# New Constructive Aspects of the Lovász Local Lemma[*]

Bernhard Haeupler[†]     Barna Saha[‡]     Aravind Srinivasan[§]

October 4, 2011

## 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 for the full asymmetric LLL. 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 a 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. This is shown via a simple union bound over the probabilities of non-core events in the conditional LLL-distribution, and automatically leads to simple and efficient Monte-Carlo (and in most cases *RNC*) algorithms.

We demonstrate this idea on several applications. We give the first constant-factor approximation algorithm for the Santa Claus problem by making a 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 show that the properties of the conditional LLL-distribution can be used in cases 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 a selected smaller core from the LLL-distribution, we can avoid a fraction of bad events that is higher than the expectation. MAX $k$-SAT is an illustrative example of this.

arXiv:1001.1231v5 [cs.DS] 2 Oct 2011

# 1 Introduction

The well-known Lovász Local Lemma (LLL) [24] is a powerful probabilistic approach to prove the existence of certain combinatorial structures. Its diverse range of applications include breakthroughs in packet-routing [33], a variety of theorems in graph-coloring including list coloring, frugal coloring, total coloring, and coloring graphs with lower-bounded girth [38], as well as a host of other applications where probability appears at first sight to have no role [10]. 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 [17, 1, 23, 37, 38, 46, 41, 39, 40] beginning with Beck [17] and culminating with the breakthrough of Moser & Tardos (MT) [40] 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 [40]. 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.

We will use standard notation: $e$ denotes the base of the natural logarithm, and ln and log denote the logarithm to the base $e$ and 2, respectively.

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 1.1) 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 [40]: at most 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 poly$(n)$ processors and run in 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 $O(n \cdot \text{polylog}(n))$), 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 $D$ [10]: 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 $\Pr[B]$: at

---

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

most $f_\mathcal{A}(B) \cdot \Pr[B]$ (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 $f_\mathcal{A}(B) \cdot \Pr[B]$ 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 very general Theorem 3.4 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 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.

## 1.1 Applications that avoid all bad events

A summary 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 $n$ items that need to be distributed among $m$ children. Each child has a utility for each item, which is either 0 or some given $p_j$ for item $j$. The objective is to assign each item to some child, so that the minimum total utility received by any child is maximized. This problem has received much attention recently [15, 14, 26, 13, 16, 20]. The problem is NP-Hard and the best-known approximation algorithm due to Bansal and Sviridenko [15] achieves an approximation factor of $O(\frac{\log \log m}{\log \log \log m})$ by rounding a certain configuration LP. Later, Feige in [26] and subsequently Asadpour, Feige and Saberi in [13] showed that the integrality gap of the configuration LP is a constant. Surprisingly, both results were obtained using two different non-constructive approaches and left the question for a constant-factor approximation algorithm open. This made the Santa Claus problem to one of the rare instances [27] in which the proof of an integrality gap did not result in an approximation algorithm with the same ratio. In this paper we resolve this by making the non-constructive LLL-based proof of Feige [26] constructive (Section 4) and giving the first constant-factor approximation algorithm for the Santa Claus problem.

*Non-repetitive Coloring of Graphs:* Given a graph $H = (V, E)$, a $k$-coloring (not necessarily proper) of the edges of $H$ is called *non-repetitive* if the sequence of colors along any simple path is not the same in the first and the second half. The smallest $k$ such that $H$ has a non-repetitive $k$-coloring is called the *Thue number* $\pi(H)$ of $H$ [47]. Alon, Grytczuk, Hauszczak and Riordan showed via the LLL that $\pi(H) \leq O(\Delta(H)^2)$ [5], where $\Delta$ is the maximum degree of any vertex in $H$. This was followed by much additional works [22, 45, 29, 32, 19, 4]. However, no efficient construction is known till date, except for special classes of graphs such as complete graphs, cycles and trees. We present a randomized algorithm for non-repetitive coloring of $H$ using at most $O(\Delta(H)^{2+\epsilon})$ colors, for every constant $\epsilon > 0$ (Section 5).

*General Ramsey-Type Graphs:* The Ramsey number $R(U_s, V_t)$ refers to the smallest $n$ such that any graph on $n$ vertices either contains a $U_s$ within any subgraph of $s$ vertices, or there exist $t$ vertices that do not contain $V_t$. Obtaining lower bounds for various special cases of $R(U_s, V_t)$ and constructing Ramsey type graphs have been studied in much detail [2, 9, 31, 7]. A predominant case for such problems is when $s$ is held fixed. We consider the general setting of $R(U_s, V_t)$ with fixed $s$, and provide efficient randomized algorithms for constructing Ramsey-type graphs (Section 6).

*Acyclic Edge-Coloring:* A proper edge-coloring of a graph is *acyclic* iff each cycle in it receives more than 2 colors. The acyclic chromatic number $a(G)$ introduced in [28] is the minimum number of colors in a proper

acyclic edge coloring of $G$ [8, 37, 11, 28, 42]. Alon, McDiarmid and Reed [8] showed that $a(G) < 64\Delta$, where $\Delta$ is the maximum degree. The constant was later improved to 16 by Molloy and Reed [37], who also mention an algorithmic version using $20\Delta$ colors. However it was conjectured that $a(G) = \Delta + 2$; Alon, Sudakov and Zaks showed indeed the conjecture is true for graphs having girth $\Omega(\Delta \log \Delta)$ [11]. Their algorithm can be made constructive using Beck's technique [17] to obtain an acyclic edge coloring using $\Delta + 2$ colors, albeit for graphs with girth significantly larger than $\Theta(\Delta \log \Delta)$ [11]. We bridge this gap by providing constructions to achieve the same girth bound as in [11], yet obtaining an acyclic edge coloring with only $\Delta + 2$ colors. For graphs with no girth bound, $16\Delta$ colors suffice to efficiently construct an acyclic edge coloring in contrast to the $20\Delta$ algorithmic bound of [37] (Section 7).

The recent result of Matthew Andrews on approximating the edge-disjoint paths problem on undirected graphs is another example, where problems (a) and (b) occur and our LLL-techniques are applied to avoid super-polynomially many bad events [12].

## 1.2   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 stated in Section 1.3, 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 1, if $\Pr[A_i] \le p$ and $A_i$ depends on at most $d$ other $A_j$ for all $i$, then $e \cdot p \cdot (d+1) \le 1$ 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 [6] 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; see Section 8. We believe this connection to be the main conceptual message of this paper, and expect further applications in the future.

## 1.3   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 1, 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 $\text{vbl}(B)$. 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 $\text{vbl}(A) \cap \text{vbl}(B) \neq \emptyset$. 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 1.1** (Asymmetric Lovász Local Lemma). *With $\mathcal{A}, \mathcal{P}$ and $\Gamma$ defined as above, if there exists an assignment of reals $x : \mathcal{A} \to (0, 1)$ such that*

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

*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 $\mathcal{P}$.*

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 $B \in \mathcal{A}$ then 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 $x$ (the maximum is attained at $x = \frac{1}{2}$).

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 $\delta$ 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.

The following subsection 1.4 reviews the MT algorithm and its analysis, which will be helpful to understand some of our proofs and technical contributions; the reader familiar with the MT algorithm may skip it.

## 1.4 Review of the MT Algorithm and its Analysis

Recall the resampling-based MT algorithm; let us now review some of the technical elements in the analysis of this algorithm, that will help in understanding our technical contributions better.

A witness tree $\tau = (T, \sigma_T)$ is a finite rooted tree $T$ together with a labeling $\sigma_T : V(T) \to \mathcal{A}$ of its vertices to events, such that the children of a vertex $u \in V(T)$ receive labels from $\Gamma(\sigma_T(u)) \cup \sigma_T(u)$. In a proper witness tree distinct children of the same vertex always receive distinct labels. The "log" $C$ of an execution of MT lists the events as they have been selected for resampling in each step. Given $C$, we can associate a witness tree $\tau_C(t)$ with each resampling step $t$ that can serve as a justification for the necessity of that correction step. $\tau_C(t)$ will be rooted at $C(t)$. A witness tree is said to occur in $C$, if there exists $t \in N$, such that $\tau_C(t) = \tau$. It has been shown in [40] that if $\tau$ appears in $C$, then it is proper and it appears in $C$ with probability at most $\Pi_{v \in V(\tau)} \Pr[\sigma_T(v)]$.

To bound the running time of the MT algorithm, one needs to bound the number of times an event $A \in \mathcal{A}$ is resampled. If $N_A$ denotes the random variable for the number of resampling steps of $A$ and $C$ is the execution log; then $N_A$ is the number of occurrences of $A$ in this log and also the number of distinct proper witness trees occurring in $C$ that have their root labeled $A$. As a result one can bound the expected value of $N_A$ simply by summing the probabilities of appearances of distinct witness trees rooted at $A$. These probabilities can be related to a Galton-Watson branching process to obtain the desired bound on the running time.

A Galton-Watson branching process can be used to generate a proper witness tree as follows. In the first round the root of the witness tree is produced, say it corresponds to event $A$. Then in each subsequent round, for each vertex $v$ independently and again independently, for each event $B \in \Gamma \sigma_T(v) \cup \sigma_T(v)$, $B$ is selected as a child of $v$ with probability $x(B)$ and is skipped with probability $(1 - x_B)$. We will use the concept of a proper witness trees and Galton-Watson process in several of our proofs.

