\bowtie_k$ child if it were not already eligible below Y in the witness tree. Next, suppose that Y has a $\bowtie_k$-child Y' in τ. Our definition of a potential time specifies that we encounter Y'' and Y after observing this Y'. In that case, going backward to form the witness tree, at the time Y is created its $\bowtie_k$-child Y' has not been added. So again Y'' would be eligible to be placed under Y.

In any of these cases, the node Y'' appears below Y in the witness tree.

Suppose that node Y'' is not saturated. Then node Y'' is of type (2) in the above listing. This implies that in any potential time, node Y'' would be have selected. But node Y'' is selected during a potential time, i.e. node Y'' is selected *after* node Y'' is selected. This is impossible.

Suppose that node Y'' is saturated by Z. As Z was placed as a child of Y'', it must be that Z occured before Y''. So, during the time that Y'' was created, Y'' would have a node of type (3). This again contradicts the definition of a potential time.

In summary, we have shown that the first potential time in which Y is eligible to be selected for some violated set $A \in B_k$, it is in fact selected. Critically, the target of this condition has a probability of at most $B_k'(Y)$ due to (11), irrespective of what the violated set A was.

It is clear that each condition of the first kind, refers to a distinct target. The second type of event must also — for if (Y, k) and (Y', k') are selected in the same event, then $k = k'$ and $Y = Y'$ and one of these nodes would have placed above the other in the witness tree. Observe that if two identical nodes appear in the witness tree, then our conditions state that one must be selected before the other.

By Lemma B.1, taking the product over all such conditions, the total probability is at most $\prod p^{Y_s} B'_{k_s}(Y_s)$. The key point in this proof is that, based on the execution of the PRA and the information in the tree τ, we can determine exactly when each Y_s, k_s should have been selected and what these should be resampled to. □

The PRA and its proof are very similar to the Moser-Tardos algorithm, and it is tempting to view this as a special case of that algorithm. However, the proof of the analogue of Lemma 2.6 for the Moser-Tardos algorithm uses a quite different argument based on coupling. In that proof, we imagine that before running the resampling algorithm we select in advance all future resamplings of every category. The witness tree τ then imposes necessary conditions on these samplings, whose probability distribution can be computed easily. While this coupling argument works for the value selected in each category, it does not appear to work for bounding the probability of selecting a given Y. For this, we appear to need the "Nostradamus Lemma": Lemma B.1 from Appendix B.

For any tree τ, we define its weight

$$w(\tau) = \prod_s p^{Y_s} B'(Y_s).$$

In order to show that this algorithm converges, we bound the total weight of the the witness trees is small. Using arguments from Moser & Tardos, we can immediately show Theorem 2.5(a):

Theorem 2.5(a). *Suppose there exists $\mu : 2^X \times [K] \to [0, \infty)$ which satisfies, for all $Y \subseteq X$ and all $k \in [K]$*

$$\mu(Y, k) \geq p^Y B'_k(Y) \Big(\prod_{(Y',k') \sim (Y,k)} (1 + \mu(Y', k')) \Big) \Big(1 + \sum_{Y'' \bowtie_k Y} \mu(Y'', k) \Big)$$

Then, the expected number of resamplings of any category i is at most $\sum_{(Y,k):\ Y\sim i} \mu(Y,k)$.

For many applications of the PRA, in which the fractional hitting-sets are relatively simple, this criterion is sufficient. However, it can be awkward to use in more general settings. The reason is that it requires us to specify yet another function μ, and check a constraint for every Y, k. In the next section, we develop an alternative approach to counting witness trees.

2.5. **A new approach to counting witness trees.** As we have seen, the standard accounting of witness trees includes a parameter μ for each bad-event. We will reduce the number of parameters dramatically by rephrasing the LLL criterion in terms of each *category*.

There are several advantages to the category-based approach. First, in the collection of all witness trees, certain subtrees always co-occur; the total weight of these fragments is more critical than the individual weight of any of them. In many cases, there is information about a category which cannot be easily localized to any individual event. (For example – jumping ahead – in the independent-transversal application of Section 4, we may know the *total* number of edges touching a block, without any bounds on the number of edges touching any particular vertex in that block.)

A second advantage that it makes it easier to show that PRA converges in polynomial time. In some applications, the number of bad events may be exponentially large while the number of categories is polynomial. In this case, we need to show an upper bound on the number of resamplings of a category.

Finally, the PRA criterion is given in terms not just of the bad-events, but includes a necessary condition on every *subset* of the bad-events. It is very cumbersome to check this condition individually for each subset. By grouping them into categories, we obtain a much more succinct and tractable condition on the PRA.

This type of accounting is useful not just for our PRA, but for the usual LLL and Moser-Tardos algorithm as well. When applied to the usual Moser-Tardos algorithm, this will give us a simplified and slightly weakened form of Pegden's criterion [25]. Nevertheless, it is stronger and more simple than the standard LLL, particularly the asymmetric LLL.

Lemma 2.8. *Suppose we construct witness trees as described. For any category i, let $T_h(i)$ denote the total weight of all witness trees of height $\leq h$ with root labeled by (Y,k) for an arbitrary k and an arbitrary $Y \subseteq X$ with $Y \sim i$. Let $S_h(k,Y)$ denote the total weight of all witness trees of height $\leq h$ with root labeled by Y, k. Then T_h, S_h satisfy the mutual recurrences*

$$S_{h+1}(k,Y) \leq B'_k(Y) p^Y \left(\prod_{i:\ i\sim Y} (1+T_h(i)) \right) \cdot \left(1 + \sum_{Y':\ Y' \bowtie_k Y} S_h(k,Y') \right) \tag{12}$$

$$T_h(i) \leq \sum_{(k,Y):\ Y\sim i} S_h(k,Y). \tag{13}$$

Proof. The bound (13) is obvious; so we just need to show (12). Let τ be a tree of height $h+1$ rooted at a node r labeled (Y,k). Then the weight of τ is $p^Y B'_k(Y)$ times the weight of all the children of r – which are all trees of height $\leq h$. Let us count these subtrees now. The node (Y,k) may contain one or zero $\bowtie_k$-children — this accounts for the factor "$1 + \sum_{Y':\ Y'\bowtie_k Y} S_h(k,Y')$". The other children are nodes labeled (Y',k'), with $Y' \sim Y$. As noted by [25] (and is easily checked by the way witness trees are constructed), such children of r must be independent under $\sim$. Thus, for each i such that $Y \sim i$, we can have at most one such child of r. This explains the term "$\prod_{i:\ i\sim Y}(1+T_h(i))$" term in (12). □

Lemma 2.9. *Suppose we are given some $S(k,Y) \in [0,\infty)$ for all k and for all $Y \subseteq X$, and some $T(i) \in [0,\infty)$ for each category i, such that for all i,k,Y we satisfy*

$$S(k,Y) \geq B'_k(Y) p^Y \left(\prod_{i:\ i \sim Y} (1 + T(i)) \right) \cdot \left(1 + \sum_{Y' : Y' \bowtie_k Y} S(k,Y') \right)$$

$$T(i) \geq \sum_{k, Y \sim i} S(k,Y)$$

Then we have $T_h(i) \leq T(i)$ and $S_h(k,Y) \leq S(k,Y)$ for all h,i,k,Y. Also, the PRA terminates with probability one, and the expected number of resamplings of any category i is at most $T(i)$.

Proof. The bounds $T_h(i) \leq T(i)$ and $S_h(k,Y) \leq S(k,Y)$ follow by induction on h.

Construct the witness-tree corresponding to each resampling of category i. These are all distinct, and each such tree occurs with probability at most its weight. Hence the expected number of resamplings is at most the sum of the weights of all trees rooted at i, which in turn is at most $T_\infty(i) \leq T(i)$. □

Hence, whatever running-time parameter κ we use, if the number of categories n and all the $T(i)$ are polynomially bounded in κ, and a single iteration of the modified Moser-Tardos algorithm can be implemented in $\text{poly}(\kappa)$ time, then we get a $\text{poly}(\kappa)$-time algorithm to find a configuration avoiding all bad events; in particular, such a configuration exists. The number of bad events and the number of possible assignments to each category may be exponentially large or even infinite, but that presents no problem for the algorithm.

Theorem 2.11 sometimes offers a useful way to employ Lemma 2.9. In many settings, the linkages due to $\bowtie$ are relatively insignificant compared to the linkages due to $\sim$. One possible reason for this is that the $\bowtie$ linkage becomes null; this always occurs in the usual Moser-Tardos algorithm (without partial resampling). Alternatively, there may be many bad events each of which has relatively small probability. We can simplify our LLL in this setting; in particular, we can avoid the mutual recursion as in (12) and (13): we get a "pure" recurrence (14), which can then be used in (15). We start with the following definition:

Definition 2.10. ***(Value $\hat{S}_k$ that depends on a vector $\vec{\lambda}$)*** *Suppose that we are given an assignment of non-negative real numbers $\vec{\lambda}_{i,j}$ to each element $\langle i,j \rangle$. For each k and Y, suppose*

$$\sum_{Y' \bowtie_k Y} B'_k(Y') \vec{\lambda}^{Y'} < 1$$

We define, for each k, the parameter $\hat{S}_k > 0$ to be

$$\hat{S}_k = \max_{Y:\ B'_k(Y) > 0} \frac{1}{1 - \sum_{Y' \bowtie_k Y} B'_k(Y') \vec{\lambda}^{Y'}} \tag{14}$$

Note that if the $\bowtie$ relation is null (i.e. for $B'(Y_1) > 0, B'(Y_2) > 0$ we have $Y_1 \not\bowtie_k Y_2$) then $\hat{S}_k = 1$. Also, we have the upper bound

$$\hat{S}_k \leq \frac{1}{1 - G^k}$$

Theorem 2.11. *Suppose we are given an assignment of non-negative real numbers $\vec{\lambda}_{i,j}$ to each element $\langle i,j \rangle$. For any category i, define $\vec{\lambda}_i = \sum_j \vec{\lambda}_{i,j}$. Suppose that*

$$\forall i,\ \vec{\lambda}_i \geq 1 + \sum_k \hat{S}_k G_i^k, \tag{15}$$

where G_i^k is as in Definition 2.4. Then the PRA terminates with probability one, and the expected number of resamplings of a category i is at most $\vec{\lambda}_i$.

Proof. Use Lemma 2.9 with $p_{i,j} = \frac{\vec{\lambda}_{i,j}}{\vec{\lambda}_i}$, $T(i) = \vec{\lambda}_i - 1$, and $S(k, Y) = \hat{S}_k B'_k(Y)\vec{\lambda}^Y$. □

Parts (b) and (c) of Theorem 2.5 now follows from Theorem 2.11.

Parts (b) and (c) of the Theorem 2.5 can be unified if we define, each element $\langle i, j\rangle$,

$$H_{i,j} = \sum_k \hat{S}_k G^k_{i,j}$$

and similarly $H_i = \sum_j H_{i,j}$. Then the criterion of Theorem 2.5 has the simple form $\vec{\lambda}_i - H_i \geq 1$. We may assume without loss of generality that $\vec{\lambda}_{i,j} \geq H_{i,j}$ for each $\langle i, j\rangle$; for if not, we may set $\vec{\lambda}_{i,j} = H_{i,j} = 0$ and still satisfy Theorem 2.5.

The parameter $H_{i,j}$ will turn out to play the crucial role in Section 2.6 which analyzes the LLL distribution.

It is instructive to compare these formulas to those conditions of Theorem 3.2.

2.6. **The LLL distribution.** If we are given $\vec{\lambda}, B'_k$ satisfying Theorem 2.11, then we know that there exists a configuration which avoids all bad events. Furthermore, such a configuration can be found by running the PRA.

We may wish to learn more about such configurations, other than that they exist. We can use the probabilistic method, by defining an appropriate distribution on the set of feasible configurations. The PRA naturally defines a probability distribution, namely, the distribution imposed on elements after the algorithm terminates.

The following theorem bounds the probability that an event E occurs in the output of the PRA:

Theorem 2.12.

(a) *Suppose that we satisfy Theorem 2.5(a). Then, for any atomic event E, the probability that E true in the output of the PRA, is at most*

$$\mathbf{P}(\textit{PRA output satisfies } E) \leq P(E) \prod_{k, Y \sim E} (1 + \mu(k, Y))$$

(b) *Suppose that we satisfy Theorem 2.5(b) or Theorem 2.5(c). Let J be a set of assignments to category i. The probability that the PRA ever selects $\langle i, j\rangle$ for $j \in J$ is bounded by*

$$\mathbf{P}(\textit{Select } \langle i, j\rangle \textit{ for some } j \in J) \leq \frac{\sum_{j \in J} \vec{\lambda}_{i,j}}{\vec{\lambda}_i - H_i + \sum_{j \in J} H_{i,j}}$$

(c) *Suppose that we satisfy Theorem 2.5(b) or Theorem 2.5(c). The probability that the PRA ever simultaneously selects $\langle i_1, j_1\rangle, \ldots, \langle i_k, j_k\rangle$, is at most*

$$\mathbf{P}(\textit{Select } \langle i_1, j_1\rangle, \ldots, \langle i_k, j_k\rangle) \leq \lambda_{i_1,j_1} \ldots \lambda_{i_k,j_k}$$

Proof. Case (a) is shown in [9]. Case (c) is similar to case (b), but easier. We will only prove case (b).

We consider the first occurence of $\langle i, j\rangle$, for $j \in J$, during the execution of the PRA. There are two cases for this occurence; we may either select $\langle i, J\rangle$ initially, or we resample some $Y \ni \langle i, \bar{J}\rangle$, and then we select some $\langle i, J\rangle$. (We will abuse notation so that $\langle i, J\rangle$ denotes all elements of the form $\langle i, j\rangle$ for $j \in J$; and similarly $\langle i, \bar{J}\rangle$ denotes the elements with $j \notin J$.)

Consider the witness tree corresponding to this event. This tree is either null or is rooted in some $Y \ni \langle i, J\rangle, k$, and cannot contain any instances of $\langle i, \bar{J}\rangle$ below the root.

Let R denote the total weight of all witness trees rooted in some $Y \ni \langle i, \bar{J}\rangle$, below which never occurs any $\langle i, J\rangle$. Consider the root node Y, k; this may have children corresponding to any $Y' \bowtie_k Y$; or it may have children from any of the other categories in Y (other than i); or it may

have another child also rooted in such a $Y \ni \langle i, \bar{J}\rangle$. Along similar lines to Lemma 2.9, if a value $r > 0$ satisfies the condition

$$r \geq \sum_k \sum_{Y \ni \langle i,\bar{J}\rangle} B'_k(Y) p^Y (1 + S(k,Y))(1+r) \prod_{i' \neq i, i' \sim Y} (1 + T(i')) \tag{16}$$

then it must be that $R \leq r$.

