Efficient Derandomization of the Lovász-Local-Lemma and Applications to Coloring and Packing Problems

Dissertation
zur Erlangung des Doktorgrades
der Technische Fakultät
der Christian-Albrechts-Universität
zu Kiel

vorgelegt von

Nitin Ahuja

Kiel
2003

Contents

1 Introduction **5**

2 Basics of Derandomization **9**
2.1 The Method of Conditional Probabilities 9
2.2 Pessimistic Estimator 11
2.3 The Lovász Local Lemma 12
2.4 Large Deviation Inequalities 14

3 Algorithmic Lovász Local Lemma **21**
3.1 Introduction 21
3.2 Beck's Algorithm 22
3.2.1 Main Result 23
3.2.2 Subsidiary Concepts and Results 28
3.3 A General Setting 29
3.4 A Class of Integer Programs 34
3.4.1 Generating Fractional Solutions 35
3.4.2 Randomized Rounding 36
3.4.3 Randomized Construction 39
3.4.4 Derandomization 42

4 Applications **45**
4.1 Constrained Hypergraph Coloring 45
4.2 Resource Constrained Scheduling 48

5 Multi-dimensional Bin Packing **51**
5.1 Preliminaries 52
5.2 The Algorithm 53
5.3 Analysis 54

6 Conclusion **57**
6.1 Open Questions 58

Bibliography **59**

Chapter 1

Introduction

For most of us coloring is fun but packing is not. So, in layman terms it might seem odd to talk about two very different activities, different at least on the fun scale. Mathematically, these two problems are not so different as the fun scale might lead you to believe.

The most commonly known coloring problem is the graph coloring problem. Given a graph, *i.e.*, a finite set of vertices $V = \{v_1, v_2, \dots\}$ and a set of edges $E = \{(v_i, v_j) \mid v_i, v_j \in V\ ,\ i \neq j\}$, color its vertices such that no two end vertices of any one of its edges get the same color. This problem can be applied to solve certain scheduling problems, among other problems, where the task is to schedule certain activities but some pairs of activities cannot be scheduled at the same time in parallel.

Among packing problems, the most famous one is the one-dimensional bin packing problem. In an instance of this problem we are given a finite list of numbers (items) in the interval $(0, 1]$ and lots of bins of size one. The aim is to pack these numbers in bins, using minimum number of bins, such that for every bin the sum of numbers in that bin is not more than its size. This problem has many applications like packing a set of commercials (items) in commercial breaks of pre-defined duration (bins), etc.

In this work we deal with multi-dimensional generalisations of the above mentioned problems. The more general forms of graphs are known as hypergraphs (see Definition 3.1.1). In a hypergraph an edge can have two or more vertices. On similar lines, in multi-dimensional bin packing we are given vectors as items instead of numbers and the bins are multi-dimensional too. Two main problems considered by us are as follows:

- Constrained Hypergraph Coloring or CHC (Chapter 4): given a hypergraph with n vertices and s edges, color its vertices using minimum number of

colors such that in each hyperedge i there are no more than b_i vertices of any color.

- Multi-dimensional Bin Packing or MDBP (Chapter 5): given n rational, d-dimensional vectors with components from the interval $[0, 1]$, pack these vectors in minimum number of bins such that in each bin, for each of the d dimensions, the sum over all vectors in that bin is at most one.

These problems can be written as integer programs and the set of problem instances of CHC, for fixed b_is, is a subset of the set of problem instances of MDBP. Let us denote by T_{opt} and T^* the values of the optimum solutions of the integer program and its LP-relaxation respectively, corresponding to the constrained hypergraph coloring problem. Also, let *opt* and m^* be the values of the optimum solutions of the integer program and its LP-relaxation respectively, corresponding to the multi-dimensional bin packing problem. We also apply our methods to a similar problem called resource constrained scheduling (Chapter 4).

Special cases of CHC are well known. For a hypergraph H with hyperedges $F_1, \ldots, F_s$, and $b_i = |F_i| - 1 \ \forall i$, CHC is closely related to the property B problem (*i.e* there exists a 2-coloring of H in which no hyperedge is monochromatic). Whenever H has property B, CHC is equivalent to the problem of finding a non-monochromatic 2-coloring of H. A coloring problem related to the CHC problem, which also generalizes the property B problem to multicolors has been studied by Lu [20]. The aim here is to color the vertices of H with k given colors such that no color appears more than b times in any edge.

Previous approximations for T_{opt} and *opt* in this context are as follows. In [12] Garey et al. use the First-Fit-Decreasing heuristic to give a polynomial time $(d + \frac{1}{3})$ approximation algorithm for MDBP problem. Subsequently De la Vega and Lueker [28] improved this result and gave a linear time algorithm which, for every $\epsilon > 0$, gives a $(d + \epsilon)$ approximate solution. Recently Chekuri and Khanna [7] further improved the long standing $(d + \epsilon)$ bound, they gave a polynomial time algorithm that, for any fixed $\epsilon > 0$, delivers a $(1 + \epsilon d + O(\log \epsilon^{-1}))$-approximate solution. For arbitrary b_i's, Srivastav and Stangier [26, 27] gave a polynomial time approximation algorithm that for every $\epsilon > 0$ delivers a coloring using at most $\lceil (1+\epsilon) T_{opt} \rceil$ colors provided that for all i, $b_i \geq 3\epsilon^{-2}(1+\epsilon)\log(8sT^*)$.

In this work we apply probabilistic methods, mainly the Lovász Local Lemma (LLL) and its algorithmic version, to obtain algorithms with better approximation ratio for the above mentioned problems. In Chapter 2 we lay down the basics of derandomization and other important tools like large deviation inequalities. Chapter 3 develops the algorithmic version of Lovász Local Lemma. We end this chapter by applying a variation of the algorithmic LLL to approximately

solve a class of integer programs which captures the CHC problem and the resource constrained scheduling problem.