# 2    LLL-Distribution

When trying to turn the non-constructive Lovász Local Lemma into an algorithm that finds a good assignment 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 non-efficient. While the celebrated algorithm of Moser (and Tardos) is 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 that is obtained when one conditions on no bad event happening. In the following, we call this distribution *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 8. The following is a well-known bound on the probability $\Pr_D[B]$ 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 non-constructive LLL-proof [10]):

**Theorem 2.1.** *If the LLL-conditions from Theorem 1.1 are met, then the LLL-distribution $D$ is well-defined. For any event $B$ that is determined by $\mathcal{P}$, the probability $\Pr_D[B]$ of $B$ under $D$ satisfies:*

$$\mathsf{Pr}_D[B] := \mathsf{Pr}\left[B \mid \bigwedge_{A \in \mathcal{A}} \overline{A}\right] \leq \mathsf{Pr}[B] \cdot \prod_{C \in \Gamma(B)} (1 - x_C)^{-1}; \tag{3}$$

*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 increase much in the conditional LLL-distribution when $B$ does not depend on "too many" $C \in \mathcal{A}$, is used critically 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 2.2.** *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 $B$ being true in the output distribution of MT obeys this upper-bound.*

*Proof.* The bound on the probability of $B$ ever happening is a simple extension of the MT proof [40]. Note that we want to prove the theorem irrespective of whether $B$ is in $\mathcal{A}$ or not. In either case we are interested in the probability that the event was true at least once during the execution, i.e., if $B$ is in $\mathcal{A}$ whether it

could have been resampled at least once. The witness trees that certify the *first time* $B$ becomes true are the ones that have $B$ as a root and all non-root nodes from $\mathcal{A} \setminus \{B\}$. Similarly as in [40], we calculate the expected number of these witness trees via a union bound. Let $\tau$ be a fixed proper witness tree with its root vertex labeled $B$. Following the proof of Lemma 3.1 and using the fact that $B$ cannot be a child of itself, it can be shown that the probability $p_\tau$ with which the Galton-Watson process that starts with $B$ yields exactly the tree $\tau$ is $p_\tau = \prod_{A \in \Gamma(B)} (1 - x(A)) \cdot \prod_{v \in V(\tau)} x'(\sigma_v)$. Here $V(\tau)$ are the non-root vertices of $\tau$ and $x'(\sigma_v) = x(\sigma_v) \prod_{C \in \Gamma(\sigma_v)} (1 - x(C))$. Plugging this in the arguments following the proof of Lemma 3.1 of [40] it is easy to see that the union bound over all these trees and therefore also the desired probability is at most $\Pr[B] \cdot (\prod_{C \in \Gamma(B)} (1 - x_C))^{-1}$ where the term "$\Pr[B]$" accounts for the fact that the root-event $B$ has to be true as well. ∎

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 2.2 and demonstrate this idea.

# 3  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 number of 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 3.1 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 3.1.** *Suppose there is an $\epsilon \in [0, 1)$ and an assignment of reals $x : \mathcal{A} \to (0, 1)$ such that:*

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

*With $\delta$ denoting $\min_{A \in \mathcal{A}} x(A) \prod_{B \in \Gamma(A)} (1 - x(B))$, we have*

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

*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 $\lambda \ge 1$, the MT algorithm terminates within $\lambda v_1$ 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 $\lambda \ge 1$, the MT algorithm terminates within $\lambda v_2$ resamplings with probability $1 - \exp(-\lambda)$.*

*Proof.* The main idea of relating the quantity $T$ to $n$ and $\delta$ is to use: (i) the fact that the variable-sharing graph $G$ is very 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

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

and note that these events form a clique in $G$. Indeed, the $m$ vertices of $G$ can be partitioned into $n$ such cliques with potentially further edges between them, and therefore has at least $n \cdot \binom{m/n}{2} = m^2/(2n) - m/2$ edges, which is high density for $m \gg n$.

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,

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

Let us now show that for $1/2 \leq x_A \leq x_{B_1} \leq 1$,

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

Fix $x_A$. We thus need to show $e^{x_{B_1}}(1 - x_{B_1}) \leq \frac{1}{x_A e^{x_A}}$. The derivative of $e^{x_{B_1}}(1 - x_{B_1})$ is negative for $x_{B_1} \geq 0$, showing that it is a decreasing function in the range $x_{B_1} \in [x_A, 1]$. Therefore the maximum value of $e^{x_{B_1}}(1 - x_{B_1})$ is obtained at $x_{B_1} = x_A$ and for (6) to hold, it is enough to show that, $x_A(1 - x_A) \leq e^{-2x_A}$ holds. The second derivative of $e^{-2x_A} - x_A(1 - x_A)$ is positive. Differentiating $e^{-2x_A} - x_A(1 - x_A)$ and equating the derivative to 0, returns the minimum in $[1/2, 1]$ at $x_A = 0.7315$. The minimum value is $0.0351 > 0$. Thus we have (6) and 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 [40] and by a simple application of Markov's inequality.

Part (2) now also follows from [40]. In section 5 of [40] it is shown that saving a $(1-\epsilon)$ 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 $\delta$ in (2), $\log(1/\delta) \leq O(n \log n)$ in all applications known to us, and is often even smaller.

**Remarks**

- The $\max_{A \in \mathcal{A}} \dfrac{1}{1 - x(A)}$ factor in the running time of part (1) of Theorem 3.1 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.

- 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 3.1 gives very good bounds on the running time of MT even for applications with $\Omega(n) \leq m \leq \text{poly}(n)$ 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 of Section 4, 5 and 6 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 3.2. (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 2.2, 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 cannot 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 3.3 shows that if the LLL-conditions are fulfilled for $\mathcal{A}$ then a non-core event $A \in \mathcal{A} \setminus \mathcal{A}'$ is violated in the produced output with probability at most $x_A$. This makes the success probability of such an approach at least $1 - \sum_{A \in \mathcal{A} \setminus \mathcal{A}'} x_A$.

**Theorem 3.3.** *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:*

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

*Then the modified MT-algorithm can be efficiently implemented with an expected number of resamplings according to Theorem 3.1. The algorithm furthermore outputs a good assignment with probability at least $1 - \sum_{A \in \mathcal{A} \setminus \mathcal{A}'} x_A$.*

*Proof.* Note that the set $\mathcal{A}'$ on which the actual MT-algorithm is run fulfills the LLL-conditions. This makes Theorem 3.1 applicable. To argue about the success probability of the modified algorithm, note that $x(A) \geq \Pr[A] \prod_{B \in \Gamma'(A)} (1 - x(B))$ where $\Gamma'(A)$ are the neighbors of $A$ in the variable sharing graph defined on $\mathcal{A}'$. Using Theorem 2.2 we get that the probability that a non-core bad event $A \in \mathcal{A} \setminus \mathcal{A}'$ holds in the assignment produced by the modified algorithm is at most $x_A$. Since core-events are avoided completely by the MT-algorithm a simple union bound over all conditional non-core event probabilities results in a failure probability of at most $\sum_{A \in \mathcal{A} \setminus \mathcal{A}'} x_A$.

Here is furthermore a direct proof of the theorem incorporating the argument from Theorem 2.2 into the proof:

Redefine the witness trees of [40] to have only events from $\mathcal{A}'$ in non-root nodes, thus getting a modification of the Galton-Watson process from Section 3 of [40]. As in [40], we grow witness trees from an execution-log starting with a root event that holds at a certain point in time. This guarantees that we capture events $A \in \mathcal{A} \setminus \mathcal{A}'$ happening even though they are never resampled (since we never check whether such events $A$ hold or not). Note that if some $A \in \mathcal{A} \setminus \mathcal{A}'$ holds after termination, then there is a witness tree with $A$ as root and with all non-root nodes belonging to $\mathcal{A}'$. Following the proof of Lemma 3.1 from [40] the probability for this to happen is at most $\sum_{A \in \mathcal{A} \setminus \mathcal{A}'} x_A$. (We do not get $\sum_{A \in \mathcal{A} \setminus \mathcal{A}'} x_A / (1 - x_A)$, since $A$ cannot be a child of itself in the witness trees that we construct.) ∎

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" $\epsilon$-slack as introduced by [21], finding such a good core is straightforward:

**Theorem 3.4.** *Suppose there is a fixed constant $\epsilon \in (0,1)$ and an assignment of reals $x : \mathcal{A} \to (0, 1 - \epsilon)$ such that:*

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

*Suppose further that $\log 1/\delta \leq \mathrm{poly}(n)$, where $\delta = \min_{A \in \mathcal{A}} x(A) \prod_{B \in \Gamma(A)} (1 - x(B))$. Then for every $p \geq \frac{1}{\mathrm{poly}(n)}$ the set $\{A_i \in \mathcal{A} : \Pr[A_i] \geq p\}$ 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 3.1 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 $\Pr[A] \leq x_A < 1 - \epsilon$ and thus get a $(1-\epsilon)^\epsilon = (1 - \Theta(\epsilon^2))$-slack; therefore Theorem 3.1 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 $\epsilon$-slack" provide an extra multiplicative $p^{-\epsilon}$ factor over the LLL-conditions in Theorem 3.1. We have $x(A)\Pr[A]^\epsilon \geq \Pr[A] \prod_{B \in \Gamma'(A)} (1 - x(B))$ where $\Gamma'(A)$ are the neighbors of $A$ in the variable sharing graph defined on $A'$. Using Theorem 2.2 and setting $p = n^{-\Theta(1/\epsilon)}$, we get that the probability that a non-core bad event $A \in \mathcal{A} \setminus \mathcal{A}'$ holds in the assignment produced by the modified algorithm is at most $x_A \Pr[A]^\epsilon \leq x_A n^{-\Theta(1)}$. Since core-events are avoided completely by the MT-algorithm, a simple union bound over all conditional non-core event probabilities results in a failure probability of at most $\frac{1}{n^{\Theta(1)}} \sum_{A \in \mathcal{A} \setminus \mathcal{A}'} x_A$. Now since, $\sum_{A \in \mathcal{A} \setminus \mathcal{A}'} x_A \leq \sum_{A \in \mathcal{A}'} x_A = T = poly(n)$ holds, 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 3.4 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 3.4 makes these proofs constructive without further complicated analysis. In most cases, only some adjustments in the bounds are needed to respect the $\epsilon$-slack in the LLL-condition.