The main term of (16) can be rewritten as

$$\begin{aligned}\sum_k \sum_{Y \ni \langle i,\bar{J}\rangle} B'_k(Y) p^Y (1 + S(k,Y)) \prod_{\substack{i' \neq i \\ i' \sim Y}} (1 + T(i')) &= \sum_k \sum_{j \notin J} \sum_{Y \ni \langle i,j\rangle} B'_k(Y) \vec{\lambda}^Y \frac{p_{i,j}}{\vec{\lambda}_{i,j}} (1 + S(k,Y)) \\ &= \sum_{j \notin J} \frac{p_{i,j}}{\vec{\lambda}_{i,j}} H_{i,j} = \frac{1}{\vec{\lambda}_i} \sum_{j \notin J} H_{i,j}\end{aligned}$$

Hence we have $R \leq \frac{H_{i,\bar{J}}}{\vec{\lambda}_i - H_{i,\bar{J}}}$. Bearing in mind that we may have initially selected $\langle i, J\rangle$, we have

$$\begin{aligned}P(\text{ever select } \langle i, J\rangle) &\leq p_{i,J}(1+R) \\ &\leq p_{i,j}(1 + \frac{H_{i,\bar{J}}}{\vec{\lambda}_i - H_{i,\bar{J}}}) \\ &= \frac{\sum_{j \in J} \vec{\lambda}_{i,j}}{\vec{\lambda}_i - H_i + \sum_{j \in J} H_{i,j}}\end{aligned}$$

□

A simple corollary of Theorem 2.12 shows a *lower bound* on the probability that the PRA terminates by selecting a given element $\langle i, j\rangle$:

Corollary 2.13. *Suppose that we satisfy Theorem 2.5(b) or Theorem 2.5(c). Let J be a set of possible assignments to category i. The probability that the PRA terminates by selecting $\langle i, j\rangle$ for $j \in J$ is bounded by*

$$\mathbf{P}(\textit{PRA finally selects } \langle i, j\rangle \textit{ for some } j \in J) \geq \frac{\sum_{j \in J} \vec{\lambda}_{i,j} - \sum_{j \in J} H_{i,j}}{\vec{\lambda}_i - \sum_{j \in J} H_{i,j}}$$

Proof. Apply Theorem 2.12(b) to bound from above the probability of ever selecting $\langle i, \bar{J}\rangle$. □

3. A probabilistic LLL variant

The standard LLL has two faces: the efficient algorithm of Moser-Tardos, which finds a valid configuration by resampling bad-events, and the original probabilistic formulation of Erdos and Lovász [7]. In the latter formulation, one selects the variables according to the indicated distribution, without any resamplings. One then has a positive, exponentially small, probability of avoiding all bad-events. This typically gives an existence proof, without a corresponding polynomial-time algorithm.

These two interpretations of the LLL lend themselves to different generalizations and there are many useful interconnections between them. Historically, many improvements and specializations of the LLL were first developed in the probabilistic framework (including the original LLL itself), and then were translated into the Moser-Tardos framework. The Partial Resampling Algorithm we have given follows the opposite path: it is a generalization of the Moser-Tardos framework, without any probabilistic interpretation.

In this section, we will describe a probabilistic variant of the LLL which closely parallels the PRA. This process does not exactly match the bounds of the PRA; it is sometimes stronger but usually weaker. We will discuss the connections between the interpretations in Section 3.3.

We begin by setting and recalling some notations. We let $[n]$ index the blocks, and let $Z_{i,j}$ be the indicator for selecting the jth element from block i. Each bad event B_k is determined by, and is an increasing function of a subset S_k of the Z variables. (Note that this differs from the formulation of the PRA, which never explicitly required that the B_k had sparse support). As before, we refer to an ordered pair $\langle i, j\rangle$ as an *element*. Recall that we let X denote the set of all elements; we let $|X| = N$.

3.1. **The assignment LLL.** Unlike in the usual LLL, we cannot simply define a probability distribution and compute the probability of the bad event occuring. We will only have partial control over this probability distribution, and the following definition will be important:

Definition 3.1. ***(UNC)*** *Given a probability distribution Ω on the underlying indicator variables Z, we say that Ω satisfies* upper negative correlation with respect to probability vector p *or simply "UNC(p)" if all entries of p lie in $[0, 1]$ and if for all elements $x_1, \ldots, x_k$, we have*

$$\mathbf{P}_\Omega(Z_{x_1} = \cdots = Z_{x_k} = 1) \leq p_{x_1} \ldots p_{x_k}.$$

For an event E and a probability vector p, we define $\mathbf{P}^*_p(E)$ to be the minimum, over all Ω satisfying UNC(p), of $\mathbf{P}_\Omega(E)$. (As the number of variables is finite, this minimum is achieved.)

Essentially, when computing $\mathbf{P}^*(E)$, we are not allowing the random variables Z to be positively correlated. For some types of events, such as large-deviation events, this allows us to control the probability very strongly; for other events, such as a union of many events, this is no better than the union bound.

Our main theorem is:

Theorem 3.2 (Assignment LLL)**.** *Suppose we are given a CSP for which there exists $\lambda \in [0, 1]^N$ such that when we sample all the Z_x independently with $\Pr[Z_x = 1] = \lambda_x$, we have*

$$\forall i \in [n],\ \sum_j \lambda_{i,j} \cdot \mathbf{P}^*_\lambda[\bigcap_{B \in \mathcal{B}_{i,j}} \overline{B} \ \big|\ Z_{i,j} = 1] > 1.$$

Then, if no bad-event $B \in \mathcal{B}$ is a tautology, the CSP is feasible.

To prove the theorem, we will study the following probabilistic process. We are given a vector $p \in [0, 1]^N$ of probabilities, one for each indicator Z_x. Each Z_x is drawn *independently* as Bernoulli-p, i.e. $P(Z_x = 1) = p_x$. (If for some event x we have $p_x > 1$, then by an abuse of notation, we take this to mean that $Z_x = 1$ with certainty). Our goal is to satisfy all the assignment constraints and avoid all the events in $\mathcal{B}$. If $\mathcal{C} \subseteq \mathcal{B}$, we use the notation $\exists\mathcal{C}$ to denote the event that some $B \in \mathcal{C}$ occurs. So in this case, we want to *avoid* the event $\exists\mathcal{B}$. For an element x we define $\mathcal{B}_x$ to index the set of all bad events $B_k \in \mathcal{B}$ which are (explicitly) affected by Z_x.

We recall a basic lemma concerning increasing and decreasing events, which follows from the FKG inequality [8]:

Lemma 3.3. *Let $X_0, X_1 \subseteq X$ be two disjoint subsets of the elements. Let E_1 be some event depending solely on variables in X_1. Let E^- be a decreasing event. Then,*

$$\mathbf{P}(\forall x \in X_0 \ \ Z_x = 1 \mid \mathcal{E}_1, E^-) \leq p^{X_0}$$

Similarly, if E^+ is increasing, then $\mathbf{P}(\forall x \in X_0 \ \ Z_x = 1 \mid E_1, E^+) \geq \prod_{x \in X_0} p^{X_0}$.

Recall that we are using the power notation so that p^{X_0} means simply $\prod_{x \in X_0} p_x$.

Proof. We will only prove the first part of this lemma; the second is analogous.

We average over all assignments to the variables Z_x, for $x \in X_1$. For any such assigment-vector $\vec{z}$, the event $\bigwedge_{x \in X_0} Z_x = 1$ is an increasing function, while E^- is a decreasing function in the remaining variables. Hence, by FKG, the probability of this event conditional on $(Z_{X_1} = \vec{z} \wedge E^-)$ is at most its value conditional on $Z_{X_1} = \vec{z}$ alone. But, the events $\bigwedge_{x \in X_0} Z_{X_0} = 1$ and $Z_{X_1} = \vec{z}$ involve disjoint sets of variables, so they are independent. Hence this probability is at most the unconditional probability of $\bigwedge_{x \in X_0} Z_x = \vec{1}$, namely p^{X_0}. □

If $A \subseteq [n]$ is any subset of the blocks, we define the event Assigned(A) to be the event that, for all $i \in A$, there is *at least one* value of j for which $Z_{i,j} = 1$. Our goal is to satisfy the constraint Assigned($[n]$). If $i \in [n]$ we write Assigned(i) as short-hand for Assigned($\{i\}$). Because all the bad events are increasing Boolean functions, if we can find a configuration in which each block has at least one value assigned, we can easily alter it to a feasible configuration in which each block has *exactly one* value assigned.

We are now ready to state the first lemma concerning this probabilistic process. We want to show that the there is a positive probability of satisfying all the assignment constraints and avoiding all bad events. We will show the following by induction, with stochastic domination playing a key role:

Lemma 3.4. *Let $\epsilon < 1$. Suppose $p \in [0, \epsilon]^N$ is a probability vector such that for all blocks i,*

$$\sum_j p_{i,j}(\mathbf{P}^*_{p/\epsilon}(\neg\exists \mathcal{B}_{i,j} \mid Z_{i,j} = 1)) - \sum_{j,j'} p_{i,j}p_{i,j'} \geq \epsilon.$$

Then for any block i, any $\mathcal{C} \subseteq \mathcal{B}$ a set of bad events, and any $A \subseteq [n]$, we have

$$\mathbf{P}(\textit{Assigned}(i) \mid \neg\exists B', \textit{Assigned}(A)) \geq \epsilon.$$

Proof. We show this by induction on $|\mathcal{C}| + |A|$. We may assume that $i \notin A$, as otherwise this is vacuous. First, suppose $|\mathcal{C}| = 0$. Then, $\mathbf{P}(\text{Assigned}(i) \mid \neg\exists \mathcal{C}, \text{Assigned}(A))$ equals $\mathbf{P}(\text{Assigned}(i))$ as these events are independent. By Inclusion-Exclusion, the latter probability is at least

$$\mathbf{P}(\text{Assigned}(i)) \geq \sum_j p_{i,j} - \sum_{j,j'} p_{i,j}p_{i,j'}$$

and it is easy to see that the lemma's hypothesized constraint implies that this is at least ϵ.

Next suppose $|\mathcal{C}| > 0$. We use Inclusion-Exclusion to estimate $\mathbf{P}(\text{Assigned}(i) \mid \neg\exists \mathcal{C}, \text{Assigned}(A))$. First, consider the probability that a distinct pair j, j' in block i are jointly chosen, conditional on all these events. For this, by Lemma 3.3 we have

$$\mathbf{P}(Z_{i,j} = Z_{i,j'} = 1 \mid \text{Assigned}(A), \neg\exists \mathcal{C}) \leq p_{i,j}p_{i,j'} \tag{17}$$

as $i \notin A$ and $\neg\exists \mathcal{C}$ is decreasing.

Let us fix j. We next need to show a lower bound on $\mathbf{P}(Z_{i,j} = 1 \mid \text{Assigned}(A), \neg\exists \mathcal{C})$. This is easily seen to equal $\mathbf{P}(Z_{i,j} = 1)$ if $\mathcal{B}_{i,j} \cap \mathcal{C} = \emptyset$, so we can assume $\mathcal{B}_{i,j} \cap \mathcal{C} \neq \emptyset$. We see by Bayes' Theorem that $\mathbf{P}(Z_{i,j} = 1 \mid \text{Assigned}(A), \neg\exists \mathcal{C})$ equals

$$\frac{\mathbf{P}(\neg\exists(\mathcal{C} \cap \mathcal{B}_{i,j}) \mid Z_{i,j} = 1, \text{Assigned}(A), \neg\exists(\mathcal{C} - \mathcal{B}_{i,j}))}{\mathbf{P}(\neg\exists(\mathcal{C} \cap \mathcal{B}_{i,j}) \mid \text{Assigned}(A), \neg\exists(\mathcal{C} - \mathcal{B}_{i,j}))} \times \mathbf{P}(Z_{i,j} = 1 \mid \text{Assigned}(A), \neg\exists(\mathcal{C} - \mathcal{B}_{i,j}))$$

since the denominator is a probability, Lemma 3.3 yields

$$\mathbf{P}(Z_{i,j} = 1 \mid \text{Assigned}(A), \neg\exists \mathcal{C}) \geq p_{i,j} \cdot \mathbf{P}(\neg\exists \mathcal{B}_{i,j} \mid Z_{i,j} = 1, \text{Assigned}(A), \neg\exists(\mathcal{C} - \mathcal{B}_{i,j})).$$

(This approach to handling a conditioning was inspired by [6].)

Consider the random variables Z **conditioned on** the events Assigned(A), $\neg\exists(\mathcal{B} - \mathcal{B}_{i,j})$, $Z_{i,j} = 1$. Our *key claim* now is that these conditional random variables Z (apart from $Z_{i,j}$ itself) satisfy

UNC(p/ϵ): note that p/ϵ is a valid probability vector since $p \in [0,\epsilon]^N$. To show this, we need to upper-bound $\mathbf{P}(\mathcal{E}_1 \mid \mathcal{E}_2)$, where

$$\mathcal{E}_1 \equiv (Z_{i'_1,j'_1} = \cdots = Z_{i'_k,j'_k} = 1) \text{ and}$$
$$\mathcal{E}_2 \equiv (\text{Assigned}(A), \neg\exists(Y - \mathcal{B}_{i,j}), Z_{i,j} = 1),$$

and where k and $i'_1, j'_1, \ldots, i'_k, j'_k$ are arbitrary. Letting $I' = \{i'_1, \ldots, i'_k\}$, we also define

$$\mathcal{E}_3 \equiv (Z_{i,j} = 1, \text{Assigned}(A - I'), \neg\exists(\mathcal{B} - \mathcal{B}_{i,j})).$$

By simple manipulations, we see that $\mathbf{P}(\mathcal{E}_1 \mid \mathcal{E}_2)$ is at most

$$\mathbf{P}(\mathcal{E}_1 \mid \mathcal{E}_3)/\mathbf{P}(\text{Assigned}(i'_1, \ldots, i'_k) \mid \mathcal{E}_3). \tag{18}$$

Note that $\mathcal{E}_1$ does not share any variables with $(Z_{i,j} = 1, \text{Assigned}(A - i'_1 - \cdots - i'_k))$, and that $\neg\exists(\mathcal{B} - \mathcal{B}_{i,j})$ is a decreasing event. Hence by Lemma 3.3 the numerator is at most $p_{i'_1,j'_1} \ldots p_{i'_k,j'_k}$. Now let us examine the denominator. The variable $Z_{i,j}$ does not affect any of the events mentioned in the denominator, so we may remove it from the conditioning:

$$\mathbf{P}(\text{Assigned}(I') \mid \mathcal{E}_3) = \mathbf{P}(\text{Assigned}(I') \mid \text{Assigned}(A - I'), \neg\exists(\mathcal{B} - \mathcal{B}_{i,j})),$$

which in turn is at least $\epsilon^{|I'|}$, by iterated application of the induction hypothesis (recall that $\mathcal{B}_{i,j} \cap \mathcal{C} \neq \emptyset$).

