Approximation Algorithms Workshop June 13-17, 2011, Princeton

Lecture: New Constructive Aspects of the Lovász Local Lemma, and their Applications — June 15, 2011

Aravind Srinivasan *Scribe: Ioana Bercea*

1 Introduction

The Lovász Local Lemma (LLL) is a powerful combinatorial result used in many applications to show that a particular event happens with positive probability. It is known that if a large number of independent events each happen with positive probability, then there is positive (possibly exponentially small) probability that they all happen at the same time. The LLL, first proved by Erdős and Lovász [EL75], extends this result to the case of rare dependencies. Specifically, the LLL in its general form is the following [AS08]:

Theorem 1. *Let $\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_n$ be events in an arbitrary probability space. A directed graph $G = (V, E)$ on the set of vertices $V = \{1, 2, \cdots, n\}$ is called a dependency graph for the events $\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_n$ if for each $i, 1 \leq i \leq n$, the event $\mathcal{A}_i$ is mutually independent of all the events $\{\mathcal{A}_j : (i,j) \notin E\}$. Suppose that $G = (V, E)$ is a dependency digraph for the above events and suppose there are real numbers $x_1, \cdots, x_n$ such that $0 \leq x_i < 1$ and $\Pr[\mathcal{A}_i] \leq x_i \prod_{(i,j)\in E}(1 - x_j)$ for all $1 \leq i \leq n$. Then*

$$\Pr\left[\bigwedge_{i=1}^{n} \overline{\mathcal{A}_i}\right] \geq \prod_{i=1}^{n}(1 - x_i).$$

1.1 The algorithmic perspective

The LLL is an important tool used in the probabilistic method applied in randomized algorithms but also in a variety of approximation algorithms. It is in this latter context that we will discuss it. The main problem that arises in any endeavor to apply it algorithmically is its non-constructive aspect. Also, the LLL states that there is an assignment such that a particular event happens with positive probability. It does not, however, guarantee that the probability of it happening is not exponentially small in the dimensions of the problem. In this talk, we will describe the celebrated result of Moser and Tardos [MT10] that address these two problems and we will improve their result. This is joint work done with Bernhard Haeupler (MIT) and Barna Saha (UMD). For details, you can consult the original paper [HSS10].

First, let us present the usual framework in which the algorithmic versions of the LLL is used. Let $\{X_1, X_2, \ldots X_n\}$ be **independent** random variables, and let $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots \mathcal{A}_m\}$ be a collection of m "bad" events, each determined by a subset of X_j's. Let us also consider the ubiquitous version of the framework, with neighborhood relation Γ on $\mathcal{A}$.

The main **question** arising from this framework is whether all $\mathcal{A}_i$ can be avoided **simultaneously**. The problem becomes therefore, of finding an algorithm that outputs an assignment to all X_j that guarantees all the "bad" events are avoided. Note that the output size is n. Also note that this framework is slightly more restrictive than the general framework in the sense that one can always transform the general setting into this framework by considering indicator variables for each of the "bad" events.

1.2 Results

In this context, the main results of [HSS10] we are going to discuss are:

- A randomized, poly(n)-time algorithm that work for all known applications of the LLL, even when $m \gg \text{poly}(n)$, if we allow a tiny slack in the LLL condition.
- An application that algorithmically involves avoiding "most" of the bad events. Such a result is useful in MAX SAT-like problems (as opposed to SAT) in which there is a dichotomy between existing approaches. On one hand, in the case in which the dependency is small enough, LLL can be used and we can avoid all bad events. On the other hand, if the overlap is larger than the LLL threshold, the only known bound on the number of bad events is the trivial one given by computing the linearity of expectation on random assignments (in which each clause is violated with probability 2^{-k} in MAX k-SAT).

2 Symmetric LLL

Before we begin discussing the main results, as a warm-up, we will present the symmetric version of the LLL. Suppose:

- $\max_i \Pr[\mathcal{A}_i] \leq p$, and
- each $\mathcal{A}_i$ has $\leq D$ neighbors in the dependency graph.