**Remarks**

- 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 probability amplification techniques.

- 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 NP-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 an MIS of violated events in each round (using MIS algorithms such as [3, 35]). Using the proof of Theorem 3.1 which is based on some ideas from the parallel LLL algorithm of MT, it is easy to see that only logarithmically many rounds of resampling these events are needed.

- Even though the derandomization of [21] also only requires an "exponential $\epsilon$-slack" in the LLL-conditions, applying the techniques used there and in general getting efficient deterministic algorithms when $m$ is superpolynomial seems hard. The derandomization in [21] either explicitly works on all $m$ events when applying the method of conditional probabilities or uses approximate $O(\log m)$-wise independent probability spaces which have an inherently $poly(m)$ size domain.

# 4   A Constant-Factor Approximation Algorithm for the Santa Claus Problem

The *Santa Claus* problem is the restricted assignment version of the max-min allocation problem of indivisible items. In this section, we present the first efficient randomized constant-factor approximation algorithm for this problem.

In the max-min allocation problem, there is a set $\mathcal{C}$ of $n$ items, and $m$ children. The value (utility) of item $j$ to child $i$ is $p_{i,j} \geq 0$. An item can be assigned to only one child. If a child $i$ 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 child, 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 [34].) This problem has received much attention recently [15, 14, 26, 13, 16, 20, 44].

A restricted version of max-min allocation is where each item has an intrinsic value, and where for every child $i$, $p_{i,j}$ 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 $P = NP$ [18]. Bansal and Sviridenko [15] 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 [26] 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" [27] for which a constructive proof

would be desirable. Using a completely different approach, Asadpour, Feige and Saberi [13] 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 [30]. Haxell's theorems are again highly non-constructive and the stated local-search problem is not known to be efficiently solvable. Thus this second non-constructive proof still left the question of a constant-factor approximation algorithm open.

In this section we show how our Theorem 3.4 can be used to easily and directly constructivize the LLL-based proof of Feige [26], giving the first constant-factor approximation algorithm for the Santa Claus problem.

It is to be noted that the more general max-min fair allocation problem appears significantly harder. It is known that for general max-min fair allocation, the configuration LP has a gap of $\Omega(\sqrt{m})$. Asadpour and Saberi [14] gave an $O(\sqrt{m}\ln^3(m))$ approximation factor for this problem using the configuration LP. Recently, Saha and Srinivasan [44] have improved this to $O(\frac{\sqrt{m\ln m}}{\ln\ln m})$. So far the best approximation ratio known for this problem due to Chakraborty, Chuzhoy and Khanna is $O(n^\epsilon)$ [20], for any constant $\epsilon > 0$; their algorithm runs in $O(n^{1/\epsilon})$ time.

## 4.1 Algorithm

We focus on the Santa Claus problem here. We start by describing the configuration LP and the reduction of it to a combinatorial problem over a set system, albeit with a constant factor loss in approximation. Next we give a constructive solution for the set system problem, thus providing a constant-factor approximation algorithm for the Santa Claus problem.

We guess the optimal solution value $T$ using binary search. An item $j$ is said to be small, if $p_j < \alpha T$, otherwise it is said to be big. Here $\alpha < 1$ is the approximation ratio, which will get fixed later. A configuration is a subset of items. The value of a configuration $C$ to child $i$ is denoted by $p_{i,C} = \sum_{j \in C} p_{i,j}$. A configuration $C$ is called valid for child $i$ if:

- $p_{i,C} \geq T$ and all the items are small; or

- $C$ contains only one item $j$ and $p_{i,j} = p_j \geq \alpha T$, that is, $j$ is a big item for child $i$.

Let $C(i,T)$ denote the set of all valid configurations corresponding to child $i$ with respect to $T$. We define an indicator variable $y_{i,C}$ for each child $i$ and all valid configurations $C \in C(i,T)$ such that it is 1 if child $i$ receives configuration $C$ and 0 otherwise. These variables are relaxed to take any fractional value in $[0,1]$ to obtain the configuration LP relaxation.

$$\forall j : \sum_{C \ni j} \sum_i y_{i,C} \leq 1 \tag{7}$$
$$\forall i : \sum_{C \in C(i,T)} y_{i,C} = 1$$
$$\forall i, C : y_{i,C} \geq 0$$

Bansal and Sviridenko showed that if the above LP is feasible, then it is possible to find a fractional allocation that assigns a configuration with value at least $(1 - \epsilon)T$ to each child in polynomial time.

The algorithm of Bansal and Sviridenko starts by solving the configuration LP (7). Then by various steps of simplification, they reduce the problem to the following instance:

*There are p groups, each group containing l children. Each child is associated with a collection of k items with a total valuation of $\frac{T}{c}$, for some constant $c > 0$. Each item appears in at most $\beta l$ sets for some $\beta \leq 3$. Such an instance is referred to as $(k, l, \beta)$-system.*

The goal is to efficiently select one child from each group and assign at least $\lfloor \gamma k \rfloor$ items to each of the chosen children, such that each item is assigned only once. If such an assignment exists, then the corresponding $(k, l, \beta)$-system is said to be $\gamma$-good $(k, l, \beta)$-system.

Feige showed that indeed the $(k, l, \beta)$-system that results from the configuration LP is $\gamma$-good, where $\gamma = O\left(\frac{1}{max(1,\beta)}\right)$ [26]. This established a *constant* factor integrality gap for the configuration LP. However, the proof being non-constructive, no algorithm was known to efficiently find such an assignment. In the remaining of this section, we make Feige's argument constructive, thus giving a constant-factor approximation algorithm for the Santa Claus problem. But before that, for the sake of completeness, we briefly describe the procedure that obtains a $(k, l, \beta)$-system from an optimal solution of the configuration LP [15].

## 4.2 From a configuration LP solution to a $(k, l, \beta)$-system

The algorithm starts by simplifying the assignment of *big* items in an optimal solution (say) $y^*$ of the configuration LP. Let $J_B$ denote the set of big items. Consider a bipartite graph $G$ with children $M$ on the right side and big items $J_B$ on the left side. An edge $(i, j), i \in M, j \in J_B$ of weight $w_{i,j} = \sum_{j \in C(i,T)} y^*_{i,C}$ is inserted in $G$ if $w_{i,j} > 0$. These $w_{i,j}$ values are then modified such that after the modification the edges of $G$ with weight in $(0, 1)$ form a forest.

**Lemma 5 [15].** *The solution $y^*$ can be transformed into another feasible solution of the configuration LP where the graph $G$ is a forest.*

The transformation is performed using the simple cycle-breaking trick. Each cycle is broken into two matchings; weights on the edges of one matching are increased gradually while the weights on the other are decreased until some weights hit 0 or 1. If a $w_{i,j}$ becomes 0 in this procedure, the edge $(i, j)$ is removed from $G$. Else if it becomes 1, then item $j$ is permanently assigned to child $i$ and the edge $(i, j)$ is removed.

Suppose $G'$ is the forest obtained after this transformation. The forest structure is then further exploited to form groups of children and big items.

**Lemma 6 [15].** *The solution $y^*$ can be transformed into another solution $y'$ such that children $M$ and big items $J_B$ can be clustered into $p$ groups $M_1, M_2, \ldots, M_p$ and $J_{B,1}, J_{B_2}, \ldots, J_{B_p}$ respectively with the following properties.*

1. *For each $i = 1, 2, \ldots, p$, the number of jobs $J_{B,i}$ in group $M_i$ is exactly $|M_i| - 1$. The group $J_{B,i}$ could possibly be empty.*

2. *Within each group the assignment of big job is entirely flexible in the sense that they can be placed feasibly on any of the $|M_i| - 1$ children out of the $|M_i|$ children.*

3. *For each group $M_i$, the solution $y'$ assigns exactly one unit of small configurations to children in $M_i$ and all the $|M_i| - 1$ units of configurations correspond to big jobs in $J_{B,i}$. Also, for each small job $j$, $\sum_{C \ni j} \sum_i y'_{i,C} \leq 2$.*

Lemma 6 implies that the assignment of big items to children in a group is completely flexible and can be ignored. We only need to choose one child from each group who will be satisfied by a configuration of small items. Let $y'$ assigns a small configuration $C$ to an extent of $y'_{m,C}$ to some child $c \in M_i, i \in [1, p]$, then we say that $M_i$ contains the small configuration $C$ for child $c \in M_i$. Without loss of generality, it can be assumed that each child in the groups is fractionally assigned to exactly one small configuration. Bansal and Sviridenko further showed that $y'$ can again be simplified such that each small configuration is

assigned to at least to an extent of $\frac{1}{l} = \frac{1}{n+m}$ to each child and for each small job $j$, $\sum_{C \ni j} \sum_i y'_{i,C} \leq 3$. This implies, if we consider all the small configurations across $p$ groups, then each small job appears in at most $\beta l$ configurations, where $\beta = 3$.

