2013 IEEE 54th Annual Symposium on Foundations of Computer Science

The Moser-Tardos Framework with Partial Resampling

David G. Harris
University of Maryland, College Park, MD 20742
Research supported in part by NSF Award CNS-1010789
Email: davidgharris29@hotmail.com.

Aravind Srinivasan
University of Maryland, College Park, MD 20742.
Research supported in part by NSF Award CNS-1010789.
Email: srin@cs.umd.edu.

***Abstract*—The resampling algorithm of Moser & Tardos is a powerful approach to develop 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 set of variables that define this event, rather than the entire set as in Moser & Tardos. 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 transversals asymptotically.**

I. Introduction

The Lovász Local Lemma (LLL) [1] is a fundamental probabilistic tool. The breakthrough of Moser & Tardos shows that a very natural *resampling* approach yields a constructive approach to the LLL [2]; this, along with a few subsequent investigations [3], [4], gives a fairly comprehensive suite of techniques to develop algorithmic versions of the LLL. The basic algorithm of [2] 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 Resampling Algorithm; 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.

In order to motivate our applications, we start with two classical problems: scheduling on unrelated parallel machines [5], and low-congestion routing [6]. In the former, we have n jobs and K machines, and each job i needs to be scheduled on any element of a given subset X_i of the machines (this is not the standard notation for job-scheduling problems). 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 if we can schedule with makespan T [5], [7]. Letting $[k]$ denote the set $\{1, 2, \ldots, k\}$, a moment's reflection leads to the following integer-programming formulation:

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

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

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

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

(Although (3) is redundant for the IP, it will be critical for the natural LP relaxation [5].)

Our class of problems. Given the above example, 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 II; Theorem 2.4 presents some general conditions under which our algorithm quickly computes a feasible solution $\{x_{i,j}\}$. The same class of problems that we study, has been investigated by us in recent work [8]. Here is a comparison:

(i) the work of [8] is *non-constructive*; all our results are constructive, and our approach based on hitting sets and partial resampling, is completely different from the "Rödl

0272-5428/13 $26.00 © 2013 IEEE
DOI 10.1109/FOCS.2013.57

Nibble" based approach of [8] and its inductive proof;
(ii) we obtain significantly-improved quantitative bounds in many cases over the results of [8] (which – again – are nonconstructive) as mentioned in a few places below (see, e.g., the comparison toward the end of Section I-A); and
(iii) our approach does not seem to be a constructive version of [8], and there appears to be no formal inclusion either way; on the one hand, there appear to be some parameter ranges for Theorem 4.2 in which [8] can give slightly better bounds, and on the other hand, some of the work of [8] can be interpreted in our language.

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 similar (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 $\delta = t/\mu - 1 \geq 0$, define

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

this 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 set $\text{Chernoff}(\mu, t) = 1$.

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

A. 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 [9] shows constructively that $b_k = b'_k + D$ suffices. Given a solution to the LP-relaxation of makespan minimization, this bound 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 [5], [7].

The second approach to our problem is randomized rounding [6]: 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 followed by a union bound" approach [6] shows that $\text{Chernoff}(b'_k, b_k) \leq 1/(2K)$ suffices. That is, there is some constant $c_0 > 0$ such that

$$b_k \geq \begin{cases} c_0 \cdot \frac{\log K}{\log(2 \log K / b'_k)} & \text{if } b'_k \leq \log K; \\ b'_k + c_0 \cdot \sqrt{b'_k \cdot \log K} & \text{if } b'_k > \log K \end{cases} \tag{6}$$

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 ($b'_k = b_k + D$ vs. (6)). 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, $b'_k = b_k + D$ gives an $O(D)$-approximation, and (6) yields an $O(\log K / \log\log K)$-approximation. Can we do better? We answer this in the affirmative in Theorem 4.2 – we are able to essentially replace K by D in (6), by showing constructively that for *any* desired constant $C_1 > 0$, there exists a constant $C_0 > 0$ such that the following suffices:

$$b_k \geq \begin{cases} C_0 \cdot \frac{\log D}{\log(2 \log D / b'_k)} & \text{if } b'_k \leq \log D \\ b'_k + b'_k D^{-C_1} + C_0 \cdot \sqrt{b'_k \cdot \log D} & \text{if } b'_k > \log D \end{cases} \tag{7}$$

Application to multi-dimensional scheduling. Consider the following D-dimensional generalization of scheduling to minimize makespan, studied by Azar & Epstein [10]. 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 [5]), 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 makespan of at most* T_ℓ *in each dimension* ℓ*?* The framework of [10] and [9] gives a $(D+1)$-approximation, while our bound (7) yields an $O(\log D / \log\log D)$-approximation, which can be a significant improvement over the $O(\log K / \log\log K)$-approximation that follows from (6).

