The Constructive LLL

Evan Hansen

November 22, 2019

1 What is the Lovász Local Lemma ?

The Lovász Local Lemma is an important theorem in probability which guarantees that under certain hypotheses for a set of random events, there is a non-zero probability of none of them occurring. We will abbreviate Lovász Local Lemma to LLL in much of what follows. It generalizes the trivial statement that if $X_1, \ldots, X_k$ are independent events and $\Pr(X_i) < 1$ for all i then $\Pr(\wedge_i \overline{X}_i) > 0$ by allowing some dependence between the variables.

Definition 1 *Let $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$ be a finite set of random events. We say that a (undirected) graph G with vertex set $[n]$ is a dependency graph for $\mathcal{A}$ if for all $i \in V(G)$, A_i is independent of all subsets of $\{A_j \mid (i,j) \notin E(G)\}$.*

The first-proven version of the lemma is

Theorem 1 ***The Symmetric Lovász Local Lemma***
Again let $\mathcal{A}$ be a finite set of random events. Let $Pr(A_i) = p_i$. If $\max_i p_i = p$ and there exists a dependency graph for $\mathcal{A}$ with maximum degree d then $4pd \leq 1 \implies \wedge_i \overline{A_i} \neq \emptyset$.

We will first motivate the lemma by showing a few classical combinatorial applications, next we will summarize the work done in [1] which provides a constructive proof for (a variant of) the lemma, and finally we will sumarize an algorithmic application found in [5].

2 Some Applications of the Lemma

In this section we motivate why the LLL is useful by using it to sketch a few proofs in combinatorics.

The LLL is used primarily in existential proofs in a method known as the *probabilistic method*. The probabilistic method is a powerful tool, Erdős especially is well-known as having rapidly proved a large number of theorems with it. The idea is that to prove existence of an object with certain properties one shows that the probability of a random object having these properties is non-zero. It follows there must be at least one object with those properties.

We now demonstrate some applications of the LLL .

Before giving the first example we must give some background. We will reference the following theorem to motivate the problem and borrow terminology but do not actually need the theorem itself.

Theorem 2 ***Ramsey's Theorem*** *Let s be a positive integer. Let $q_i \geq 2$ for $i \in [s]$ be given. Then there exists a minimal positive integer $N(q_1, q_2, \ldots, q_s)$ with the following property.*

If $n \geq N(q_1, q_2, \ldots, q_s)$ then for any coloring of the edges of K_n into s colors $c_1, c_2, \ldots, c_s$, there is an $i \in [s]$ and a subset S of $V(G)$ such that $|S| = s_i$ and all edges in the subgraph induced by G on S have the color c_i.

The numbers $N(q_1, q_2, \ldots, q_s)$ are called Ramsey numbers.

We do not prove the theorem as that would digress from the main topic too much.

For instance, the theorem states that there is a number $N(3,3)$ such that if $n \geq N(3,3)$ then any coloring of the edges of K_n with two colors will include at least one monochromatic triangle. In fact $N(3,3) = 6$ which is relatively easy to prove: one need only provide a coloring of the edges of K_5 with two colors such that there is no monochromatic triangle, then show that every coloring of K_6 has a monochromatic triangle. The coloring on K_5 is what is known as a Ramsey graph since it is a constructive proof for a lower bound on the Ramsey number. Unfortunately, relatively few Ramsey numbers are known since finding large Ramsey graphs is very difficult and not many better techniques are known to provide lower bounds.

Nevertheless, we can provide general lower bounds.

Consider trying to provide a lower bound on $N(p,p)$. The following elementary argument provides a weak bound.