Finally, the following lemma shows that by losing a constant factor in the approximation, one can assume that all the small jobs have same size.

**Lemma 8** [15]**.** *Given the algorithmic framework above, by losing a constant factor in the approximation, each small job can be assumed to have size $\frac{\epsilon T}{n}$.*

As a consequence of the above lemma, we now have the following scenario.

*There are $p$ groups $M_1, M_2, \ldots, M_p$, each containing at most $l$ children. Each child is associated with a set that contains $k = \Theta(\frac{n}{\epsilon})$ items. Each item belongs to at most $\beta l$ sets. The goal is to pick one child from each group and assign at least a constant fraction of the items in its set such that each item is assigned exactly once.*

Therefore, we arrive at what is referred as a $(k, l, \beta)$-system.

## 4.3   Construction of a $\gamma$-good solution for a $(k, l, \beta)$-system

We now point out the main steps in Feige's algorithm, and in detail, describe the modifications required to make Feige's algorithm constructive.

**Feige's Nonconstructive Proof for $\gamma$-good $(k, l, \beta)$-system:**   Feige's approach is based on a systematic reduction of $k$ and $l$ in iterations, finally arriving to a system where $k$ or $l$ are constants. For constant $k$ or $l$ the following lemma asserts a constant $\gamma$.

**Lemma 4.1** (Lemma 2.1 and 2.2 of [26])**.** *For every $(k, l, \beta)$-system a $\gamma$-good solution with $\gamma$ satisfying, $\gamma = \frac{1}{k}$ or $\gamma k = \lfloor \frac{k}{\lceil \beta l \rceil} \rfloor$ can be found efficiently.*

The reduction of $(k, l, \beta)$-system to constant $k$ and $l$ involves two main lemmas, which we refer to as *Reduce-l* lemma and *Reduce-k* lemma respectively.

**Lemma 4.2** (Lemma 2.3 of [26], Reduce-l)**.** *For $l > c$ ($c$ is a sufficiently large constant), every $\gamma$-good $(k, l, \beta)$-system with $k \leq l$ can be transformed into a $\gamma$-good $(k, l', \beta')$-system with $l' \leq \log^5 l$ and $\beta' \leq \beta(1 + \frac{1}{\log l})$.*

**Lemma 4.3** (Lemma 2.4 of [26], Reduce-k)**.** *Every $(k, l, \beta)$-system with $k \geq l \geq c$ can be transformed into a $(k', l, \beta)$-system with $k' \leq \frac{k}{2}$ and with the following additional property: if the original system is not $\gamma$-good, then the new system is not $\gamma'$-good for $\gamma' = \gamma(1 + \frac{3 \log k}{\sqrt{\gamma k}})$. Conversely, if the new system is $\gamma'$-good, then the original system was $\gamma$-good.*

If $\beta$ is not a constant to start with, then by applying the following lemma repeatedly, $\beta$ can be reduced below 1.

**Lemma 4.4** (Lemma 2.5 of [26])**.** *For $l > c$, every $\gamma$-good $(k, l, \beta)$-system can be transformed into a $\gamma$-good $(k', l, \beta')$-system with $k' = \lfloor \frac{k}{2} \rfloor$ and $\beta' \leq \frac{\beta}{2}\left(1 + \frac{\log \beta l}{\sqrt{\beta l}}\right)$.*

However in our context, $\beta \leq 3$, thus we ignore Lemma 2.5 of [26] from further discussions.

Starting from the original system, as long as $l > c$, Lemma Reduce-l is applied when $l > k$ and Lemma Reduce-k is applied when $k \geq l$. In this process $\beta$ grows at most by a factor of 2. Thus at the end, $l$ is a constant and so is $\beta$. Thus by applying Lemma 4.1, the constant integrality gap for the configuration LP is established.

**Randomized Algorithm for $\gamma$-good $(k, l, \beta)$-system:** There are two main steps in the algorithm.

1. Show a constructive procedure to obtain the reduced system through Lemma Reduce-l and Lemma Reduce-k.

2. Map the solution of the final reduced system back to the original system.

We now elaborate upon each of these.

### 4.3.1 Making Lemma Reduce-l Constructive

This follows quite directly from [40]. The algorithm picks $\lfloor \log^5 l \rfloor$ sets uniformly at random and independently from each group. Thus while the value of $k$ remains fixed, $l$ is reduced to $l' = \lfloor \log^5 l \rfloor$. Now in expectation the value of $\beta$ does not change and the probability that $\beta' > \beta(1 + \frac{1}{\log l})$, and hence $\beta' l' > \beta l(1 + \frac{1}{\log l})$, is at most $e^{-\beta' l'/3 \log^2 l} \le e^{-\log^3 l} = l^{-\log^2 l}$. We define a bad event corresponding to each element:

- $A_j$: Element $j$ has more than $\beta' l'$ copies.

Now noting that the dependency graph has degree at most $kl\beta l \le 6l^3$, the uniform (symmetric) version of the LLL applies. Now it is easy to check if there exists a violated event: we simply count the number of times an element appears in all the sets. Thus we directly follow [40]; setting $x_{A_j} = \frac{1}{el^{\log^2 l}}$, we get the expected running time to avoid all the bad events to be $O(plk/l^{\log^2 l}) = O(p) = O(m)$.

### 4.3.2 Making Lemma Reduce-k Constructive

This is the main challenging part. The random experiment involves selecting each item independently at random with probability $\frac{1}{2}$. To characterize the bad events, we need a structural lemma from [26]. Construct a graph on the sets, where there is an edge between two sets if they share an element. A collection of sets 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 set has less than $k' = \left(1 - \frac{\log k}{\sqrt{k}}\right) \frac{k}{2}$ items surviving, and

2. $B_i$ for $i \ge 2$: there is a connected collection of $i$ sets from distinct groups whose union 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 $\gamma'$-goodness ($\gamma' = \delta' \frac{k}{2} \frac{1}{k'} \le \gamma \left(1 + \frac{3 \log k}{\sqrt{\gamma k}}\right)$) of the new system certifies that the original system was $\gamma$-good. That this is indeed the case follows directly from Hall's theorem as proven by Feige:

**Lemma 4.5** (Lemma 2.7 of [26]). *Consider a collection of $n$ sets and a positive integer $q$.*

1. *If for some $1 \le i \le n$, 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 $i$, $1 \le i \le n$, 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.*

14

Feige showed in [26] 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 1.3: 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 [26] that with the choice $x_i = 2^{-10i \log k}$ for all events in $B_i$, we have

$$\forall (i \geq 1) \ \forall (B \in B_i), \ \Pr[B] \leq 2^{-20i \log k} \leq x_i \prod_{j \geq 1} \prod_{A \in (B_j \cap \Gamma(B))} (1 - x_j). \tag{8}$$

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 3.4 again directly makes this proof constructive and gives an efficient Monte Carlo algorithm for producing a reduce-k system with high probability.

**Lemma 4.6.** *There is a Monte Carlo algorithm that produces a valid reduce-k system with probability at least $1 - 1/m^2$.*

*Proof.* Note from (8) that we can take $\delta = 2^{-20m \log k}$. So, we get that $\log 1/\delta = O(m \log k) = O(n \log n)$ where $n$ is the number of items and $m \leq n$ is the number of children. We furthermore get that all events with probability larger than a fixed inverse-polynomial involve only connected subsets of size $O(\frac{\log m}{\log k})$ and Theorem 3.4 implies that there are only polynomially many such "high" probability events. (This can also be seen directly since the degree of a subset is bounded by $k\beta l \leq 6k^2$ and the number of connected subcollections is therefore at most $(6k^2)^{O(\frac{\log m}{\log k})} = m^{O(1)} = n^{O(1)}$.) The connected collections of subsets are easy to enumerate using, e.g., breadth-first search and are therefore efficiently verifiable (in fact, even in parallel). Theorem 3.4 thus applies and directly proves the lemma. ∎

### 4.3.3 Mapping the solution of the final reduced system back to the original system

By repeatedly applying algorithms to produce Reduce-l or Reduce-k system, we can completely reduce down the original system to a system with a constant number of children per group, where $\beta$ can increase from 3 to at most 6 due to Lemma Reduce-l. This involves at most $\log m$ Reduce-l reductions and at most $\log n$ Reduce-k reductions. We can furthermore assume that $n < 2^m$ since otherwise simply all combinations of one child per group could be tried in time polynomial in $n$. Since, each Reduce-l or Reduce-k operation produces a desired solution with probability at least $1 - \frac{1}{m^2}$, by union bound, with probability at least $1 - O(\log n \log m/m^2) = 1 - O(\log m/m)$ a final $(k, l, \beta)$-system is produced that is $\gamma$-good for some constant $\gamma$ by Lemma 4.1. Using Lemma 4.1, we can also find a $\gamma$-good selection of children. Now, once one child from each group is selected, we can construct a standard network flow instance to assign items to these chosen children (Lemma 4.8). This finishes the process of mapping back a solution of the reduced system to the original $(k, l, \beta)$-system. While checking whether an individual reduction failed seems to be a NP-hard task, it is easy to see in the end whether a good enough assignment is produced. This enables us to rerun the algorithm in the unlikely event of a failure. Thus, the Monte Carlo algorithm can be strengthened to an algorithm that always produces a good solution and has an expected polynomial running-time.