Then, $e \cdot p \cdot (D+1) \leq 1$ implies $\Pr[$ no $\mathcal{A}_i$ holds $] > 0$.

Note: The typical case in which the symmetric version is applied is that in which $D \ll m$, in which case the problem we encounter is that $Pr[\bigwedge_i \overline{\mathcal{A}_i}]$ is inevitably small. Since the dependency graph is sparse, we can find a large independent set I of $\mathcal{A}_i$ such that $|I| \geq m/(D+1)$. We therefore get that $\Pr[\bigwedge_i \overline{\mathcal{A}_i}] \leq \Pr[\bigwedge_{i \in I} \overline{\mathcal{A}_i}] = (1-p)^{m/(D+1)} \approx exp(-mp/D)$.

2.1 Domatic partitions

Now we are going to present an application of the symmetric LLL to the **domatic number problem**, as can be found in [FHKS02]. A set of vertices in a graph is a dominating set if every vertex outside the set has a neighbor in the set. The domatic number problem is that of partitioning the vertices of a graph into the maximum number of disjoint dominating sets. Let the domatic number of a graph G be denote by $D(G)$. As an interesting application of this problem to the problem of wireless coordination, please consult [CJBM02].

Another way of viewing the domatic number problem is that of "coloring" vertices with a maximum number of colors such that, for any vertex v, all colors are visible in $N^+(v)$= inclusive neighborhood of vertex v. On the other hand, we can think of assignments in the LLL as being coloring of the vertices in the dependency graph. Using this approach, [FHKS02] provide an algorithm with a logarithmic approximation threshold, marking the rare incident of a maximization problem having such a threshold and no better.

Before we begin describing the algorithm, however, let us remark that every graph G satisfies $D(G) \geq 1$ and unless G contains an isolated node, $D(G) \geq 2$ (consider a spanning tree in which the odd levels consist of one dominating set and the even levels of another one, therefore producing at least 2 dominating sets). At the same time, let $(\delta, \Delta) = (\min, \max)$ degrees in G. Then we get that $D(G) \leq \delta + 1$ since a node of minimum degree must have some neighbor (or itself) in each of the disjoint dominating sets.

2.2 Domatic partitions assuming d-regularity

Now we can proceed to present a restricted version of the algorithm in [FHKS02] that achieves a $3 \ln d$-approximation when G is d-regular.

Independently color each vertex randomly with one of $\ell \sim d/(3 \ln d)$ colors. For each (v, c) pair, define a Boolean variable $\mathcal{A}_{v,c}$ to be true if there is no vertex of color c in $N^+(v)$, and to be false otherwise. Note that the events $\mathcal{A}_{v,c}$ are "bad" events for us: if $\mathcal{A}_{v,c}$ holds for some pair (v, c), then the coloring is not a domatic partition.
For each pair (v, c), $p = \Pr[\mathcal{A}_{v,c}] = (1 - 1/\ell)^{d+1} \sim 1/d^3$. Now let us fix a $\mathcal{A}_{v,c}$ an let us look at its dependence. We note that each event $\mathcal{A}_{v,c}$ is independent of all events $\mathcal{A}_{w,c'}$ with $dist(v, w) \geq 3$ since vertices of distance at least 3 from v are completely irrelevant for v. Since the number of vertices at distance ≤ 2 is $< d^2 + 1$ and the number of possible colors c' is $\leq \ell$, we get that $D < d^3/(3 \ln d)$. We also get that $e \cdot p \cdot (D + 1) \leq 1$ and therefore, a good coloring exists.

2.3 Improving the Constant

Improving the constant from "3" to "1" can be done by using an **iterated** approach of the LLL, a powerful methodology which has been successfully used in a variety of breakthrough results. For more information on this subject, we recommend [MR02].

The main difference from regular methods is that we apply a "slow"(two-stage) partitioning that helps prune the dependencies in applying the LLL, leading to our improvement. The first partition is small-sized, but has properties much stronger than being domatic; the second partition refines the first, and crucially benefits from these useful properties of the first partition.