Lemma 1 $N(p,p) \geq 2^{p/2}$

Proof: There are $2^{\binom{n}{2}}$ colorings of K_n's edges into red and blue. For any particular $K_p \subseteq K_n$, $2^{\binom{n}{2}-\binom{p}{2}+1}$ of these colorings make K_p monochromatic. Then there are at most $\binom{n}{p}2^{\binom{n}{2}-\binom{p}{2}+1}$ colorings of K_n such that there is at least one monochromatic K_p. But $\binom{n}{p} < n^p/p!$ such that if $n \leq 2^{p/2}$ we find

$$\begin{aligned}\binom{n}{p}2^{\binom{n}{2}-\binom{p}{2}+1} &< (n^p/p!)2^{\binom{n}{2}-\binom{p}{2}+1} \\ &\leq 2^{p^2/2}2^{\binom{n}{2}-\binom{p}{2}+1}/p! \\ &= 2^{\binom{n}{2}+1}/p! \\ &< 2^{\binom{n}{2}}\end{aligned} \tag{1}$$

where in the last line we assumed that $p > 2$, but the $p \leq 2$ case is easily verified since a graph with fewer than 2 nodes has no edges. Since there are fewer colorings that produce at least one monochromatic K_p than total colorings it follows that there always exists some colorings with no monochromatic K_p. QED

Note that the above argument is probabilistic. We could make this apparent with some rephrasing. Simply dividing all the quantities by $2^{\binom{n}{2}}$ makes them probabilities (with uniform distribution) and the bound given for colorings with at least one monochromatic K_p is the union bound. The result shows that the probability is bounded by a value less than 1, and hence there is a positive probability all these "bad events" are avoided.

We can provide an asymptotically better bound by using the LLL .

Lemma 2 *$N(p,p)$ is $\Omega(p2^{p/2})$.*

Proof sketch: Give the edges K_n a random coloring from the uniform distribution. We let

$$\mathcal{A} = \{A_S \mid A_S \text{ is the event that the induced subgraph on } S \subseteq V(K_n) \text{ with } |S| = k \text{ is colored monochromatically}\}. \tag{2}$$

The dependency graph G has $(S, S') \in E(G)$ if S and S' share an edge. For each edge e of the $\binom{k}{2}$ edges in S, there are $\binom{n}{k-2}$ complete k-subgraphs of G that also contain e. Thus $d = \binom{k}{2}\binom{n}{k-2}$ bounds the degree of G. $\Pr(A_S) = p = 2^{-\binom{k}{2}+1}$ for any S so that the LLL implies there exists a coloring of K_n with no monochromatic K_p subgraphs whenever

$$4 \cdot 2^{-\binom{k}{2}+1}\binom{k}{2}\binom{n}{k-2} < 1 \tag{3}$$

The rest of the proof is a little messy. One applies Stirling's approximation and then wrangles the resulting expression until the result is attained.

We provide one more example of a typical use of the LLL .

First, we give the following fact without proof:

If we replace the condition that A_i is independent of every subset of $\{A_j \mid (i,j) \notin E(G)\}$ with $\Pr(A_i \mid \overline{A_{j_1}}, \ldots, \overline{A_{j_k}}) \leq \Pr(A_i)$ for any set of j_i such that $j_i \in \{j_i \mid (i, j_i) \notin E(G)\}$ for all j_i then the statement of the LLL remains true and this variant of the lemma is known as the Lopsided LLL [3].

With this variant of the lemma one can prove the following statement.

Definition 2 *Let A be an $n \times n$ matrix. If σ is a permutation of $[n]$ such that $\{A_{i\sigma(i)}\}_{i\in[n]}$ is pairwise distinct then σ is said to be a Latin transversal of A.*

Lemma 3 *If A is an $n \times n$ matrix in which any given entry appears no more than $k \leq (n-1)/16$ times, then there is a Latin transversal of A.*