The details of the above are given in two lemmas, Lemma 4.7 and Lemma 4.8. Theorem 4.9 follows from the two lemmas.

Suppose we start with a $(k_1, l_1, \beta_1)$-system and after repeated application of either Lemma Reduce-l or Lemma Reduce-k reach at a $(k_s, l_s, \beta_s)$-system, where $l_s < c$, a constant. We then employ Lemma 4.1 to obtain a $\gamma_s$-good $(k_s, l_s, \beta_s)$-system, where $\gamma_s$ satisfies $\gamma_s k_s = \lfloor \frac{k_s}{\lceil \beta_s l_s \rceil} \rfloor$. Since $l_s$ is a constant and $\beta_s \leq 6$, $\gamma_s$ is also a constant. Lemma 4.1 also gives a choice of a child from each group, denoted by a function

$f : \{1, \ldots, p\} \to \{1, \ldots, l_s\}$ that serves as a witness for $\gamma_s$-goodness of $(k_s, l_s, \beta_s)$-system. We use this same mapping for the original system. The following lemma establishes the goodness of the $(k_1, l_1, \beta_1)$-system.

**Lemma 4.7.** *Given a sequence of reductions of $k$, $(k_1, l_1, \beta_1) \to \ldots \to (k_s, l_s, \beta_s)$, interleaved with reductions of $l$, let for all $s \geq 2$, $\gamma_s = \gamma_{s-1}(1 + \frac{3 \log k_{s-1}}{\sqrt{\gamma_{s-1} k_{s-1}}})$. Then if the final reduced system is $\gamma_s$-good and the function $f : \{1, \ldots, p\} \to \{1, \ldots, l_s\}$ serves as a witness for its $\gamma_s$-goodness, then $f$ also serves as a witness of $\gamma$-goodness of $(k_1, l_1, \beta_1)$ system with high probability. In other words, we can simply use the assignment given by $f$ to select one child from each group and that assignment serves as a witness of $\gamma$-goodness of the original system with high probability.*

*Proof.* Suppose there exists a function $f$ that serves as a witness for $\gamma_s$-goodness of the $(k_s, l_s, \beta_s)$-system, but does not serve as a witness that $(k_{s-1}, l_{s-1}, \beta_{s-1})$-system is $\gamma_{s-1}$-good. Then there must exist a connected collection of $i, i > 0$ sets chosen from $p$ groups according to $f$, such that their union contains less than $\gamma_{s-1} k_{s-1} i$ items. However in the reduced system, their union has $\gamma_s k_{s-1} i$ elements. Call such a function $f$ bad. Thus every bad function is characterized by a violation of event of type $B_i, i \geq 1$, described in Section 4.3.2. However, by Lemma 4.6 we have $\Pr[\exists \text{ a bad function } f] \leq \Pr[\text{an event of type } B_i, i \geq 1 \text{ happens}] \leq \frac{1}{m^2}$.

Now the maximum number of times the Reduce-k step is applied is at most $\log k_1 \leq \log n$. Thus if the Reduce-l step is not applied at all, then by a union bound, function $f$ is $\gamma$-good for the $(k_1, l_1, \beta_1)$-system with probability at least $1 - \frac{\log m \log n}{m^2}$. We can assume without loss of generality that $n \leq 2^m$. (Otherwise in polynomial time we can guess the children who receive small items and thus know $f$. Once $f$ is known, an assignment of small items to the children chosen by $f$ can be done in polynomial time through Lemma 4.8.) Since $n \leq 2^m$, function $f$ is $\gamma$-good for the $(k_1, l_1, \beta_1)$-system with probability at least $1 - \log m/m$. Now since the Reduce-l step only reduces $l$ and keeps $k$ intact, it does not affect the goodness of the set system. ∎

Once we know the function $f$, using Lemma 4.8, we can get a valid assignment of $\lfloor k\gamma \rfloor$ items to each chosen child:

**Lemma 4.8.** *Given a function $f : \{1, \ldots, p\} \to \{1, \ldots, l\}$, and parameter $\gamma$, there is a polynomial time algorithm to determine, whether $f$ is $\gamma$-good and we can determine the subset of $\lfloor k\gamma \rfloor$ items received by each child $f(i), i \in [1, p]$.*

*Proof.* We construct a bipartite graph with a set of vertices $U = \{1, \ldots, p\}$ corresponding to each chosen child from the $p$ groups, a set of vertices $V$ corresponding to the small items in the sets of the chosen children, a source $s$ and a sink $t$. Next we add a directed edge of capacity $\lfloor \gamma k \rfloor$ from source $s$ to each vertex in $U$. We also add directed edges $(u, v), u \in U, v \in V$, if the item $u$ belongs to the set of $v$. These edges have capacity 1. Finally we add a directed edge from each vertex in $V$ to the sink $t$ with capacity 1. We claim that this flow network has a maximum flow of $\lfloor k\gamma \rfloor p$ iff $f$ is $\gamma$-good:

For the one direction let $f$ be $\gamma$-good. Thus there exists a set of $\lfloor \gamma k \rfloor$ elements that can be assigned to each child $u \in U$. Send one unit of flow from each child to these items that it receives. The outgoing flow from each $u \in U$ is exactly $\lfloor \gamma k \rfloor$. Since each item is assigned to at most one child, flow on each edge $(v, t), v \in V$ is at most 1. Thus all the capacity constraints are maintained and the flow value is $\lfloor \gamma k \rfloor p$.

For the other direction consider an integral maximum flow of $\lfloor k\gamma \rfloor p$. Since the total capacity of all the edges emanating from the source is $\lfloor k\gamma \rfloor p$, they must all be saturated by the maxflow. Since the flow is integral, for each child $u$ there are exactly $\lfloor \gamma k \rfloor$ edges with flow 1 corresponding to the items that it receives. Also since no edge capacity is violated, each item is assigned to exactly one child. Therefore $f$ is $\gamma$-good.

To check a function $f$ for $\gamma$-goodness and obtain the good assignment we construct the flow graph and run a max flow algorithm that outputs in an integral flow. As proven above a max flow value of $\lfloor k\gamma \rfloor p$ indicates $\gamma$-goodness and for a $\gamma$-good $f$ the assignment can be directly constructed from the flow by considering only the flow carrying edges. ∎

**Theorem 4.9.** *There exists a constant $\alpha > 0$ and a randomized algorithm for the Santa Claus problem that runs in expected polynomial time and always assigns items of total valuation at least $\alpha \cdot \mathrm{OPT}$ to each child.*

# 5 Non-repetitive Coloring of Graphs

In this section, we give an efficient Monte-Carlo construction for non-repetitive coloring of graphs. Call a word (string) $w$ "squarefree" or "non-repetitive" if there does not exist any string of the form $w = xx$, where $x \neq \emptyset$. Let us refer to graphs using the symbol $H$ instead of $G$, to not confuse with our dependency graphs $G$. Recall from Section 1 that a $k$-coloring of the edges of $H$ is called *non-repetitive* if the sequence of colors along any path in $H$ is squarefree: i.e., we want a coloring in which no path has a color-sequence of the form $xx$. (All paths here refer to simple paths.) The smallest $k$ such that $H$ has a non-repetitive coloring using $k$ colors is called the *Thue number* of $H$ and is denoted by $\pi(H)$. The Thue number was first defined by Alon, Grytczuk, Hauszczak and Riordan in [5]: it is named after Thue who proved in 1906 that if $H$ is a simple path, then $\pi(H) = 3$ [47]. While the method of Thue is constructive, no efficient construction is known for general graphs. Alon et al. showed through application of the asymmetric LLL that $\pi(H) \leq c\Delta(H)^2$ for some absolute constant $c$. Their proof was non-constructive. The number of bad events is exponential. Not only that, checking whether a given coloring is non-repetitive is coNP-Hard, even when the number of colorings is restricted to 4 [36]. Thus checking if some "bad event" holds in a given coloring is coNP-Hard. Since the work of Alon et al., the non-repetitive coloring of graphs has received a good deal of attention in the last few years [22, 45, 29, 32, 19, 4]. Yet no efficient construction is known till date, except for some special classes of graphs such as complete graphs, cycles and trees.

## 5.1 Randomized algorithm for obtaining a non-repetitive coloring

Suppose we are given a graph $H$ with maximum degree $\Delta$. We first give the proof of Alon et al. which shows that $\pi(H) \leq c\Delta^2$, and then show how to convert this proof directly into a constructive algorithm (with the loss of a $\Delta^\epsilon$ factor in the number of colors used):

**Theorem 5.1** (Theorem 1 of [5]). *There exists an absolute constant $c$ such that $\pi(H) \leq c\Delta^2$ for all graphs $H$ with maximum degree at most $\Delta$.*

*Proof.* Let $C = (2e^{16} + 1)\Delta^2$. Randomly color each edge of $H$ with colors from $C$. Consider the following types of bad events $B_i$, for $i \geq 1$: "there exists a path $P$ of length $2i$, such that the second half of $P$ is colored identically to its first half".

