New Constructive Aspects of the Lovász Local Lemma

Bernhard Haeupler Barna Saha Aravind Srinivasan

***Abstract*—The Lovász Local Lemma (LLL) is a powerful tool that gives sufficient conditions for avoiding all of a given set of "bad" events, with positive probability. A series of results have provided algorithms to efficiently construct structures whose existence is non-constructively guaranteed by the LLL, culminating in the recent breakthrough of Moser & Tardos. We show that the output distribution of the Moser-Tardos algorithm well-approximates the *conditional LLL-distribution* – the distribution obtained by conditioning on all bad events being avoided. We show how a known bound on the probabilities of events in this distribution can be used for further probabilistic analysis and give new constructive and non-constructive results.**

We also show that when an LLL application provides a small amount of slack, the number of resamplings of the Moser-Tardos algorithm is nearly linear in the number of underlying independent variables (not events!), and can thus be used to give efficient constructions in cases where the underlying proof applies the LLL to super-polynomially many events. Even in cases where finding a bad event that holds is computationally hard, we show that applying the algorithm to avoid a polynomial-sized "core" subset of bad events leads to a desired outcome with high probability.

We demonstrate this idea on several applications. We give the first constant-factor approximation algorithm for the Santa Claus problem by making an LLL-based proof of Feige constructive. We provide Monte Carlo algorithms for acyclic edge coloring, non-repetitive graph colorings, and Ramsey-type graphs. In all these applications the algorithm falls directly out of the non-constructive LLL-based proof. Our algorithms are very simple, often provide better bounds than previous algorithms, and are in several cases the first efficient algorithms known.

As a second type of application we consider settings beyond the critical dependency threshold of the LLL: avoiding all bad events is impossible in these cases. As the first (even non-constructive) result of this kind, we show that by sampling from the LLL-distribution of a selected smaller core, we can avoid a fraction of bad events that is higher than the expectation. MAX k-SAT is an example of this.

haeupler@mit.edu; CSAIL, Dept. of Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139. Part of this work was done while visiting the University of Maryland.

barna@cs.umd.edu; Dept. of Computer Science, University of Maryland, College Park, MD 20742

srin@cs.umd.edu; Dept. of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. Supported in part by NSF ITR Award CNS-0426683 and NSF Award CNS-0626636.

I. Introduction

The well-known Lovász Local Lemma (LLL) [14] is a powerful probabilistic approach to prove the existence of certain combinatorial structures. Its diverse range of applications include breakthroughs in packet-routing [19], a variety of theorems in graph-coloring including list coloring, frugal coloring, total coloring, and coloring graphs with lower-bounded girth [23], as well as a host of other applications where probability appears at first sight to have no role [4]. Furthermore, almost all known applications of the LLL have no alternative proofs known. While the original LLL was non-constructive – it was unclear how the existence proofs could be turned into polynomial-time algorithms – a series of works [1], [9], [13], [22]–[26], [28] beginning with Beck [9] and culminating with the breakthrough of Moser & Tardos (MT) [25] have led to efficient algorithmic versions for most such proofs. However, there are several LLL applications to which these approaches inherently cannot apply; our work makes progress toward bridging this gap, by uncovering and exploiting new properties of [25]. We also obtain what are, to our knowledge, the first algorithmic applications of the LLL where a few of the bad events have to happen, and where we aim to keep the number of these small.

Essentially all known applications of the LLL use the following framework. Let $\mathcal{P}$ be a collection of n mutually independent random variables $\{P_1, P_2, \ldots, P_n\}$, and let $\mathcal{A} = \{A_1, A_2, \ldots, A_m\}$ be a collection of m ("bad") events, each determined by some subset of $\mathcal{P}$. The LLL (Theorem I.1) shows sufficient conditions under which, with positive probability, none of the events A_i holds: i.e., that there is a choice of values for the variables in $\mathcal{P}$ (corresponding to a discrete structure such a suitable coloring of a given graph) that avoids all the A_i. Under these same sufficient conditions, MT shows the following very simple algorithm to make such a choice: (i) initially choose the P_i independently from

their given distributions; (ii) *while* the current assignment to $\mathcal{P}$ does not avoid all the A_i, *repeat*: arbitrarily choose a currently-true A_i, and resample, from their product distribution, the variables in $\mathcal{P}$ on which A_i depends. The amazing aspect of MT is that the expected number of resamplings is small [25]: at most $\mathrm{poly}(n, m)$ in all known cases of interest. However, there are two problems with implementing MT, that come up in some applications of the LLL:

(a) the number of events m can be superpolynomial in the number of variables n; this can result in a superpolynomial running time in the "natural" parameter n [1]; and, even more seriously,

(b) given an assignment to $\mathcal{P}$, it can be computationally hard (e.g., NP-hard or yet-unknown to be in polynomial time) to either certify that no A_i holds, or to output an index i such that A_i holds.

Since detection and resampling of a currently-bad event is the seemingly unavoidable basic step in the MT algorithm, these applications seemed far out of reach. We deal with a variety of applications wherein (a) and/or (b) hold, and develop Monte Carlo (and in many cases, RNC) algorithms whose running time is polynomial in n: some of these applications involve a small loss in the quality of the solution. (We loosely let "RNC algorithms" denote randomized parallel algorithms that use $\mathrm{poly}(n)$ processors and run in $\mathrm{polylog}(n)$ time, to output a correct solution with high probability.) First we show that the MT algorithm needs only $O(n^2 \log n)$ many resampling steps in all applications that are known (and in most cases $O(n \cdot \mathrm{polylog}(n))$), even when m is superpolynomial in n. This makes those applications constructive that allow an *efficient* implicit representation of the bad events (in very rough analogy with the usage of the ellipsoid algorithm for convex programs with exponentially many constraints but with good separation oracles). Still, most of our applications have problem (b). For these cases, we introduce a new proof-concept based on the *(conditional) LLL-distribution* – the distribution D on $\mathcal{P}$ that one obtains when conditioning on no A_i happening. Some very useful properties are known for D [4]: informally, if B depends "not too heavily" on the events in $\mathcal{A}$, then the probability placed on B by D is "not much more than" the unconditional probability $\mathsf{Pr}\{B\}$: at most $f_{\mathcal{A}}(B) \cdot \mathsf{Pr}\{B\}$ (see (3)). Such bounds in combination with further probabilistic analysis can be used to give interesting (nonconstructive) results. Our next main contribution is that the MT algorithm has an output distribution (say D') that "approximates" the LLL-distribution D: in that for every B, the *same* upper bound $f_{\mathcal{A}}(B) \cdot \mathsf{Pr}\{B\}$ as above, holds in D' as well. This can be used to make probabilistic proofs that use the LLL-condition constructive.