Specifically, we will color the vertices in two stages. Let χ_1 be the first coloring and χ_2 the second one. In the end, our coloring will be a pair of the two colors, one assigned by χ_1, and the other component assigned by χ_2. In the first stage, our goal is to color the vertices such that we can obtain a good range for the fraction of vertices in $N^+(u)$ that receives a particular color c, for any such color Note that this property is much stronger than the coloring being domatic, since being domatic just means that every color is visible in $N^+(u)$ and doesn't say anything about the fraction of neighbors that exhibit any color.
By applying the LLL in the first stage, we ensure that we obtain such a "good" coloring. Then, in the second coloring, we can consider $\mathcal{B}_{u,c_1,c_2}$ to be the bad event that there is no vertex of color (c_1, c_2) in $N^+(u)$. We will use the LLL to show that all these bad events can be avoided with positive probability. The main ingredient is bounding the number of dependencies of a fixed $\mathcal{B}_{u,c_1,c_2}$. Let $N^+_{u,c} = \{v \in N^+(u) : \chi_1(u) = c\}$. Then we get that $\mathcal{B}_{u,c_1,c_2}$ simply says that all the elements of $N^+(u, c_1)$ got a $\chi_2(\cdot)$ value different from c_2. Thus, we can check that $\mathcal{B}_{u,c_1,c_2}$ only depends on the events in:

$$S(u, c_1, c_2) = \{\mathcal{B}_{v,c_1',c_2'} : v \in N^+(N^+_{u,c_1}) \text{ and } c_1' = c_1\}.$$

In the analysis, we get that $|S(u, c_1, c_2)|$ is only $\Delta^{1+o(1)}$ as compared to the previous dependence of $\Delta^{3+o(1)}$, each of the constraints in the definition of $S(u, c_1, c_2)$ saving us a factor of $\Delta^{1-o(1)}$.

3 Asymmetric LLL

Now let us get back to the general version of LLL and review its statement. We have that if $\exists x : \mathcal{A} \to (0, 1)$ such that

$$\forall i : \Pr[\mathcal{A}_i \le x(\mathcal{A}_i) \prod_{\mathcal{A}_j \in \Gamma(\mathcal{A}_i)} (1 - x(\mathcal{A}_j)),$$

then $\Pr[\bigwedge_i \overline{\mathcal{A}_i}] \ge \prod_i (1 - x(\mathcal{A}_i)) > 0$.

The asymmetric version has numerous applications, some of them including:

- (hyper-)graph colorings and Ramsey numbers
- routing ([LMR94])
- LP integrality gaps ([Fei08], [LLRS01])
- edge-disjoint paths ([And10]).

3.1 Short proof of the general LLL

We will review a short proof of the lemma, as it is given in [AS08], because it will give us insight into the main theorem of [FHKS02].

The idea of the proof is to split $\Pr[\bigwedge_i \overline{\mathcal{A}_i}]$ into a product of conditional probabilities that we can prove a lower bound on. Notice that:

$$\Pr[\bigwedge_{i=1}^{n} \overline{\mathcal{A}_i}] = (1 - \Pr[\mathcal{A}_1]) \cdot (1 - \Pr[\mathcal{A}_2 \mid \overline{\mathcal{A}_1}]) \cdots (1 - \Pr[\mathcal{A}_n \mid \bigwedge_{i=1}^{n-1} \overline{\mathcal{A}_i}]).$$

Hence, the first step is proving by induction on s, that for any $S \subseteq \{1, \cdots n\}, |S| = s < n$ and any $i \notin S$,

$$\Pr[\mathcal{A}_i \mid \bigwedge_{j \in S} \overline{\mathcal{A}_j}] \le x_i.$$

The base case is trivial so now let's assume, for the inductive step, that the above holds for all $s' < s$ and prove that it also holds for s. Let $S_1 = \{j \in A : (i,j) \in E\}$ and $S_2 = S \backslash S1$. For clarity, we assume that $G = (V, E)$ is the dependency graph. Then:

$$Pr[\mathcal{A}_i \mid \bigwedge_{j \in S} \overline{\mathcal{A}_j}] = \frac{Pr[\mathcal{A}_i \wedge (\bigwedge_{j \in S_1} \overline{\mathcal{A}_j}) \mid \bigwedge_{l \in S_2} \overline{\mathcal{A}_l}]}{Pr[\bigwedge_{j \in S_1} \overline{\mathcal{A}_j} \mid \bigwedge_{l \in S_2} \overline{\mathcal{A}_l}]}.$$