We have for a path $P$ of length $2i, i \geq 1$, $\Pr\left[\text{P has coloring of the form xx}\right] = \frac{1}{C^i}$. Also, a path of length $2i$ intersects at most $4ij\Delta^{2j}$ paths of length $2j$. Thus, for any bad event $A$ of type $i$, we have $\Pr\left[A\right] = \frac{1}{C^i}$ and that each bad event of type $i$ share variables with at most $4ij\Delta^{2j}$ bad events of type $B_j$. Set $x_i = \frac{1}{2^i \Delta^{2i}}$. We have $(1 - x_j) \geq e^{-2x_j}$; this, along with the fact that $\sum_{j \geq 1} j/2^j = 2$, shows that

$$x_i \prod_j (1 - x_j)^{4ij\Delta^{2j}} \geq x_i e^{-8i \sum_j x_j j \Delta^{2j}} \geq \frac{1}{2^i \Delta^{2i}} e^{-8i \sum_j \frac{j}{2^j}} = (2e^{16}\Delta^2)^{-i}.$$

Since $C = (2e^{16} + 1)\Delta^2$, the condition of the LLL is satisfied and we are guaranteed the existence of such a non-repetitive coloring. ∎

Now we see that using just a slightly higher number of colors suffices to make Theorem 3.4 apply.

**Theorem 5.2.** *There exists an absolute constant $c$ such that for every constant $\epsilon > 0$ there exists a Monte Carlo algorithm that given a graph $H$ with maximum degree $\Delta$, produces a non-repetitive coloring using at*

*most $c\Delta^{2+\epsilon}$ colors. The failure probability of the algorithm is an arbitrarily small inverse polynomial in the size of $H$.*

*Proof.* We apply the LLL using the same random experiments and bad events as in Theorem 5.1 but with $C' = C^{\frac{1}{1-\epsilon'}}$ colors such that $C' < c\Delta^{2+\epsilon}$. Using the same settings for $x_A$ gives an exponential $\epsilon'$ slack in the LLL-conditions since the probability of a bad event of type $i$ is now at most $\frac{1}{C'^n} = \left(\frac{1}{C^n}\right)^{\frac{1}{1-\epsilon'}}$. Recall Theorem 3.4. Clearly, $\log 1/\delta = O(n^2)$ and so the last thing to check to apply Theorem 3.4 is that for any inverse polynomial $p$, the bad events with probability at least $p$ are efficiently verifiable. Here these events consist of paths smaller than a certain length (of the form $O((1/\epsilon)\log n/\log \Delta)$, where $n$ is the number of vertices), and Theorem 3.4 guarantees that there are only polynomially many of these. Using breadth-first-search to go through these paths and checking each of them for non-repetitiveness is efficient and thus Theorem 3.4 directly applies. ∎

# 6 Ramsey-type Bounds

In this section, we briefly sketch another application of our method, namely the construction of Ramsey-type graphs.

The Ramsey number $R(K^s, K^t)$ is the smallest number $\ell$ such that for any $n \geq \ell$ and in any red-blue coloring of the edges of $K^n$, there either exists a $K^s$ with all red edges or a $K^t$ with all blue edges. Here, $K^a$ for any integer $a$ denotes a clique of size $a$ as usual. The fact that these numbers are finite for all $s, m$ is a special case of Ramsey's well-known theorem (see e.g. [43]). In one of the first applications of probabilistic methods in combinatorics Erdös showed the lower bound of $R(K^m, K^m) = \Omega(m2^{m/2})$ [25]. Since, then obtaining lower bounds on $R(K^s, K^t)$ and constructing Ramsey graphs avoiding "large" cliques as well as "large" independent sets simultaneously has attracted much attention [2, 9, 31, 7]. The case of fixed $s$ is the main example case for *off-diagonal* Ramsey numbers. Alon and Pudlák gave an explicit deterministic construction for off-diagonal Ramsey graphs in [2]. They showed constructively for some $\epsilon > 0$, $R(K^s, K^t) \geq t^{\epsilon\sqrt{\log s/\log\log s}}$. The best known bound for $R(K^s, K^t)$ can be obtained using LLL, $R(K^s, K^t) = O\left(\frac{t}{\log t}\right)^{\frac{s+1}{2}}$ [7]. Krivelevich gave a Monte Carlo algorithm matching this bound through large deviation inequalities [31]. In addition, Krivelevich considered related Ramsey type problems, for example, he showed that there exists a $K^4$-free graph on $n$ vertices in which any $o(n^{3/5}\log^{1/2} n)$ set of vertices does not contain a $K^3$. The problem of finding constructions for Ramsey type graphs matching the best known bounds is of great interest and may have algorithmic applications as well.

Using our method, we can achieve the best known bound for off-diagonal Ramsey numbers, that is, for fixed $s$, by directly making the LLL-based proof [7] constructive. More importantly, we can provide randomized (Monte Carlo) constructions of graphs on $n$ vertices of the form: "there is no subgraph $U$ in any set of $s$ vertices and no subgraph $W$ in any set of $t$ vertices", where $t$ can be large, typically $n^{\Theta(1)}$ – the existence of which can be proved using the LLL (often using appropriate random-graphs $G(n,p)$). When $U = K^s$, $W = K^t$ and $s$ is fixed, we get the off-diagonal Ramsey number. We refer to these as general Ramsey-type graphs. This is a direct generalizations of related Ramsey type problems considered by Krivelevich [31].

When $U$ and $W$ are some special subgraphs, few results are known. As mentioned earlier Krivelevich considered the case where $U = K^4$, $W = K^3$, $s = 4$, and $t = o(n^{3/5}\log^{1/2} n)$. In addition, he also showed constructions for arbitrary $U$, but when $W = K^t$ [31]. Alon and Krivelevich in [7] and Krivelevich in [31] considered a Ramsey-type bound $R'(K^s, K^r(t))$: the smallest number $\ell$ such that for any $n \geq \ell$, any graph on $n$ vertices either contains a $K^s$ or there exists a set of $t$ vertices containing a $K^r$. When $r = 2$, $R(K^s, K^t) = R'(K^s, K^2(t))$. However, to the best of our knowledge, no general algorithmic result avoiding any subgraph $U$ and $W$ on any set of $s$ and $t$ vertices respectively is known till date.

Briefly, the idea is as follows. Suppose, as in the typical existence-proofs for such graphs, we are able to show using the (asymmetric) LLL that for a suitable $p = p(n)$, the random graph with $n$ vertices and independent edge-probability $p$, satisfies all the required properties with positive probability. Theorem 3.4 will typically immediately apply if we allow an exponential $\epsilon$-slack. When $s$ is fixed, another related approach is to apply Theorem 3.3 and to only verify the events that correspond to $s$-sized subgraphs; since $s$ is fixed, these can be enumerated and verified efficiently. Note that as pointed out in [10], the LLL may be much more significant in improving the bounds with fixed $s$ and this is generally the case of interest while applying the LLL based arguments.

# 7   Acyclic Edge Coloring

Given a graph $G$, an acyclic edge coloring is a proper edge coloring of $G$ where each cycle receives more than 2 colors. In a proper edge coloring, no two incident edges receive the same color. In addition, here we require that no cycle receives only 2 colors. The goal is to use a minimum number of colors (known as the acyclic chromatic number $a(G)$) and obtain an acyclic edge coloring. The concept of $a(G)$ was introduced way back in 1973 [28] and has been studied by a series of researchers over the years [8, 37, 11, 28, 42]. In all these works, the asymmetric LLL is applied to achieve the best non-constructive bounds. Thus an algorithmic version of the local lemma strikes as the first choice to obtain an acyclic edge coloring.

Alon, McDiarmid and Reed [8] showed that $a(G) < 64\Delta$, where $\Delta$ is the maximum degree. The constant was later improved to 16 by Molloy and Reed [37], but the proof still was non-constructive. Both the methods are essentially the same: *randomly color each edge from a pool of colors* $\{1, 2, \dots, C\}$. They define a series of bad events, where *Type 1* bad event corresponds to two incident edges $e, f$ receiving the same color and *Type $k$* bad event implies a cycle of length $2k$ getting 2 colors. A cycle of odd length automatically gets 3 colors, if the coloring is proper. Note that the number of Type $k$ events for non-constant $k$ is super-polynomial in the number of edges of $G$. The probability of Type 1 event is $1/C$ and the probability of Type $k$ event is $1/C^{2(k-1)}$. Let $\Delta$ be the maximum degree of $G$. It is now an easy exercise to verify that each Type $k$ event depends on at most $4k\Delta$ Type 1 events and $2k\Delta^{2(l-1)}$ Type $l$ events. With this dependency, setting $C = 16\Delta$, $x_e = \frac{2}{C}$ for each edge $e$ and $x_k = \left(\frac{2}{C}\right)^{2(k-1)}$ for each cycle of length $2k$ satisfies the asymmetric LLL condition 1.1. We can turn this proof to an algorithm using $16\Delta$ colors as a direct corollary of Theorem 3.1.