In Chapter 4 we use the result of Chapter 3 to obtain the following result: given a hypergraph H with n vertices and s hyperedges we show that for every $\epsilon \in (0,1)$, H can be colored in polynomial time using at most $\lceil (1+\epsilon)T_{opt} \rceil$ colors provided that $b_i = \Omega\left(\epsilon^{-2}(1+\epsilon)\log \mathcal{D}\right)$ for all edges $i \in [s]$. The parameter $\mathcal{D} \leq d\eta\tau = O((sT^*)^2)$ where d is the maximum vertex degree of the hypergraph and η, τ (defined in Section 3.4.2) depend on the structure of inequality constraints in the integer program. But depending on the problem instance, $d\eta\tau$ could be very small in comparison to sT^*. This improves the previous best lower bounds on b_i's given by Srivastav and Stangier [27]. For $\eta\tau = O(\text{poly}(r))$ we are able to construct a $\lceil (1+\epsilon)T_{opt} \rceil$-coloring provided that $b_i = \Omega\left(\epsilon^{-2}(1+\epsilon)\log(dr)\right)$ $\forall i$. We also give a negative result [1, 7] in this chapter. All these results also hold for resource constrained scheduling.

In Chapter 5 we modify the algorithm of [7]. Let ν be the minimum component among all components in the given n vectors. The modified algorithm achieves a $(1+\epsilon q + O(\log \epsilon^{-1}))$-approximation ratio for any fixed $\epsilon > \nu$, where $q = \min\{d, \frac{1}{\nu}\Theta\left(\frac{\log dm^*}{\log\log dm^*}\right)\}$. This leads to a classification of problem instances for which our approximation ratio is better than the previous best. In the end we conclude with Chapter 6 which contains the summary of the methods described in the following chapters and some open questions.

Notation

Here are some standard symbols used by us:

$\mathbb{Z}$: set of integers
$\mathbb{Z}_+$: set of positive integers
$\mathbb{Q}$: set of rational numbers
$\mathbb{Q}_+$: set of positive rational numbers
$\mathbb{N}$: set of natural numbers
$\mathbb{R}$: set of real numbers
$\mathbb{R}_+$: set of positive real numbers
$[n]$: the set $\{1, \ldots, n\}$
ξ : a simple or compound event
ξ^c : the complement event of ξ
[5] : bibliographic reference to item 5
$G(V,E)$: a graph with vertex set V and edge set E
$G(\mu,\delta)$: Chernoff bound
$H(V,E)$: a hypergraph with vertex set V and edge set E

Acknowledgments

I would like to sincerely thank Prof. Dr. Anand Srivastav for his support and guidance. My heartfelt thanks goes to my family members, numerous friends and colleagues for their constant support and encouragement. In particular, I must thank Andreas Baltz and Gudrun Thiel for making life easier for me. I also thank the DFG for granting me a scholarship through the graduate school "Effiziente Algorithmen und Mehrskalenmethoden".

Chapter 2

Basics of Derandomization

Often the probabilistic method yields efficient randomized algorithms for many problems, specially in combinatorial optimization. Sometimes these randomized algorithms can be derandomized to yield polynomial time deterministic algorithms. It is well known that for various problems randomized algorithms out perform the best deterministic algorithms in terms of simplicity, quality of solution provided, running time and ease of implementation. Thus, derandomizing some of these randomized algorithms often gives us a deterministic algorithm with performance ratio much better than that of the prviously known deterministic algorithms. On the other hand, a randomized algorithm after undergoing through the process of derandomization often looses its simplicity and charm in terms of running time. Roughly speaking, a randomized algorithm transforms the underlying search space into an appropriate probability space and finds out points of interest in this space, which yield near optimal solutions, with a reasonably high probability. Derandomization then, is the process of finding out the points of interest in this space deterministically. In this chapter we explain the basics of derandomization.

2.1 The Method of Conditional Probabilities

The method of conditional probabilities can be best explained with the help of an example. Suppose we have a random vector $X = (x_1, x_2, x_3, x_4)$ where each x_j is a 0/1 random variable for $j = 1, 2, 3, 4$ and is defined as follows :

$$x_j = \begin{cases} 0 & \text{with probability } 0.5 \ ; \\ 1 & \text{with probability } 0.5 \ . \end{cases}$$

Let E be the event "either there is not a single 1 in vector X or there are at least two 1s in X" which is of interest to us. We consider the complement event E^c of event E. E^c is the event "vector X has exactly one component 1 and the other three components are 0". Obviously $\mathbb{P}[E^c] = 0.25 < 1$ and

we want that E^c should not occur, *i.e.* event E should occur. The idea now is to sequentially assign a 0 or 1 to each of the x_js such that the conditional probability of event E^c decreases monotonically at each step. During the first step we calculate $\mathbb{P}[E^c \mid x_1 = 1]$ and $\mathbb{P}[E^c \mid x_1 = 0]$. They turn out to be 0.125 and 0.375 respectively, so we assign $x_1 = 1$. At the second step we calculate $\mathbb{P}[E^c \mid x_1 = 1\ ,\ x_2 = 1]$ and $\mathbb{P}[E^c \mid x_1 = 1\ ,\ x_2 = 0]$, and they come out to be 0 and 0.25 respectively. At this point we assign 1 to x_2 and we are done because no matter what values x_3 and x_4 take E^c is not going to occur.

The credit of inventing the method of conditional probabilities goes to Erdös and Selfridge [10]. As is usual, the method was first applied sporadically to specefic problems and later it was picked up, extended and extensively applied to many problems by other researchers. We now give a more general framework for the method of conditional probabilities and the method of conditional expectations.