Proof sketch: Let σ be drawn uniformly from S_n. Let G be the dependency graph and $V(G) = T$ be defined by the set (i, j, i', j') with $A_{ij} = A_{i'j'}$ yet $i \neq i', j \neq j'$ where $i < i'$. Let $A_{iji'j'}$ denote the event $\sigma(i) = j$ and $\sigma(i') = j'$. We would like to show the probability that $\Pr(\wedge_{(i,j,i',j')\in T}\overline{A_{iji'j'}}) > 0$. Clearly $\Pr(A_{iji'j'}) = 1/n(n-1)$. The dependency graph G has $(i, j, i'j')$ adjacent to (x, y, x', y') when either of (i, j) or (i', j') are in the same row or column as (x, y) or (x', y') since only one entry from each row and each column will be $A_{i\sigma(i)}$ for some i. There are less than $4n$ such entries in the matrix, and each has at most k other entries which it is equal to. This shows the degree of G is less than or equal to $4nk$. The proof is completed by showing that the conditional probability of A_t for any $t \in T$ is at most its own probability when conditioned on all $\overline{A_{t'}}$ such that $(t, t') \notin E(G)$, because then the Lopsided LLL applies since $4dp \leq 4 \cdot 4nk/(n(n-1)) \leq 1$.

We do not complete the proof.

As can be seen, the LLL is a powerful tool for existence proofs.

3 Why a Constructive Form of the LLL is Useful

Although the probabilistic method is powerful, it has a major shortcoming. It does not provide any way to actually find even a single object with the desired properties. This means that these objects cannot be studied in more detail and it can be frustrating to know that such objects exist yet not be able to produce any examples. There are many existential proofs that have been given for which no examples have been found.

The constructive proof of the lemma provides an efficient algorithm that in theory allows such examples to be generated. The proof of correctness for the algorithm does not depend on the LLL itself, so it is also a new constructive proof of the lemma. The algorithm can also be used as a subroutine to solve other problems.

4 A More General Variant of the Lemma

There is a more general variant known as the *Asymmetric* LLL which is the one that the constructive proof/algorithm uses. We present the lemma then show how it is (usually) more powerful than the symmetric version.

Theorem 3 ***The Asymmetric LLL***

Suppose that $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$ *is a set of events.*

Suppose there is a function $\Gamma : \mathcal{A} \to 2^{\mathcal{A}}$ *such that for all* A, A *is independent of* $\mathcal{A} - (\{A\} \cup \Gamma(A))$. *If there exists a function* $x : \mathcal{A} \to (0, 1)$ *such that*

$$\forall A \in \mathcal{A}, Pr(A) \leq x(A) \prod_{A_j \in \Gamma(A_k)} (1 - x(A_j)) \tag{4}$$

then $Pr(\wedge_{i \in [n]} \overline{A_i}) > 0$.

Note that we still refer to the dependency graph G where $\Gamma(A)$ is the set of vertices A is adjacent to. This is well-defined since Γ must be symmetric.

This nearly implies the symmetric version of the lemma since letting $x(A) = \frac{1}{d+1}$ for all $A \in \mathcal{A}$ we note that $|\Gamma(A)| \leq d$ for all A and hence we must only require

$$\Pr(A) \leq p \leq x(A) \prod_{A_j \in \Gamma(A)} (1 - x(A_j)) \leq \frac{1}{d+1}\left(1 - \frac{1}{d+1}\right)^d \tag{5}$$

Since $e^{-1} \geq (1 - \frac{1}{d+1})^d$ it follows $ep(d+1) \leq 1$ suffices. This is usually a stronger statement than our original statement of the symmetric variant of the lemma, but is weaker for $d < 3$. For instance Lemma 2 remains unchanged since it is an asymptotic result where $d \to \infty$ and Lemma 3 remains unchanged since $d = 4nk$ and $n, k \geq 1$ implies $d \geq 4$.

Additionally, it is worth noting that the condition $ep(d + 1) \leq 1$, although not the original form of the lemma, is now more often called the LLL than the form we gave in Theorem 1.

The primary reason why the asymmetric variant is more useful is that it allows certain events to be fairly likely if this is "counterbalanced" by less dependence between events and other events being less likely, a flexibility not given in the symmetric bounds.

When we refer to the LLL going forward we will always mean the Asymmetric LLL unless otherwise stated.

5 The Algorithm Given by Moser and Tardos

5.1 Introduction

To make the task of providing an algorithm for the above possible, we need to search through the probability space within which the events in $\mathcal{A}$ are defined.