Problem (b), in all cases known to us, comes from problem (a): it is easy to test if any *given* A_i holds currently (e.g., if a given subset of vertices in a graph is a clique), with the superpolynomiality of m being the apparent bottleneck. To circumvent this, we develop our third main contribution: the general Theorem III.4 that is simple and directly applicable in all LLL instances that allow a small slack in the LLL's sufficient conditions. This theorem proves that a small $\mathrm{poly}(n)$-sized core-subset of the events in $\mathcal{A}$ can be selected and avoided efficiently using the MT algorithm. Using the LLL-distribution and a simple union bound over the non-core events, we get efficient (Monte Carlo and/or RNC) algorithms for these problems.

We develop two types of applications, as sketched next. Due to page limitations, we omit several details here; please see the extended version of this paper for these details and all omitted proofs [17].

A. Applications that avoid all bad events

A list of four applications follows; all of these have problem (a), and all but the acyclic-coloring application have problem (b). Most such results have RNC versions as well.

The Santa Claus Problem: The Santa Claus problem is the restricted assignment version of the max-min allocation problem of indivisible goods. The Santa Claus has s gifts that need to be distributed among t children. Each child has a utility for each gift, which is either 0 or some given p_j for gift j. The objective is to assign each gift to some child, so that the minimum total utility received by any child is maximized. This problem has received much attention recently [5]–[8], [11], [15]. The problem is NP-Hard and the best-known approximation algorithm due to Bansal and Sviridenko [7] achieves an approximation factor of $O(\frac{\log\log s}{\log\log\log s})$ by rounding a certain configuration LP. Later, Feige in [15] and Asadpour, Feige and Saberi in [5] showed that the integrality gap of the configuration LP is a constant. These results were obtained using two different *nonconstructive* approaches, and left the question of a constant-factor approximation algorithm open. This made the Santa Claus problem one of the rare instances

[1] n is the parameter of interest since the output we seek is one value for each of $P_1, P_2, \ldots, P_n$.

[16] in which the proof of an integrality gap did not result in a approximation algorithm with the same ratio. We resolve this by making the nonconstructive LLL-based proof of Feige [15] constructive (Section IV) and giving the first constant-factor approximation algorithm for the Santa Claus problem.

Please see the full version [17] for details about the other three applications: *Non-repetitive Coloring of Graphs*, *General Ramsey-Type Graphs*, and *Acyclic Edge-Coloring*.

B. Applications that avoid many bad events

Many settings require "almost all" bad events to be avoided, and not necessarily all; e.g., consider MAX-SAT as opposed to SAT. However, in the LLL context, essentially the only known general applications were "all or nothing": either the LLL's sufficient conditions hold, and we are able to avoid all bad events, or the LLL's sufficient conditions are violated, and the only known bound on the number of bad events is the trivial one given by the linearity of expectation (which does not exploit any "almost-independence" of the bad events, as does the LLL). This situation is even more pronounced in the algorithmic setting. We take what are, to our knowledge, the first steps in this direction, interpolating between these two extremes.

While our discussion here holds for all applications of the symmetric LLL, let us take MAX-k-SAT as an illustrative example. (The LLL is defined in Section I-C, but let us recall its well-known "symmetric" special case: in the setting of MT with $\mathcal{P}$ and A as defined near the beginning of Section I, if $\mathsf{Pr}\{A_i\} \leq p$ and A_i depends on at most d other A_j for all i, then $e \cdot p \cdot (d+1) \leq 1$ suffices to avoid all the A_i.) Recall that in MAX-k-SAT, we have a CNF formula on n variables, with m clauses each containing exactly k literals; as opposed to SAT, where we have to satisfy all clauses, we aim to maximize the number of satisfied clauses here. The best general upper-bounds on the number of "violated events" (unsatisfied clauses) follow from the probabilistic method, where each variable is set to True or False uniformly at random and independently. On the one hand, the linearity of expectation yields that the expected number of unsatisfied clauses is $m \cdot 2^{-k}$ (with a derandomization using the method of conditional probabilities). On the other hand, if each clause shares a variable with at most $2^k/e - 1$ other clauses, a simple application of the symmetric LLL shows that all clauses can be satisfied (and made constructive using MT). No interpolation between these was known before; among other results, we show that if each clause shares a variable with at most $\sim \alpha 2^k/e$ other clauses for $1 < \alpha < e$, then we can efficiently construct an assignment to the variables that violates at most $(e \ln(\alpha)/\alpha + o(1)) \cdot m \cdot 2^{-k}$ clauses for large k. (This is better than the linearity of expectation iff $\alpha < e$: it is easy to construct examples with $\alpha = e$ where one cannot do better than the linearity of expectation. See [3] for the fixed-parameter tractability of MAX-k-SAT above $(1 - 2^{-k})m$ satisfied clauses.)

The above and related results for applications of the symmetric LLL, follow from the connection to the "further probabilistic analysis using the remaining randomness of LLL-distributions" that we alluded to above.

C. Preliminaries & Algorithmic Framework

We follow the general algorithmic framework of the Local Lemma due to MT. As in our description at the beginning of Section I, let $\mathcal{P}$ be a finite collection of mutually independent random variables $\{P_1, P_2, \ldots, P_n\}$ and let $\mathcal{A} = \{A_1, A_2, \ldots, A_m\}$ be a collection of events, each determined by some subset of $\mathcal{P}$. For any event B that is determined by a subset of $\mathcal{P}$ we denote the smallest such subset by $\mathrm{vbl}(B)$. For any event B that is determined by the variables in $\mathcal{P}$, we furthermore write $\Gamma(B) = \Gamma_{\mathcal{A}}(B)$ for the set of all events $A \neq B$ in $\mathcal{A}$ with $\mathrm{vbl}(A) \cap \mathrm{vbl}(B) \neq \emptyset$.[2] This neighborhood relation induces the following standard *dependency graph* or *variable-sharing graph* on $\mathcal{A}$: For the vertex set $\mathcal{A}$ let $G = G_{\mathcal{A}}$ be the undirected graph with an edge between events $A, B \in \mathcal{A}$ iff $A \in \Gamma(B)$. We often refer to events in $\mathcal{A}$ as *bad events* and want to find a point in the probability space, or equivalently an assignment to the variables $\mathcal{P}$, wherein none of the bad events happen. We call such an assignment a **good assignment**.