Notice that since $\mathcal{A}_i$ is mutually independent of the events in $\{\mathcal{A}_l : l \in S_2\}$, we get that the numerator is upper bounded by $\Pr[\mathcal{A}_i \mid \bigwedge_{l \in S_2} \overline{\mathcal{A}_l}] = \Pr[\mathcal{A}_i] \le x_i \cdot \prod_{(i,j) \in E} (1 - x_j)$.

On the other hand, we can bound the denominator by the induction hypothesis. Let $S_1 = j_1, j_2, \cdots, j_r$. If $r = 0$, the the denominator is 1 and our conclusion follows. Otherwise:

$$\begin{aligned}
\Pr[\overline{\mathcal{A}_{j_1}} \wedge \overline{\mathcal{A}_{j_2}} \wedge \cdots \wedge \overline{\mathcal{A}_{j_r}} \mid \bigwedge_{l \in S_2} \overline{\mathcal{A}_l}] &= (1 - Pr[\mathcal{A}_{j_1} \mid \bigwedge_{l \in S_2} \overline{\mathcal{A}_l}]) \cdot (1 - Pr[\mathcal{A}_{j_2} \mid \overline{\mathcal{A}j_1} \wedge \bigwedge_{l \in S_2} \overline{\mathcal{A}_l}]) \cdots \\
&\cdots (1 - Pr[\mathcal{A}_{j_r} \mid \overline{\mathcal{A}_{j_1}} \wedge \cdots \wedge \overline{\mathcal{A}_{j_{r-1}}} \wedge \bigwedge_{l \in S_2} \overline{\mathcal{A}_l}]) \\
&\ge (1 - x_{j_1})(1 - x_{j_2}) \cdots (1 - x_{j_r}) \\
&\ge \prod_{(i,j) \in E} (1 - x_j)
\end{aligned}$$

Combining the two inequalities gives us the desired bound and completes the proof.

4 The Algorithmic Framework

In trying to make the proof constructive, it is natural to consider the following trivial algorithm:

```
repeat
    pick random assignments for all $X_j$
until no $\mathcal{A}_i$ holds
```

The LLL theorem tells us that is the LLL conditions holds, then the algorithm finds a satisfying assignment with positive probability. However, the running time might be exponential in m, let alone n.

4.1 The Moser-Tardos breakthrough

Given such a situation, several algorithmic versions of the LLL have emerged, ([Bec91], [Alo91], [MR98], [CS00], [Sri08], [Mos09]) culminating in the **Moser-Tardos(MT) breakthrough** [MT10]:

```
start with an arbitrary assignment
while $\exists$ event $\mathcal{A}_i$ that holds do
    assign new random values to the variables of $\mathcal{A}_i$
```

The analysis of the algorithm is based on the following theorem from [MT10]:

Theorem 2. *If the LLL-conditions hold, then the above algorithm finds a satisfying assignment within an expected* $\sum_i \frac{x(\mathcal{A}_i)}{1-x(\mathcal{A}_i)}$ *iterations.*

Proof. In order to prove this, [MT10] describe the process of obtaining such an assignment as a branching process that corresponds to decisions of resampling some variables. The log of the execution is the list of events in $\mathcal{A}$ as they have been selected for resampling at each step. They associate with each resampling step a witness tree that can serve as a "justification" for the necessity of the correction step. A particular witness tree occurs in the log of the execution if there exists a step t such that the witness tree is rooted at it. The idea of the proof is to upper bound the number of resamplings each event $A \in \mathcal{A}$ needs (remember that the process finishes when no such resampling is needed). The authors do that in two steps:

- first, they upper bound the probability that a fixed witness tree occurs in a random log produced by the algorithm
- second, they obtain a probability distribution over different witness trees. For each such witness tree, they consider a Galton-Watson branching process that attempts to construct it. They start with the singleton event A as the root and then, in each round, they add to the vertex v a child for the event B (v or adjacent to v in the dependency graph) with probability $x(B)$ or skip it with probability $1 - x(B)$.