Comparison with other known bounds. As described above, our bound (7) improves over the two major approaches here. However, two related results deserve mention. First, a bound similar to (7) is shown in [11], [8], 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 [11] 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 [8] is nonconstructive.

B. 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 [12], and has received a considerable amount of attention (see, e.g., [13], [14], [15], [16], [17], [18], [19], [20], [21]). Furthermore, such transversals serve as building blocks for other graph-theoretic parameters such as the linear arboricity and the strong chromatic number [14], [15]. We improve (algorithmically) a variety of known sufficient conditions for the existence of good transversals, in Section III. In particular, Szabó & Tardos present a conjecture on how large the blocks should be, to guarantee the existence of transversals that avoid K_s [20]; we show that this conjecture is true asymptotically for large s. We also study weighted transversals, as considered by Aharoni, Berger & Ziv [13], and show that near-optimal (low- or high-) weight transversals exist, and can be found efficiently. In particular, we improve the quantitative bounds of [8] and show that "large-weight" (existentially-optimal) independent transversals exist, once the smallest block-size becomes reasonably large.

C. 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 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 $T \geq (1 + \Omega(1)) \cdot (C + D)$ [22]. A seminal result of [23] 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 [24], [25], our work [8] 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 Resampling Algorithm. To understand the intuition behind our 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 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.

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 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. Further specific comparisons with the standard LLL are made in Sections III and IV-A.

We omit several proofs in this version due to the lack of space.

II. The Resampling Algorithm

A. Notation

We begin by discussing some basic definitions. We have n categories, which we identify with the set $[n] = \{1, \ldots, n\}$. Each category has a set of possible assignments X_i. We specify a probability distribution p_i on each category i, with $\sum_{j \in X_i} p_{i,j} = 1$. We will usually not be explicit about the set of possible assignments, so we would write simply

$\sum_j p_{i,j} = 1$. We refer to any ordered pair $\langle i,j\rangle$ where $j \in X_i$ as an *element*; we sometimes refer to an element as (i,j) as well. We let X denote the set of all elements. Given any vector $\vec{\lambda} = (\vec{\lambda}_{i,j})$ indexed by elements $\langle i,j\rangle$, we define, for any set $Y \subseteq X$, $\vec{\lambda}^Y = \prod_{\langle i,j\rangle \in Y} \vec{\lambda}_{i,j}$.

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,

$$B_k = \{Y \subseteq X \mid A_{k,1} \subseteq Y \vee \cdots \vee A_{k,m_k} \subseteq Y\}. \quad (8)$$

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.

B. 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 (8), we have for all $A \in B$ that $\sum_{Y \subseteq A} C(Y) \geq 1$.

C. Resampling Algorithm and the Main Theorem

We first present our partial resampling algorithm:

- 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 [2], 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 2.2: **(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 there is some atomic event $A'' \in B_k$ with $Y, Y' \subseteq A''$.

Relation $\approx$ is defined between pairs (Y,k): the only admissible pairs (Y,k) here are those for which $Y \subseteq A$ for some atomic $A \in B_k$. We define $(Y,k) \approx (Y',k')$ iff: (i) $Y \sim Y'$, or (ii) $k = k'$ and $Y \bowtie_k Y'$.

We now also define:

Definition 2.3: **(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, we define

$$G^k_{i,j}(\vec{\lambda}) = \sum_{Y \ni \langle i,j\rangle} B'_k(Y)\vec{\lambda}^Y$$
$$G^k_i(\vec{\lambda}) = \sum_{Y \sim i} B'_k(Y)\vec{\lambda}^Y, \quad \text{and}$$
$$G^k(\vec{\lambda}) = \sum_i G^k_i(\vec{\lambda}).$$

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 2.4: **(Main Theorem)** In each of the following three cases, the Resampling Algorithm 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 $k \in [K]$,

$$\mu(Y,k) \geq p^Y B'_k(Y) \prod_{(Y',k') \approx (Y,k)} (1 + \mu(Y',k')).$$

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 Resampling Algorithm with $p_{i,j} = \vec{\lambda}_{i,j} / \sum_{j'} \vec{\lambda}_{i,j'}$ for all i,j. We will use Definition 2.3 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 implied by $\sim$: i.e., for all k, $Y \bowtie_k Y'$ implies that $Y \sim Y'$. 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 $\vec{\lambda}_i$.