With these definitions the general ("asymmetric") version of the LLL simply states:

Theorem I.1 (Asymmetric Lovász Local Lemma)**.** *With $\mathcal{A}, \mathcal{P}$ and Γ defined as above, if there exists an assignment of reals $x : \mathcal{A} \to (0,1)$ such that*

$$\forall A \in \mathcal{A} : \mathsf{Pr}\{A\} \leq x(A) \prod_{B \in \Gamma(A)} (1 - x(B)); \qquad (1)$$

then the probability of avoiding all bad events is at least $\Pi_{A \in \mathcal{A}}(1 - x(A)) > 0$ and thus there exists a good assignment to the variables in $\mathcal{P}$.

[2] It is important to note that B itself may not be an element of $\mathcal{A}$.

We study several LLL instances where the number of events to be avoided, m, is super-polynomial in n; our goal is to develop algorithms whose running time is polynomial in n which is also the size of the output - namely a good assignment of values to the n variables. We introduce a key parameter:

$$\delta := \min_{A \in \mathcal{A}} x(A) \prod_{B \in \Gamma(A)} (1 - x(B)). \tag{2}$$

Note that without loss of generality $\delta \leq \frac{1}{4}$ because otherwise all $A \in \mathcal{A}$ are independent, i.e., defined on disjoint sets of variables. Indeed if $\delta > \frac{1}{4}$ and there is an edge in G between $A \in \mathcal{A}$ and $B \in \mathcal{A}$ than we have $\frac{1}{4} > x(A)(1 - x(B))$ and $\frac{1}{4} > x(B)(1 - x(A))$, i.e., $\frac{1}{4} \cdot \frac{1}{4} > x(A)(1 - x(A)) \cdot x(B)(1 - x(B))$ which is a contradiction because $x(1 - x) \leq \frac{1}{4}$ for all x.

We allow our algorithms to have a running-time that is polynomial in $\log(1/\delta)$; in all applications known to us, $\delta \geq \exp(-O(n \log n))$, and hence, $\log(1/\delta) = O(n \log n)$. In fact because δ is an upper bound for $\min_{A \in \mathcal{A}} P(A)$ in any typical encodings of the domains and the probabilities of the variables, $\log(1/\delta)$ will be at most linear in the size of the input or the output.

II. LLL-Distribution

When trying to make the non-constructive Lovász Local Lemma constructive, the following straightforward approach comes to mind: draw a random sample for the variables in $\mathcal{P}$ until one is found that avoids all bad events. If the LLL-conditions are met, this rejection-sampling algorithm certainly always terminates but because the probability of obtaining a good assignment is typically exponentially small, it takes an expected exponential number of resamplings and is therefore inefficient. While the celebrated results of [24], [25] are much more efficient, the above rejection-sampling method has a major advantage: it does not just produce an arbitrary assignment but provides a randomly chosen assignment from the distribution obtained when one conditions on no bad event happening. In the following, we call this distribution the *LLL-distribution* or *conditional LLL-distribution*.

The LLL-conditions and further probabilistic analysis can be a powerful tool to obtain new results (constructive or otherwise) like the constructive one in Section V. The following is a well-known bound on the probability $\mathsf{Pr}_D\{B\}$ that the LLL-distribution D places on *any* event B that is determined by variables in $\mathcal{P}$ (its proof is an easy extension of the standard LLL-proof [4]):

Theorem II.1. *If the LLL-conditions from Theorem I.1 are met, then the LLL-distribution D is well-defined. For any event B that is determined by $\mathcal{P}$, the probability $\mathsf{Pr}_D\{B\}$ of B under D equals:*

$$\mathsf{Pr}\left\{B \mid \bigwedge_{A \in \mathcal{A}} \overline{A}\right\} \leq \mathsf{Pr}\{B\} \prod_{C \in \Gamma(B)} (1 - x_C)^{-1} \tag{3}$$

here, $\mathsf{Pr}\{B\}$ is the probability of B holding under a random choice of $P_1, P_2, \ldots, P_n$.

The fact that the probability of an event B does not change much in the conditional LLL-distribution when B does not depend on "too many" $C \in \mathcal{A}$ is used a lot in the rest of the paper.

More importantly, the following theorem states that the output distribution D' of the MT-algorithm approximates the LLL-distribution D and has the very nice property that it essentially also satisfies (3):

Theorem II.2. *Suppose there is an assignment of reals $x : \mathcal{A} \to (0, 1)$ such that (1) holds. Let B be any event that is determined by $\mathcal{P}$. Then, the probability that B was true* at least once *during the execution of the MT algorithm on the events in $\mathcal{A}$, is at most $\mathsf{Pr}\{B\} \cdot (\prod_{C \in \Gamma(B)}(1 - x_C))^{-1}$. In particular, the probability of B in the output distribution of MT obeys this upper-bound.*

Using this theorem we can view the MT algorithm as an *efficient* way to obtain a sample that comes approximately from the conditional LLL-distribution. This efficient sampling procedure makes it possible to make proofs using the conditional LLL-distribution constructive and directly convert them into algorithms. All constructive results of this paper are based on Theorem II.2 and demonstrate this idea.

III. LLL Applications with Super-Polynomially Many Bad Events

In several applications of the LLL, the number of bad events is super-polynomially larger than the underlying variables. In these cases we aim for an algorithm that still runs in time polynomial in the number of variables, and it is not efficient to have an explicit representation of all bad events. Surprisingly, Theorem III.1 shows that the number of resamplings done by the MT algorithm remains quadratic and in most cases even near linear in the number of variables n.