With these two ingredients, the authors are able to upper bound the expectation of the number of occurrences of A in the log simply by summing the bounds for a fixed witness tree rooted at A on the probabilities of the occurrences of the different witness trees. The bound they obtain for a particular A is $\frac{x(A)}{1-x(A)}$. Summing over all such A therefore gives us the upper bound on the expected number of total resampling. □

4.2 LLL-distribution and the MT- algorithm

One observation that needs to be made is that the MT algorithm doesn't just provide an assignment in which all bad events are avoided. In fact, it provides a randomly chosen assignment from the distribution that is obtained when one conditions on no bad events holding. [HSS10] uses this observation to introduce a new proof-concept based on the **(conditional) LLL-distribution** $\mathcal{D}$.

Some very useful properties are known for $\mathcal{D}$:

Theorem 3. $\Pr_{\mathcal{D}}[B] = \Pr[B \mid \bigwedge_i \overline{\mathcal{A}_i}] \leq \Pr(B) \cdot \left(\prod_{\mathcal{A}_j} \in \Gamma(B)(1 - x(\mathcal{A}_j)) \right)^{-1}$

Proof. Looking at the proof for the general LLL, one notices that it can the induction can be applied to any event B conditioned on a the bad events not happening, where B is determined by X_i's. Consider any $S \subseteq \{1, \cdots, n\}, |S| = s < n$, then let $S_1 = S \cap \Gamma(B)$ and $S_2 = S \backslash S_1$. Then we get as before that:

$$\Pr[B \mid \bigwedge_{j \in S} \overline{\mathcal{A}_j}] = \frac{\Pr[B \wedge (\bigwedge_{j \in S_1} \overline{\mathcal{A}_j}) \mid \bigwedge_{l \in S_2} \overline{\mathcal{A}_l}]}{\Pr[\bigwedge_{j \in S_1} \overline{\mathcal{A}_j} \mid \bigwedge_{l \in S_2} \overline{\mathcal{A}_l}]}.$$

Following the calculations just as before, we get that the numerator is smaller than $\Pr[B]$ and the denominator is bigger than $\bigwedge_{A_j \in \Gamma(B)} (1 - x(\mathcal{A}_j))$.

□

Informally, if B depends "not too heavily" on the events $\mathcal{A}_j$, then the probability places on B by $\mathcal{D}$ is "not much more than" the unconditional probability $\Pr(B)$. Such bounds in combination with further probabilistic analysis can be used to give interesting(nonconstructive) results. The main result of [HSS10] is that the MT algorithm has an output distribution that "approximates" the LLL-distribution $\mathcal{D}$: in that for every B, the *same* upper bound as above holds as well. This can be used to make the probabilistic proofs that use the LLL-condition constructive.

In particular, we can modify the analysis of the MT algorithm to upper bound the probability that an event B was true at least once during the execution of the MT algorithm. Note that B does not have to be in $\mathcal{A}$. The witness trees that certify the first time B become true are the ones that have B as a root and all non-root nodes from $\mathcal{A} \backslash B$. Just as in the MT analysis, we calculate the probability that the Galton-Watson process builds a witness tree rooted at B and then calculate expected number of these witness trees via a union bound. The upper bound that we get is exactly $\Pr[B] \cdot (\prod_{\mathcal{A}_j \in \Gamma(B)} (1 - x(\mathcal{A}_j)))^{-1}$, where the term $\Pr[B]$ accounts for the fact that the root-event B has to be true as well.
Such insight will prove significant in developing a more refined version of the MT algorithm, as it is described in the following sections.