D. Proof ingredient: Witness Trees

A key component of our proofs will be the notion of *witness trees* similar to [2]. Just as in the proof of [2], 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 [2], but driven by the relation $\approx$: we start with the event of interest (for example, the final resampling), which goes at the root of the tree. This, and all nodes of the tree, will be labeled by the corresponding pair (Y,k). Stepping backward in time, suppose the current event being processed is labeled (Y,k). If there is no (Y',k') in the current tree such that $(Y,k) \approx (Y',k')$, then we skip over (Y,k) (and move on to the previous time-step). If not, let v labeled (Y',k') be a node of the lowest level (i.e., highest depth) in the current witness tree such that $(Y,k) \approx (Y',k')$; make the new node labeled (Y,k) a child of this node v. Continue this process going backward in time to complete the construction of the witness tree.

We next introduce our main Lemma 2.5, which parallels the analysis of [2], connecting the witness trees to the execution logs. However, its proof is much more involved.

Lemma 2.5: Let τ be any witness 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

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

Proof: 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 $(Y,k) \approx (Y',k')$, 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. The probability of this is $p_{i,j}$.

We next consider the probability of selecting set Y. Now, consider all the nodes of the witness tree which are at a lower level than Y and are adjacent under $\approx$ to (Y,k); denote these by $(Y_1,k_1),(Y_2,k_2),\ldots,(Y_x,k_x)$. These are not necessarily distinct, but in that case they would be listed with their multiplicity.

Again, because of the way τ is formed, we must select Y *after* selecting these sets. After we select $Y_1,\ldots,Y_x$, we must encounter some atomic event $A \in B_k$ for which Y is eligible. We now claim that the *first time* after selecting $Y_1,\ldots,Y_x$ that we encounter such an A, then we must select Y. For, suppose we select some other $Y' \subseteq A$. In order for the tree to contain (Y,k), we must sometime later in the execution log select Y. In this case, (Y',k) will appear lower in the witness tree than our node s, and we will have $(Y,k) \approx (Y',k')$. Regardless of the exact position of Y' in the witness tree, we will thus not see the tree τ as a subtree of the witness tree. This gives us a condition which is more complicated than the one previously: the first time after selecting $Y_1,\ldots,Y_x$ that we encounter some atomic event for which (Y,k) is eligible, we select it. This condition has a probability of at most $B'_k(Y)$, irrespective of what the violated set A was.

If all these events had a fixed time-ordering, then the system's stochasticity would immediately imply that their joint probability is the product of their individual probabilities, at most $\prod p^{Y_s} B'_{k_s}(Y_s)$. Unfortunately, the time sequence of these events is not fully determined. As we show in the full paper, these events all have a specific form, which allows us to still conclude that the total probability can be obtained by multiplying individual probabilities. ■

The Resampling Algorithm and its proof are very similar to [2], and it is tempting to view this as a special case of that algorithm. However, the proof of the analogue of Lemma 2.5 for the Moser-Tardos algorithm uses a quite different argument based on coupling. This type of argument does not appear to work for bounding the probability of selecting a given Y.

In order to show that this algorithm converges, we show that the total weight of all large witness trees is small. Using arguments from Moser & Tardos, we can immediately show Theorem 2.4(a). For many applications of the Resampling Algorithm, 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.

E. 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*.

This type of accounting is useful not just for our Resampling Algorithm, 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 [26]. Nevertheless, it is stronger and simpler than the standard LLL, particularly the asymmetric LLL.

Definition 2.6: Given k and a set Y with $Y \subseteq A$ for some atomic $A \in B_k$, we define a family $\mathcal{Y} \subseteq 2^X$ to be a *k-neighborhood of Y* if the following properties hold:

- (P1) for all $Y' \in \mathcal{Y}$, we have $Y' \bowtie_k Y$ and $Y' \not\sim Y$; and
- (P2) for all distinct $Y', Y'' \in \mathcal{Y}$, we have $Y' \not\bowtie_k Y''$.

Note that $\emptyset$ is always a k-neighborhood of Y.