Theorem III.1. *Suppose there is an $\epsilon \in [0,1)$ and an assignment of reals $x : \mathcal{A} \to (0,1)$ such that:*

$$\forall A \in \mathcal{A} : \mathsf{Pr}\{A\} \leq (1-\epsilon)x(A) \prod_{B \in \Gamma(A)} (1-x(B)).$$

With δ denoting $\min_{A \in \mathcal{A}} x_A$, we have

$$T := \sum_{A \in \mathcal{A}} x_A \leq n \log(1/\delta). \tag{4}$$

Furthermore:

1) *if $\epsilon = 0$, then the expected number of resamplings done by the MT algorithm is at most $v_1 = T \max_{A \in \mathcal{A}} \frac{1}{1-x(A)}$, and for any parameter $\lambda \geq 1$, the MT algorithm terminates within λv_1 resamplings with probability at least $1 - 1/\lambda$.*
2) *if $\epsilon > 0$, then the expected number of resamplings done by the MT algorithm is at most $v_2 = O(\frac{n}{\epsilon} \log \frac{T}{\epsilon})$, and for any parameter $\lambda \geq 1$, the MT algorithm terminates within λv_2 resamplings with probability $1 - \exp(-\lambda)$.*

Proof: The main idea of relating the quantity T to n and δ is to use: (i) the fact that the variable-sharing graph G is dense, and (ii) the nature of the LLL-conditions which force highly connected events to have small probabilities and x-values. To see that G is dense, consider for any variable $P \in \mathcal{P}$ the set of events

$$\mathcal{A}_P = \{A \in \mathcal{A} | P \in \mathrm{vbl}(A)\},$$

and note that these events form a clique in G.

Let us first prove the bound on T. To do so, we fix any $P \in \mathcal{P}$ and show that $\sum_{B \in \mathcal{A}_P} x_B \leq \log(1/\delta)$, which will clearly suffice. Recall from the discussion following (2) that we can assume w.l.o.g. that $\delta \leq \frac{1}{4}$. If $|A_P| = 1$, then of course $\sum_{B \in \mathcal{A}_P} x_B \leq 1 \leq \log(1/\delta)$. If $|A_P| > 1$, let $A \in \mathcal{A}_P$ have the smallest x_A value. Note that by definition

$$\delta \leq x_A \prod_{B \in \mathcal{A}_P \setminus A} (1-x_B) = \frac{x_A}{1-x_A} \prod_{B \in \mathcal{A}_P} (1-x_B).$$

If $x_A \leq 1/2$, then $\delta \leq \prod_{B \in \mathcal{A}_P}(1-x_B) \leq e^{-\sum_{B \in \mathcal{A}_P} x_B}$, and we get $\sum_{B \in \mathcal{A}_P} x_B \leq \ln(1/\delta) < \log(1/\delta)$ as required. Otherwise, if $x_A > 1/2$, let $B_1 \in \mathcal{A}_P \setminus A$. Then,

$$\begin{aligned} \delta &\leq x_A \cdot \prod_{B \in \mathcal{A}_P \setminus A} (1-x_B) \\ &= x_A(1-x_{B_1}) \prod_{B \in \mathcal{A}_P \setminus (A \cup B_1)} (1-x_B) \\ &\leq x_A(1-x_{B_1}) e^{-\sum_{B \in \mathcal{A}_P \setminus (A \cup B_1)} x_B}. \end{aligned} \tag{5}$$

Now it can be argued (see the full version [17] for details) that for $1/2 \leq x_A \leq x_{B_1} \leq 1$,

$$x_A(1-x_{B_1}) \leq e^{-(x_A + x_{B_1})}. \tag{6}$$

So we get

$$x_A(1-x_{B_1}) e^{-\sum_{B \in \mathcal{A}_P \setminus (A \cup B_1)} x_B)} \leq e^{-\sum_{B \in \mathcal{A}_P} x_B};$$

using this with (5), we obtain $\sum_{B \in \mathcal{A}_P} x_B \leq \ln(1/\delta) < \log(1/\delta)$ as desired.

Given the bound on T, part (1) follows directly from the main theorem of [25] and by a simple application of Markov's inequality.

Part (2) now also follows from [25]. In section 5 of [25] it is shown that saving an $1-\epsilon$ factor in the probability of every resampling step implies that with high probability, no witness tree of size $\Omega(\frac{1}{\epsilon} \log \sum_{A \in \mathcal{A}} \frac{x_A}{1-x_A})$ occurs. This easily implies that none of the n variables can be resampled more often. It is furthermore shown that without loss of generality all x-values can be assumed to be bounded away from 1 by at least $O(\epsilon)$. This simplifies the upper bound on the expected running time to $n \cdot O(\frac{1}{\epsilon} \log \frac{T}{\epsilon})$. ∎

As mentioned following the introduction of δ in (2), $\log(1/\delta) \leq O(n \log n)$ in all applications known to us, and is often even smaller.

a) Remarks: 1. The $\max_{A \in \mathcal{A}} \frac{1}{1-x(A)}$ factor in the running time of part (1) of Theorem III.1 corresponds to the expected number of times the event A gets resampled until one satisfying assignment to its variables is found. It is obviously unavoidable for an algorithm that has only black-box resampling and evaluation access to the events. If one alters the algorithm to pick a random assignment that satisfies A (which can for example be computed using rejection sampling, taking an expected $\Theta(\frac{1}{1-x(A)})$ trials each time), this factor can be avoided.

2. The estimation $T = \sum_{A \in \mathcal{A}} x_A = O(n \log 1/\delta)$ is tight and can be achieved, e.g., by having an isolated event with constant probability for each variable. In many cases with $\log 1/\delta = \omega(\log n)$ it is nevertheless an overestimate, and in most cases the running time is $O(n \log n)$ even for $\epsilon = 0$.

While Theorem III.1 gives very good bounds on the running time of MT even for applications with $\Omega(n) \leq m \leq \mathrm{poly}(n)$ many events, it unfortunately often fails to be directly applicable when m becomes super-polynomial in n. The reason is that maintaining bad events implicitly and running the resampling process requires an efficient

way to find violated events. In many examples like those discussed in Section I (except acyclic edge-coloring) with super-polynomially many events, finding violated events or even just verifying a good assignment is not known to be in polynomial time (often even provably NP-hard). To capture the sets of events for which we can run the MT algorithm efficiently we use the following definition:

Definition III.2. ***(Efficient verifiability)*** *A set $\mathcal{A}$ of events that are determined by variables in $\mathcal{P}$ is* efficiently verifiable *if, given an arbitrary assignment to $\mathcal{P}$, we can efficiently find an event $A \in \mathcal{A}$ that holds or detect that there is no such event.*

Because many large $\mathcal{A}$ of interest are not efficiently verifiable a direct application of the MT-algorithm is not efficient. Nevertheless we show in the rest of this section that using the randomness in the output distribution of the MT-algorithm characterized by Theorem II.2, it is still practically always possible to obtain efficient Monte Carlo algorithms that produce a good assignment with high probability.