Consider a probability space $(\Omega, \mathbb{P})$, where $\Omega = \{1, 2, \ldots, m\}^n = [m]^n$ and $m, n \in \mathbb{Z}_+$. $\mathbb{P}$ is a probability measure on Ω. Let $E \subset \Omega$ be an event with $\mathbb{P}[E] > 0$ and let E^c be the complement event. For $X \in \Omega$ let $X = (x_1, \ldots, x_n)$ and let $w_1, w_2, \ldots, w_n \in [m]$. For $l \in [n]$, let $\mathbb{P}[E^c \mid w_1, w_2, \ldots, w_l]$ be the conditional probability of E^c provided that $x_1 = w_1$, $x_2 = w_2, \ldots$, $x_l = w_l$. In other words, it is the probability that E^c still occurs after fixing the first l components of the random vector X. Since we are interested in event E, we do the following: at step l, $l = 1, 2, \ldots, n$, we find $w \in [m]$ such that $\mathbb{P}[E^c \mid w_1, w_2, \ldots, w_{l-1}, x_l = w]$ is minimum. So, the first step is to find w such that $\mathbb{P}[E^c \mid x_1 = w]$ is minimized. Let this value of w be w_1. Subsequent steps are executed in a similar fashion. The fact that guarantees a monotonically decreasing sequence of conditional probabilities is that at each step the conditional probability can be written as a convex combination

$$\mathbb{P}[E^c \mid w_1, w_2, \ldots, w_{l-1}] = \sum_{w \in [m]} \mathbb{P}[E^c \mid w_1, w_2, \ldots, w_{l-1}, x_l = w] \cdot \mathbb{P}[x_l = w] \ .$$

By following the procedure described above we are able to satisfy the following chain of inequalities

$$1 > \mathbb{P}[E^c] \geq \mathbb{P}[E^c \mid w_1] \geq \cdots \geq \mathbb{P}[E^c \mid w_1, w_2, \ldots, w_n] \in \{0, 1\} \ .$$

Thus, we end up finding $w_1, w_2, \ldots, w_n$ such that $\mathbb{P}[E^c \mid w_1, w_2, \ldots, w_n] = 0$, *i.e.*, E^c does not occur but E, the event of our interest does.

The method of conditional expectations is based on similar arguments. We have a function $f : \Omega \to \mathbb{Q}$ and our aim is to find a vector (or a point) $X \in \Omega$ such that $f(X) \leq \mathbb{E}[f]$. The first step is to find a $w \in [m]$ such that $\mathbb{E}[f \mid x_1 = w]$ is minimized. In the same way for $l = 2, \ldots, n$, in step l, when we have already set $x_1 = w_1, \ldots, x_{l-1} = w_{l-1}$, we try to find a $w \in [m]$ such that

$\mathbb{E}[f \mid w_1, \ldots, w_{l-1}, x_l = w]$ is minimized. As described in the case of conditional probability method, at each step we are able to maintain monotonicity because

$$\mathbb{E}[f \mid w_1, \ldots, w_{l-1}] = \sum_{w \in [m]} \mathbb{E}[f \mid w_1, \ldots, w_{l-1}, x_l = w] \cdot \mathbb{P}[x_l = w] \; .$$

Since the following inequalities hold

$$\mathbb{E}[f] \geq \mathbb{E}[f \mid w_1] \geq \cdots \geq \mathbb{E}[f \mid w_1, w_2, \ldots, w_n] = f(w_1, w_2, \ldots, w_n) \; ,$$

in the end we get the required vector X.

Notice that in these methods we examine all m^n possibilities in order to find a suitable n-dimensional vector X. So, in general, we cannot hope to apply derandomization and obtain efficient polynomial time algorithms. But all is not lost because as the intricacies of these methods show, it suffices to have an upper bound on the corresponding conditional probabilities. Although the idea of exploiting upper bounds was used before, it became popular after Raghavan [23] formally defined and propogated it as pessimistic estimators in the context of packing integer programs. Pessimistic estimators are explained in the following section.

2.2 Pessimistic Estimator

Again, we denote by $(\Omega, \mathbb{P})$ a probability space, where $\Omega = [m]^n$ and $m, n \in \mathbb{Z}_+$. $\mathbb{P}$ is a probability measure on Ω. Let $E \subset \Omega$ be an event with $\mathbb{P}[E] \geq \delta$ for some $0 < \delta < 1$ and let E^c be the complement event. We are now ready to define a pessimistic estimator.

Definition 2.2.1 *For $l = 1, \ldots, n$, let $\mathcal{P}$ be a family of functions $PE_l : [m]^l \to \mathbb{Q}$ containing a constant function $PE_0 \leq 1 - \delta$. $\mathcal{P}$ is called a pessimistic estimator for event E, if for each $l \in [n]$ the following conditions are satisfied :*

1. *$\mathbb{P}(E^c \mid w_1, \ldots, w_l) \leq PE_l(w_1, \ldots, w_l)$ for all $w_1, \ldots, w_l \in [m]$.*
2. *Given $w_1, \ldots, w_{l-1}$ there exists $w_l \in [m]$ such that $PE_l(w_1, \ldots, w_l) \leq PE_{l-1}(w_1, \ldots, w_{l-1})$.*
3. *$PE_l(w_1, \ldots, w_l)$ can be computed in time no more than a polynomial of $\log \delta^{-1}$, m and n.*

Notice that condition 2 is automatically satisfied if there exist positive numbers $a_1, a_2, \ldots, a_n$ with $\sum_{w=1}^{m} a_w = 1$ such that for each $l \in [n]$

$$PE_{l-1}(w_1, \ldots, w_{l-1}) \geq \sum_{w=1}^{m} a_w \cdot PE_l(w_1, \ldots, w_{l-1}, w) \; . \tag{2.1}$$

Such a family $\mathcal{P}$, satisfying equation (2.1) instead of condition 2, is called a *convex pessimistic estimator*. Furthermore a family of functions $\mathcal{P}$ is called a *weak pessimistic estimator* if it does not satisfy condition 3 in Definition 2.2.1 above. As mentioned above, a pessimistic estimator relieves us from the burden of calculating conditional probabilities (resp. expectations) during the process of derandomization. The following theorem states the result formally

Theorem 2.2.2 *Given an event $E \subset \Omega$ with $\mathbb{P}[e] \geq \delta > 0$ and a pessimistic estimator $\mathcal{P}$ for E, a vector $X \in E$ can be found in time bounded by a polynimial of $\log \delta^{-1}$, m and n.*