Putting this all together, we have that the probability of the event $Z_{i'_1} = \cdots = Z_{i'_k}$ is at most $p^k/\epsilon^{|I'|} \leq (p/\epsilon)^k$. So the random variables Z satisfy UNC(p/ϵ) and we have

$$\mathbf{P}(\neg\exists\mathcal{B}_{i,j} \mid Z_{i,j} = 1, \text{Assigned}(A), \neg\exists(\mathcal{C} - \mathcal{B}_{i,j})) \geq \mathbf{P}^*_{p/\epsilon}(\neg\exists\mathcal{B}_{i,j} \mid Z_{i,j} = 1)$$

The right-hand side is substantially simpler, as there is *no conditioning to link the variables*. Substituting this into (17) and (3.1), we get

$$\mathbf{P}(\text{Assigned}(i) \mid \text{Assigned}(A), \neg\exists\mathcal{C}) \geq \sum_j p_{i,j}\mathbf{P}^*_{p/\epsilon}(\neg\exists\mathcal{B}_{i,j} \mid Z_{i,j} = 1) - \sum_{j,j'} p_{i,j}p_{i,j'}$$

and by our hypothesis the right-hand side is at least ϵ. □

We can now allow all entries of p to tend to 0 at the same rate, which simplifies our formulae:

Theorem 3.2 (Assignment LLL – restated)**.** *For any element $x \in X$ and any vector of probability $\lambda \in [0,1]^N$ define*

$$h_x(\lambda) = \mathbf{P}^*_\lambda(\neg\exists\mathcal{B}_x \mid Z_x = 1)$$

For any block i define

$$H_i(\lambda) = \sum_j \lambda_{i,j} h_{i,j}(\lambda)$$

Suppose that for all blocks $i \in [n]$ we satisfy the constraint $H_i(\lambda) > 1$. Then the corresponding CSP has a feasible solution.

Proof. Let $p = \alpha\lambda$ and let $\epsilon = \alpha$ for some $\alpha > 0$. In order to use Lemma 3.4, it suffices to satisfy the constraint for all i

$$\sum_j p_{i,j} h_{i,j}(p/\epsilon) - \sum_{j,j'} p_{i,j}p_{i,j'} \geq \epsilon. \tag{19}$$

Let us fix a block i. Suppose we allow $\alpha \to 0$. In this case, (19) will be satisfied for some $\alpha > 0$ sufficiently small if we have

$$\sum_j \lambda_{i,j} \cdot \mathbf{P}^*_\lambda(\neg\exists\mathcal{B}_{i,j} \mid Z_{i,j} = 1) > 1$$

As there are only finitely many blocks, there is $\alpha > 0$ sufficiently small which satisfies all constraints simultaneously.

In this case, we claim that there is a positive probability of satisfying $\text{Assigned}(i), \overline{B}_k$ for all blocks i and all bad events B_k, when we assign variables Z independently Bernoulli-p. First, $\mathbf{P}(\neg\exists\mathcal{B}) \geq \prod_{x\in X} \mathbf{P}(Z_x = 0)$, since no B_k is a tautology; the latter product is clearly positive for small-enough α. Next, by Lemma 3.4 and Bayes' Theorem,

$$\mathbf{P}(\text{Assigned}(1) \wedge \cdots \wedge \text{Assigned}(n) \mid \neg\exists Y) \geq \prod_{i=1}^{n} \epsilon > 0.$$

In particular, there is a configuration of the Z_x which satisfies all the constraints simultaneously. □

3.2. **Computing $\mathbf{P}^*$.** In the usual LLL, one can fully specify the underlying random process, so one can compute the probability of a bad event fairly readily. In the assignment LLL, we know that the random variables must satisfy their UNC constraints, but we do not know the full distribution of these variables. This can make it much harder to bound the probability of a bad event.

Roughly speaking, the UNC constraints force the underlying variables to be negatively correlated (or independent). For some types of bad events, this is enough to give strong bounds:

Lemma 3.6. *For random variables $Z_{x_1}, \ldots, Z_{x_k}$, let $\mu = \lambda_{x_1} + \cdots + \lambda_{x_k}$. Then*

$$\mathbf{P}^*_\lambda(Z_{x_1} + \cdots + Z_{x_k} \leq \mu(1+\delta)) \geq 1 - \Big(\frac{e^\delta}{(1+\delta)^{1+\delta}}\Big)^\mu$$

Proof. The Chernoff upper-tail bound applies to negatively correlated random variables [24]. □

Suppose we have an increasing bad event B which depends on $Z_{x_1}, \ldots, Z_{x_k}$. We are given $\lambda \in [0,1]^N$. Note that Ω is a probability distribution on $Z_1, \ldots, Z_N$, but we abuse notation to view it as a distribution on $Z_{x_1}, \ldots, Z_{x_k}$ as well. We describe a generic algorithm to compute $\mathbf{P}^*_\lambda(\overline{B})$ (we sometimes just denote $\mathbf{P}^*_\lambda(\cdot)$ as $\mathbf{P}^*(\cdot)$).

As B is an increasing event, we can write $B = a_1 \vee \cdots \vee a_n$, where each $a_i \in B$ is an atomic event, and is minimal in the sense that for all $a' < a_i$ we have $a' \notin B$. We assume $0 \notin B$, as otherwise B is a tautology and $\mathbf{P}^*(B) = 1$.

We say that a probability distribution Ω on the variables $Z_{x_1}, \ldots, Z_{x_k}$ is *worst-case* if $\mathbf{P}^*_\lambda(B) = \mathbf{P}_\Omega(B)$. By finiteness, such Ω exists. The basic idea of our algorithm is to view each $\mathbf{P}_\Omega(\omega)$ as an unknown quantity, where $\omega \in \Omega$ is an atomic event. We write $q_\omega = \mathbf{P}_\Omega(\omega)$ for simplicity. In this case, $\mathbf{P}_\Omega(B)$ is the sum

$$\mathbf{P}_\Omega(B) = \sum_{\omega\in B} q_\omega$$

Furthermore, the UNC constraints can also be viewed as linear constraints in the variable q_ω. For each $x'_1, \ldots, x'_{k'}$, we have the constraint

$$\sum_{\omega_{x'_1} = \cdots = \omega_{x'_{k'}} = 1} q_\omega \leq \lambda_{x'_1} \ldots \lambda_{x'_{k'}}$$

This defines a linear program, in which we maximize the objective function $\mathbf{P}^*(B) = \sum_{\omega\in B} q_\omega$ subject to the UNC constraints. The size of this linear program may be enormous, potentially including 2^k variables and 2^k constraints. However, in many applications k is a parameter of the problem which can be regarded as constant, so such a program may still be tractable.

In general, we can reduce the number of variables and constraints of this linear program with the following observations:

Proposition 3.7. *There is a distribution Ω such that $\mathbf{P}_\Omega(B) = \mathbf{P}^*(B)$ and such that Ω is only supported on the atomic events $\{0, a_1, \ldots, a_n\}$.*

Proof. Suppose Ω satisfies the UNC constraints. Then define Ω' as follows. For each atomic event ω, if $\omega \notin B$, then shift the probability mass of ω to 0; otherwise, shift the probability mass of ω to any minimal event a_i underneath it. This preserves all UNC constraints as well as the objective function. □

For some types of bad events, there are certain symmetries among classes of variables. In this case, one can assume that the distribution Ω is symmetric in these variables; hence the probabilities of all such events can be collapsed into a single variable.

Proposition 3.8. *Given a group $G \subseteq S_k$, where S_k is the symmetric group on k letters, define the group action of G on a probability distribution Ω by permutation of the indices and on λ by permutation of the coordinates. Suppose B, λ are closed under the action of G. Then there is a worst-case probability distribution Ω which is closed under G. For this probability distribution Ω, we only need to keep track of a single unknown quantity q' for each* orbit *of G.*

Proof. Given a worst-case distribution Ω, let $\Omega' = \frac{1}{|G|}\sum_{g\in G} g\Omega$. As B is closed under G, each of the distributions $g\Omega$ has the same probability of the bad event B. As λ is closed under G, all the UNC constraints are preserved in each $g\Omega$. □

3.3. **Comparing the Assignment LLL and PRA.** Although it is not obvious in the form we have stated it, there are connections between the PRA, and in particular Theorem 2.11, Assignment LLL of Theorem 3.2. Of course, the latter is nonconstructive, while the former is usually a polynomial-time algorithm. Moreover, for all the *applications* in this paper, the PRA leads to efficient algorithms, and in most cases, to bounds which are (significantly) improved compared to the Assignment LLL.

However, we have not been able to prove a general theorem to the effect that the PRA is a constructive form of the Assignment LLL or is always better. Indeed, there appear to be some parameter regimes of Theorem 5.2 in which Theorem 3.2 can give slightly better bounds. In order to explain some links between the Assignment LLL and our resampling approach, we next give some intuitive connections between the two.

Note that in Theorem 2.11, the values of p and T are not significant by themselves, only their product $\vec{\lambda} = p(1+T)$. A way to interpret this result is that we are "oversampling" each element x. We imagine that there is a process in which individual indicator variable Z_x is Bernoulli-$\vec{\lambda}_x$, and the Z_x variables are negatively correlated, but we relax the restriction that exactly one element is selected from each category.

A fractional hitting-set B' provides an upper bound on the probability of the bad event B. Furthermore, this upper bound depends only on the negative correlation of these variables:

Proposition 3.9. *For each element $x = \langle i, j\rangle \in X$, we define the indicator variable Z_x which is 1 is x is selected and 0 otherwise. Suppose that the indicator variables Z_x are each individually Bernoulli-$\vec{\lambda}$ and are negatively correlated. (We are provided no other information on their distribution). Then the probability of any bad event B is at most $\mathbf{P}(B) \leq \sum_{Y\subseteq B} B'(Y)\vec{\lambda}^Y$.*

Furthermore, let p^* denote the maximum possible value of $\mathbf{P}(B)$, over all distributions Ω on the indicator variables which satisfy these two properties, namely that the indicator variables are individually Bernoulli-$\vec{\lambda}$ and are negatively correlated. Then there is a fractional hitting-set B' with $p^* = \sum_{Y\subseteq B} B'(Y)\vec{\lambda}^Y$.

Proof. Consider the following process: we draw the variables Z as indicated. If the bad event B occurs, we select some violated subset $A \subseteq B$ and draw $Y \subseteq A$ with probability proportional to $B'(Y)$. Otherwise we do not draw. The probability of selecting a subset is $\mathbf{P}(B)$. But, we can also

count it as

$$\begin{aligned}\mathbf{P}(B) &= \sum_{Y \subseteq B} \mathbf{P}(Y \mid B) \\ &\leq \sum_{Y} \mathbf{P}(\forall y \in Y \quad Z_y = 1) \sum_{Y \subseteq A \in B} \mathbf{P}(\text{select } A \text{ from } Y) \cdot \mathbf{P}(\text{select } A) \\ &\leq \sum_{Y} \vec{\lambda}^Y B'(Y)\end{aligned}$$

We prove that there is a hitting-set B' achieving $p^* = \sum_{Y \subseteq B} B'(Y)\vec{\lambda}^Y$ by LP duality. We can view the probability of each atomic event $\omega \in \Omega$ as a linear unknown. Then the constraint that the variables are negatively correlated is a linear constraint, and the objective function $\mathbf{P}(B)$ is linear. It is not hard to see that a feasible dual corresponds to a fractional hitting-set. □

We can imagine that we are drawing the variables Z_x, not by selecting exactly one element from each category, but according to some more complicated distribution of which we know two things: the marginal distribution of each element is $\vec{\lambda}_x$; and the elements are negatively correlated.

In this case, the term G^k is measuring the probability of a bad event. The term G^k_x is measuring how much of this probabiltiy is "due to" the variable x. If the variable x does not affect the bad event at all, then $G^k_x = 0$. If the bad event is equivalent to x being selected (i.e. $B = \{x\}$), then $G^k_x = \vec{\lambda}_x$. The standard LLL would not distinguish between a variable affecting the bad event by "a little" or "a lot", but the Partial Resampling Algorithm interpolates smoothly between these extremes.

One important difference between the Assignment LLL and the Partial Resampling Algorithm is that for the Assignment LLL, we *compute* $\mathbf{P}^*$ based on the structure of the bad-events. For the PRA, we *choose* the fractional hitting-set B'. For a given bad-event B, there is not necessarily a "best" choice of B'; a choice of B' can affect the whole dependency structure of the bad-events, and this can have complicated interactions. In general, the optimal choice for the Assignment LLL is not necessarily optimal for the PRA.

Furthermore, although the assignment LLL does (implicitly) choose a fractional hitting-set for each bad-event B_k, in fact this hitting-set may depend not only on k but on the category i it impacts. The PRA can only select a *single* fractional hitting-set for each B_k. Although the PRA is generally stronger, in this one regard it can be slightly weaker than the assignment LLL.

4. Transversals with omitted subgraphs

Suppose we are given a graph $G = (V, E)$ with a partition of its vertices into sets $V = V_1 \sqcup V_2 \sqcup \cdots \sqcup V_l$, each of size b. We refer to these sets as *blocks* or *classes*. We wish to select exactly one vertex from each block. Such a set of vertices $A \subseteq V$ is known as a *transversal*. There is a large literature on selecting transversals such that the graph induced on A omits certain subgraphs. (This problem was introduced in a slightly varying form by [5]; more recently it has been analyzed in [32, 15, 33, 16, 13]). For example, when A is an independent set of G (omits the 2-clique K_2), this is referred to as an *independent transversal*.

It is well-known that a graph G with n vertices and average degree d has an independent set of size at least $n/(d+1)$. For an independent transversal, a similar criterion exists. Alon gives a short LLL-based proof that a sufficient condition for such an independent transversal to exist is to require $b \geq 2e\Delta$ [2], where Δ is the maximum degree of any vertex in the graph. Haxell provides an elegant topological proof that a sufficient condition is $b \geq 2\Delta$ [14]. The condition of [14] is existentially optimal, in the sense that $b \geq 2\Delta - 1$ is not always admissible [16, 33, 32]. The work of [15] gives a similar criterion of $b \geq \Delta + \lfloor \Delta/r \rfloor$ for the existence of a transversal which induces no connected component of size $> r$. (Here $r = 1$ corresponds to independent transversals.) Finally,

the work of [22] gives a criterion of $b \geq \Delta$ for the existence of a transversal omitting K_3; this is the optimal constant but the result is non-constructive.

These bounds are all given in terms of the *maximum degree* Δ, which can be a crude statistic. The proof of [14] adds vertices one-by-one to partial transversals, which depends very heavily on bounding the maximum degree of any vertex. It is also highly non-constructive. Suppose we let d denote the maximum *average* degree of any class V_i (that is, we take the average of the degree (in G) of all vertices in V_i, and then maximize this over all i). This is a more flexible statistic than Δ. We present our first result here in parts (R1), (R2), and (R3) of Theorem 4.1. As shown in [16, 33, 32], the result of (R1) cannot be improved to $b \geq 2\Delta - 1$ (and hence, in particular, to $b \geq 2d - 1$). As shown in [22], the result of (R3) cannot be improved to $b \geq cd$ for any constant $c < 1$. The result (R1) for independent transversals can also be obtained using the LLL variant of [25], but (R2) and (R3) appear new to our knowledge.

Theorem 4.1. *Suppose we have a graph G whose vertex set is partitioned into blocks of size at least b. Suppose that the average degree of the vertices in each block is at most d. Then:*

(R1): *If $b \geq 4d$, then G has an independent transversal;*
(R2): *If $b \geq 2d$, then G has a transversal which induces no connected component of size > 2;*
(R3): *If $b \geq (4/3)d$, then G has a transversal which induces no 3-clique K_3.*

Furthermore, these transversals can be constructed in expected polynomial time.

Proof. In the framework of our PRA, we associate each block to a category. For each forbidden subgraph which appears in G, we associate an atomic bad event. (Note that the atomic bad events for (R2) are all the path of length two in G; for (R3) they are the triangles of G.)