We therefore assume there is a given finite set $\mathcal{P}$ of random variables which determine the events in $\mathcal{A}$. For each $A_i \in \mathcal{A}$ we define $\text{vbl}(A_i)$ to be the set of random variables in $\mathcal{P}$ that A_i is determined by and assume this is given to the algorithm as well. Also, since we are trying to avoid all of the events in $\mathcal{A}$ we will say an event $A_i \in \mathcal{A}$ is "violated" by some assignment to the random variables in $\mathcal{P}$ if it occurs under this assignment.

We will also let $m = |\mathcal{A}|$ and $n = |\mathcal{P}|$.

In their paper, Moser and Tardos provide a randomized algorithm to generate a set of values for the variables in $\mathcal{P}$ such that none of the events in $\mathcal{A}$ are violated.

The method is very simple, but to be precise we provide pseudocode.

Algorithm 1 MT Algorithm

```
procedure LLL(P, A)
    for all P ∈ P do
        v_P ← a random evaluation of P
    end for
    while ∃A_i ∈ A which is violated in the current assignment do
        Let A ∈ A occur
        for all P ∈ vbl(A) do
            v_P ← a new random evaluation of P
        end for
    end while
    output v⃗_P
end procedure
```

In the paper the following statement is proven:

Theorem 4 *If the algorithm terminates it is clearly correct. Additionally, if all the hypotheses of the Asymmetric LLL are met the MT Algorithm runs in time* $O\big(\sum_{i\in[n]} \frac{x((A_i)}{1-x(A_i)}\big)$ *with respect to the number of calls to an oracle which resamples all the random variables that some event* $A_i \in \mathcal{A}$ *depends on.*

The analysis of the algorithm they present is non-trivial, but we provide a (brief) overview.

5.2 Execution Logs and Witness Trees

The analysis centers around bounding $\mathbb{E}[N_A]$ for all $A \in \mathcal{A}$ where N_A denotes the number of times that A is resampled.

Definition 3 *We define the execution log of a run of the algorithm to be a function* $C : \mathbb{N} \to \mathcal{A}$ *where* $C(i)$ *is the event* A *that was resampled on the* i^{th} *resampling step of the algorithm.* C *is not defined for values larger than the number of iterations of the algorithm.*

Definition 4 *We define a witness tree to be any finite rooted tree with vertex set a subset of* $\mathcal{A} \times \mathbb{N}$ *such that if* (u, k) *is a child of* (v, l) *then either* $u = v$ *or* $(u, v) \in G$*, where* G *is the dependency graph as usual.*

Note: We used $\mathcal{A} \times \mathbb{N}$ simply because we want to allow repetitions of elements of $\mathcal{A}$. From now on we will refer to the vertices of a witness tree simply by their first coordinate, i.e. like a multiset.

It will be useful to define $\Gamma^+(v)$, the inclusive neighborhood, as $\Gamma(v) \cup \{v\}$.

We now associate a witness tree with each time-step of the algorithm using the execution log C. We define $\tau_C^{(t)}(t)$ to be an isolated root $C(t)$. We inductively define $\tau_C^{(k)}(t)$ for $k < t$ by adding $C(k)$ to the tree $\tau_C^{(k+1)}(t)$ as the child of the node v of greatest depth such that $C(k) \in \Gamma^+(v)$ (ties broken arbitrarily). If no such vertex exists in $\tau_C^{(k+1)}(t)$ then we set $\tau_C^{(k)}(t) = \tau_C^{(k+1)}(t)$. We define $\tau_C(t) = \tau_C^{(1)}(t)$. We say that "$\tau$ appears in C" if $\tau = \tau_C(t)$ for some t.

The paper goes on to define a proper witness tree as one in which all nodes have pairwise distinct children. It then shows any witness tree which appears in C must be proper by proving the set of nodes at a given depth are independent.

We then do two things:

Firstly, we bound the probability that any given proper witness tree τ appears in C.

In the paper it is proven $\prod_{v \in V(\tau)} \Pr(v)$ is such a bound.