4.3 LLL- Applications with exponentially many events

For now, however, let us backtrack and look at the greater picture. The main question that arises in the algorithmic aspect of the LLL is the situation in which m is exponentially large. Such situations can be encountered in a variety of problems such as:

- Acyclic edge coloring
- Non-repetitive coloring
- Santa Claus problem

- Edge-disjoint paths

In fact, when running MT, we encounter the following problems:

1. the expected number of resamplings is $\sum_i \frac{x(\mathcal{A}_i)}{1-x(\mathcal{A}_i)}$
2. we need to represent the "bad" events
3. we need to verify a solution/ find some $\mathcal{A}_i$ that currently holds

The first problem can be solved by proving an upper bound:
Let $\delta = \min_i \Pr[\mathcal{A}_i]$. Then,

$$\begin{aligned}
\mathrm{E}[\text{ \# iterations of MT}] &\leq \textstyle\sum_i \frac{x(\mathcal{A}_i)}{1-x(\mathcal{A}_i)} \\
&\leq (\sum_i x(\mathcal{A}_i)) \cdot \max_i \frac{1}{1-x(\mathcal{A}_i)} \\
&\leq O(n \log(1/\delta)) \cdot \max_i \frac{1}{1-x(\mathcal{A}_i)}
\end{aligned}$$

In all applications known to us, $\log(\frac{1}{\delta}) \leq O(n \log n)$.

While we have good bounds on the running time of MT even for application with $\Omega(n) \leq m \leq \mathrm{poly}(n)$ many events, it unfortunately often fails to be directly applicable when m becomes super-polynomial in n. The reason is that maintaining bad events implicitly and running the resampling process requires an efficient way to find violated events. In many examples with super-polynomially many events, finding violated events or even just verifying a good assignment is hard.

The solution that [HSS10] give to this problem is:

- find a **core-subset** of the events, of polynomial size, and run MT on that
- bound the probabilities of non-core events using Theorem 3.
- use a union bound over these probabilities to prove that with high probability all of the $\mathcal{A}_i$ are avoided

Essentially what this algorithm does is it uses the randomness in the output distribution of the MT-algorithm run on the smaller core. Here is where the theorem about the LLL-distribution becomes helpful. 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.

As another way to look at the proof, one can imagine redefining the witness trees to have only core events as non-root nodes, just like we did in the proof of Theorem 3. Then, with this modified Galton-Watson process, we can grow witness trees from a log starting with a root event that holds at a certain point in time. This guarantees that we capture non-core events happening even though they are never resampled (since we never check whether such events hold or not).

The following theorem summarizes the idea:

Theorem 4. *If* $\exists \epsilon \in (0,1)$ *such that for all* $\mathcal{A}_i$*,*

$$\Pr[\mathcal{A}]^{1-\epsilon} \leq x(\mathcal{A}_i) \cdot \prod_{\mathcal{A}_j \in \Gamma(\mathcal{A}_i)} (1 - x(\mathcal{A}_j)),$$

then:

- *for any* $p \geq \frac{1}{poly(n)}$, $|\{\mathcal{A}_i : \Pr[\mathcal{A}_i] \geq p\}| \leq poly(n)$*;*
- *If* $\log \frac{1}{\delta} \leq poly(n)$ *and the above core is "checkable", then for any desired constant* $c > 0$, $\exists$ *a Monte Carlo algorithm(with* $p \sim n^{-c/\epsilon}$*) that terminates within* $O(\frac{n}{\epsilon} \log \frac{n}{\epsilon})$ *resamplings and returns a good assignment with probability at least* $1 - n^{-c}$*.*

The idea is that, for every $p \geq \frac{1}{\text{poly}(n)}$, the set $\{\mathcal{A}_i : \Pr[\mathcal{A}_i] \geq p\}$ has size at most poly(n), and is thus essentially always an efficiently verifiable core subset of $\mathcal{A}$. In order to deal with the non-core events, we can use Theorem 3, in the sense that the underlying dependency graph G is very dense (the m vertices of G can be partitioned into n cliques according to the variables the bad events depend on) and the nature of the LLL-conditions force highly connected events to have small probabilities and x-values. The ϵ-slack provides an extra multiplicative factor over the LLL-conditions but by choosing $p = n^{-O(1/\epsilon)}$, we can make the union bound small, at most n^{-c}.