There is a single bad event $B = B_1$, which is that one of the forbidden subgraphs appears in the transversal. We define a fractional hitting set B' as follows. For each edge $f = \langle u, v \rangle \in E$ we assign weight $B'(\{u, v\}) = 1/r$, where r is a parameter depending on the structure we are avoiding. For case (R1), we assign $r = 1$. For case (R2), we assign $r = 2$; for case (R3) we assign $r = 3$. (B' is zero everywhere else.) Now, note that in any of the three cases, the atomic bad events all involve exactly r edges, so the fractional hitting set is valid. Furthermore, any pair of such edges overlap in at least one vertex, so the $\bowtie$ relation is null in this case – i.e., all 1-neighborhoods are empty (recall that the only index k for a bad event here is $k = 1$).

Note that the precondition of Theorem 2.5(c) holds here. We apply Theorem 2.5(c) with all entries of $\vec{\lambda}$ being α, for some scalar α to be determined. Let d_v denote the degree of vertex v. Then, in order to prove (15) for category i (i.e., for block V_i), what we need is

$$b\alpha - \sum_{v \in V_i} d_v \alpha^2 / r \geq 1, \text{ i.e., } b\alpha - bd\alpha^2/r \geq 1 \text{ suffices.}$$

This has a solution $\alpha > 0$ iff

$$b \geq \frac{4d}{r},$$

which gives us the three claimed results. □

4.1. **Avoiding large cliques.** For avoiding cliques of size $s > 3$, the above approach based on the maximum average degree d no longer works; we instead give a bound in terms of the maximum degree Δ. We will be interested in the case when both s and Δ is large. That is, we will seek to show a bound of the form $b \geq \gamma_s \Delta + o(\Delta)$, where γ_s is a term depending on s and s is large. Clearly we must have $\gamma_s \geq 1/(s-1)$; e.g., for the graph $G = K_s$, we need $b \geq 1 = \Delta/(s-1)$. An argument of [32] shows the slightly stronger lower bound $\gamma_s \geq \frac{s}{(s-1)^2}$; intriguingly, this is conjectured in [32] to be exactly tight. On the other hand, a construction in [22] shows that $\gamma_s \leq 2/(s-1)$. This is non-constructive, even for fixed s; this is the best upper-bound on γ_s previously known.

We show that the lower-bound of [32] gives the correct *asymptotic* rate of growth, up to lower-order terms; i.e., we show in Theorem 4.2 that $\gamma_s \leq 1/s + o(1/s)$. In fact, we will show that when $b \geq \Delta/s + o(\Delta)$, we can find a transversal which avoids any s-star; that is, all vertices have degree $< s$. This implies that the transversal avoids K_s. Furthermore, such transversals can be found in polynomial time.

Comparison with the standard LLL: Before we give our construction based on the PRA, we discuss how one might approach this problem using the standard LLL, and why this approach falls short. As in [2], we make the natural random choice for a transversal: choose a vertex randomly and independently from each V_i. Suppose we define, for each s-clique H of the graph, a separate bad event. Each bad event has probability $(1/b)^s$. We calculate the dependency of an s-clique as follows: for each vertex $v \in H$, we choose another v' in the category of v, and v' may be involved in up to $\Delta^{s-1}/(s-1)!$ other s-cliques. This gives us the LLL criterion

$$e \times (1/b)^s \times sb\Delta^{s-1}/(s-1)! \leq 1$$

which gives us the criterion

$$b/\Delta \geq (\frac{es}{(s-1)!})^{\frac{1}{s-1}} = e/s + o(1/s)$$

Now note that when we are calculating the dependency of a bad event, we must take the worst-case estimate of how many other s-cliques a given vertex v may participate in. We bound this in terms of the edges leaving v, so the number of such s-cliques is at most $\Delta^{s-1}/(s-1)!$. However, heuristically, this is a big over-estimate; there is an additional constraint that the endpoints of all $s-1$ such edges are themselves connected, which is very unlikely. Unfortunately, for any given vertex v, or even a whole partition V_i, we cannot tighten this estimate; this estimate can only be tightened in a *global* sense. The LLL is focused on the very local neighborhood of a vertex, and so it cannot "see" this global condition.

In implementing the PRA, it is simpler to enforce the stronger condition that the traversal produced omits s-stars. (That is, in the traversal T, no vertex may have induced degree $\geq s$). The PRA, gives us a way to "localize" global information to the neighborhood of a vertex. We will resample only $r \leq s$ of the vertices in an s-star, chosen uniformly. In addition to all the local information about the given position, we also know the global information that a neighborhood of an s-star must contain $\binom{s}{r}$ separate r-stars.

We now present our theorem here:

Theorem 4.2. *There is a constant $c > 0$ such that, whenever*

$$b \geq \Delta/s + cs^{-3/2} \log s$$

then there is a transversal which omits any s-stars. Furthermore, such a transversal can be found in polynomial time.

Proof. We will use Theorem 2.5(c), assigning the same vector $\vec{\lambda} = \alpha$ where α is a constant to be chosen. There is a single bad event, which is that a K_s appears. We use the following fractional hitting, which assigns weight $\binom{s}{r}^{-1}$. In this case, $\bowtie$ is null: for any two r-stars H, H' which both correspond to the same s-star, will overlap in their central vertex.

Then the condition of Theorem 2.5(c) becomes

$$b\alpha - b\Big(\binom{\Delta}{r} + \Delta\binom{\Delta-1}{r-1}\Big)\binom{s}{r}^{-1}\alpha^{r+1} \geq 1$$

Routine algebra shows that this is satisfied when $b \geq \Delta/s + cs^{-3/2}\log s$, for some sufficiently large constant c.

To implement a step of the PRA, one must search the graph for any s-star in the current candidate transversal; this can be done easily in polynomial time. □

We note that this result improves on [22] in three distinct ways: it gives a better asymptotic bound; it is fully constructive; it finds a transversal omitting not only s-cliques but also s-stars.

4.2. **Weighted independent transversals.** We next study *weighted* transversals, as considered by [1]. We use our weighting condition to lower- and upper- bound the weights of independent transversals, which is not possible using [14] or [25].

Suppose that we are given weights $w(v) \geq 0$ for each vertex of G. There is a simple argument that $G = (V, E)$ has an independent set of weight at least $\frac{w(V)}{\Delta+1}$ and that G has a transversal (not necessarily independent) of weight at least $\frac{w(V)}{b}$. Likewise, G has independent sets or transversals with weight at most $w(V)/(\Delta+1)$ or $w(V)/b$, respectively. Note also that we cannot always expect independent transversals of weight more (or less) than $w(V)/b$: e.g., consider the case of all weights being equal. Our theorems 4.3 and 4.4 improve quite a bit upon the (nonconstructive) bounds of [10]; among other things, we show next that weight at least $\frac{w(V)}{b}$ is in fact achievable if $b \geq 4.5\Delta$, a result that was shown to be true asymptotically for large b in [10].

Theorem 4.3. *Suppose $4\Delta \leq b \leq 4.5\Delta$. Then there is an independent transversal $I \subseteq V$ with weight*

$$w(I) \geq w(V)\Big(\frac{\sqrt{b}+\sqrt{b-4\Delta}}{\sqrt{b}(2b-1)+\sqrt{b-4\Delta}}\Big) \geq \frac{w(V)}{8\Delta-1}.$$

Suppose $b \geq 4.5\Delta$. Then there is an independent transversal $I \subseteq V$ with weight

$$w(I) \geq w(V) \cdot \min(1/b, \frac{4}{27\Delta-2}).$$

Furthermore, independent transversals with weight at least $(1-n^{-\Theta(1)})$ times these lower-bounds, can be found in polynomial time with high probability.

Proof. The first result follows in a straightforward manner from Theorem 2.12 when we set each entry of $\vec{\lambda}$ to the scalar constant $\alpha = \frac{b-\sqrt{b}\sqrt{b-4\Delta}}{2b\Delta}$; we use a single bad-event B_1 with the trivial hitting-set, whose atomic events correspond to every edge separately. Then the $\bowtie_1$-relation is null and we have $\hat{S}_1 = 1$, and the condition of Theorem 2.11 requires

$$b\alpha - b\Delta\alpha^2 \geq 1$$

which is easily checked. Now for each vertex $v \in V_i$ we have $H_{i,v} \leq \alpha^2\Delta$; so the probability of selecting this vertex in the LLL distribution is

$$\mathbf{P}(\text{select } v) \geq \frac{\vec{\lambda}_{i,j} - H_{i,v}}{\vec{\lambda}_i - H_{i,v}} \geq \frac{\alpha - \alpha^2\Delta}{b\alpha - \alpha^2\Delta} = \frac{\sqrt{b}+\sqrt{b-4\Delta}}{\sqrt{b}(2b-1)+\sqrt{b-4\Delta}}$$

To obtain the second result, in each block V_i, we discard all but the $b' = \lfloor 9/2\Delta \rfloor$ highest-weight vertices. To simplify the proof, consider only the case when Δ is even (the odd Δ is similar).

In this case, we assign $\vec{\lambda} = \alpha = \frac{1}{3\Delta}$ for each of the b' highest-weight vertices, and $\vec{\lambda} = 0$ for the remaining. Let us fix a block V_i, consisting of vertices $v_1, \ldots, v_b$ sorted by weight so that $w(v_1) \geq w(v_2) \geq \cdots \geq w(v_b)$. By Theorem 2.12, each of the high-weight vertices is selected with probability $\geq \frac{\alpha-\Delta\alpha^2}{b'\alpha-\Delta\alpha^2} = \frac{4}{27\Delta-2}$. Hence the expected weight of the independent transversal selected is at least $(w(v_1)+\cdots+w(v_{b'}))\frac{4}{27\Delta-2}$. By concavity, subject to a fixed value of $w(V_i)$, the choices of weights $w(v_1), \ldots, w(v_b)$ which minimizes this assigns constant weight x to all vertices, except for an additional vertex which receives an additional weight of $w(V_i) - xb$. The expected weight of this block then becomes

$$\mathbf{E}[w(V_i \cap I)] = x + (w(V_i) - bx)\frac{4}{27\Delta-2}$$

This achieves its minimum at either $x = 0$ or $x = 1/b$, yielding

$$\mathbf{E}[w(I)] \geq w(V) \cdot \min(\frac{1}{b}, \frac{4}{27\Delta - 2}).$$

Finally, the high-probability bound follows by standard repetition of this basic randomized algorithm. □

We show a matching upper bound on weights:

Theorem 4.4. *Suppose $4\Delta \leq b \leq 8\Delta$. Then there is an independent transversal $I \subseteq V$ with weight*

$$w(I) \leq w(V)\frac{2}{4\sqrt{\Delta}\sqrt{b - 4\Delta} + b}$$

Suppose $b \geq 8\Delta$. Then there is an independent transversal $I \subseteq V$ with weight

$$w(I) \leq \frac{w(V)}{b}$$

Furthermore, independent transversals with weight at most $(1+n^{-\Theta(1)})$ times these upper-bounds, can be found in polynomial time with high probability.

Proof. Suppose we discard all but the lowest-ranking b' vertices in each block, where $b' = 4\Delta$. For these vertices v, we set $\vec{\lambda}_v = \alpha = \frac{1}{2\Delta}$, and we set $\vec{\lambda}_v = 0$ for the remaining vertices. We have a single bad-event with trivial hitting-set and atomic bad-events for each edge. Then we have $\hat{S}_1 = 1$, and the condition of Theorem 2.11 is satisfied.

Fix a block V_i, in which the vertices are sorted in increasing order of their weight $w(v_1) \leq w(v_2) \leq \cdots \leq w(v_b)$. Then we can write the expected weight of the resulting block as

$$\begin{aligned}\mathbf{E}[w(V_i \cap I)] &= w(v_1) + (w(v_2) - w(v_1))(1 - \mathbf{P}(v_1 \text{ selected})) + (w(v_3) - w(v_2))(1 - \mathbf{P}(v_1 \text{ or } v_2 \text{ selected})) \\ &\quad + \cdots + (w(v_{b'}) - w(v_{b'-1})\mathbf{P}(v_{b'} \text{ selected}) \\ &\leq w(v_1) + (w(v_2) - w(v_1))(1 - \frac{\alpha - \Delta\alpha^2}{b'\alpha - \Delta\alpha^2}) + (w(v_3) - w(v_2))(1 - \frac{2(\alpha - \Delta\alpha^2)}{b'\alpha - 2\Delta\alpha^2}) + \dots\end{aligned}$$

Subject to the constraints that $w(v_1) \leq w(v_2) \leq \dots$ and $w(v_1) + \cdots + w(v_b) = w(V_i)$, the choice of $w(v_1), \dots, w(v_b)$ which maximizes this is the following: for some $2 \leq k \leq b'$, all the vertices $v_k, \dots, v_b$ have weight x, while vertex v_{k-1} has weight $y \leq x$, and $(b - k + 1)x + y = w(V_i)$. In this case, we have

$$\begin{aligned}\mathbf{E}[w(V_i \cap I)] &\leq \max_{x \geq y \in \mathbf{R}, k \in \{2,\dots,b'\}} y\mathbf{P}(v_{k-1}, \dots, v_{b'} \text{ selected}) + (x - y)\mathbf{P}(v_k, \dots v_{b'} \text{ selected}) \\ &\leq \max_{x \geq y \in \mathbf{R}, k \in \{2,\dots,b'\}} y\frac{(b' - k + 2)\alpha}{b'\alpha - (k-2)\alpha\Delta^2} + (x - y)\frac{(b' - k + 1)\alpha}{b'\alpha - (k-1)\alpha\Delta^2}\end{aligned}$$

We now relax the restriction that k is an integer in the range $\{2, \dots, b'\}$ to allow k to be a real number in the interval $[1, b']$. When k is relaxed in this way, the maximum of the above expression occurs at $y = 0$ and $x = \frac{w(V_i)}{b-k+1}$; we thus have

$$\mathbf{E}[w(V_i \cap I)] \leq \max_{k \in [1,b']} \frac{w(V_i)}{b - k + 1}\frac{(b' - k + 1)\alpha}{b'\alpha - (k-1)\alpha\Delta^2}$$

When $b \geq 8\Delta$, this is decreasing function of k on the interval $[1, b']$, hence achieves its maximum value at $k = 1$, yielding $\mathbf{E}[w(V_i \cap I)] \leq \frac{1}{b}$. When $4\Delta < b \leq 8\Delta$, this achieves its maximum at the critical point $k = b' + 1 - \frac{\left(\sqrt{b'-4\Delta}+\sqrt{b'}\right)\sqrt{b'(b-b')}}{2\sqrt{\Delta}}$; this yields $\mathbf{E}[w(V_i \cap I)] \leq \frac{2}{4\sqrt{\Delta}\sqrt{b-4\Delta}+b}$. Finally, at $b = 4\Delta$, then we again restrict k to range over the integers; in this case it achieves a maximum value at $k = b$ yielding $\mathbf{E}[w(V_i \cap I)] \leq \frac{2}{1+4\Delta}$.

Putting all these cases together gives us the claimed result. □

We can give similar bounds for independent transversals omitting other subgraphs. Note that such a bound cannot be specified in terms of the average degree, because we might add vertices of small degree and weight.