The following is an alternative model to how the algorithm runs which is a sketch of how the bound is proven:

1. The algorithm visits the nodes of $\tau_C(t)$ in order from maximum depth $\rightsquigarrow$ the root (all nodes of a given depth visited before any node of a smaller depth).

2. When the algorithm visits a node, it checks if it is violated, and if it is then it resamples all the variables in $\mathcal{P}$ that determine that event.

The observation is that whenever the algorithm checks if the node it visits is violated under its current valuation the answer is yes. Further the underlying variables that determine that event had been sampled randomly, and since all the vertices at the same level of the tree are independent, the random sampling was only used once (since at all lower levels where an event was violated these were resampled), and hence not correlated with whether other nodes were violated. So the probability of all the nodes being violated is simply the product of their individual probabilities.

Secondly, we analyze a seemingly unrelated probabilistic process to generate proper witness trees because it will happen to cancel later.

The paper describes a random Galton-Watson process that grows proper witness trees. This process iteratively grows the tree. In each time step, for each vertex v added in the last step, each $u \in \Gamma^+(v)$ is added as a child of v with probability $x(u)$. The process halts whenever there is a round with no children added.

We define $x'(A) = x(A) \prod_{B \in \Gamma(A)} (1 - x(A))$. Note this is the quantity that appears in the inequality required by the lemma.

The paper shows through just a few steps that the probability this process randomly generates some given proper witness tree τ is

$$p_\tau = \frac{1 - x(A)}{x(A)} \prod_{v \in V(\tau)} x'(v). \tag{6}$$

5.3 Finishing the Analysis

The paper then uses the following string of inequalities to bound $\mathbb{E}[N_A]$. Let $\mathcal{T}_A$ denote the set of all proper witness trees rooted at A.

$$\mathbb{E}[N_A] = \sum_{\tau \in \mathcal{T}_A} \Pr(\tau \text{ appears in the log } C) \leq \sum_{t \in \mathcal{T}_A} \prod_{v \in V(\tau)} \Pr(v) \leq \sum_{\tau \in \mathcal{T}_A} \prod_{v \in V(\tau)} x'(v). \tag{7}$$

The first equality follows from the fact that for every occurrence of A in the execution log there is one associated witness tree rooted at A that occurs in the log, the first inequality follows from the calculation bounding the probability a given τ occurs in the log, and the last step follows from the hypothesis in the Asymmetric LLL .

Then it finishes with

$$\mathbb{E}[N_A] \leq \sum_{\tau \in \mathcal{T}_A} \prod_{v \in V(\tau)} x'(v) = \frac{x(A)}{1 - x(A)} \sum_{\tau \in \mathcal{T}_A} p_\tau \leq \frac{x(A)}{1 - x(A)} \tag{8}$$

where the equality is from the calculation of p_τ and the last inequality follows from the fact that $\mathcal{T}_A$ is the probability space for mutually exclusive events p_τ.

Since every step of the algorithm is to resample some event $A \in \mathcal{A}$ it follows that

$$\begin{aligned} \mathbb{E}[\#\text{ steps of MT Algorithm}] &= \sum_{A \in \mathcal{A}} \mathbb{E}[N_A] \\ &\leq \sum_{A \in \mathcal{A}} \frac{x(A)}{1 - x(A)} \end{aligned} \tag{9}$$

which concludes the proof for the runtime of the algorithm.

Remark 1 *If there exists $\varepsilon > 0$ not dependent on m such that $1 - x(A) \geq \varepsilon$ for all $A \in \mathcal{A}$ then $\sum \frac{x(A)}{1-x(A)}$ is bounded by $m(1-\varepsilon)/\varepsilon$ which makes the algorithm $O(m)$.*

5.4 Parallel Variant

The algorithm can be parallelized as follows.