In many settings, the linkages due to $\bowtie$ are relatively insignificant compared to the linkages due to $\sim$. One possible reason for this are that the $\bowtie$ linkage becomes null (any pair of sets $Y_1 \bowtie_k Y_2$ for any k, already have $Y_1 \sim Y_2$ – and so $\emptyset$ is the only k-neighborhood); 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 a mutual recursion as in Theorem 2.4(a): we get a "pure" recurrence (9), which can then be used in (10).

Definition 2.7: **(Value $\hat{S}_k$ depending on vector $\vec{\lambda}$)** Suppose further that we are given an assignment of non-negative real numbers $\vec{\lambda}_{i,j}$ to each element $\langle i, j\rangle$. We define, for each k, a real $\hat{S}_k$ (if it exists) to be any value that satisfies the condition

$$\hat{S}_k \geq \max_{Y:\ B'_k(Y)>0} \sum_{\substack{\mathcal{Y}:\ \mathcal{Y} \text{ is a} \\ k\text{-neighborhood of } Y}} (\hat{S}_k)^{|\mathcal{Y}|} \prod_{Y' \in \mathcal{Y}} B_k(Y') \vec{\lambda}^{Y'}. \tag{9}$$

Note that $\hat{S}_k \geq 1$ because of the neighborhood $\mathcal{Y} = \emptyset$. Furthermore, note that if the $\bowtie$ relation is null (i.e. $Y_1 \bowtie_k Y_2 \Rightarrow Y_1 \sim Y_2$), then $\hat{S}_k = 1$.

Theorem 2.8: 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 further that for each k, there is some real $\hat{S}_k$ that satisfies (9, and that

$$\forall i,\ \vec{\lambda}_i \geq 1 + \sum_k \hat{S}_k G_i^k, \tag{10}$$

where $G_i^k = G_i^k(\vec{\lambda})$ as in Definition 2.3. Then the Resampling Algorithm terminates with probability one, and the expected number of resamplings of a category i is at most $\vec{\lambda}_i$.

The worst case for $\hat{S}_k$ occurs when all the atomic events within B_k are connected, yielding:

Proposition 2.9: Given a vector $\vec{\lambda}$ with $G^k(\vec{\vec{\lambda}}) < 1$, we can set $\hat{S}_k = \frac{1}{1-G^k}$ to satisfy (9).

Parts (b) and (c) of Theorem 2.4 can be unified. Suppose we are given some $\hat{S}_k$ which satisfies (9). In this case, we define for each $\langle i, j\rangle$ $H_{i,j} = \sum_k \hat{S}_k G_{i,j}^k$ and similarly $H_i = \sum_j H_{i,j}$. Then the criterion of Theorem 2.4 has the simple form $\vec{\lambda}_i - H_i \geq 1$.

Next, if we are given $\vec{\lambda}, B'_k$ satisfying Theorem 2.8, then we know that there exists a configuration which avoids all bad events. Furthermore, such a configuration can be found by running the Resampling Algorithm. 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 Resampling Algorithm 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 Resampling Algorithm:

Theorem 2.10: (a) Suppose that we satisfy Theorem 2.4(a). Then, for any atomic event E, the probability that E is true in the output of the Resampling Algorithm, is at most $P(E) \prod_{k, Y \sim E} (1 + \mu(k, Y))$.

(b) Suppose that we satisfy Theorem 2.4(b) or Theorem 2.4(c). Let $J \subseteq X_i$. The probability that the Resampling Algorithm ever selects $\langle i, j\rangle$ for $j \in J$ is at most $\frac{\sum_{j \in J} \vec{\lambda}_{i,j}}{\vec{\lambda}_i - H_i + \sum_{j \in J} H_{i,j}}$

(c) Suppose that we satisfy Theorem 2.4(b) or Theorem 2.4(c). The probability that the Resampling Algorithm ever simultaneously selects $\langle i_1, j_1\rangle, \ldots, \langle i_k, j_k\rangle$, is at most $\lambda_{i_1,j_1} \ldots \lambda_{i_k,j_k}$.

A simple corollary of Theorem 2.10 shows a *lower bound* on the probability of selecting a given element $\langle i, j\rangle$:

Corollary 2.11: Suppose that we satisfy Theorem 2.4(b) or Theorem 2.4(c). Let $J \subseteq X_i$. The probability that the Resampling Algorithm terminates by selecting $\langle i, j\rangle$ for $j \in J$ is at least $\frac{\sum_{j \in J} \vec{\lambda}_{i,j} - \sum_{j \in J} H_{i,j}}{\vec{\lambda}_i - \sum_{j \in J} H_{i,j}}$.