4.4 Algorithmic results

This approach has led to a number of efficient algorithms for:

- $O(1)$-approximation for the Santa Claus problem(Feige's proof [AFS08] made constructive)
- non-repetitive coloring(proof of Alon-Grytczuk-Hauszczak-Riordan [AGHR02] made constructive)
- acyclic edge coloring
- edge-disjoint paths([And10])

By the Santa Claus problem we mean the restricted assignment version of the max-min allocation problem for indivisible goods. In the max-min allocation problem, there is a set of n items and m platers. The value(utility) of item j to player i is $p_{i,j} \geq 0$. An item can be assigned to one player and the total valuation of the items received by a player is the sum of the individual valuations of each item, $p(i,j)$, for player i and item j. The goal is to maximize the minimum total valuation of the items received by any player, that is, to maximize $\min_i \sum_{j \in S_i} p(i,j)$, where S_i is the subset of items player i receives. In the restricted version of the max-min allocation problem, each item has an intrinsic value, for every player i, $p_{i,j}$ is either p_j or 0. [BS06] considered an LP relaxation to the problem known as the configuration LP. [AFS08] showed that the LP has a constant integrality gap but the proof cannot be made constructive because it is heavily based on repeated reductions that apply the asymmetric LLL to exponentially many events. In this context, Theorem 4 can be easily and directly applied to constructivize the proof.

5 Allowing Some $\mathcal{A}_i$ to Hold

As mentioned before, another result refers to interpolating between the LLL and the linearity of expectation. [HSS10] obtain the following result:

Theorem 5. *In the symmetric LLL with p and D, if $D \leq \alpha \cdot (1/(ep) - 1)$ where $1 < \alpha < e$, then we can make at most $\sim (e \ln(\alpha)/\alpha) \cdot mp$ of the $\mathcal{A}_i$ to hold, in randomized poly(m) time.*

The main observation is that, when $\alpha = 1$, we have the standard LLL and all bad events can be avoided. If $\alpha > e$, then the best one can do is linearity of expectation. In the case in which $1 < \alpha < e$, the core is chosen at random but the number of non-core events is not too big so that linearity of expectation can be applied. As mentioned before, allowing "most" bad events to be avoided can be very useful in MAX k-SAT problems.

6 A Different Framework

[KS11] introduce a different framework of looking at LLL, based on the work of Shearer ([She85]). Shearer has found an exact characterization of the sequence of probabilities for which all "bad" events can be avoided. The difference is that the dependency graph in the MT framework(variable, algorithmic version) is more restrictive than the dependency graph based on probabilistic independence (general version). [KS11] make the link by relating Shearer's framework (coming from the general version) with the analysis of the MT algorithm (coming from the variable version). The main technique is to construct Shearer condition equivalents that correspond to steps in the analysis of the MT algorithm. For example, instead of the Galton-Watson branching processes, they provide bounds using the spectral norm of the Stable Set Matrix.

They obtain results on both sides. On one hand, they prove that the sequential and parallel versions of the MT algorithm are efficient up to Shearer's bound and extend our result by showing that the running time is not only polynomial in the number of events but also in the number of variables. In the opposite direction, they reprove Shearer's result for the general LLL, making a tighter connection between the general and variable versions of LLL.

7 Conclusion and Open Problems

As we have seen, the LLL is an elegant statement that provides successful results in randomized algorithms and surprising results in approximation algorithms. The main problem we have encountered is the difficulty of making it constructive and efficient in the context of algorithms. The MT framework is a celebrated result in that direction. In addition, various problems inspire further refined approaches, such as considering a polynomially sized core and allowing some "bad" events to hold.

In the end, we would like to leave the reader with a few open questions that highlight potential directions in which the application of LLL can be taken:

- Is the term "$e \ln(\alpha)/\alpha$" tight?
- Can the algorithm be derandomized?
- Can we further analyze the dependencies among non-core events, such that we use something else instead of a union-bound?
- How much slack is really needed?
- Can we use [KS11] as a possible guideline for generalizing MT ?
- Our approach doesn't work in giving results about the Lopsided Local Lemma (a variant of the LLL that also distinguishes between negative and positive correlation [AS08]). However, the framework used by [KS11] is more abstract than the MT framework which heavily relies on the underlying assignment. Can a full understanding of the setting extend some results?