5. Sums of random variables, and column-sparse packing

Different types of bad events call for different hitting-sets, and the best choice may depend on "global" information about the variables it contains, in addition to local parameters. However, there is a natural and powerful option for upper-tail bad events B_k of the form $\sum_\ell Z_\ell \geq k$, which is what we discuss next. As above, we work in our usual setup of elements, categories, and increasing bad events B_k. In the discussion below, elements will often be referred to as x, x_r etc.; note that an element is always some pair of the form $\langle i, j\rangle$.

Theorem 5.1. *Suppose we are given an assignment of non-negative real numbers $\vec{\lambda}_{i,j}$ to each element $\langle i, j\rangle$. Let $x_1, \ldots, x_v$ be a set of elements. Define $\mu = \sum_t \vec{\lambda}_{x_t}$, and for each category i let*

$$\mu_i = \sum_{x_t \text{ is in category } i} \vec{\lambda}_{x_t}$$

Suppose that in our usual setting of categories and bad events, there is a bad event B_k that $Z_{x_1} + \cdots + Z_{x_v} \geq \mu(1+\delta)$, where $\delta > 0$ and $\mu(1+\delta)$ is an integer. Let $d \leq \mu(1+\delta)$ be a positive integer. Then, recalling Definition 2.4, there is a fractional hitting-set B'_k with the property

$$G^k \leq \frac{\mu^d}{d!\binom{(1+\delta)\mu}{d}};\ G^k_i \leq (\mu_i/\mu) \cdot d \cdot (1 - (\mu_i/\mu))^{d-1} \cdot \frac{\mu^d}{d!\binom{(1+\delta)\mu}{d}}.$$

Also, we refer to the parameter d as the width *of this hitting-set.*

Proof. Assign the following fractional hitting-set: for each subset $Y = \{x_{r_1}, \ldots, x_{r_d}\}$ of cardinality d in which all the elements $x_{r_1}, \ldots, x_{r_d}$ come from distinct categories, assign weight $B'_k(Y) = \frac{1}{\binom{(1+\delta)\mu}{d}}$. Note that exactly one element gets assigned from each category; furthermore, an atomic bad event is one in which $\mu(1+\delta)$ elements, all from different categories, get assigned. These two facts easily help show that B'_k is a valid fractional hitting-set.

We now have

$$\begin{aligned} G^k &= \frac{\sum_{\substack{x_{r_1} < \cdots < x_{r_d} \\ \text{from distinct categories}}} \vec{\lambda}_{r_1} \ldots \vec{\lambda}_{r_d}}{\binom{(1+\delta)\mu}{d}} \\ &= \frac{\sum_{0 \leq i_1 < \cdots < i_d < n} \mu_{i_1} \cdots \mu_{i_d}}{\binom{(1+\delta)\mu}{d}} \\ &\leq \frac{\binom{n}{d}(\mu/n)^d}{\binom{(1+\delta)\mu}{d}} \leq \frac{\mu^d}{d!\binom{(1+\delta)\mu}{d}}. \end{aligned}$$

For a category i, we have

$$\begin{aligned} G_i^k &\leq \frac{\sum_{i_1,\ldots,i_{d-1}\neq i} \mu_{i_1}\ldots\mu_{i_{d-1}}\mu_i}{\binom{(1+\delta)\mu}{d}} \\ &\leq \frac{\mu_i\binom{n-1}{d-1}(\frac{\mu-\mu_i}{n-1})^{d-1}}{\binom{(1+\delta)\mu}{d}} \\ &\leq \frac{\mu_i(\mu-\mu_i)^{d-1}}{(d-1)!\binom{(1+\delta)\mu}{d}} \\ &\leq (\mu_i/\mu)d(1-(\mu_i/\mu))^{d-1}\frac{\mu^d}{d!\binom{(1+\delta)\mu}{d}}. \end{aligned}$$

□

Note that by setting $d = \lceil \mu\delta \rceil$, one can achieve the Chernoff bounds [30]:

$$\frac{\mu^d}{d!\binom{(1+\delta)\mu}{d}} \leq \Big(\frac{e^\delta}{(1+\delta)^{1+\delta}}\Big)^\mu.$$

5.1. **LP rounding for column-sparse packing problems.** In light of Theorem 5.1, consider the family of CSPs where we have a series of linear packing constraints of the form "$\sum_x a_{k,x}y_x \leq b_k$", with non-negative coefficients a, b. (Here and in what follows, x will often refer to some element (i,j).) In addition, there is the usual assignment constraint, namely a series of disjoint blocks $X_1,\ldots,X_n$ with the constraint that $\sum_j y_{i,j} = 1$. When does such an integer linear program have a feasible solution?

Suppose we wish to solve this via LP relaxation. One technique is to solve the simpler linear program where the integrality constraints on $y \in \{0,1\}$ are relaxed to $y' \in [0,1]$, and in addition the packing constraints are tightened to $\sum_x a_{k,x}y_x \leq b'_k$ for some $b'_k \leq b_k$. We assume that each $a_{k,x} \in [0,1]$ and that for each x we have $\sum_k a_{k,x} \leq D$.

We note that there are constructions using the standard LLL and standard Moser-Tardos algorithm that can yield results qualitatively similar to the ones in this section. Two examples are [12] and [19]. The analysis of [12] appears to only work in the regime $b'_k = 1$; the analysis of [19] is phrased in terms of the total *number* of constraints each variable participates in (as opposed to the $\updownarrow_1$ sum of the corresponding coefficients). Tthe case in which b'_k have differing magnitudes appears to be beyond either of these analyses.

In addition, these papers are quite difficult technically; in [12], there is a quantization argument, in which one must handle separately coefficients of different magnitudes. In [19], there is an iterated application of the LLL, in which one must track quite carefully the sizes of the relevant papers as they are reduced from the original system.

The PRA provides however, a simple and comprehensive framework to obtain an integral solution. We have just a single application of the PRA which directly produces our desired solution, and we handle fractional coefficients almost automatically.

Our condition on the separation between b_k and b'_k is based on the Chernoff bound. To state this theorem in the simplest and broadest form, recall the Chernoff separation function from Definition 1.1.

Theorem 5.2. *There is some universal constant $C > 0$ with the following property. Suppose that we have an LP parametrized by $a_{k,x}, b'_k$, where $D = \max_x \sum_{k,x} a_{k,x} \geq 1$.*

Now let $\epsilon > 0$, b_k be given such that:

(C1): *For all k we have $\frac{b_k}{b'_k}$ Chernoff$(b'_k(1+\epsilon), b_k) \leq \frac{C\epsilon}{D}$*

(C2): *For all k we have $b_k \geq 1$*

(C3): *For all k we have $b_k \geq b'_k(1+\epsilon)$*

Then if the linear program

$$\sum_j y_{i,j} \geq 1, \qquad \sum_x a_{k,x} y_x \leq b'_k, \qquad y_x \in [0,1]$$

is satisfiable, then so is the ***integer*** *program*

$$\sum_j y_{i,j} \geq 1, \qquad \sum_x a_{k,x} y_x \leq b_k, \qquad y_x \in \{0,1\}.$$

Furthermore, suppose we have a separation oracle *for the LP and the IP. (That is, given a variable assignment, we can either find a violated linear constraint, or determine that all constraints are satisfied). Then such a satisfying assignment can be found in polynomial time.*

Proof. We will prove this Theorem under the assumption that $\epsilon \in (0,1]$. As will be shown in Proposition 5.3, we can then adjust the constant C so that the theorem holds for any $\epsilon \in (0,\infty)$.

Using the separation oracle, one can solve the LP in polynomial time. Suppose we have a feasible solution y to the relaxed linear program. Then we set $\vec{\lambda} = (1+\epsilon)y$. We associate a bad event to each packing constraint. Let us analyze a constraint k. Define $\mu = b'_k(1+\epsilon)$, define $t = b_k$, and define $\delta = t/\mu - 1$. This corresponds to a bad-event $\sum_x a_{k,x} y_x > b_k$. We use the fractional hitting set which assigns, to each set of d elements $x_1, \ldots, x_d$ from distinct categories, the weight

$$B'_k(\{x_1, \ldots, x_l\}) = \frac{a_{k,x_1} \ldots a_{k,x_l}}{\binom{b_k}{l}}$$

where $d = \lceil \mu\delta \rceil$.

We first claim that this is a valid hitting set. For, suppose we have a set of d' elements such that $a_{k,x_1} + \cdots + a_{k,x_{d'}} > b_k$. As $a \in [0,1]$ we must have $d' > b_k \geq d$. Furthermore, summing over all d-subsets of $\{x_1, \ldots, x_{d'}\}$ we have

$$\sum_{s_1 < s_2 < \cdots < s_l} B'_k(\{x_{s_1}, \ldots, x_{s_d}\}) = \sum_{s_1 < s_2 < \cdots < s_d} \frac{a_{k,x_{s_1}} \ldots a_{k,x_{s_d}}}{\binom{b_k}{d}}$$

This can be regarded as a polynomial in the weights a. Subject to the constraint $a_{k,x_1} + \cdots + a_{k,x_d} \geq b_k$ and $a \in [0,1]$, the numerator achieves its minimium value when there are are $\lfloor b_k \rfloor$ elements of weight $a = 1$ and one further element of weight $a = b_k - \lfloor b_k \rfloor$. In particular, the numerator is at least $\binom{b_k}{d}$, and this sum is at least 1 as desired.

We will next bound the contribution of each bad event. For a constraint k, we have

$$\begin{aligned} G^k &= \sum_{x_1 < \cdots < x_d} \vec{\lambda}_{x_1} \ldots \vec{\lambda}_{x_d} B'_k\{x_1, \ldots, x_d\} \\ &= \frac{\sum_{x_1 < \cdots < x_d} \vec{\lambda}_{x_1} a_{k,x_1} \ldots \vec{\lambda}_{x_d} a_{k,x_l}}{\binom{b_k}{d}} \end{aligned}$$

Now, note that $\sum_x \vec{\lambda}_x a_{k,x} = \sum_t a_{k,x} y_{k,x}(1+\epsilon) \leq b'_k(1+\epsilon)$. By concavity, the numerator is maximized when all $|X|$ terms $\vec{\lambda}_t a_{k,t}$ are equal to $b'_k(1+\epsilon)/|X|$. (Recall that X is the set of all possible ordered pairs $\langle i,j \rangle$.) Using a similar argument to Theorem 5.1, we have

$$G^k \leq \frac{\binom{|X|}{d}\left(\frac{b'_k(1+\epsilon)}{|X|}\right)^d}{\binom{b_k}{l}} \leq \frac{\mu^d}{d!\binom{\mu(1+\delta)}{d}}$$

For $d = \lceil \mu\delta \rceil$, this expression is bounded by the Chernoff bound

$$\begin{aligned} G^k &\le \text{Chernoff}(b'_k(1+\epsilon), b_k) \\ &\le \frac{b_k}{b'_k(1+\epsilon)} \text{Chernoff}(b'_k(1+\epsilon), b_k) \qquad \text{by (C3)} \\ &\le \frac{C\epsilon}{D(1+\epsilon)} \qquad \text{by (C1)} \\ &\le C; \end{aligned}$$

for C sufficiently small this yields $G^k \le 1 - \Omega(1)$.

Similarly, along the lines of Theorem 5.1, for a category i, set $\mu_i = \sum_j a_{k,i,j} \vec{\lambda}_{i,j}$ so that we have

$$\begin{aligned} G_i^k &\le \frac{\mu_i d}{\mu} \frac{\mu^d}{d! \binom{(1+\delta)\mu}{d}} \\ &\le \mu_i(\delta + 1/\mu) \text{Chernoff}(\mu, t) \\ &\le \mu_i(\frac{b_k + 1}{b'_k(1+\epsilon)} - 1) \text{Chernoff}(\mu, t) \\ &= O(C\mu_i \frac{\epsilon}{D(1+\epsilon)}) \qquad \text{by (C1), (C2)} \end{aligned}$$

Now, by Theorem 2.5(b), we sum over all j obtaining

$$\begin{aligned} \sum_j \vec{\lambda}_{i,j} - \sum_k \frac{G_i^k}{1 - G^k} &\ge (1+\epsilon) - \sum_k \frac{\mu_{k,i} O(C\epsilon/(D(1+\epsilon))}{\Omega(1)} \\ &= (1+\epsilon) - \sum_j \sum_k O(a_{k,i,j} \vec{\lambda}_{i,j} C \frac{\epsilon}{D(1+\epsilon)}) \\ &\ge (1+\epsilon) - C\frac{\epsilon}{1+\epsilon} O(\sum_j \vec{\lambda}_{i,j}) \\ &\ge 1 + \epsilon - O(C\epsilon); \end{aligned}$$

for C sufficiently small, this is ≥ 1 as desired.

Furthermore, we have $\sum_j \vec{\lambda}_{i,j} \le 1+\epsilon \le 2$, so the PRA terminates to such a configuration after an expected constant number of iterations. Although the number of constraints may be exponential, it is not hard to see that one can efficiently implement a single step of the PRA using the separation oracle. So this gives a polynomial-time algorithm.

□

To complete this proof, we show that, by adjusting C slightly, one can extend the above proof to cover the case when $\epsilon \to \infty$:

Proposition 5.3. *Let C be an arbitrary constant. There is a constant $C' < C$ with the following property. Suppose that, for some $\epsilon' \in (0, \infty)$, we satisfy*

$$\forall k \qquad \frac{b_k}{b'_k} \mathit{Chernoff}(b'_k(1+\epsilon'), b_k) \le \frac{C'\epsilon}{D}$$

Then, with $\epsilon = \min(1, \epsilon')$ we satisfy

$$\forall k \qquad \frac{b_k}{b'_k} \mathit{Chernoff}(b'_k(1+\epsilon), b_k) \le \frac{C\epsilon}{D}$$

Now, if we replace the constant C determined in the proof of Theorem 5.2 (which was only valid in the range $\epsilon \in (0,1]$), with the constant C', then it should be clear that Theorem 5.2 holds for any $\epsilon \in (0,\infty)$.

Proof. This theorem obviously holds when $\epsilon' < 1$, so it will suffice to show that there is some constant $\phi > 0$ such that, for all real numbers x, y with $y \geq 1, y \geq 2x$ we have

$$\min_{\epsilon \in [1, y/x-1]} \frac{\text{Chernoff}(x(1+\epsilon), y)}{\epsilon} \geq \phi \text{Chernoff}(2x, y)$$

(Here x plays the role of b'_k and y plays the role of b_k; note we must have $b_k \geq (1+\epsilon)b'_k \geq 2b_k$, which explains the constraints $y \geq 2x$ and $\epsilon \leq y/x - 1$).

Let $\epsilon^* = \text{argmin}_{\epsilon \in [1, y/x-1]} \frac{\text{Chernoff}(x(1+\epsilon), y)}{\epsilon}$. Using simple calculus, we see that $\epsilon^* = 1, y/x - 1$ or the critical point $\frac{-\sqrt{(x-y+1)^2 - 4x} - x + y - 1}{2x}$; the latter is only possible when it is a real number, i.e. $y \geq 1 + 2\sqrt{x} + x$.