The main idea is to judiciously select an efficiently verifiable *core subset* $\mathcal{A}' \subseteq \mathcal{A}$ of bad events and apply the MT-algorithm to it. Essentially instead of looking for violated events in $\mathcal{A}$ we only resample events from $\mathcal{A}'$ and terminate when we can not find one such violated event. The non-core events will have small probabilities and will be sparsely connected to core events and as such their probabilities in the LLL-distribution and therefore also the output distribution of the algorithm does not blow up by much. There is thus hope that the non-core events remain unlikely to happen even though they were not explicitly fixed by the algorithm. Theorem III.3 the proof of which can be found in the full version [17] shows that if the LLL-conditions are fulfilled for $\mathcal{A}$ then a non-core event $A \in \mathcal{A}'$ is violated in the produced output with probability at most x_A. This makes the success probability of such an approach at least $1 - \sum_{A \in \mathcal{A}'} x_A$.

Theorem III.3. *Let $\mathcal{A}' \subseteq \mathcal{A}$ be an efficiently verifiable core subset of $\mathcal{A}$. If there is an $\epsilon \in [0,1)$ and an assignment of reals $x : \mathcal{A} \to (0,1)$ such that:*

$$\forall A \in \mathcal{A} : \mathsf{Pr}\{A\} \leq (1-\epsilon)x(A) \prod_{B \in \Gamma(A) \cap \mathcal{A}'} (1 - x(B)).$$

Then the modified MT-algorithm can be efficiently implemented with an expected number of resamplings according to Theorem III.1. The algorithm furthermore outputs a good assignment with probability at least $1 - \sum_{A \in \mathcal{A} \setminus \mathcal{A}'} x_A$.

While the concept of an efficiently verifiable core is easy to understand, it is not clear how often and how such a core can be found. Furthermore having such a core is only useful if the probability of the non-core events is small enough to make the failure probability, which is based on the union bound over those probabilities, meaningful. The following main theorem shows that in all applications that can tolerate a small "exponential" ϵ-slack as introduced by [12] , finding such a good core is straightforward:

Theorem III.4. *Suppose $\log 1/\delta \leq \mathrm{poly}(n)$. Suppose further that there is a fixed constant $\epsilon \in (0,1)$ and an assignment of reals $x : \mathcal{A} \to (0, 1-\epsilon)$ such that:*

$$\forall A \in \mathcal{A} : \mathsf{Pr}\{A\}^{1-\epsilon} \leq x(A) \prod_{B \in \Gamma(A)} (1 - x(B)).$$

Then for every $p \geq \frac{1}{\mathrm{poly}(n)}$ the set $\{A_i \in \mathcal{A} : \mathsf{Pr}\{A_i\} \geq p\}$ has size at most $\mathrm{poly}(n)$, and is thus essentially always an efficiently verifiable core subset of $\mathcal{A}$. If this is the case, then there is a Monte Carlo algorithm that terminates after $O(\frac{n}{\epsilon^2} \log \frac{n}{\epsilon^2})$ resamplings and returns a good assignment with probability at least $1 - n^{-c}$, where $c > 0$ is any desired constant.

Proof: For a probability $p = 1/\mathrm{poly}(n)$ to be fixed later we define $\mathcal{A}'$ as the set of events with probability at least p. Recall from Theorem III.1 that $\sum_{A \in \mathcal{A}} x_A \leq O(n \log(1/\delta))$. Since $x_A \geq p$ for $A \in \mathcal{A}'$, we get that $|\mathcal{A}'| \leq O(n \log(1/\delta)/p) = \mathrm{poly}(n)$. By assumption $\mathcal{A}'$ is efficiently verifiable and we can run the modified resampling algorithm with it.

For every event we have $\mathsf{Pr}\{A\} \leq x_A < 1 - \epsilon$ and thus get an $(1-\epsilon)^\epsilon = (1 - \Theta(\epsilon^2))$-slack; therefore Theorem III.1 applies and guarantees that the algorithm terminates with high probability after $O(\frac{n}{\epsilon^2} \log \frac{n}{\epsilon^2})$ resamplings.

To prove the failure probability note that for every non-core event $A \in \mathcal{A} \setminus \mathcal{A}'$ the LLL-conditions with the "exponential ϵ-slack" provide an extra multiplicative $p^{-\epsilon}$ factor over the LLL-conditions in Theorem III.1. While $\sum_{A \in \mathcal{A} \setminus \mathcal{A}'} x(A) \leq \sum_{A \in \mathcal{A}} x_A = T = \mathrm{poly}(n)$ holds in this setting we can make this union bound at most n^{-c} by choosing $p = n^{-O(1/\epsilon)}$ small enough. Now as in Lemma III.3 we get that we fail with probability at most n^{-c} on non-core events while safely avoiding the core. This completes the proof of the theorem. ∎

The last theorem nicely completes this section; it shows

that in practically all applications of the general LLL it is possible to obtain a fast Monte Carlo algorithm with arbitrarily high success probability. The conditions of Theorem III.4 are very easy to check and are usually directly fulfilled. That is, in all LLL-based proofs (with a large number of events A_i) known to us, the set of high-probability events forms a polynomial-sized core that is trivially efficiently verifiable, e.g., by exhaustive enumeration. Theorem III.4 makes these proofs constructive without further complicated analysis. Only in cases where the LLL-condition is used are adjustments in the bounds needed, to respect the ϵ-slack.

Note that the failure probability can be made an arbitrarily small inverse polynomial. This is important since for problems with non-efficiently verifiable solutions, the success probability of Monte Carlo algorithms cannot be boosted using standard amplification.