References

[AFS08] Arash Asadpour, Uriel Feige, and Amin Saberi. Santa Claus Meets Hypergraph Matchings. In *APPROX-RANDOM*, pages 10–20, 2008.

[AGHR02] Noga Alon, Jaroslaw Grytczuk, Mariusz Haluszczak, and Oliver Riordan. Nonrepetitive colorings of graphs. *Random Struct. Algorithms*, 21(3-4):336–346, 2002.

[Alo91] Noga Alon. A Parallel Algorithmic Version of the Local Lemma. *Random Struct. Algorithms*, 2(4):367–378, 1991.

[And10] Matthew Andrews. Approximation Algorithms for the Edge-Disjoint Paths Problem via Raecke Decompositions. In *FOCS*, pages 277–286, 2010.

[AS08] Noga Alon and Joel H. Spencer. *The Probabilistic Method*. Wiley, New York, 2008.

[Bec91] József Beck. An Algorithmic Approach to the Lovász Local Lemma. i. *Random Struct. Algorithms*, 2(4):343–366, 1991.

[BS06] Nikhil Bansal and Maxim Sviridenko. The Santa Claus problem. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, STOC '06, pages 31–40, New York, NY, USA, 2006. ACM.

[CJBM02] Benjie Chen, Kyle Jamieson, Hari Balakrishnan, and Robert Morris. Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks. *Wireless Networks*, pages 481–494, 2002.

[CS00] Artur Czumaj and Christian Scheideler. Coloring non-uniform hypergraphs: a new algorithmic approach to the general Lovász local lemma. In *SODA*, pages 30–39, 2000.

[EL75] Paul Erdős and László Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. In R. Rado A. Hajnal and V.T. Sós, editors, *Infinite and Finite Sets*, pages 609–628. North-Holland, Amsterdam, 1975.

[Fei08] Uriel Feige. On allocations that maximize fairness. In *SODA*, pages 287–293, 2008.

[FHKS02] Uriel Feige, Magnús M. Halldórsson, Guy Kortsarz, and Aravind Srinivasan. Approximating the Domatic Number. *SIAM J. Comput.*, 32(1):172–195, 2002.

[HSS10] Bernhard Haeupler, Barna Saha, and Aravind Srinivasan. New Constructive Aspects of the Lovasz Local Lemma. In *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, FOCS '10, pages 397–406, Washington, DC, USA, 2010. IEEE Computer Society.

[KS11] Kashyap Babu Rao Kolipaka and Mario Szegedy. Moser and Tardos meet Lovász. In *STOC*, pages 235–244, 2011.

[LLRS01] Frank Thomson Leighton, Chi-Jen Lu, Satish Rao, and Aravind Srinivasan. New Algorithmic Aspects of the Local Lemma with Applications to Routing and Partitioning. *SIAM J. Comput.*, 31(2):626–641, 2001.

[LMR94] F. T. Leighton, Bruce M. Maggs, and Satish B. Rao. Packet routing and job-shop scheduling in o(congestion+dilation) steps. *Combinatorica*, 14:167–186, 1994. 10.1007/BF01215349.

[Mos09] Robin A. Moser. A constructive proof of the Lovász local lemma. In *STOC*, pages 343–350, 2009.

[MR98] Michael Molloy and Bruce A. Reed. Further Algorithmic Aspects of the Local Lemma. In *STOC*, pages 524–529, 1998.

[MR02] M. Molloy and B. Reed. Graph colouring and the probabilistic method. 2002.

[MT10] Robin A. Moser and Gábor Tardos. A constructive proof of the general lovász local lemma. *J. ACM*, 57(2), 2010.

[She85] J. Shearer. On a problem of spencer. *Combinatorica*, 5:241–245, 1985. 10.1007/BF02579368.

[Sri08] Aravind Srinivasan. Improved algorithmic versions of the Lovász Local Lemma. In *SODA*, pages 611–620, 2008.