Proof. At each step l, where $l = 1, 2, \ldots, n$, having fixed $x_1 = w_1, \ldots, x_{l-1} = w_{l-1}$ we try to find $w \in [m]$ which minimizes $PE_l(w_1, \ldots, w_{l-1}, x_l = w)$ and set $w_l = w$. We can find the minimum by checking out all m possibilities in polynomial time because by Definition 2.2.1 PE_l can be computed in polynomial time. Since there are at most n steps to be performed, this procedure can be completed in polynomial time. The correctness of this procedure follows from the following inequalities which use the fact that at each step l the value of PE_l is at most PE_{l-1}

$$\begin{aligned} 1 - \delta \geq PE_0 &\geq PE_1(w_1) \\ &\vdots \\ &\geq PE_{n-1}(w_1, \ldots, w_{n-1}) \\ &\geq PE_n(w_1, \ldots, w_{n-1}, w_n) \geq \mathbb{P}[E^c \mid w_1, \ldots, w_{n-1}, w_n] = 0 \,. \end{aligned}$$

Thus, in the end we get the required vector $X = (x_1, \ldots, x_n) \in E$ in polynomial time. □

2.3 The Lovász Local Lemma

As mentioned at the beginning of this chapter, we have a probability space $(\Omega, \mathbb{P})$ and we are mainly interested in efficiently finding the points of interest corresponding to the event $\mathcal{E} \subset \Omega$. Sometimes $\mathcal{E}$ is large, *i.e.*, the points of interest to us are abundantly present and scattered in Ω and thus, it is not a difficult task to efficiently find these points. In fact, this is most of what randomized algorithms do. But, sometimes $\mathcal{E}$ is very small, containing very few points which are of interest to us. In these cases it is hard to show the existence of points in Ω corresponding to event $\mathcal{E}$ (showing $\mathbb{P}[\mathcal{E}] > 0$), let alone find them efficiently.

For instance, let $\mathcal{E}$ be the conjunction of n independent events $\mathcal{E}_1, \ldots, \mathcal{E}_n$ with $\mathbb{P}[\mathcal{E}_i] = 0.5$ for each $i \in [n]$. For large n the probability $\mathbb{P}[\mathcal{E}] = 2^{-n}$ is very small but non-zero. The same thing happens if $\mathcal{E} = \cap_{i=1}^{n} \mathcal{E}_i^c$ and $\mathbb{P}[\mathcal{E}_i] \leq 0.9$ for each

$i \in [n]$ because

$$\mathbb{P}[\mathcal{E}] = \prod_{i=1}^{n} \mathbb{P}[\mathcal{E}_i^c] = \prod_{i=1}^{n} (1 - \mathbb{P}[\mathcal{E}_i]) \geq 10^{-n} .$$

Here we were able to show that $\mathcal{E} \neq \emptyset$ inspite of it being small (*rare* event).

The Lovász Local Lemma (LLL) is a powerful tool which is used to show the existence of rare, compound events even when the basic constituting events, $\mathcal{E}_i$s, are dependent on each other to some extent. The local lemma was first proved and applied by Erdös and Lovász [9]. Surprisingly, in some cases, the Lovász Local Lemma can be converted into a polynomial time algorithm. We will discuss these aspects of the local lemma in the next chapter, here we give the basic (non-algorithmic) version.

Suppose we are given events $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_n$ in an arbitrary probability space. Each of these events is mutually independent of all but a few other events. Consider a digraph $G = (V, E)$ where there is a vertex v_i representing each event $\mathcal{E}_i$ and an event $\mathcal{E}_i$ is mutually independent of all the events $\{\mathcal{E}_j \,|\, (i,j) \notin E\ ,\ i \neq j\}$. This digraph G is called the *dependency graph*. The Lovász Local Lemma is:

Lemma 2.3.1 *(General Case) Let $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_n$ be events in an arbitrary probability space and let $G = (V, E)$ be the corresponding dependency graph. Suppose there exist $x_i \in [0,1]$ for $i \in [n]$ such that*

$$\mathbb{P}[\mathcal{E}_i] \leq x_i \prod_{(i,j) \in E} (1 - x_j) ,$$

then

$$\mathbb{P}[\cap_{i=1}^{n} \mathcal{E}_i^c] \geq \prod_{i=1}^{n} (1 - x_i) . \tag{2.2}$$

Proof. We prove that for any $S \subset [n]$ where $|S| = s < n$ and any $i \notin S$,

$$\mathbb{P}\left[\mathcal{E}_i \mid \cap_{j \in S} \mathcal{E}_j^c\right] \leq x_i . \tag{2.3}$$

This can be used to prove the lemma because

$$\begin{aligned} \mathbb{P}[\cap_{i=1}^{n} \mathcal{E}_i^c] &= (1 - \mathbb{P}[\mathcal{E}_1]) \cdot (1 - \mathbb{P}[\mathcal{E}_2 \mid \mathcal{E}_1^c]) \cdot \ldots (1 - \mathbb{P}[\mathcal{E}_n \mid \cap_{i=1}^{n-1} \mathcal{E}_i^c]) \\ &\geq \prod_{i=1}^{n} (1 - x_i) . \end{aligned}$$

To prove the statement of (2.3) we use induction. It holds for $s = 0$. So, let us assume that it holds for all $s' < s$. Let $S_1 = \{j \in S \,|\, (i,j) \in E\}$ and let $S_2 = S \S_1$. By the definition of conditional probability we have

$$\mathbb{P}\left[\mathcal{E}_i \mid \cap_{j \in S} \mathcal{E}_j^c\right] = \frac{\mathbb{P}\left[\mathcal{E}_i \cap (\cap_{j \in S_1} \mathcal{E}_j^c) \mid \cap_{k \in S_2} \mathcal{E}_k^c\right]}{\mathbb{P}\left[\cap_{j \in S_1} \mathcal{E}_j^c \mid \cap_{k \in S_2} \mathcal{E}_k^c\right]} . \tag{2.4}$$