III. 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 [12]; more recently it has been analyzed in [20], [17], [21], [18], [27]). 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 $b \geq 2e\Delta$ [14], where Δ is the maximum degree of any vertex in the graph. Haxell provides an elegant topological proof that a sufficient condition is $b \geq 2\Delta$ [16]. The condition of [16] is existentially optimal, in the sense that $b \geq 2\Delta - 1$ is not always admissible [18], [21], [20]. The work of [17] 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 [19] 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 [16] 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 3.1. As shown in [18], [21], [20], 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 [19], 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 [26], but (R2) and (R3) appear new to our knowledge.

Theorem 3.1: 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 Resampling Algorithm, 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 paths of length two in G; for (R3) they are the triangles of G.) We apply Theorem 2.4(c) with all entries of $\vec{\lambda}$ equal to a scalar α. This has a solution $\alpha > 0$ iff $b \geq \frac{4d}{r}$. ■

A. 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 [20] shows the slightly stronger lower bound $\gamma_s \geq \frac{s}{(s-1)^2}$; intriguingly, this is conjectured in [20] to be exactly tight. On the other hand, a construction in [19] 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 [20] gives the correct *asymptotic* rate of growth, up to lower-order terms; i.e., we show in Theorem 3.2 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 will have degree $< s-1$, and hence we avoid K_s. Furthermore, such transversals can be found in polynomial time.

Comparison with the standard LLL: We now discuss how one might approach this problem using the standard LLL, and why this approach falls short. As in [14], 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 can be solved when $b/\Delta \geq e/s + o(1/s)$. 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. The Resampling Algorithm gives us a way to "localize" this global information to the neighborhood of a vertex. We now present our theorem here:

Theorem 3.2: 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: Apply Theorem 2.4(c), assigning the same vector $\vec{\lambda} = \alpha$ where $\alpha > 0$ is a scalar. Such an α exists under the conditions of Theorem 3.2. ■

We note that this result improves on [19] 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.

We next study *weighted* transversals, as considered by [13]. We use our weighting condition to lower- and upper-bound the weights of independent transversals, which is not possible using [16] or [26].

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 3.3 and 3.4 improve quite a bit upon the (nonconstructive) bounds of [8]; 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 [8].

Theorem 3.3: 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

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

We show a matching upper bound on weights:

Theorem 3.4: 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 $w(I) \leq \frac{w(V)}{b}$.

We can give similar bounds for independent transversals omitting other subgraphs.

IV. 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 4.1: 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 2.3, 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 by setting $d = \lceil \mu\delta \rceil$, one can achieve the Chernoff bounds [28]:

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

A. LP rounding for column-sparse packing problems

In light of Theorem 4.1, 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 are the usual assignment constraints: 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 LP 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 $\sum_k a_{k,x} \leq D$. The condition on the separation between b_k and b'_k is based on the Chernoff separation function from Definition 1.1.

Theorem 4.2: There is a constant $C > 0$ with the following property. Suppose we have an LP parameterized 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 $\frac{b_k}{b'_k}\text{Chernoff}(b'_k(1+\epsilon), b_k) \leq \frac{C\epsilon}{D}$

(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, such a satisfying assignment can be found in polynomial time.

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 4.2; 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 would be impossible 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. Of course, this dependency might be very small, but the LLL cannot exploit this.

Theorem 4.2 is given a generic setting, which is intended to handle a wide range of sizes for the parameters b_k, b'_k, D. We illustrate some typical applications.

Proposition 4.3: Suppose we are given an LP satisfying the requirement of Theorem 4.2; let $c > 0$ be any desired constant. Then we can set b_k as follows so as to satisfy Theorem 4.2:

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.

The multiplicative factor $(1 + D^{-c})$ in Proposition 4.3 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 4.4: Suppose we are given an LP satisfying the requirement of Theorem 4.2; 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 4.2.

The multi-dimensional scheduling application from the introduction follows as an easy application of Proposition 4.3. 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 4.3, we can now set $b_k = O(\log D / \log \log D)$.

B. Packet routing

We begin by reviewing the basic strategy of [24], its improvements by [25] and [8]. [24] is a very readable overview of our basic strategy, and we will not include all the details which are covered there. We note that [25] 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. 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 [23] has shown that in fact a schedule of makespan $O(C + D)$ is possible. [24] 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 [25] and improved in [8] to $8.84(C + D)$. A non-constructive probabilistic argument in [8] showed the existence of schedules with makespan $7.26(C + D)$.