In all applications known to us, the core above has further nice structure: usually the probability of an event A_i is exponentially small in the number of variables it depends on. Thus, each event in the core only depends on $O(\log n)$ many A_i, and hence is usually trivial to enumerate. This makes the core efficiently verifiable, even when finding a general violated event in $\mathcal{A}$ is hard. The fact that the core consists of polynomially many events with usually logarithmically many variables each, makes it often even possible to enumerate the core in parallel and to evaluate each event in parallel. If this is the case, one can get an RNC algorithm by first building the dependency graph on the core and then computing a maximal independent set (MIS) of violated events in each round, using MIS algorithms such as [2], [21]. Using the proof of Theorem III.1, it is easy to see that only logarithmically many rounds of resampling these events are needed.

We also note that although the derandomization of [12] also only requires an exponential ϵ-slack in the LLL-conditions, applying the techniques of [12] seems difficult when m is superpolynomial.

IV. A Constant-Factor Approximation Algorithm for the Santa Claus Problem

In the max-min allocation problem, there is a set $\mathcal{C}$ of n items, and m players. The value (utility) of item j to player i is $p_{i,j} \geq 0$. An item can be assigned to only one player. If a player i receives a subset of the items $S_i \subseteq \mathcal{C}$, then the total valuation of the items received by i is $\sum_{j \in S_i} p(i,j)$. The goal is to maximize the minimum total valuation of the items received by any player, that is, to maximize $\min_i \sum_{j \in S_i} p(i,j)$. (The "minmax" version of this "maxmin" problem is the classical problem of makespan minimization in unrelated parallel machine scheduling [20].) This problem has received much attention recently [5]–[8], [11], [15], [27].

A restricted version of max-min allocation is where each item has an intrinsic value, and where for every player i, $p_{i,j}$ is either p_j or 0. This is known as the Santa Claus problem. The Santa Claus problem is NP-hard and no efficient approximation algorithm better than $1/2$ can be obtained unless $P = NP$ [10]. Bansal and Sviridenko [7] considered a linear-programming (LP) relaxation of the problem known as the configuration LP, and showed how to round this LP to obtain an $O(\log\log\log m/\log\log m)$-approximation algorithm for the Santa Claus problem. They also showed a reduction to a crisp combinatorial problem, a feasible solution to which implies a constant-factor integrality gap for the configuration LP.

Subsequently, Feige [15] showed that the configuration LP has a constant integrality gap. Normally such a proof immediately gives a constant-factor approximation algorithm that rounds an LP solution along the line of the integrality-gap proof. In this case Feige's proof could not be made constructive because it was heavily based on repeated reductions that apply the asymmetric version of the LLL to exponentially many events. Due to this unsatisfactory situation, the Santa Claus problem was the first on a list of problems reported in the survey "Estimation Algorithms versus Approximation Algorithms" [16] for which a constructive proof would be desirable. Using a completely different approach, Asadpour, Feige and Saberi [5] could show that the configuration LP has an integrality gap of at most $\frac{1}{5}$. Their proof uses local-search and hypergraph matching theorems of Haxell [18]. Haxell's theorems are again highly nonconstructive and the stated local-search problem is not known to be efficiently solvable and in fact the conclusion in [5] suggests that finding a local optimum could be potentially PLS-complete. Thus this second nonconstructive proof still left the question of a constant-factor approximation algorithm open.

We now sketch how Theorem III.4 can be used to easily and directly constructivize the LLL-based proof of Feige [15], giving the first constant-factor approximation algorithm for the Santa Claus problem. A complete discussion of the proof can be found in the full version [17].

A. Reduction to (k, l, β) systems and Feige's proof

As explained above, Bansal and Sviridenko [7] show how to reduce the question of finding a good solution to the Santa Claus problem to solving the following kind of combinatorial matching called (k, l, β) system:

A (k, l, β) system consists of p groups of l players. Each player values a set of k items and each item is valued by at most βl players (we will always have $1 \leq \beta \leq 3$). A (k, l, β) system is γ-good if there is a choice of one player per group and $\lfloor \gamma k \rfloor$ items valued by each such player such that all chosen items are disjoint.

The following theorem establishes the connection to the Santa Claus problem:

Theorem IV.1 ([7]). *If there exists a $\gamma = \Theta(1)$ such that for every (k, l, β) system with $\beta = \Theta(1)$ a γ-good solution can be found than this can be turned into a solution for the Santa Claus problem that is within a constant-factor of the optimum.*

Feige shows that finding a good matching of items to one player from each group is always possible by systematically reducing either the number k of items valued by each player or the number l of players per group until both are small enough constants. For this much simpler situation when there are only a constant number of players in each group and each player only values a constant number of items the following lemma asserts a constant goodness.

Lemma IV.2 (Lemma 2.1 and 2.2 of [15]). *Every (k, l, β) system a γ-good solution for γ satisfying, $\gamma = \frac{1}{k}$ or $\gamma k = \lfloor \frac{k}{\lceil \beta l \rceil} \rfloor$ can be found efficiently.*

The reduction of a (k, l, β) system to an equivalent system with constant k and l involves two main lemmas, which we refer to as *Reduce-l* lemma and *Reduce-k* lemma respectively.

Lemma IV.3 (Lemma 2.3 of [15], Reduce-l). *For $l > c$ (c a sufficiently large constant), every (k, l, β) system with $k \leq l$ can be transformed into a (k, l', β') system with $l' \leq \log^5 l$ and $\beta' \leq \beta(1 + \frac{1}{\log l})$.*

Lemma IV.4 (Lemma 2.4 of [15], Reduce-k). *Every (k, l, β) system with $k \geq l \geq c$ can be transformed into a (k', l, β) system with $k' \leq \frac{k}{2}$ and with the following additional property: if the original system is not γ-good, then the new system is not γ'-good for $\gamma' = \gamma(1 + \frac{3 \log k}{\sqrt{\gamma k}})$. Conversely, if the new system is γ'-good, then the original system was γ-good.*

Starting from the original system, as long as $l > c$, Lemma Reduce-l is applied and when $k \geq l$, Lemma Reduce-k is applied. In this process β grows at most by a factor of 2. Thus at the end, l and k are constants and so is β. Then applying Lemma IV.2 finishes the proof.