The numerator of equation (2.4) can be bounded from above by using the facts that $\mathbb{P}[\mathcal{E}_i \cap \mathcal{E}_j] \leq \mathbb{P}[\mathcal{E}_i]$ and $\mathcal{E}_i$ is mutually independent of the set of events $\{\mathcal{E}_k \mid k \in S_2\}$. Thus,

$$\mathbb{P}\left[\mathcal{E}_i \cap (\cap_{j\in S_1}\mathcal{E}_j^c) \mid \cap_{k\in S_2}\mathcal{E}_k^c\right] \leq \mathbb{P}[\mathcal{E}_i \mid \cap_{k\in S_2}\mathcal{E}_k^c] = \mathbb{P}[\mathcal{E}_i] \leq x_i \prod_{(i,j)\in E}(1-x_j)\ .$$

To get a lower bound for the denominator of (2.4) let us suppose that $S_1 = \{j_1, \ldots, j_r\}$. Note that the denominator is 1 if $r = 0$, so assume $r > 0$. Since $S_2 \subset S$ we use the induction hypothesis to get the lower bound as follows:

$$\begin{aligned}
&\mathbb{P}\left[\mathcal{E}_{j_1}^c \cap \cdots \cap \mathcal{E}_{j_r}^c \mid \cap_{k\in S_2}\mathcal{E}_k^c\right] \\
&= (1 - \mathbb{P}[\mathcal{E}_{j_1} \mid \cap_{k\in S_2}\mathcal{E}_k^c]) \cdot \ldots \cdot (1 - \mathbb{P}[\mathcal{E}_{j_r} \mid \mathcal{E}_{j_1}^c \cap \cdots \cap \mathcal{E}_{j_{r-1}}^c \cap_{k\in S_2}\mathcal{E}_k^c]) \\
&\geq (1-x_{j_1})(1-x_{j_2})\ldots(1-x_{j_r}) \geq \prod_{(i,j)\in E}(1-x_j)\ .
\end{aligned}$$

This proves (2.3), hence, completing the proof. □

While applying the local lemma $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_n$ are considered as *bad* events, *i.e.*, we would not like any of these events to occur. The Lovász Local Lemma is usually applied in the following form.

Corollary 2.3.2 *(Symmetric LLL) Let $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_n$ be events in an arbitrary probability space with $\mathbb{P}[\mathcal{E}_i] \leq p$ for all i. Suppose that each event $\mathcal{E}_i$ is mutually independent of all but at most d other events. If $ep(d+1) \leq 1$ then $\mathbb{P}[\cap_{i=1}^n \mathcal{E}_i^c] > 0$.*

Proof. If $d = 0$ the result is obvious, so let $d > 0$. The proof follows by applying Lemma 2.3.1 after setting $x_i = 1/(d+1)$ for all i and showing that for each event $\mathcal{E}_i$, $x_i \prod_{(i,j)\in E}(1-x_j) \geq p$. □

Sometimes the condition $ep(d+1) \leq 1$ is replaced by $4pd \leq 1$.

2.4 Large Deviation Inequalities

Probabilistic methods followed by derandomization are often used to tackle hard problems. Usually these problems have a complex combinatorial structure which is captured or modelled using different techniques like casting the problems in the form of (non)linear optimization problem. After obtaining a satisfactory formulation of the problem at hand randomness is introduced by considering various parameters and/or variables of the model to be random variables. Thus, the events of our interest, those which correspond to good approximate solutions, are expressed in terms of these random variables. As we have already seen in the previous section, to show the existence of a rare event $\mathcal{E}$, we need to have an upper

bound on the probability of basic events $\mathcal{E}_i$. Such upper bounds on probabilities of random variables or events are known as tail inequalities or large deviation inequalities.

As the name itself suggests, the tail inequalities bound the probability of a random variable deviating too far from its mean. Given below are some such inequalities which are often used in analysing randomized algorithms and derandomization. We begin with the most basic Markov inequality which is later used to derive more complex upper bounds.

Lemma 2.4.1 *(Markov Inequality) If X is a random variable assuming non-negative values x only, then for all $r \in \mathbb{R}_+$,*

$$\mathbb{P}[X \geq r] \leq \frac{\mathbb{E}[X]}{r} .$$

Proof. Define a function $f(x)$ as follows:

$$f(x) = \begin{cases} 1 & \text{if } x \geq r , \\ 0 & \text{otherwise} . \end{cases}$$

Note that $\mathbb{P}[X \geq r] = \mathbb{E}[f(X)]$ and $x/r \geq 1 \geq f(x)$ for all x. Therefore

$$\mathbb{E}[f(X)] \leq \mathbb{E}\left[\frac{X}{r}\right] \leq \frac{\mathbb{E}[X]}{r} ,$$

and this completes the proof. □

Notice that Markov inequality is a very general result assuming that the random variable X takes just non-negative values. If we have some more information about X then we can obtain better upper bounds. For instance if we know the mean $\mu = \mathbb{E}[X]$ and variance $\sigma^2 = \mathbb{E}[(X - \mu)^2]$ then we can apply Chebyshev inequality which is as follows

Lemma 2.4.2 *(Chebyshev Inequality) If X be a random variable with mean μ and standard deviation σ $(= +\sqrt{\sigma^2})$, then for any $r \in \mathbb{R}_+$,*

$$\mathbb{P}[|X - \mu| \geq r\sigma] \leq r^{-2} . \tag{2.5}$$

Proof. Notice that random variable $Y = (X - \mu)^2$ has expectation σ^2 and $\mathbb{P}[|X - \mu| \geq r\sigma] = \mathbb{P}[Y \geq (r\sigma)^2]$. Since Y takes on only non-negative values, one can apply Markov inequality and prove inequality (2.5). □

We are now ready to state and prove much sharper and very useful *Chernoff bounds* on probabilities of large deviations. Till now we have been concerned with tail inequalities of simple random variables but usually, for solving a hard

problem algorithmically, we need large deviation inequalities for more complex and compound random variables. Now we will try to bound the probabilities of large deviations of sums of independent random variables. These bounds are heavily used while analysing various probabilistic algorithms or arguments. Let $X_1, \ldots, X_n$ be n independent random variables with

$$X_i = \begin{cases} a_i & \text{with probability } p_i \ , \\ 0 & \text{with probability } 1 - p_i \ , \end{cases} \tag{2.6}$$

where $0 < p_i < 1$ and $a_i \in (0, 1]$ for each $i \in [n]$. Let $X = \sum_{i=1}^{n} X_i$ with $\mu = \mathbb{E}[X] = \sum_{i=1}^{n} a_i p_i$.