Let us first deal with the case $\epsilon^* = y/x - 1$. For this case, we have

$$\frac{\text{Chernoff}(x(1+\epsilon^*), y)}{\epsilon^* \text{Chernoff}(2x, y)} = \frac{1}{\text{Chernoff}(2x, y)(y/x - 1)} \geq \frac{1}{\text{Chernoff}(2x/y, 1)(y/x - 1)};$$

the latter is a decreasing function of y/x, which approaches to a limit of $1/(2e)$ as $y/x \to \infty$. Hence for $\epsilon^* = y/x - 1$ we have

$$\frac{\text{Chernoff}(x(1+\epsilon^*), y)}{\epsilon^*} \geq \frac{\text{Chernoff}(2x, y)}{2e}.$$

We next consider $\epsilon^* = \frac{-\sqrt{(x-y+1)^2 - 4x} - x + y - 1}{2x}$. In order to have the latter expression be a real number in the range $[1, \infty)$, we must have

$$0 < x \leq 1 \qquad 1 + 2\sqrt{x} + x \leq y \leq 2 + 2x$$

Through simple calculus, one can verify that the minimum value of $\frac{\text{Chernoff}(x(1+\epsilon^*), y)}{\epsilon^*}$ in this domain is obtained at $x \to 0, y \to 1$ yielding

$$\frac{\text{Chernoff}(x(1+\epsilon^*), y)}{\epsilon^*} \geq \frac{\text{Chernoff}(2x, y)}{2}.$$

This completes the proof. □

In a typical application of this theorem, one is given some fixed LP which specifies a, b'_k, D. One wants to choose b_k, ϵ to satisfy Theorem 5.2; typically, it is important to make the b_k as small as possible so as to get a good approximation ratio to the LP, while ϵ is not important except inasmuch as it satisfies the Theorem.

Comparison with the standard LLL: We note that it is quite difficult for the standard LLL to approach this result, at least in its full generality. The reason is that variable $y_{i,j}$ affects constraint k if $a_{k,i,j} > 0$, and it is possible that every variable affects every constraint. To get around this in [12], there is a complex quantization scheme in which coefficients which are small are ignored at certain stages of the construction. By contrast, the PRA is able to handle all sizes of coefficients in a unified and simple way.

Example parameters: Theorem 5.2 is given a very generic setting, which is intended to handle a wide range of sizes for the parameters b_k, b'_k, D. When we can restrict these parameters, we can obtain much more explicit and straightforward bounds. We illustrate some typical results.

Proposition 5.4. *Suppose we are given an LP satisfying the requirement of Theorem 5.2; let $c > 0$ be any desired constant. Then we can set b_k as follows so as to satisfy Theorem 5.2:*

(1) *For each k with $b'_k = O(1)$, we set $b_k = O(\frac{\log D}{\log \log D})$;*

(2) *For each k with $b'_k \geq \Omega(\log D)$, there is some constant $c' > 0$ such that we may set $b_k = b'_k(1 + D^{-c}) + c'\sqrt{b'_k \log D}$.*

Values of b'_k which are not covered by these cases, will have a similar but more complicated value of b_k.

Proof. To simplify the notation, we suppose $D \geq D_0$ for some sufficiently large constant D_0. We set $\epsilon = D^{-c}$.

We first prove case (1). In this case, let $b_k = b'_k(1 + \epsilon) + \phi(\frac{\log(D}{\log\log D})$ for some constant ϕ to be specified. In this case, the critical constraint is (C1); we bound it as

$$\begin{aligned}
\frac{b_k}{b'_k}\text{Chernoff}(b'_k(1+\epsilon), b_k) &= O\Big(\frac{\phi \log D}{\log\log D}\Big)\Big(\frac{e^\delta}{(1+\delta)^{1+\delta}}\Big)^{-O(1)} \\
&\qquad \text{where } \delta = \Omega(\phi\frac{\log D}{\log\log D}) \\
&= O\Big(\frac{\phi \log D}{\log\log D}\Big)\Big(D^{-\Omega(\phi)}\Big)^{-O(1)} \\
&\leq \phi D^{-x\phi} \qquad \text{where } x \text{ is a constant} \\
&\leq \frac{C\epsilon}{(1+\epsilon)D} \qquad \text{for } \phi \text{ sufficiently large}
\end{aligned}$$

Case (2) is analogous. □

The multi-dimensional scheduling application from the introduction follows as an easy application of Proposition 5.4. First, given $(T_1, T_2, \ldots, T_D)$, we can, motivated by (3), set $x_{i,j} := 0$ if there exists some ℓ for which $p_{i,j,\ell} > T_\ell$. After this filtering, we solve the LP relaxation. If it gives a feasible solution, we scale the LP so that all r.h.s. values b'_k equal 1; our filtering ensures that the coefficient matrix has entries in $[0, 1]$ now, as required. By Proposition 5.4, we can now set $b_k = O(\log D/\log\log D)$. This result is not new to this paper, but it is obtained in a particularly straightforward way.

The multiplicative factor $(1 + D^{-c})$ in Proposition 5.4 is required when the right-hand sides b'_k have different magnitudes. When they all have the same magnitude, a simpler bound is possible:

Proposition 5.5. *Suppose we are given an LP satisfying the requirement of Theorem 5.2; suppose that there is some $T \geq \Omega(\log D)$ such that, for all k, we have $b'_k \leq T$. Then setting $b_k = T + O(\sqrt{T \log T})$ suffices to satisfy Theorem 5.2.*

Proof. Set $\epsilon = T^{-1/2}$; the remainder of the proof is similar to Proposition 5.4. □

6. Packet routing

6.1. **Review of background and known approaches.** We begin by reviewing the basic strategy of [29], and its improvement by [26]. [29] is a very readable overview of our basic strategy, and we will not include all the details which are covered there. Our choice of parameters will be slightly improved from [29] and [26]. We note that [26] studied a more general version of the packet-routing problem, so their choice of parameters was not (and could not be) optimized.

We are given a graph G with N packets. Each packet has a simple path, of length at most D, to reach its endpoint vertex (we refer to D as the *dilation*). In any timestep, a packet may wait at its current position, or move along the next edge on its path. Our goal is to find a schedule of smallest makespan in which, in any given timestep, an edge carries at most a single packet.

We define the *congestion* C to be the maximum, over all edges, of the number of packets scheduled to traverse that edge. It is clear that D and C are both lower bounds for the makespan, and [20] has shown that in fact a schedule of makespan $O(C + D)$ is possible. [29] provided an explicit

constant bound of $39(C+D)$, as well as describing an algorithm to find such a schedule. This was improved to $23.4(C+D)$ in [26] as will be described below.

While the final schedule only allows one packet to cross an edge at a time, we will relax this constraint during our construction. We consider "infeasible" schedules, in which arbitrarily many packets pass through each edge at each timestep. We define an *interval* to be a consecutive set of times in our schedule, and the *congestion* of an edge in a given interval to be the number of packets crossing that edge. If we are referring to intervals of length i, then we define a *frame* to be an interval which starts at an integer multiple of i.

From our original graph, one can easily form an (infeasible) schedule with delay D and overall congestion C. Initially, this congestion may "bunch up" in time, that is, certain edges may have very high congestion in some timesteps and very low congestion in others. So the congestion is not bounded on any smaller interval than the trivial interval of length D. During our construction, we will "even out" the schedule, bounding the congestion on successively smaller intervals.

Ideally, one would eventually finish by showing that on each each individual timestep (i.e. interval of length 1), the congestion is roughly C/D. In this case, one could turn such an infeasible schedule into a feasible schedule, by simply expanding each timestep into C/D separate timesteps.

As [26] showed, it suffices to control the congestion on intervals of length 2. Given our infeasible schedule, we can view each interval of length 2 as defining a new subproblem. In this subproblem, our packets start at a given vertex and have paths of length 2. The congestion of this subproblem is exactly the congestion of the schedule. Hence, if we can schedule problems of length 2, then we can also schedule the 2-intervals of our expanded schedule.

We quote the following result from [26].

Proposition 6.1 ([26])**.** *Suppose there is a instance G with delay $D = 2$ and congestion C. Then there is a schedule of makespan $C + 1$. Furthermore, such a schedule can be formed in polynomial time.*

[26] used this to improve the bound on the makespan to $23.4(C + D)$. [26] speculated that by examining the scheduling for longer, but still small, delays, one could further improve the general packet routing. Unfortunately, we are not able to show a general result for small delays such as $D = 3$. However, as we will see, the schedules that are produced in the larger construction of [29] are far from generic, but instead have relatively balanced congestion across time. We will see how to take advantage of this balanced structure to improve the scheduling.

We begin by first reviewing the general construction of [29].

6.2. **Using the LLL to find a schedule.** The general strategy for this construction is to add random delays to each packet, and then allowing the packet to move through each of its edges in turn without hesitation. This effectively homogenizes the congestion across time. We have the following lemma:

Lemma 6.2. *Let $i' < i$, let m, C' be non-negative integers. Suppose there is a schedule S of length L such that every interval of length i has congestion at most C. Suppose that we have*

$$e \times \mathbf{P}(\mathit{Binomial}(C, \frac{i'}{i-i'}) > C') \times (Cmi^2 + 1) < 1$$

Then there is a schedule S' of length $L' = L(1 + 1/m) + i$, in which every interval of length i' has congestion $\leq C'$. Furthermore, this schedule can be constructed in expected polynomial time.

Proof. We break the schedule S into frames of length $F = mi$, and refine each separately. Within each frame, we add a random delay of length $i - i'$ to each packet separately.

Let us fix an F-frame for the moment. Associate a bad event to each edge f and i'-interval I, that the congestion in that interval and edge exceeds C'.

For each I, e, there are at most C possible packets that could cross, and each does so with probability $p = \frac{i'}{i-i'}$. Hence the probability of the bad event is at most the probability that a Binomial random variable with C trials and probability p exceeds C'.

Next consider the dependency. Given an edge f and i'-interval I, there are at most C packets crossing it, each of which may pass through up to mi other edges in the frame. We refer to the combination of a specific packet passing through a specific edge as a *transit*. Now, for each transit, there are (depending on the delay assigned to that packet) at most i other i'-intervals in which this transit could have been scheduled. Hence the dependency is at most Cmi^2.

By the LLL, the condition in the hypothesis guarantees that there is a positive probability that the delays avoid all bad events. In this case, we refine each frame of S to obtain a new schedule S' as desired. We can use the algorithmic LLL to actually find such schedules S' in polynomial time.

So far, this ensures that *within each frame*, the congestion within any interval of length i' is at most C'. In the refined schedule S' there may be intervals that cross frames. To ensure that these do not pose any problems, we insert a delay of length i' between successive frames, during which no packets move at all. This step means that the schedule S' may have length up to $L(1+1/m)+i$. □

Using this Lemma 6.2, we can transform the original problem instance (in which C, D may be unbounded), into one in which C, D are small finite values. In order to carry out this analysis properly, one would need to develop a series of separate bounds depending on the sizes of C, D. To simplify the exposition, we will assume that C, D are very large, in which case certain rounding effects can be disregarded. When C, D are smaller, we can show stronger bounds but doing this completely requires extensive case analysis of the parameters.

Lemma 6.3. *Assume $C + D \geq 2^{896}$. There is a schedule of length at most $1.004(C + D)$ and in which the congestion on any interval of length 2^{24} is at most 17040600. Furthermore, this schedule can be produced in polynomial time.*

Proof. Define the sequence a_k recursively as follows.

$$a_0 = 256 \qquad a_{k+1} = 2^{a_k}$$

There is a unique k such that $a_k^{3.5} \leq (C+D) < a_{k+1}^{3.5}$. By a slight variant on Lemma 6.2, one can add delays to obtain a schedule of length $C + D$, in which the congestion on any interval of length $i' = a_k^3$ is at most $C' = i'(1 + 4/a_k)$.

At this point, we use Lemma 6.2 repeatedly to ensure to control the congestion intervals of length a_j^3, for $j = k - 1, \ldots 0$. At each step, this increases the length of the resulting schedule from L_j to $L_j(1 + 1/a_{j+1}) + a_j$, and increases the congestion on the relevant interval from $i(1 + 4/a_k)$ to

$$i(1 + 4/a_k) \prod_{j=0}^{k-1} (1 + 4/a_j)(\frac{1}{1 - (a_j/a_{j+1})^3})$$

(We use the Chernoff bound to estimate the binomial tail in Lemma 6.2.)

For $C + D \geq a_k^{3.5}$, it is a simple calculation to see that the increase in length is from $C + D$ (after the original refinement) to at most $1.004(C + D)$. In the final step of this analysis, we are bounding the congestion of intervals of length $a_0^3 = 2^{24}$, and the congestion on such an interval is at most 17040600.

Furthermore, since all of these steps use the LLL, one can form all such schedules in polynomial time.

See [29] for a much more thorough explanation of this process. □

Now that we have reduced to constant-sized intervals, we are no longer interested in asymptotic arguments, and come down to specific numbers.

Lemma 6.4. *There is a feasible schedule of length at most* $10.92(C+D)$*, which can be constructed in polynomial time.*

Proof. For simplicity, we assume $C + D \geq 2^{896}$.

By Lemma 6.3, we form a schedule S_1, of length $L_1 \leq 1.004(C + D)$, in which each interval of length 2^{24} has congestion at most 17040600.

Now apply Lemma 6.2 to S_1, with $m = 64, i' = 1024, C' = 1385$ to obtain a schedule S_2, of length $L_2 \leq 1.0157L_1 + 2^{24}$, in which each interval of length 1024 has congestion at most 1385.

Now apply Lemma 6.2 to S_2 with $m = 64, i' = 2, C' = 20$, to obtain a schedule S_3 of length $L_3 \leq 1.0157L_2 + 1024$, in which each frame of length 2 has congestion at most 20.

Now apply Proposition 6.1 to S_3, expanding each 2-frame to a *feasible* schedule of length 21. The total length of the resulting schedule is at most $\frac{21}{2}L_3 \leq 10.92(C + D)$. □

6.3. **Better scheduling of the final 2-frame.** Let us examine the last stage in the construction more closely. In this phase, we are dividing the schedule into intervals of length 2, and we want to control the congestion of each edge in each 2-frame.

For a given edge f and time t, we let $c_t(f)$ denote the number of packets scheduled to cross that edge in the four time steps of the original (infeasible) schedule.

Suppose we have two consecutive 2-frames starting at time t. The reason for the high value of C' in the final step of the above construction is that it is quite likely that $c_t + c_{t+1}$ or $c_{t+2} + c_{t+3}$ are much larger than their mean. However, it would be quite rare for *both* these bad events to happen simultaneously. We will construct a schedule in which we insert an "overflow" time between the 2-frames. This overflow handles cases in which either $c_t + c_{t+1}$ is too large *or* $c_{t+2} + c_{t+3}$ is too large.

Our goal will be to modify either of the 2-frames so as to ensure that the congestion is at most T. In order to describe our modification strategy, we first fix, for every packet and frame, a "first edge" and "second edge" in this frame. Some packets may only transit a single edge, which we will arbitrarily label as first or second. As we modify the schedule, some packets that initially had two transits scheduled will be left with only one; in this case, we retain the label for that edge. So, we may assume that every edge is marked as first or second and this label does not change.

We do this by shifting transits into the overflow time. For each 2-frame, there are two overflow times, respectively earlier and later. If we want to shift an edge to the later overflow time, we choose any packet that uses that edge as a second edge (if any), and reschedule the second transit to the later overflow time; similarly if we shift an edge to the earlier overflow time. See Figure 1.