**Theorem 7.1.** *There is a randomized algorithm that produces a valid acyclic coloring of any graph with $n$ edges and maximum degree $\Delta$ in expected polynomial time using $16\Delta$ colors.*

*Proof.* Recall from Section 1.3 that $\delta$ needs to be at least as high as the smallest upper bound on a probability of a bad event which is $1/C^{2(k-1)}$ for an event of Type $k$. This gives that $\log 1/\delta$ is at most $O(n \log \Delta)$. Further, it is easy to check for a violated bad event in $O(\Delta^2 n)$ time: consider subgraphs on every pair of colors and check if there is a cycle in it. Therefore Theorem 3.1 directly applies (we can set $\epsilon = 0$ in it) and we get that the expected running time is $O(\Delta^2 n) \cdot n \cdot O(n \log \Delta) = O(n^3 \Delta^2 \log \Delta)$. Note that this is far from tight. We can, for example, exploit that there is already a $\epsilon$-slack in the analysis to get a smaller number of resamplings from Theorem 3.1. ∎

Whereas this gives an efficient way to obtain acyclic edge coloring using $16\Delta$ colors and thus matching the bound known non-constructively so far; the conjectured bound for $a(G)$ is $\Delta + 2$. Alon, Sudakov and Zaks showed indeed the conjecture is true for graphs having girth $\Omega(\Delta \log \Delta)$ [11]. Their algorithm can be made constructive using Beck's technique [17] to obtain an acyclic edge coloring using $\Delta + 2$ colors, albeit for graphs with girth significantly larger than $\Theta(\Delta \log \Delta)$. We bridge this gap by providing constructions to achieve the same girth bound as in [11], yet obtaining an acyclic edge coloring with only $\Delta + 2$ colors.

The proof of Alon, Sudakov and Zaks again relies on the asymmetric LLL, but their procedure for random coloring is different from [8, 37]. They first perform a proper coloring of the edges of $G$ using $\Delta + 1$ colors [48]. Next each edge is switched to color $\Delta + 2$ with probability $1/32\Delta$. Three types of bad events are defined. In type 1 events, two incident edges $e, f$ are colored with $(\Delta+2)$th color. Type 2 events correspond to the case where no edge of a previously bichromatic cycle switches its coloring to the $(\Delta+2)$th color. Type 3 events correspond to the case where a cycle with half of its edges (every other edge) having the same color after the first step, receives $(\Delta + 2)$th color on half of its remaining edges resulting in a bichromatic cycle. It is sufficient to avoid these three types of events to ensure an acyclic edge coloring. It has been shown in [11] that by setting the values of $x$ variables to $1/512\Delta^2$, $1/128\Delta^2$ and $1/2^k\Delta^k$ for events of Type 1, 2, and 3 (with cycles of length $2k$), conditions of the asymmetric LLL (Theorem 1.1) satisfy. Converting this non-constructive proof into an algorithm using our method is an easy exercise. We state the theorem below, whose proof is similar to Theorem 7.1 above, and is left as an exercise.

**Theorem 7.2.** *There is a randomized algorithm that produces a valid acyclic coloring in expected polynomial time using $(\Delta + 2)$ colors for graphs having girth $\Omega(\Delta \log \Delta)$.*

Further non-asymptotic results are known for graphs with sufficiently large girth. Muthu, Narayanan and Subramanian showed $a(G) < 6\Delta$ for graphs with girth at least 9 and $a(G) < 4.52\Delta$ for graphs with girth at least 220 [42]. Their proofs can also be made constructive using essentially the same proof of Theorem 7.1.

# 8    Beyond the LLL Threshold

This section sketches another application of using the properties of the conditional LLL-distribution introduced in Section 2 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 2.2. 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 wish 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 proof 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 happens when the number of clauses sharing a variables is more than $2^k/e$? Lemma 8.1 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 8.1.** *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*

poly$(n, |C|)$, *satisfies all clauses in $C$, and violates at most $n$ $(1/\rho) \cdot 2^{-k} e^{\gamma}$ fraction of clauses from $\overline{C}$. This can be useful if $|C| \ll m$.*)

*Proof.* Briefly, the idea above is as follows. Suppose we do the obvious random assignment to the variables: each is set to "True" or "False" uniformly at random and independently. For any clause $C_i$, let $A_i$ be the bad event that it is violated in such an assignment. It is well-known that we can take $x(A_i) = e/2^k$ for all $C_i \in C$, and avoid all of these events with positive probability: this can be made constructive using MT. Suppose we run the MT algorithm for up to $n^c$ times its expected number of resamplings. By Markov's inequality, the probability of MT not terminating by then is at most $n^{-c}$. Furthermore, the probability that at the end of this process, some clause $C_i \in \overline{C}$ is violated can be bounded (using part (ii) of Theorem 2.2) by the following:

$$2^{-k} \cdot (1 - e/2^k)^{-\gamma(2^k/e-1)} \le e^{\gamma} \cdot 2^{-k}.$$

Thus, the expected fraction of clauses from $\overline{C}$ that are violated in the end, is at most $2^{-k} e^{\gamma}$. Markov's inequality and a union bound (for a sufficiently large choice of $c$) complete the proof. ∎

Along these lines we aim to develop a general result that can be applied in cases where the number of dependencies are (slightly) beyond the LLL-threshold. For this, suppose we have 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 $G = G_{\mathcal{A}}$ as in the introduction. Let us consider the symmetric case in which $\Pr[A_i] \le p$ for each $i$. Again there are only two types of constructive results known in general, in terms of only allowing a "small" number of the bad events to happen. It is easy to have only about $mp$ of the $A_i$ happen – without any assumptions about $G$ – just by using the linearity of expectation. On the other hand, if the maximum degree of $G$ is at most $1/(ep) - 1$, the conditions of the symmetric LLL and the algorithm of [40], guarantee that we can efficiently ensure that *none* of the $A_i$ happen. Interpolating between these two extremes Theorem 8.3 characterizes the fraction $\lambda$ of events that can be avoided if the maximum degree of $G$ is by a factor of $\alpha > 1$ larger than the LLL-threshold $1/(ep) - 1$. To the best of our knowledge virtually nothing was known (even non-constructively) in this setting. Theorem 8.3 is obtained by using the probabilistic method to construct a sparsely connected core that satisfies the LLL-conditions with a sufficiently large gap. Using the linearity of expectation in the analysis of the LLL-distribution with respect to this core the existence of a good assignment can be proven:

**Definition 8.2.** *For any $\alpha \ge 0$, let $\lambda(\alpha)$ be the smallest number satisfying the following:*
*For any setting of the standard form "variables $\mathcal{P}$ and bad events $\mathcal{A}$" in which*
*(a) the probability of any event $A \in \mathcal{A}$ is at most $p = o(1)$ and*
*(b) the maximum degree in the variable-sharing (dependency) graph is at most $d = \alpha(1/(ep) - 1)$,*
*there exists an assignment to variables in $\mathcal{P}$ that violates at most $(1 + o(1))\lambda(\alpha) \cdot mp$ events.*

**Theorem 8.3.** *The fraction $\lambda(\alpha)$ is upper bounded as follows:*

- $\forall \alpha \le 1 : \lambda(\alpha) = 0$;

- $\forall 1 < \alpha < e: \lambda(\alpha) \le e \ln(\alpha)/\alpha < 1$;

- $\forall \alpha > e : \lambda(\alpha) = 1$.

*Proof.* If $\alpha \le 1$ then the standard symmetric LLL and the MT algorithm ensure that no bad event holds. On the other hand if $\alpha > e$, then consider $m = 1/p$ events given by a single random variable $X$ which is uniformly distributed in $\{1, 2, \ldots, 1/p\}$; the $i$th event holds iff $X = i$. We have $d = 1/p - 1$ here, and exactly one bad event holds with probability one. Thus, we cannot do better than the obvious bound of $mp$ if $\alpha > e$.

For our main case where the constant $\alpha$ satisfies $1 < \alpha < e$, we employ the probabilistic method to first determine a core subset of the bad events, and then apply Theorem 2.2. We give a proof sketch here. Since

$p = o(1)$, $d = \omega(1)$. We will pick a suitable $\epsilon = o(1)$ and an appropriate constant $\beta \geq 1$; $d$ can be assumed sufficiently larger than $\beta$ since $d = \omega(1)$. Choose a random subset $\mathcal{A}'$ of the events $A_i$ by choosing each event independently with probability $(1 - \epsilon)/\alpha\beta$ and then eliminate all events from $\mathcal{A}$ that have more than $d/(\alpha\beta)$ neighbors in $\mathcal{A}'$. The Chernoff bound shows that with probability $1 - \exp(-\Omega(d\epsilon^2/\beta))$ (which is $1 - o(1)$ for a suitable choice of $\epsilon$ and $\beta$), the core $\mathcal{A}'$ has at least an $\frac{(1-\epsilon)^2}{\alpha\beta}$ fraction of the events, and at most an $\exp(-d\epsilon^2/\beta) = o(p)$ fraction of the events get eliminated from $\mathcal{A}$. Therefore, there is a core $\mathcal{A}'$ of size at least $(1 - o(1))m/(\alpha\beta)$ to which all events that are not eliminated (a fraction of $(1 - o(p))$ events) have at most $d/(\alpha\beta)$ neighbors. If we take $x_A \sim \gamma\alpha/d$ for all $A \in \mathcal{A}'$ for a suitable $\gamma \in [0, 1]$ then the core $\mathcal{A}'$ satisfies the LLL-conditions. This is the case if $\gamma$ satisfies

$$\alpha/(ed) < (\gamma\alpha/d) \cdot (1 - \gamma\alpha/d)^{d/(\beta\alpha)};$$

i.e.,

$$1/e < \gamma e^{-\gamma/\beta} \tag{9}$$

suffices for $d$ large enough.