For a $\delta > 0$, we are interested in bounding $\mathbb{P}[X > (1+\delta)\mu]$ from above. To get good upper bounds we will apply Markov inequality to e^{tX} ,$t \in \mathbb{R}_+$, instead of X. As a result we will have the moment generating function $\mathbb{E}[e^{tX}]$ of X on the right hand side. The advantage we get here is that $\mathbb{E}[e^{tX}] = \prod_{i=1}^{n} \mathbb{E}[e^{tX_i}]$ because X_is are independent random variables. We then simplify each $\mathbb{E}[e^{tX_i}]$ and choose t so as to get the sharpest possible upper bound. This technique can be applied in other cases too, so in that sense it is a general tool to get good upper bounds.

Lemma 2.4.3 *(Chernoff bound) Let $X_1, \ldots, X_n$ be n independent random variables satisfying (2.6) and let $X = \sum_{i=1}^{n} X_i$ then, for $\mu = \mathbb{E}[X]$ and any $\delta > 0$,*

$$\mathbb{P}[X \geq \mu(1+\delta)] < G(\mu, \delta) = \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}} \right)^{\mu} . \tag{2.7}$$

Proof. For $t \in \mathbb{R}_+$,

$$\mathbb{P}[X \geq \mu(1+\delta)] = \mathbb{P}\left[e^{tX} \geq e^{t\mu(1+\delta)}\right] \leq \frac{\mathbb{E}[e^{tX}]}{e^{t\mu(1+\delta)}} , \tag{2.8}$$

where the inequality follows by applying Markov inequality. Now, using the independence of the random variables X_i we can replace $\mathbb{E}[e^{tX}]$ by

$$\mathbb{E}[e^{tX}] = \mathbb{E}\left[\prod_{i=1}^{n} e^{tX_i}\right] = \prod_{i=1}^{n} \mathbb{E}[e^{tX_i}] \tag{2.9}$$

in (2.8) and get:

$$\mathbb{P}[X \geq \mu(1+\delta)] \leq \frac{\prod_{i=1}^{n} \mathbb{E}[e^{tX_i}]}{e^{t\mu(1+\delta)}} . \tag{2.10}$$

According to (2.6),

$$\mathbb{E}[e^{tX_i}] = p_i e^{ta_i} + 1 - p_i \leq p_i e^{t} + 1 - p_i \ . \tag{2.11}$$

Substituting this in (2.10) we obtain

$$\mathbb{P}[X \geq \mu(1+\delta)] \leq \frac{\prod_{i=1}^{n}(1+p_i(e^t-1))}{e^{t\mu(1+\delta)}} \ . \tag{2.12}$$

Setting $y = p_i(e^t - 1)$ and using $1 + y < e^y$ results in

$$\begin{aligned}\mathbb{P}[X \geq \mu(1+\delta)] &< \frac{\prod_{i=1}^{n} e^{(p_i(e^t-1))}}{e^{t\mu(1+\delta)}} \\ &= \frac{e^{(\sum_{i=1}^{n} p_i(e^t-1))}}{e^{t\mu(1+\delta)}} \\ &= \frac{e^{\mu(e^t-1)}}{e^{t\mu(1+\delta)}} \ .\end{aligned}$$

Notice that the right hand side is minimum for $t = \ln(1+\delta)$ and that minimum is $G(\mu, \delta)$. This proves the lemma. □

So now we have a good upper bound in the form of Chernoff bound but this bound $G(\mu, \delta)$ is a bit inconvenient to handle. Thus, for our convenience and for the sake of simplifying the analyses appearing later, we give the following definition and the subsequent corollary.

Definition 2.4.4 *For any $\mu > 0$ and $p \in (0,1)$, $\delta = H(\mu, p) > 0$ is the smallest value of δ satisfying $G(\mu, \delta) \leq p$.*

Corollary 2.4.5 *For $\mu > 0$ and $p \in (0,1)$, there exists $\delta = H(\mu, p) > 0$ such that $G(\mu, H(\mu, p)) \leq p$ and*

$$H(\mu, p) = \begin{cases} \Theta\left(\sqrt{\frac{\log p^{-1}}{\mu}}\right) & \text{if } \mu \geq \frac{\log p^{-1}}{2} \ ; \\ \\ \Theta\left(\frac{\log p^{-1}}{\mu \log(\log(p^{-1})/\mu)}\right) & \text{otherwise} \ . \end{cases}$$

Again recall that the key to getting tight upper bounds for probabilities of large deviations of $X = \sum_{i=1}^{n} X_i$ is to get good upper bounds on the moment generating function $\mathbb{E}[e^{tX}]$, where $t \in \mathbb{R}_+$. Note that each X_i is a binary random variable and

$$e^{tX} = \sum_{j=0}^{\infty} \frac{(tX)^j}{j!} \ .$$

Now, assuming $a_i = 1$ for all i in (2.6), let us see what happens to X^2.

$$X^2 = (\sum_{i=1}^{n} X_i)^2 = \sum_{i=1}^{n} X_i^2 + \sum_{1 \leq i \leq j \leq n} X_i X_j = \sum_{i=1}^{n} X_i + \sum_{1 \leq i \leq j \leq n} X_i X_j \ .$$