B. Randomized Algorithm for (k, l, β) systems

The two main steps required for obtaining an algorithm that produces a γ-good solution for every (k, l, β) system are: (i) showing a constructive procedure to obtain the reduced system through Lemmas Reduce-l and Reduce-k, and (ii) mapping the solution of the final reduced system back to the original system. We now discuss these in some more detail.

The proof for Lemma Reduce-l only applies the symmetric version of the LLL and can easily be made constructive using the MT-algorithm [25]. As Feige points out, Lemma Reduce-k is more problematic: "the main source of difficulty in this respect is Lemma 2.4, because there the number of bad events is exponential in the problem size, and moreover, there are bad events that involve a constant fraction of the random variables." In the following we recapitulate the proof for the Reduce-k lemma and show how Theorem III.4 can be directly applied to circumvent these problems and make it constructive.

1) Making Lemma Reduce-k Constructive: The random experiment used to prove Lemma Reduce-k selects each item independently at random with probability $\frac{1}{2}$. To characterize the bad events in the application of the LLL, we need a structural lemma from [15]. Construct a graph on the players, where there is an edge between two players if they have an item they both value. A collection of players is said to be connected if and only if the subgraph induced by this collection is connected.

We consider two types of bad events:

1) B_1: some player has less than $k' = \left(1 - \frac{\log k}{\sqrt{k}}\right) \frac{k}{2}$ items surviving; and
2) B_i for $i \geq 2$: there is a connected collection of i players from distinct groups whose union of items valued originally contained at most $i\gamma k$ items, of which more than $i\delta' \frac{k}{2}$ items survive, where $\delta' = \gamma \left(1 + \frac{\log k}{\sqrt{\gamma k}}\right)$.

If none of the above bad events happen, then we can consider the first k' items from each set and yet the second type of bad events do not happen. These events are chosen such that γ'-goodness ($\gamma' = \delta' \frac{k}{2} \frac{1}{k} \leq \gamma \left(1 + \frac{\log k}{\sqrt{\gamma k}}\right)$)

of the new system certifies that the original system was γ good. That this is indeed the case follows directly from Hall's theorem:

Lemma IV.5 (Lemma 2.7 of [15])**.** *Consider a collection of n sets and a positive integer q.*

1) *If for some $1 \leq i \leq n$, there is a connected subcollection of i sets whose union contains less than iq items, then there is no choice of q items per set such that all items are distinct.*
2) *If for every i, $1 \leq i \leq n$, the union of every connected subcollection of i sets contains at least iq (distinct) items, then there is a choice of q items per set such that all items are distinct.*

Feige showed in [15] that for bad events of type $B_i, i \geq 1$, taking $x_i = 2^{-10i \log k}$ is sufficient to satisfy the condition (1) of the asymmetric LLL. More precisely, suppose we define, for any bad event $B \in \bigcup_{i \geq 1} B_i$, $\Gamma(B)$ to be as in Section I-C: i.e., $\Gamma(B)$ is the set of all bad events $A \neq B$ such that A and B both depend on at least one common random variable in our "randomly and independently selecting items" experiment. Then, it is shown in [15] that with the choice $x_i = 2^{-10i \log k}$ for all events in B_i, we have for all $i \geq 1$ and for all $B \in B_i$,

$$\mathsf{Pr}\{B\} \leq 2^{-20i \log k} \leq x_i \prod_{j \geq 1} \prod_{A \in (B_j \cap \Gamma(B))} (1 - x_j) \quad (7)$$

Thus by the LLL, there exists an assignment that avoids all the bad events. However, no efficient construction was known here, and as Feige points out, "the main source of difficulty in this respect is Lemma 2.4, because there the number of bad events is exponential in the problem size, and moreover, there are bad events that involve a constant fraction of the random variables." Our Theorem III.4 again directly makes this proof constructive and gives an efficient Monte Carlo algorithm for producing a reduce-k system with high probability:

Lemma IV.6. *There is a Monte Carlo algorithm that produces a valid reduce-k system with probability at least $1 - 1/m^2$.*

2) Mapping the solution of the final reduced system back: Please see [17] for the proof of this mapping, and of Lemma IV.6.

We can easily check if the algorithm finally produces a good solution, thus leading to a Las Vegas algorithm for our problem:

Theorem IV.7. *There exists a constant $\alpha > 0$ and a randomized algorithm for the Santa Claus problem that runs in expected polynomial time and assigns items of total valuation at least $\alpha \cdot \mathrm{OPT}$ to each player.*

V. Beyond the LLL Threshold

This section sketches another application of using the properties of the conditional LLL-distribution introduced in Section II in a slightly different way. While all results presented so far rely on a union bound over events in the LLL-distribution we use here the linearity of expectation for further probabilistic analysis of events in the LLL-distribution. This already leads to new non-constructive results. Similar to the other proofs involving the LLL-distribution in this paper this upper bound can be made constructive using Theorem II.2. Considering that the LLL-distribution approximately preserves other quantities such as higher moments, we expect that there is much more room to use more sophisticated probabilistic tools like concentration bounds to give both new non-constructive and constructive existence proofs of discrete structures with additional strong properties.

The setting we want to concentrate on here is when a set of bad events is given from which not necessarily all but as many as possible events are to be avoided. The exemplifying application is the well known MAX-k-SAT problem which in contrast to k-SAT asks not for a satisfying assignment of a k-CNF formula but for an assignment that violates as few clauses as possible. Given a k-CNF formula with m clauses a random assignment to its variables violates each clause with probability 2^{-k} and thus using linearity of expectation it is easy to find an assignment that violates at most $m2^{-k}$ clauses. If on the other hand each clause shares variables with at most $2^k/e - 1$ other clauses then the LLL can be used to prove the existence of a satisfying assignment (which violates 0 clauses) and the MT algorithm can be used to find such an assignment efficiently. But what can be achieved when the number of clauses sharing a variables is more than $2^k/e - 1$? Lemma V.1 (proof in the full version [17]) shows that a better assignment can be constructed if it is possible to find a sparsely connected sub-formula that satisfies the LLL-condition.