In the initial graph, the 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 form a feasible schedule by expanding each timestep into C/D separate timesteps.

Although the details of this construction are too complicated to discuss fully here, the following intuition is still useful. By choosing the delays for each packet, the congestion within each time-step can be regarded as a sum of random variables, with mean $O(1)$. We want to ensure

that this congestion is in fact bounded by $O(1)$, over all timesteps and all edges. (This is a simplification; in fact, in the final stages of the construction of [8], the necessary condition is a complicated and non-linear function of the congestion within four adjacent time-steps.) So, the bounds we have developed in Theorem 4.1 apply.

Theorem 4.5: There is a schedule of makespan at most $5.70(C + D)$, which can be constructed in expected polynomial time.

Acknowledgment. We thank the FOCS 2013 referees for their very helpful suggestions.

REFERENCES

[1] P. Erdős and L. Lovász, "Problems and results on 3-chromatic hypergraphs and some related questions," in *Infinite and Finite Sets*, ser. Colloq. Math. Soc. J. Bolyai. North-Holland, 1975, vol. 11, pp. 609–627.

[2] R. Moser and G. Tardos, "A constructive proof of the general Lovász Local Lemma," *Journal of the ACM*, vol. 57, no. 2, pp. 1–15, 2010.

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

[4] K. Kolipaka and M. Szegedy, "Moser and Tardos meet Lovász," in *Proceedings of ACM STOC*, 2011, pp. 235–244.

[5] J. K. Lenstra, D. B. Shmoys, and E. Tardos, "Approximation algorithms for scheduling unrelated parallel machines," *Mathematical Programming*, vol. 46, pp. 259–271, 1990.

[6] P. Raghavan and C. D. Thompson, "Randomized rounding: a technique for provably good algorithms and algorithmic proofs," *Combinatorica*, vol. 7, pp. 365–374, 1987.

[7] M. Singh, "Iterative methods in combinatorial optimization," Ph.D. dissertation, Tepper School of Business, Carnegie-Mellon University, 2008.

[8] D. G. Harris and A. Srinivasan, "Constraint satisfaction, packet routing, and the Lovász Local Lemma," in *Proc. ACM Symposium on Theory of Computing*, 2013.

[9] 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*, vol. 2, pp. 113–129, 1987.

[10] 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*. ACM, 2005, pp. 331–337.

[11] 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*, vol. 31, pp. 626–641, 2001.

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

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

[14] N. Alon, "The linear arboricity of graphs," *Israel Journal of Mathematics*, vol. 62, pp. 311–325, 1988.

[15] ——, "The strong chromatic number of a graph," *Random Structures and Algorithms*, vol. 3, pp. 1–7, 1992.

[16] P. E. Haxell, "A note on vertex list colouring," *Combinatorics, Probability, and Computing*, vol. 10, pp. 345–348, 2001.

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

[18] G. Jin, "Complete subgraphs of r-partite graphs," *Combin. Probab. Comput.*, vol. 1, pp. 241–250, 1992.

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

[20] T. Szabó and G. Tardos, "Extremal problems for transversals in graphs with bounded degree," *Combinatorica*, vol. 26, pp. 333–351, 2006.

[21] R. Yuster, "Independent transversals in r-partite graphs," *Discrete Math.*, vol. 176, pp. 255–261, 1997.

[22] 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.

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

[24] C. Scheideler, "Universal routing strategies for interconnection networks," in *Lecture Notes in Computer Science*. Springer, 1998, vol. 1390.

[25] B. Peis and A. Wiese, "Universal packet routing with arbitrary bandwidths and transit times," in *IPCO*, 2011, pp. 362–375.

[26] W. Pegden, "An extension of the Moser-Tardos algorithmic Local Lemma," *Arxiv 1102.2583*, 2011, To appear in the SIAM J. Discrete Mathematics.

[27] P. Haxell and T. Szabó, "Odd independent transversals are odd," *Comb. Probab. Comput.*, vol. 15, no. 1-2, pp. 193–211, Jan. 2006.

[28] J. P. Schmidt, A. Siegel, and A. Srinivasan, "Chernoff-Hoeffding bounds for applications with limited independence," *SIAM J. Discrete Math.*, vol. 8, pp. 223–250, 1995.