One can also verify what happens to X^j for $j \geq 3$. It is not hard to see that e^{tX} can be expressed simply as

$$e^{tX} = \sum_{k=0}^{n} \alpha_k S_k(X_1, \ldots, X_n) ,$$

where for $k = 0, 1, \ldots, n$, $\alpha_k \in \mathbb{R}_+$,

$$S_k(X_1, \ldots, X_n) = \sum_{1 \leq i_1 < i_2 < \cdots < i_k \leq n} X_{i_1} X_{i_2} \cdots X_{i_k} ,$$

and $S_0(X_1, \ldots, X_n) = 1$. Also, since X_is are independent

$$\mathbb{E}[S_k(X_1, \ldots, X_n)] = \sum_{1 \leq i_1 < i_2 < \cdots < i_k \leq n} p_{i_1} p_{i_2} \cdots p_{i_k}$$

for all $k \in [n]$ and therefore the moment generating function can be expressed as

$$\mathbb{E}\left[e^{tX}\right] = \sum_{k=0}^{n} \alpha_k S_k(p_1, \ldots, p_n) . \tag{2.13}$$

Thus, in the case of 0/1, independent random variables it makes sense to find upper bounds for the linear combinations $\sum_{k=0}^{n} \alpha_k S_k(p_1, \ldots, p_n)$ to bound $\mathbb{E}[e^{tX}]$ from above. But, with the help of the following two facts we can get rid of $\mathbb{E}[e^{tX}]$.

1. The class of functions $\{\sum_{k=0}^{n} \alpha_k S_k(X_1, \ldots, X_n) \,|\, \alpha_1, \ldots, \alpha_n \in \mathbb{R}_+\}$ contains the class $\{e^{tX} \mid t > 0\}$.

2. Let $q = \mu(1+\delta)$ be integral. Then, for any non-negative integer $l \leq n$, $X = l$ if and only if

$$\sum_{k=0}^{n} \alpha_k S_k(X_1, \ldots, X_n) = \sum_{i=0}^{l} \alpha_i \binom{l}{i} .$$

The second point implies that

$$\begin{aligned}
\mathbb{P}[X \geq q] &= \mathbb{P}\left[\sum_{k=0}^{n} \alpha_k S_k(X_1, \ldots, X_n) \geq \sum_{i=0}^{q} \alpha_i \binom{q}{i}\right] \\
&\leq \frac{\mathbb{E}\left[\sum_{k=0}^{n} \alpha_k S_k(X_1, \ldots, X_n)\right]}{\sum_{i=0}^{q} \alpha_i \binom{q}{i}} \qquad \text{(by Markov inequality)} \\
&= \frac{\sum_{k=0}^{n} \alpha_k S_k(p_1, \ldots, p_n)}{\sum_{i=0}^{q} \alpha_i \binom{q}{i}} .
\end{aligned}$$

The first point implies that on minimizing the fraction obtained above we can get better upper bounds for $\mathbb{P}[X \geq \mu(1+\delta)]$, in fact, these bounds will not be worse than the Chernoff bound $G(\mu, \delta)$.

These polynomials also come into play even when the random variables X_is satisfy (2.6). Before stating and proving the main result (Lemma 2.4.7) which is also a large deviation inequality [24, 25] and which will be of use later on, let us state a simple Lemma.

Lemma 2.4.6 *Suppose $r = (r_1, \ldots, r_n) \in [0,1]^n$, $q \geq 0$ and $\sum_{i=1}^n r_i \geq q$. Then, for a non-negative integer $k \leq q$,*

$$S_k(r) = S_k(r_1, \ldots, r_n) \geq \binom{q}{k} \ . \tag{2.14}$$

Proof. The case $\sum_{i=1}^n r_i \geq q$ directly follows from the case $\sum_{i=1}^n r_i = q$, so let us just prove the Lemma assuming the later case. Suppose $S_k(r)$ is minimum at the point $r^* \in [0,1]^n$ under the constraint $\sum_{i=1}^n r_i = q$. It can be proved that at most one component $r_i^* \in (0,1)$. To see this assume that $r_i^*, r_j^* \in (0,1)$ and that w.l.o.g. $r_i^* < r_j^*$. Reset $r_i^* := r_i^* - \epsilon$ and $r_j^* := r_j^* + \epsilon$, where $\epsilon = \min\{r_i^*, 1 - r_j^*\}$. Obviously $S_k(r^*)$ decreases, thus, contradicting our assumption that for the original r^*, $S_k(r^*)$ is minimum. Now, if q is integral then $r_i^* \in \{0,1\}$ for all i and thus, $S_k(r^*) = \binom{q}{k}$. Otherwise, if q is not integral then $r_j^* = q - \lfloor q \rfloor$ for a $j \in [n]$ and $r_i^* \in \{0,1\}$ for $i \in [n] - \{j\}$. Also

$$S_k(r^*) = \binom{\lfloor q \rfloor}{k} + (q - \lfloor q \rfloor) \binom{\lfloor q \rfloor}{k-1} \ .$$

By induction on k it can be easily shown that

$$\binom{\lfloor q \rfloor}{k} + (q - \lfloor q \rfloor) \binom{\lfloor q \rfloor}{k-1} \geq \binom{q}{k} \ ,$$

thus completing the proof. □

We are now ready for the following useful Lemma.