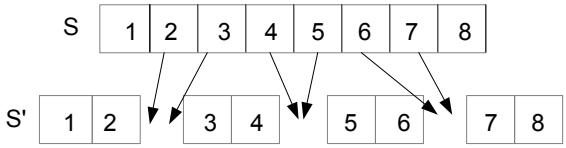

FIGURE 1. The packets in the original schedule S are shifted into overflow times in the schedule S'.

Note that in the analysis of [26], the only thing that matters is the *total* congestion of an edge in each 2-frame. In deciding how to shift packets into the overflow times, we need to be careful to account for how often the edge appears as the first or second transit. If an edge appears exclusively as a "first edge", we will only be able to shift it into the earlier overflow, and similarly if an edge appears exclusively as a "second edge".

Keeping this constraint in mind, our goal is to equalize as far as possible the distribution of edges into earlier and later overflows. We do this by the following scheme:

1. For each edge f and every odd integer $t = 1, 3, 5, \ldots, L$, repeat while $c_t(f) + c_{t+1}(f) > T$:
   2. If $c_t(f) = 0, c_{t+1}(f) > T$, then shift one packet into the later overflow time.
   3. Else if $c_t(f) > T, c_{t+1}(f) = 0$, then shift one packet into the earlier overflow time.
   4 Else if $c_t(f) + c_{t+1}(f) > T, c_t(f) > 0, c_{t+1}(f) > 0, c_t(f) + c_{t+1}(f) =$ odd, then shift one packet into the earlier overflow time.
   5. Else if $c_t(f) + c_{t+1}(f) > T, c_t(f) > 0, c_{t+1}(f) > 0, c_t(f) + c_{t+1}(f) =$ even, then shift one packet into the later overflow time.

Suppose we fix t to be some odd integer. If we let c' denote the congestions at the end of this overflow-shifting process, then we have $c'_t(f) + c'_{t+1}(f) \leq T$, and the number of packets shifted into the earlier (respectively later) overflow time can be viewed as a function of the original values of the congestions c_t, c_{t+1}. We denote these functions by $O^-(c_t, c_{t+1}; T)$ and $O^+(c_t, c_{t+1}; T)$ respectively.

Specifically we get the following condition:

Proposition 6.5. *Suppose that we have a schedule of even length L, and let $c_t(f)$ for $t = 1, \ldots, L$ denote the number of times f is scheduled as the tth edge of a packet. Suppose that for all edges $f \in E$ and all $t = 1, 3, 5, \ldots$ we satisfy the constraint*

$$O^+(c_t(f), c_{t+1}(f); T) + O^-(c_{t+2}(f), c_{t+3}(f); T) \leq T'$$

as well as the boundary constraints

$$O^-(c_1(f), c_2(f)) \leq T' \qquad O^+(c_{L-1}(f), c_L(f)) \leq T'$$

Then there is a schedule for the graph of makespan $L \times \frac{T+T'+2}{2} + T'$, which can be constructed in polynomial time.

Proof. After the modification, each 2-frame has congestion at most T, while each overflow time has congestion at T'. Each overflow time has delay at most 2, since for any packet x, there may be at most two edges scheduled into that overflow time, namely the edge that had been originally marked as the second edge of the earlier 2-frame, and the edge that had been originally marked as the first edge of the latter 2-frame. Hence each 2-frame can be scheduled in time $T + 1$ and each overflow can be scheduled in time $T' + 1$. As there are $L/2$ 2-frames in the original schedule, there are $L/2 + 1$ overflow periods. Hence the total cost is at most $L\frac{T+T'+2}{2} + T'$. □

Note that the conditions required by this Proposition 6.5 are local, in the sense that any violation is any event which affects an individual edge and a 4-interval which starts at an odd time t. We refer to such an interval for simplicity as an *aligned 4-interval*. We refer to the conditions required by this Proposition as the *4-conditions*; these conditions can be viewed as either pertaining to an entire schedule, or to an individual aligned 4-interval. We also note that the 4-conditions are *monotone*, in the sense that if a configuration violates them, then it will continue to do so if the congestion of any edge at any time is increased.

We can use this to improve our bound:

Theorem 6.6. *There is a schedule of length $8.84(C + D)$, which can be found in polynomial time.*

Proof. We assume $C + D \geq 2^{896}$ for simplicity.

As in the proof of Lemma 6.4, one obtains a schedule S of length $L = 1.0158 \times 1.004 \times 1.004(C+D)$ in which the congestion on any interval of length $i = 1024$ is at most $C = 1385$.

We divide S into frames of length $F = 1024m$ where $m = 64$, and add a random delay of length up to 1020 to each packet separately. This increases the length of the schedule up to $L' \leq 1.0157L + 1024$. We will first show that each frame individually satisfies the 4-conditions. So we may concentrate on a single such frame.

We associate a bad event to each aligned 4-interval I and edge f, that it violates the 4-conditions. It is not hard to see that the dependence of each such bad event is at most $Cmi^2/2$.

Now consider the probability of a bad event, that a given edge and interval has $O^+(c_t, c_{t+1}; T) + O^-(c_{t+2}, c_{t+3}, T) > T'$. (The boundary cases are similar and are omitted). There are up to C packets which could affect the given f, I. For each such packet and each position within the interval, there is a probability of at most $1/(i-4)$ that the packet x is scheduled at that time (conditional on any allocation to the other 4 positions). As the bad event is an increasing event, it suffices to suppose that the distribution of each $c_t, \ldots, c_{t+3}$ is iid Binomial with C trials and probability $1/(i-4)$.

Now, one can simply enumerate over all possible values and count the total probability of satisfying Proposition 6.5. This is possible because we are dealing with a fixed, finite and relatively small choice of C. A computer program calculates this probability to be 3.9×10^{-12} for the choice $T = 8, T' = 7$.

One can verify that these parameters satisfy the LLL condition. In particular, such a schedule S' exists and can be found in polynomial time.

In order to ensure that the entire schedule satisfies the 4-frame conditions, one may insert a delay of length 2 between consecutive frames. This ensure that the overflow at the boundaries of the separate frames do not interfere with each other. Doing this inflates the schedule from length L' to length $L'(1 + 2/F) + 2 \leq 1.0158L + 1027$.

By Proposition 6.5, this schedule S' can be scheduled in makespan $8.5L' + 7 \leq 8.84(C + D)$. Note that all the constructions used here can be implemented in polynomial time. □

6.4. **The PRA applied to packet routing.** So far, all of the improvements we have made to the packet routing problem used nothing more than the conventional LLL. We now show how to modify this construction to use the PRA in the appropriate places.

Let us examine more closely the process used to refine a schedule in which each interval of length C has congestion at most i. We break the schedule S into frames of length $F = mi$, and refine each separately. Within each frame, we add a random delay of length $b = i - i'$ to each packet separately. Let us fix an F-frame for the moment.

This is an assignment problem, in which we must assign a delay to each packet. Our bad events here correspond to an edge receiving an excessive congestion in some time interval. The precise meaning of an excessive congestion depends on which stage of the construction we are at. In the intermediate stages, all we care about is the total number of packets passing through that interval. In the final stage, we must satisfy the 4-conditions, which is a more delicate condition depending on the exact location of the packets within 4-intervals. In either case, these are local and increasing constraints.

We can modify Lemma 6.4 and Theorem 6.6 to use Theorem 2.5 instead of the LLL.

Proposition 6.7. *Let $i' < i$, let m, C', k be non-negative integers. Suppose there is a schedule S of length L such that every interval of length i has congestion at most C for some C.*

For a given choice of $d \leq C'$, $\vec{\lambda} \in [0, 1]$ define

$$\mu = Ci'\vec{\lambda}$$

and

$$p = \frac{\mu^d}{d!\binom{C'+1}{d}}$$

Suppose we have $p < 1$ and

$$(i - i')\vec{\lambda} - mi^2 i'\vec{\lambda}(d/\mu)\frac{p}{1-p} \geq 1$$

Then there is a schedule S' of length $L' = L(1 + 1/m) + i$, in which every interval of length i' has congestion $\leq C'$. Furthermore, such a schedule can be found in polynomial time.

Proof. Suppose we add delays in the range $b = i - i'$ uniformly to each packet within each frame of length $F = mi$. In this case, the categories correspond to each packet x, and for each delay t we assign $\vec{\lambda}_{x,t} = \vec{\lambda}$. For each edge f and i'-interval I, we introduce a bad event $B_{f,I}$ that the congestion in the interval exceeds C'. For this bad event we use a fractional hitting-set of width d as described in Theorem 5.1.

Fix a bad event $B_{f,I}$. This is an upper-tail event. There are at most C packets which could be scheduled to pass through the given edge, and there are i' possible delays which would contribute to the congestion of the given edge-interval. So, in all, the mean number of packets passing through the edge-interval is $\mu = Ci'\vec{\lambda}$. The bad event is that this exceeds C', so we have here $\delta = \frac{C'+1}{Ci'\vec{\lambda}} - 1$. This gives $G^{f,I} \leq (\frac{\mu^d}{d!\binom{C'+1}{d}})$

Now consider a fixed packet x. This packet x may pass through up to mi edges in the F-frame, and each of these can affect at most i intevals. Hence the packet x affects at most mi^2 of these bad events. For each such bad event f, I, there are at most i' possible delays that could be assigned to the given packet x to contribute to congestion of f, I. Hence, for each bad event $B_{f,I}$, we have $\mu_x = i'\vec{\lambda}$ and hence $G_x^{f,I} \leq i'\vec{\lambda}d/\mu(\frac{\mu^d}{d!\binom{C'+1}{d}})$.

By Theorem 2.11, this suffices to show a good configuration exists. □

Using this, we can replace all but the last step of the construction.

Theorem 6.8. *Suppose $C + D \geq 2^{896}$. Then there is a schedule of length $\leq 1.0157(C + D)$, in which every interval of length 1024 has congestion at most 1312.*

Proof. By Lemma 6.3, we form a schedule S_1, of length $L_1 \leq 1.004(C + D)$, in which each interval of length 2^{24} has congestion at most 17040600.

Apply Proposition 6.7 with $\vec{\lambda} = 5.985 \times 10^{-8}, C' = 1312, d = 247, m = 64$ to obtain a schedule S_2 of length $L_2 \leq 1.0157L_1 + 2^{24}$, in which each interval of length 1024 has congestion at most 1312. □

The final schedule is the most difficult to bound.

Proposition 6.9. *Let $m = 64, C = 1312, i = 1024, T = 5, T' = 4$ be given. Suppose there is a schedule S of length L such that every interval of length i has congestion at most C. There is a schedule of length $L' \leq (1 + 1/m)L + i$, which satisfies the 4-conditions with respect to T, T'. This schedule can be produced in polynomial time.*

Proof. For each edge f, and each aligned 4-interval I starting at time t, we introduce a bad event $B_{f,I}$ that

$$O^+(c_t, c_{t+1}) + O^-(c_{t+2}, c_{t+3}) > T'$$

For this edge f, I, and any packet x with delay t, we say that $\langle x, t\rangle$ has *type* j, if that packet-delay assignment would cause the given packet x to land at position $t + j$ within the bad event, for $j = 0, ..3$. If that assignment x, t does not contribute to $B_{f,I}$, then $\langle x, t\rangle$ has no type. For each bad event, there are at most C variables of each type.

For a bad event $B_{f,I}$ and a fractional hitting-set B', we define the score $s_j(f, I)$, for $j = 0, 1, 2, 3$ to be the maximum over all delays x, t of type j, of the quantity

$$\sum_{Y \subseteq B, \langle x,t\rangle \in Y} B'(Y)\vec{\lambda}^Y$$

Similarly, we define the overall-score to be $s(f, I) = g^{f,I} = \sum_{Y \subseteq B} B'(Y)\vec{\lambda}^Y$. For a collection of all bad events $B_{f,I}$, we define the score s_j to be the maximum $s_j(f, I)$ over all f, I.

By Theorem 2.11, if we satisfy the condition

$$(i-4)\vec{\lambda} - \frac{mi}{2}\frac{s_0+s_1+s_2+s_3}{1-s} \geq 1$$

then there is a schedule of length $L' \leq (1+1/m)L + i$, which satisfies the 4-conditions with T, T'.

Hence it suffices to produce a series of hitting-sets, for each of the bad events $B_{f,I}$, whose score is bounded.

Now let us fix a bad event $B_{f,I}$, and suppose that we have fixed $\vec{\lambda} = 1.23 \times 10^{-3}$. We will describe how to produce a good hitting-set. Although we have stated the proposition for a particular choice of parameters, we will walk through the algorithm we use to construct and find such a hitting-set.

The bad event depends solely on the number of assigned variables of each type, of which there are at most C. To simplify the notation, we suppose there are exactly C. Our hitting-set assigns weights to any subset of the $4C$ variables involved in the bad event. We will also select a symmetric hitting-set, in the sense that the weight assigned to any $Y \subseteq [C] \times [4]$ depends solely on the number of variables of each type in Y. So, for any $y_0, y_1, y_2, y_3 \leq C$, we will assign $B'(Y) = b(y_0, y_1, y_2, y_3)$ for any $Y \subseteq [C] \times [4]$ which has $|Y \cap [C] \times \{j\}| = y_j$, that is, for any subset Y which has exactly y_j variables of each type. In this case, we will have

$$s_0 = \sum_{y_0,y_1,y_2,y_3} \binom{C-1}{y_0-1}\binom{C}{y_1}\binom{C}{y_2}\binom{C}{y_3} b(y_0,y_1,y_2,y_3)\vec{\lambda}^{y_0+y_1+y_2+y_3}$$

and similarly for s_1, s_2, s_3, s.

In order to be valid, we must have $\sum_{Y \subseteq A} B'(Y) \geq 1$ for any atomic bad event A. By symmetry, this means that if we have k_0, k_1, k_2, k_3 minimal such that

$$O^+(k_0, k_1) + O^-(k_2, k_3) > T',$$

then we require

$$\sum_{y_0,y_1,y_2,y_3} \binom{k_0}{y_0}\binom{k_1}{y_1}\binom{k_2}{y_2}\binom{k_3}{y_3} b(y_0,y_1,y_2,y_3) \geq 1$$

We are trying to satisfy $(s_0+s_1+s_2+s_3)/(1-s) \leq t$, where t is some target value; here $t = 2.803 \times 10^{-6}$. For a fixed value of t, this is equivalently to minimizing $s_0+s_1+s_2+s_3+ts$. If we view the collection of all values $b(y_0, y_1, y_2, y_3)$ as linear unknowns, then we can view both the objective function and the constraints as linear. Hence this defines a linear program, which we can solve using standard linear programming algorithms.

For any y_0, y_1, y_2, y_3, we will set $b(y_0, y_1, y_2, y_3) = 0$ unless there is some such minimal bad $k_0, k_1, k_2, k_3 \geq y_0, y_1, y_2, y_3$. This greatly reduces the number of variables we need to consider, to something which is very large but tractable. For $T = 5, T' = 4$, for instance, the linear program has 5990 variables and 156 constraints. This is too large to write explicitly, but we wrote computer code which generates this system and solves it.

The resulting hitting-set achieves a bound of

$$\frac{s_0+s_1+s_2+s_3}{1-s} \leq 2.81 \times 10^{-6}$$

which satisfies the conditions of Theorem 2.11. We have listed this hitting-set in full in Appendix A. Note that this hitting-set gives a compact witness for Proposition 6.9; using it, one could verify the proposition directly without any need to use the algorithm we have just described. □

We now apply this construction to replace the two final steps in the construction of Section 6.3.