Algorithm 2 Parallel MT Algorithm

procedure LLL($\mathcal{P}, \mathcal{A}$)
  for all $P \in \mathcal{P}$ **do**
    $v_P \leftarrow$ a random evaluation of P
  end for
  while $\exists A_i \in \mathcal{A}$ which is violated in the current assignment **do**
    Find $S \subseteq \mathcal{A}$ a maximal independent set in the induced subgraph of G on the set of violated events
    for all $A \in S$ **do**
      for all $P \in \text{vbl}(A)$, in parallel **do**
        $v_P \leftarrow$ a new random evaluation of P
      end for
    end for
  end while
  output $\vec{v_P}$
end procedure

Theorem 5 *Assume the same hypotheses as for the sequential version of the MT algorithm. If additionally there exists $\varepsilon > 0$ such that*

$$\forall A \in \mathcal{A}, Pr(A) \leq (1-\varepsilon)x(A) \prod_{A_j \in \Gamma(A_k)} (1 - x(A_j)) \tag{10}$$

then the parallel MT algorithm terminates in expected time $O(\frac{1}{\varepsilon} \log \sum_{A \in \mathcal{A}} \frac{x(A)}{1-x(A)})$.

The proof is very similar to that of the sequential algorithm. We use the term "round" to refer to a run of the while loop. We now define the log to be a function $C : \mathbb{N} \to \mathcal{A}$ such that all the events resampled in a given round k are listed before all events resampled in round $k+1$ (and there is no gap).

We now define S_j to be the segment of the log that corresponds to round j of the algorithm ($S_j \subseteq \mathbb{N}$). Then we define witness trees the same as before, with our iterative definition for $\tau_C^{(j)}(t)$. We define the depth of a tree to be the maximal depth of any vertex in it (note that a tree with only a root is considered depth 0).

We outline the proof of Theorem 5.

First we require

Lemma 4 *If $t \in S_j$ then the depth of $\tau_C(t)$ is $j-1$.*

Proof: Consider growing the tree by our iterative definition. We can divide this process into rounds, where round k corresponds to the time we are adding vertices from $C(S_k)$; note that we start the process in round j and finish when round 1 is complete. Note all events added to the tree in a given round are independent by construction of S in the algorithm. It follows the depth of the tree grows by at most one in each round since we only add nodes as children if they are adjacent in the dependency graph. Then suppose there is a round k in which the depth of the tree does not grow. Then let v be a maximal-depth node in the tree at the end of round $k+1$. Because v appears in the tree during round $k+1$, it must have been violated at the beginning of round $k+1$ of the algorithm. Then since the tree does not grow in depth during round k it follows that no event that shares any variable with v was resampled during round k of the algorithm. But then v must have also been violated during round k of the algorithm. It follows that the set S used during round k of the algorithm was not maximal, a contradiction. This completes the proof because the depth of the tree must then grow by exactly one for each of the j rounds (except round j because we defined the depth of the tree with only root to be 0).

Note that a consequence of the above lemma is that every witness tree in round k has at least k vertices.

Using a similar string of inequalities as in the proof of Theorem 4 we can find the probability of the algorithm running for k rounds is at most

$$(1-\varepsilon)^k \sum_{A \in \mathcal{A}} \frac{x(A)}{1-x(A)} \tag{11}$$

and the bound falls out from comparing the expectation with the Taylor series for the log function.

5.5 Derandomization

The paper also presents a derandomization of the algorithm which requires a few more conditions. The derandomization of the algorithm is interesting because the algorithm is guaranteed to terminate in polynomial time w.r.t. an appropriate measure of the size of the problem.

Theorem 6 *Let $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ be a finite set of mutually independent random variables, with P_i taking values in just a finite domain D_i. Let $\mathcal{A}$ be a set of events determined by $\mathcal{P}$. Let $s = m+n+\sum_i |D_i|$ be the "size" of the problem. Suppose also that there is an oracle $\mathcal{O}$ that can compute the conditional probability $Pr(A \mid \forall i \in I : P_i = v_i)$ in time polynomial in s for any $I \subseteq [n]$ and v_i's in their respective D_i's. Suppose that the maximum degree of the dependency graph G is bounded by some k. Then if also*

$$\forall A \in \mathcal{A} : Pr(A) \leq (1-\varepsilon)x(A) \prod_{B \in \Gamma(A)} (1-x(B)), \tag{12}$$

for some $x : \mathcal{A} \to (0,1)$ and $\varepsilon > 0$, a deterministic algorithm can find an evaluation of the variables in $\mathcal{P}$ such that no event in $\mathcal{A}$ is violated.