Now we can apply Theorem 2.1 to obtain bounds on the probability of events in the conditional LLL-distribution $D$ that avoids all events in $\mathcal{A}'$. It implies that a random assignment that avoids all core-events in $\mathcal{A}'$ makes an event $A \in (\mathcal{A} \setminus \mathcal{A}')$ that is not eliminated true with probability at most

$$\Pr_D[A] = \Pr[A]/(1 - \gamma\alpha/d)^{-d/(\alpha\beta)} \sim e^{\gamma/\beta} \cdot p. \tag{10}$$

By the linearity of expectation applied to all "non-core" events $A \in (\mathcal{A} \setminus \mathcal{A}')$ and using (10), the expected total number of events $A_i$ that happen in such an assignment is at most

$$(1 + o(1)) \cdot mp \cdot (1 - 1/(\alpha\beta)) \cdot e^{\gamma/\beta}, \tag{11}$$

assuming that (9) holds and that $\epsilon = o(1)$ can be chosen suitably. From (9), we can take $1/\beta = (1 - o(1)) \cdot (1 + \ln(\gamma))/\gamma$. Plugging this into (11), we see that the optimal choice of $\gamma$ is $1/\alpha$. (Any choice of $\epsilon = o(1)$ that satisfies $\exp(-d\epsilon^2/\beta) = o(p)$, will suffice for this argument.) Substituting these choices into (11) yields the theorem. $\blacksquare$

**Theorem 8.4.** *Theorem 8.3 can be made constructive for any $\alpha > 0$ and any efficiently verifiable $\mathcal{A}$ (the verification in this case is allowed to take $\mathrm{poly}(n, m)$ time) that satisfies the conditions from Definition 8.2. That is, there is a $\mathrm{poly}(n, m)$-time randomized algorithm to set values for the variables in $\mathcal{P}$, such that the expected number of events $A_i$ that hold is at most $(1 + o(1))\lambda(\alpha) \cdot mp$.*

*Proof.* If $\alpha \leq 1$ then the theorem follows directly from [40] and for $\alpha \geq e$ a random assignment suffices. For $1 < \alpha < e$ we make the proof from Theorem 8.3 constructive. For it suffices to see that the success probability of the random experiment that creates the core can be made arbitrarily high by choosing $\epsilon$ and $\beta$ accordingly. This makes the probabilistic method used there directly constructive. Finally we use again our main Theorem 2.2 from Section 2 which states that the MT algorithm can be used to efficiently sample the LLL-distribution used in the proof of Theorem 8.3. We simply output the assignment which is produced by the MT algorithm in time $\mathrm{poly}(n, m)$ and the proof of Theorem 8.3 guarantees that the expected number of violated bad events in this assignment is at most $(1 + o(1))\lambda(\alpha) \cdot mp$ as desired. $\blacksquare$

**Remark.** Interestingly, we can use our LLL framework instead to construct the core in the proof of Theorem 8.3. This gives a larger core than what is obtained by uniform random core selection and thus slightly sharper results. Briefly, the idea is as follows. For parameters $\alpha$, $\beta$, and $\gamma$ that are similar to those in that proof, we start with essentially the same random process for constructing the core: for a sufficiently large constant $c_1$ (even something slightly smaller than 3 will suffice), we set $\epsilon = c_1\sqrt{(\ln d)/d}$, and place each event

$A_i$ in the core independently with probability $\theta = (1-\epsilon)/(\alpha\beta)$. Letting $X_i$ denote the (random) number of neighbors that $A_i$ has in the core, note that the expected value of $X_i$ is at most $d\theta = (1-\epsilon)d/(\alpha\beta)$. Now consider the system of bad events $C_i$ (one corresponding to each original event $A_i$): $C_i$ is the event that $X_i > d/(\alpha\beta)$. Note that each $C_i$ depends on at most $d^2$ others; a Chernoff bound shows that for $c_1$ large enough, $\Pr[C_i] \le 1/(ed^3)$. Thus, the LLL shows that there exists a choice of the core that avoids all the $C_i$, with a common value $x = x(C_i)$ for all $i$ such that $x = \Theta(1/d^3)$; run the Moser-Tardos algorithm on the system of bad events $C_i$ to efficiently get a core that avoids all of the $C_i$, and let $N_i$ be the indicator random variable for $i$ not belonging to the core at the end of this run. We now apply Theorem 2.1 to upper-bound the expected size $\sum_i \Pr[N_i = 1]$ of the non-core. Since $N_i$ depends on at most $1 + d + d(d-1) = d^2 + 1$ events $C_i$, Theorem 2.1 gives

$$\Pr[N_i = 1] \le \frac{1-\theta}{(1-x)^{d^2+1}} = (1 + O(1/d)) \cdot (1 - \theta).$$

Thus the expected size of the non-core is at most $(1 + O(1/d)) \cdot mp \cdot (1 - \theta)$ after the above run of Moser-Tardos, similar to what the alteration argument in the proof of Theorem 8.3 gives. We can now proceed (with one more run of Moser-Tardos) as in the proof of Theorem 8.3.

# References

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

[2] N. Alon. Explicit Ramsey graphs and orthonormal labelings. *The Electronic Journal of Combinatorics*, 1:12–8, 1994.

[3] 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.

[4] N. Alon and J. Grytczuk. Breaking the rhythm on graphs. *Discrete Mathematics*, 308(8):1375–1380, 2008.

[5] N. Alon, J. Grytczuk, M. Haluszczak, and O. Riordan. Nonrepetitive colorings of graphs. *Random Structures & Algorithms*, 21(3-4):336–346, 2002.

[6] 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.

[7] N. Alon and M. Krivelevich. Constructive bounds for a Ramsey-type problem. In *Graphs and Combinatorics 13*, pages 217–225, 1997.

[8] N. Alon, C. McDiarmid, and B. Reed. Acyclic coloring of graphs. *Random Structures & Algorithms*, 2(3):277–288, 2007.

[9] N. Alon and P. Pudlak. Constructive lower bounds for off-diagonal Ramsey numbers. *Israel Journal of Mathematics*, 122:243–251, 1999.

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

[11] N. Alon, B. Sudakov, and A. Zaks. Acyclic edge colorings of graphs. *Journal of Graph Theory*, 37(3):157–167, 2001.

[12] M. Andrews. Approximation algorithms for the edge-disjoint paths problem via Raecke decompositions. In *FOCS*, pages 277–286, 2010.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

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

[19] B. Bresar, J. Grytczuk, S. Klavzar, S. Niwczyk, and I. Peterin. Nonrepetitive colorings of trees. *Discrete Mathematics*, 307(2):163–172, 2007.

[20] 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.

[21] K. Chandrasekaran, N. Goyal, and B. Haeupler. Deterministic Algorithms for the Lovász Local Lemma. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, pages 992–1004, 2010.

[22] J. D. Currie. Pattern avoidance: themes and variations. *Theor. Comput. Sci.*, 339(1):7–18, 2005.

[23] 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.

[24] 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.

[25] P. Erdős. Some remarks on the theory of graphs. *Bulletin of the American Mathematical Society*, 53:292–294, 1947.

[26] 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.

[27] 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.

[28] B. Grünbaum. Acyclic colorings of planar graphs. *Israel Journal of Mathematics*, 14:390–408, 1973.

[29] J. Grytczuk. Thue type problems for graphs, points, and numbers. *Discrete Mathematics*, 308(19):4419–4429, 2008.

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

[31] M. Krivelevich. Bounding Ramsey numbers through large deviation inequalities. *Random Structures & Algorithms*, 7(2):145–155, 1995.

[32] A. Kündgen and M. J. Pelsmajer. Nonrepetitive colorings of graphs of bounded tree-width. *Discrete Mathematics*, 308(19):4473–4478, 2008.

[33] 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.

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

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

[36] D. Marx and M. Schaefer. The complexity of nonrepetitive coloring. *Discrete Appl. Math.*, 157(1):13–18, 2009.

[37] 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.

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

[39] 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.

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

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

[42] R. Muthu, N. Narayanan, and C. R. Subramanian. Improved bounds on acyclic edge colouring. *Discrete Mathematics*, 307(23):3063–3069, 2007.

[43] J. H. S. R.L. Graham, B.L. Rothschild. *Ramsey Theory, Second Edition*. Wiley, 1990.

[44] 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.

[45] M. Schaefer and C. Umans. Completeness in the polynomial-time hierarchy: A compendium. *SIGACT News*, 33(3):32–49, Sept. 2002.

[46] 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.

[47] A. Thue. Über unendliche Zeichenreihen. *Norske Vid Selsk. Skr. I. Mat. Nat. Kl. Christiana*, 7:1–22, 1906.

[48] V. G. Vizing. On an estimate of the chromatic class of a p graph (in russian). *Metody Diskret. Analiz.*, 3:25–30, 1964.