Theorem 6.10. *There is a feasible schedule of makespan at most* $5.70(C+D)$*, which can be constructed in expected polynomial time.*

Proof. For simplicity, we assume $C + D \geq 2^{896}$. By Theorem 6.8, we obtain a schedule S_1 of length $L_1 \leq 1.0157(C + D)$, in which each interval of length 1024 has congestion at most 1312.

Apply Proposition 6.9. This gives a schedule S_2 of length $L_2 \leq 1.0158L_1 + 1024$ satisfying the 4-conditions with $T = 5, T' = 4$. By Proposition 6.5, this yields a schedule whose makespan is $5.5L_2 + 5 \leq 5.70(C + D)$. □

Acknowledgments. We thank Tom Leighton and Satish Rao for valuable discussions long ago, which served as the foundation for this work; but for their declining, they would be coauthors of this paper. We are thankful to Noga Alon, Venkatesan Guruswami, Bernhard Haeupler, Penny Haxell, and Jeff Kahn for helpful discussions, as well as to the STOC 2013 and FOCS 2013 referees for their valuable comments.

Aravind Srinivasan dedicates this work to the memory of his late grandmother Mrs. V. Chellammal (a.k.a. Rajalakshmi): memories of her remain a great inspiration.

References

[1] R. Aharoni, E. Berger, and R. Ziv. Independent systems of representatives in weighted graphs. *Combinatorica*, 27:253–267, 2007.

[2] N. Alon. The linear arboricity of graphs. *Israel Journal of Mathematics*, 62:311–325, 1988.

[3] N. Alon. The strong chromatic number of a graph. *Random Structures and Algorithms*, 3:1–7, 1992.

[4] Y. Azar and A. Epstein. Convex programming for scheduling unrelated parallel machines. In *STOC '05: Proceedings of the 36th annual ACM Symposium on Theory of Computing*, pages 331–337. ACM, 2005.

[5] B. Bollobás, P. Erdős, and E. Szemerédi. On complete subgraphs of r-chromatic graphs. *Discrete Math.*, 1:97–107, 1975.

[6] R. B. Boppana and J. H. Spencer. A useful elementary correlation inequality. *Journal of Combinatorial Theory, Ser. A*, 50:305–307, 1989.

[7] P. Erdős and L. Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. In *Infinite and Finite Sets*, volume 11 of *Colloq. Math. Soc. J. Bolyai*, pages 609–627. North-Holland, 1975.

[8] C. M. Fortuin, J. Ginibre, and P. N. Kasteleyn. Corelational inequalities for partially ordered sets. *Communications of Mathematical Physics*, 22:89–103, 1971.

[9] B. Haeupler, B. Saha, and A. Srinivasan. New Constructive Aspects of the Lovász Local Lemma. *Journal of the ACM*, 58, 2011.

[10] D. G. Harris and A. Srinivasan. Constraint satisfaction, packet routing, and the Lovász Local Lemma. In *Proc. ACM Symposium on Theory of Computing*, pages 685–694, 2013.

[11] David G. Harris and Aravind Srinivasan. The Moser-Tardos Framework with Partial Resampling. In *FOCS*, pages 469–478, 2013.

[12] N. Harvey. A note on the discrepancy of matrices with bounded row and column sums. http://arxiv.org/abs/1307.2159, 2013.

[13] P. Haxell and T. Szabó. Odd independent transversals are odd. *Comb. Probab. Comput.*, 15(1-2):193–211, January 2006.

[14] P. E. Haxell. A note on vertex list colouring. *Combinatorics, Probability, and Computing*, 10:345–348, 2001.

[15] P. E. Haxell, T. Szabó, and G. Tardos. Bounded size components – partitions and transversals. *Journal of Combinatorial Theory, Series B*, 88:281–297, 2003.

[16] G. Jin. Complete subgraphs of r-partite graphs. *Combin. Probab. Comput.*, 1:241–250, 1992.

[17] R. M. Karp, F. T. Leighton, R. L. Rivest, C. D. Thompson, U. V. Vazirani, and V. V. Vazirani. Global wire routing in two-dimensional arrays. *Algorithmica*, 2:113–129, 1987.

[18] K. Kolipaka and M. Szegedy. Moser and Tardos meet Lovász. In *Proceedings of ACM STOC*, pages 235–244, 2011.

[19] F. T. Leighton, C.-J. Lu, S. B. Rao, and A. Srinivasan. New Algorithmic Aspects of the Local Lemma with Applications to Routing and Partitioning. *SIAM Journal on Computing*, 31:626–641, 2001.

[20] F. T. Leighton, B. M. Maggs, and S. B. Rao. Packet routing and jobshop scheduling in O(congestion + dilation) steps. *Combinatorica*, 14:167–186, 1994.

[21] J. K. Lenstra, D. B. Shmoys, and É. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46:259–271, 1990.

[22] P.-S. Loh and B. Sudakov. Independent transversals in locally sparse graphs. *Journal of Combinatorial Theory, Series B*, 97:904–918, 2007.

[23] R. Moser and G. Tardos. A constructive proof of the general Lovász Local Lemma. *Journal of the ACM*, 57(2):1–15, 2010.
[24] A. Panconesi and A. Srinivasan. Randomized Distributed Edge Coloring via an Extension of the Chernoff-Hoeffding Bounds. *SIAM Journal of Computing*, 26(2):350–368, 1997.
[25] W. Pegden. An extension of the Moser-Tardos algorithmic Local Lemma. *Arxiv 1102.2583*, 2011. To appear in the SIAM J. Discrete Mathematics.
[26] B. Peis and A. Wiese. Universal packet routing with arbitrary bandwidths and transit times. In *IPCO*, pages 362–375, 2011.
[27] P. Raghavan and C. D. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365–374, 1987.
[28] T. Rothvoss. A simpler proof for O(congestion+dilation) packet routing. In *Proc. Conference on Integer Programming and Combinatorial Optimization*, 2013. URL: http://arxiv.org/pdf/1206.3718.pdf.
[29] C. Scheideler. Universal routing strategies for interconnection networks. In *Lecture Notes in Computer Science*, volume 1390. Springer, 1998.
[30] J. P. Schmidt, A. Siegel, and A. Srinivasan. Chernoff-Hoeffding bounds for applications with limited independence. *SIAM J. Discrete Math.*, 8:223–250, 1995.
[31] M. Singh. *Iterative methods in combinatorial optimization*. PhD thesis, Tepper School of Business, Carnegie-Mellon University, 2008.
[32] T. Szabó and G. Tardos. Extremal problems for transversals in graphs with bounded degree. *Combinatorica*, 26:333–351, 2006.
[33] R. Yuster. Independent transversals in r-partite graphs. *Discrete Math.*, 176:255–261, 1997.

Appendix A. The fractional hitting-set for Proposition 6.9

The following table lists the fractional hitting-set used in Proposition 6.9. All sets which do not appear in this list have $b(y_1, y_2, y_3, y_4) = 0$.

y_0, y_1, y_2, y_3	$b(y_0, y_1, y_2, y_3)$	y_0, y_1, y_2, y_3	$b(y_0, y_1, y_2, y_3)$	y_0, y_1, y_2, y_3	$b(y_0, y_1, y_2, y_3)$
0 0 4 7	5.099206e-03	0 0 5 6	9.778912e-04	0 0 6 5	3.996599e-03
0 0 7 4	2.579365e-03	0 0 8 0	2.222222e-02	0 5 2 5	4.385269e-04
0 5 3 4	1.359760e-04	0 5 4 3	3.833186e-04	0 5 5 2	9.740495e-05
0 5 6 0	1.539957e-03	0 6 3 2	4.658320e-05	0 6 3 3	9.241610e-07
0 6 5 0	1.893794e-04	0 7 0 4	4.585073e-04	0 7 1 3	1.992969e-03
0 7 2 2	1.199898e-03	0 7 3 0	2.093047e-05	0 7 3 1	1.698355e-03
0 7 4 0	9.174968e-04	0 8 0 0	2.222222e-02	1 3 7 0	3.316187e-04
2 2 7 0	3.883201e-04	2 3 2 6	1.818870e-04	2 3 3 5	6.708826e-05
2 3 4 4	1.277478e-04	2 3 5 3	1.369389e-04	2 3 7 0	8.113415e-04
2 4 2 5	5.158908e-05	2 4 3 4	8.499152e-05	2 4 4 3	4.975736e-06
2 4 5 2	1.133274e-04	3 1 7 0	2.672980e-04	3 2 2 6	4.410280e-05
3 2 3 5	7.970540e-05	3 2 4 4	4.896451e-05	3 2 5 3	7.392384e-05
3 2 6 2	5.816254e-06	3 2 7 0	1.742220e-04	3 3 2 5	6.943956e-06
3 3 5 2	3.800255e-05	3 5 1 4	1.369777e-04	3 5 2 3	9.535644e-05
3 5 3 2	1.282475e-04	3 5 4 0	8.297563e-05	3 5 4 1	5.296674e-05
3 6 2 2	9.002473e-06	4 1 2 6	1.767087e-04	4 1 3 5	5.212549e-05
4 1 4 4	1.093016e-04	4 1 5 3	9.464293e-05	4 1 6 2	3.789534e-05
4 1 7 0	1.125084e-03	4 2 2 5	7.353027e-05	4 2 3 4	1.587323e-05
4 2 4 3	1.237762e-05	4 2 5 2	8.670195e-05	4 3 2 4	5.232234e-05
4 3 3 3	8.595063e-05	4 3 4 2	2.209769e-05	4 3 5 0	1.041270e-04
4 4 1 4	1.052873e-04	4 4 2 3	1.437882e-05	4 4 3 2	4.431716e-05
4 4 4 0	7.896302e-05	4 4 4 1	4.184396e-05	4 5 1 3	5.225282e-05
4 5 2 2	5.006599e-05	4 5 3 1	3.748908e-05	5 1 2 5	2.980766e-05
5 1 3 4	8.802145e-05	5 1 4 3	4.768986e-05	5 1 5 2	7.014272e-05
5 3 1 4	6.990688e-05	5 3 2 3	1.155525e-05	5 3 3 2	3.894932e-05
5 3 4 1	3.980330e-05	5 4 1 3	3.277361e-05	5 4 2 2	7.791839e-05
5 4 3 1	2.503639e-05	5 6 0 0	1.549509e-03	6 2 1 4	1.448414e-04
6 2 2 3	6.887231e-05	6 2 3 2	1.145665e-04	6 2 4 0	1.365900e-04
6 2 4 1	4.324878e-05	6 2 5 0	4.360905e-05	6 3 1 3	7.156338e-05
6 3 2 2	3.895600e-05	6 3 3 1	5.915442e-05	6 6 0 0	2.267574e-04
7 2 1 3	3.562315e-05	7 2 2 2	7.950298e-05	7 2 3 1	1.473170e-05
7 5 0 0	1.289683e-03	8 4 0 0	3.756614e-03		

Appendix B. The Nostradamus Lemma

In the proof of Lemma 2.6, we use the following general "Nostradamus Lemma." To justify this somewhat cryptic name, consider the following analogy. In a dusty book of prophecy, one reads that "sometime in the future, you will meet a man named John Doe. The first time you meet such a man, he will flip a coin a hundred times, all of which come up heads. Also, you will eventually meet a woman Jane Doe; the first such woman whom you meet will also flip a hundred coins, all of which come up heads." Now, you do not know when these meetings will come or which will come first, if ever, but you can confidently say that the probability that this prophecy comes true is at most 2^{-200}.

This probabilistic principle seems fairly intuitive, but we note that there are two ways it can go wrong. Suppose that we predict that John Doe will flip a hundred heads and also a red-haired man will flip a hundred heads. This probability could be just 2^{-100}, because the two men may be the same person. Another possibility: suppose we predict that "sometime in the future you will meet a man named John Doe who flips a hundred heads." The probability of this event could be one, if we encounter an infinite series of men with the same name.

Lemma B.1 (Nostradamus Lemma)**.** *Suppose one has a stochastic process indexed $\langle X_t \mid t \in \mathbf{N} \rangle$. Let S denote the countably infinite set of possible histories for this process; for notational simplicity we suppose that the states X_t themselves lie in S (i.e., that the state also includes the history thus far). The process begins in state $s_0 \in S$, which represents the null history. There are ℓ Boolean functions $A_i, B_i : S \to \{0, 1\}$, for some finite ℓ. Suppose that, for all $t \in \mathbf{N}$ and all $s \in S$, we have that*

$$\mathbf{P}(B_i(X_{t+1}) = 1 \mid X_t = s_t) \leq p_i.$$

Now define the event E that the following conditions are all satisfied:

1. *For each $i \in [l]$, there is exactly one time t_i such that $A_i(X_{t_i}) = 1$;*
2. *For all $i \neq i'$ we have $t_i \neq t_{i'}$;*
3. *For all $i \in [l]$ we have $B_i(X_{t_i+1}) = 1$.*

Then the event E is measurable, and its probability is at most $\mathbf{P}(E) \leq p_1 p_2 \dots p_\ell$.

Proof. Define the event E_T, which is that event E holds and also that $t_1, \dots, t_\ell \leq T$. We will prove that $\mathbf{P}(E_T) \leq p_1 \dots p_\ell$ by induction on T. For a given value of T, the induction will apply across all possible stochastic systems and all possible values of ℓ.

First, suppose $T = 0$. If $\ell > 0$, then the event E_T is impossible; if $\ell = 0$, then event E_T is certain. Either way the inequality holds.

Next, suppose $T > 0$. Count how many values of i are there such that $A_i(s_0) = 1$. If there is more than one such value, then event E_T is impossible, and the statement holds.

Suppose first that for all i we have $A_i(s_0) = 0$. Then the event E_T is equivalent to the event E'_{T-1}, where E' is an event similar to E except that it is defined on the stochastic process which starts at state X_1. By induction hypothesis, the event E'_{T-1} has probability at most $p_1 \dots p_\ell$ for any X_1. Integrating over X_1, we have the E_T has at most this probability as well.

Finally, suppose that there is exactly one value of i such that $A_i(s_0) = 1$; without loss of generality say it is $i = \ell$. Then the event E_T is equivalent to the event that $B_\ell(X_1) = 1$ *and* that the event E'_{T-1} occurs, where E' is an event similar to E except that it is defined on the stochastic process which starts at state X_1 and only includes Boolean functions $A_1, B_1, \dots, A_{\ell-1}, B_{\ell-1}$. The probability of $B_\ell(X_1) = 1$ is at most p_ℓ. By induction hypothesis, for any such X_1, the probability of the event E' is at most $p_1 \dots p_{\ell-1}$. Hence, integrating over all X_1, the probability of this event is at most $p_1 \dots p_\ell$.

So we have shown that $\mathbf{P}(E_T) \leq p_1 \dots p_\ell$ for all $T \in \mathbf{N}$. Now note that $E_0 \subseteq E_1 \subseteq E_2 \subseteq \dots$ and $E = \cup_{T \in \mathbf{N}} E_T$. Each set E_T is cylindrical (it is determined by the first T coordinates), hence measurable. This implies that E is measurable as well with $\mathbf{P}(E) \leq p_1 \dots p_\ell$. □