We do not prove this theorem.

The idea for the algorithm is to greedily generate a source of valuations for $\mathcal{P}$ that minimizes the expected number of witness trees, then run the standard algorithm with this sequence of valuations in place of a random number generator.

5.6 Conclusion

The constructive algorithm given by Moser and Tardos is useful in constructing examples of combinatorial structures which have been proven to exist and in the development of new algorithms based off of proofs involving the LLL , we give an example of an algorithmic application in a later section.

However, there are some issues not addressed by the paper: the authors note that the parallel algorithm may be bottlenecked by calls to Luby's algorithm for finding maximal sets of independent subproblems that can be solved independently; the paper does not address the runtime when one instead only finds approximate maximal independent sets.

In [5] it is noted that there are sometimes other issues with attempting to apply the MT algorithm. Namely:

(a) There may be super-polynomially many events we wish to avoid w.r.t. the number of underlying variables.

For instance, in Lemma 2 there are $\binom{n}{p}$ events (one event for each K_p subgraph of K_n), yet only $\binom{n}{2}$ underlying variables (the coloring of each edge of K_n).

A further analysis found in [5] that we do not discuss further shows that in fact if

$$\forall A \in \mathcal{A}, \Pr(A) \leq (1-\varepsilon)x(A) \prod_{B \in \Gamma(B)} (1 - x(B)) \tag{13}$$

holds for some $\varepsilon > 0$ then letting $\delta = \min x'(A)$ over $A \in \mathcal{A}$ the MT algorithm actually terminates with expected runtime less than $-n \log(\delta) \max_A \frac{1}{1-x(A)}$.

This shows that some applications with super-polynomially events can still be solved efficiently with this algorithm, and in [5] it is claimed that in most applications known $\log(\delta)$ is $O(n \log n)$.

(b) The problem of certifying no event is violated or outputing such an event may be computationally hard.

For sake of completeness, note that in [1] an extension of the algorithm to a lopsided variant of the Asymmetric LLL is also given.

6 An Algorithmic Application

In the following we describe some results found in [5] that used the constructive algorithm described above.

The Santa Claus Problem Consider the following algorithmic problem: Santa claus has n presents he needs to deliver to m children. Each present has an intrinstic value p_j. Each child i has a utility value p_{ij} for present j, and $p_{ij} = p_j$ or 0 for all i, j. Santa Claus doesn't want any child to be unhappy, so needs to determine an assignment of presents to children that maximizes the minimum utility that any child has for their presents, which is the sum of the valuations that child has for the individual presents they received. The Santa Claus Problem is known to be NP-complete, the goal is to give good approximation algorithms for the problem. It had already been shown that an LP-relaxation of the problem gave an n-approximation algorithm, and then later it was shown that the integrality gap of this LP-relaxation was a constant. One might expect this would lead directly to a constant-factor approximation algorithm, but the proof used the Assymetric LLL in

a non-constructive way on an exponential number of events. In [5] the constructive version of the algorithm is used to provide a polynomial-time algorithm for the Santa Claus Problem with constant approximation factor.

References

[1] *Robin A. Moser and Gábor Tardos. A constructive proof of the general LLL* . arXiv:0903.0544, 2009.

[2] *Robin A. Moser. A constructive proof of the LLL* . arXiv:0810.4812, 2008.

[3] *Paul Erdős and Joel Spencer. Lopsided LLL and Latin transversals.* Courant Institute, 251 Mercer Street, New York, NY 10012, USA, 1990.

[4] *J.H. Van Lint and R.M. Wilson. A Course In Combinatorics (Second Edition).* Cambridge University Press, The Edinburgh Building, Cambridge, UK, 2001.

[5] *Bernard Haeupler, Barna Saha, and Aravind Srinivasan. New Constructive Aspects of the LLL* . arXiv:1001.1231, 2011.