Lemma V.1. *Suppose F is a k-CNF formula in which there exists a set of core clauses C with the property that: (i) every clause in C shares variables with at most $d \leq 2^k/e - 1$ clauses in C, and (ii) every clause in $\overline{C}$ shares variables with at most $\gamma(2^k/e - 1)$ many clauses in C, for some $\gamma \geq 0$. Let n and m denote the total number of variables and clauses in F,*

respectively. Then, for any $\theta \geq 1/\mathrm{poly}(n,m)$, there is a randomized $\mathrm{poly}(n,m)$-time algorithm that produces, with high probability, an assignment in which all clauses in C are satisfied and at most an $(1+\theta)2^{-k}e^{\gamma}$ fraction of clauses from $\overline{C}$ are violated. (If we are content with success-probability $\rho - n^{-c}$ for some constant c, then there is also a randomized algorithm that runs in time $\mathrm{poly}(n,|C|)$, satisfies all clauses in C, and violates at most an $(1/\rho)\cdot 2^{-k}e^{\gamma}$ fraction of clauses from $\overline{C}$. This can be useful if $|C| \ll m$.)

As mentioned in Section I-B, we are also able to prove the following. Suppose we have, as usual, a system of independent random variables $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ and bad events $\mathcal{A} = \{A_1, A_2, \ldots, A_m\}$, with dependency graph $G = G_{\mathcal{A}}$. Let us consider the symmetric case in which $\Pr[A_i] \leq p = o(1)$ for each i. Suppose the maximum degree of G is at most $\alpha(1/(ep) - 1)$, for $1 < \alpha < e$. We prove that one can construct, in randomized time polynomial in n and m, an assignment to the P_i such that the number of A_j that hold is at most $(1 + o(1)) \cdot (e(\ln \alpha)/\alpha) \cdot mp$.

Acknowledgments: We thank Nikhil Bansal for helpful clarifications about the Santa Claus problem. The first author thanks Michel Goemans and Ankur Moitra for discussing the Santa Claus problem with him in an early stage of this work. Our thanks are also due to Bill Gasarch and Mohammad Salavatipour for their helpful comments.

REFERENCES

[1] N. Alon. A parallel algorithmic version of the Local Lemma. *Random Structures & Algorithms*, 2:367–378, 1991.

[2] N. Alon, L. Babai, and A. Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of Algorithms*, 7:567–583, 1986.

[3] N. Alon, G. Gutin, E. J. Kim, S. Szeider, and A. Yeo. Solving MAX-r-SAT above a tight lower bound. In *SODA '10: Proceedings of the 21th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2010.

[4] N. Alon and J. H. Spencer. *The Probabilistic Method, Third Edition*. John Wiley & Sons, Inc., 2008.

[5] A. Asadpour, U. Feige, and A. Saberi. Santa claus meets hypergraph matchings. In *APPROX '08 / RANDOM '08: Proceedings of the 11th international workshop, APPROX 2008, and 12th international workshop, RANDOM 2008 on Approximation, Randomization and Combinatorial Optimization*, pages 10–20, 2008.

[6] A. Asadpour and A. Saberi. An approximation algorithm for max-min fair allocation of indivisible goods. In *STOC '07: Proceedings of the 39th annual ACM Symposium on Theory of Computing*, pages 114–121, 2007.

[7] N. Bansal and M. Sviridenko. The Santa Claus problem. In *STOC '06: Proceedings of the 38th annual ACM Symposium on Theory of Computing*, pages 31–40, 2006.

[8] M. Bateni, M. Charikar, and V. Guruswami. Maxmin allocation via degree lower-bounded arborescences. In *STOC '09: Proceedings of the 41st annual ACM Symposium on Theory of Computing*, pages 543–552, 2009.

[9] J. Beck. An algorithmic approach to the Lovász Local Lemma. *Random Structures & Algorithms*, 2(4):343–365, 1991.

[10] I. Bezáková and V. Dani. Allocating indivisible goods. *SIGecom Exch.*, 5(3):11–18, 2005.

[11] D. Chakrabarty, J. Chuzhoy, and S. Khanna. On allocating goods to maximize fairness. In *FOCS '09: 50th Annual IEEE Symposium on Foundations of Computer Science*, 2009.

[12] K. Chandrasekaran, N. Goyal, and B. Haeupler. Deterministic Algorithms for the Lovász Local Lemma. *SODA '10: Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms*, 2010.

[13] A. Czumaj and C. Scheideler. Coloring non-uniform hypergraphs: A new algorithmic approach to the general Lovász local lemma. In *SODA '00: Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 30–39, 2000.

[14] 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.

[15] U. Feige. On allocations that maximize fairness. In *SODA '08: Proceedings of the 19th annual ACM-SIAM Symposium on Discrete Algorithms*, pages 287–293, 2008.

[16] U. Feige. On estimation algorithms vs approximation algorithms. In R. Hariharan, M. Mukund, and V. Vinay, editors, *FSTTCS*, volume 2 of *LIPIcs*, pages 357–363. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2008.

[17] B. Haeupler, B. Saha, and A. Srinivasan. New constructive aspects of the lovasz local lemma. *CoRR*, abs/1001.1231, 2010.

[18] P. Haxell. A condition for matchability in hypergraphs. *Graphs and Combinatorics*, 11(3):245–248, 1995.

[19] 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.

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

[21] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM Journal of Computing*, 15(4):1036–1053, 1986.

[22] M. Molloy and B. Reed. Further algorithmic aspects of the Local Lemma. In *STOC '98: Proceedings of the 30th annual ACM Symposium on Theory of Computing*, pages 524–529, 1998.

[23] M. Molloy and B. Reed. *Graph Colouring and the Probabilistic Method*. Springer-Verlag, 2001.

[24] R. Moser. A constructive proof of the Lovász Local Lemma. In *STOC '09: Proceedings of the 41st annual ACM Symposium on Theory of Computing*, pages 343–350, 2009.

[25] R. Moser and G. Tardos. A constructive proof of the general Lovász Local Lemma. *Journal of the ACM*, 57(2):1–15, 2010.

[26] R. A. Moser. Derandomizing the Lovász Local Lemma more effectively. *CoRR*, abs/0807.2120, 2008.

[27] B. Saha and A. Srinivasan. A new approximation technique for resource-allocation problems. In *ICS '10: Proceedings of the first annual Symposium on Innovations in Computer Science*, pages 342–357, 2010.

[28] A. Srinivasan. Improved algorithmic versions of the Lovász Local Lemma. In *SODA '08: Proceedings of the 19th annual ACM-SIAM Symposium on Discrete algorithms*, pages 611–620, 2008.