Lemma 2.4.7 *([25])*

(i) For any $\delta > 0$, any non-empty event Z and any non-negative integer $k \leq \mu(1+\delta)$

$$\mathbb{P}[X \geq \mu(1+\delta) \mid Z] \ \leq \ \frac{\mathbb{E}[S_k(X_1, \ldots, X_n) \mid Z]}{\binom{\mu(1+\delta)}{k}} \ .$$

(ii) If $k = \lceil \mu\delta \rceil$, then for any $\delta > 0$

$$\mathbb{P}[X \geq \mu(1+\delta)] \leq \frac{\mathbb{E}[S_k(X_1, \ldots, X_n)]}{\binom{\mu(1+\delta)}{k}} \leq G(\mu, \delta) .$$

Proof. *(i)* From Lemma 2.4.6 we have that if $X = \sum_{i=1}^{n} X_i \geq \mu(1+\delta)$ then

$$S_k(X_1, \ldots, X_n) \geq \binom{\mu(1+\delta)}{k} .$$

This holds inspite of the occurence of any non-empty event Z. Thus, by applying Markov inequality we get the desired result:

$$\begin{aligned} \mathbb{P}[X \geq \mu(1+\delta) \mid Z] &= \mathbb{P}\left[S_k(X_1, \ldots, X_n) \geq \binom{\mu(1+\delta)}{k} \mid Z \right] \\ &\leq \frac{\mathbb{E}[S_k(X_1, \ldots, X_n) \mid Z]}{\binom{\mu(1+\delta)}{k}} . \end{aligned}$$

(ii) Note that X_is are independent, so

$$\mathbb{E}[S_k(X_1, \ldots, X_n)] = S_k(a_1 p_1, \ldots, a_n p_n) .$$

Since $(a_1 p_1, \ldots, a_n p_n) \in \mathbb{R}^n$ and $\sum_{i=1}^{n} a_i p_i = \mu$, it can be shown that

$$S_k(a_1 p_1, \ldots, a_n p_n) \leq S_k(r_1, \ldots, r_n) = \binom{n}{k} \left(\frac{\mu}{n}\right)^k ,$$

where $r_1 = \cdots = r_n = \frac{\mu}{n}$. Further it is not hard to see [24] that for $k = \lceil \mu\delta \rceil$ and for any $\delta > 0$,

$$\frac{\binom{n}{k} \left(\frac{\mu}{n}\right)^k}{\binom{\mu(1+\delta)}{k}} \leq G(\mu, \delta) .$$

□

Chapter 3

Algorithmic Lovász Local Lemma

3.1 Introduction

The Lovász Local Lemma is a powerful sieve method that can be used to prove the existence of certain events (see Section 2.3). On the other hand, although the local lemma shows the existence of a certain rare event, it provides us no answer to the question: how to find a point corresponding to such an event in the huge probability space without examining all points of this space? Usually, the sample space is (sub-)exponential in size and examining even a constant sub-portion of this space requires a lot of time. In other words, even after showing the existence of an event with the help of the Lovász Local Lemma, we get no clue about how to find a point corresponding to our event in at most polynomial time.

In 1991 Beck [6] made the first breakthrough. He gave the first algorithmic version of the local lemma and applied it to obtain constructive results for the *property B* problem. Before proceeding further we take a small digression to define the property B problem and for this purpose we need the following definitions.

Definition 3.1.1 *Let $V = \{v_1, \ldots, v_n\}$ be a finite set and let $E = \{E_1, \ldots, E_m\}$ be a finite set such that for each $i \in [m]$, $E_i \subseteq V$. A set system $H = (V, E)$ is called a* hypergraph *if*

1. *for each $i \in [m]$, $E_i \neq \varnothing$, and*
2. *$\cup_{i=1}^{m} E_i = V$.*

Elements of V and E are called vertices and (hyper)edges of the hypergraph respectively.

Definition 3.1.2 *A hypergraph $H = (V, E)$ is called k-uniform if $|E_i| = k$ for each $i \in [m]$.*

We are now ready to state the property B problem.

Definition 3.1.3 *(Property B) A hypergraph $H = (V, E)$ has property B if the vertices of H can be colored with two colors such that no edge of H is monochromatic.*

The property B problem is to efficiently find such a 2-coloring. We discuss Beck's method of obtaining such 2-colorings in Section 3.2.

Beck's algorithmic version was put into a simpler probabilistic setting by Alon [4]. Molloy and Reed [21] gave a general randomized algorithm which seems to capture all applications of Beck's algorithm. This randomized algorithm can be derandomized, thus, yielding a deterministic polynomial time algorithm in many cases including the property B problem. This general algorithm is described in Section 3.3. Molloy and Reed also extend their algorithmic version of the local lemma to the problem of coloring a graph frugally, among other problems. We swerve again to define frugal graph coloring.

A *simple graph* $G = (V, E)$ is nothing but a 2-uniform hypergraph. A *coloring* of the vertices of G is called *proper* if no two adjacent vertices (vertices having a common edge) get the same color. We can now define frugal graph coloring.

Definition 3.1.4 *For a positive integer b, a proper vertex-coloring of a simple graph $G = (V, E)$ is called* b-frugal *if for each vertex $v \in V$ and color c, the number of vertices neighboring v and having color c is at most b.*

Naturally, the ideal is to find a b-frugal graph coloring with minimum number of colors.

Inspite of these initial breakthroughs obstacles still existed because the algorithmic version did not seem to apply to other NP-hard problems like solving integer programs, etc. But, this changed with the appearance of a series of papers [25, 20, 17]. In Section 3.4 we describe the main result of this Chapter in the form of an application of the algorithmic version of Lovász Local Lemma to a class of integer programs. This application is in turn based on the application of the algorithmic version to *minimax integer programs* by Leighton, et al. [17].

3.2 Beck's Algorithm

We start this section by stating the first application of Lovász Local Lemma in the form of a corollary.

Corollary 3.2.1 *Let $H = (V, E)$ be a k-uniform hypergraph with $|V| = n$ and $|E| = m$. If every edge $E_i \in E$ intersects at most 2^{k-3} other edges E_j then H has property B.*

Proof. For each vertex $v \in V$, independnet of other vertices, assign it the color red with probability $1/2$ and blue with probability $1/2$. Let $\mathcal{E}_i$ be the event that edge E_i becomes monochromatic, *i.e.*, either all k vertices of E_i are colored blue

or all of them are colored red. For any $i \in [m]$ we have $\mathbb{P}[\mathcal{E}_i] = p = 2^{1-k}$. Consider the dependency graph of these events. Vertices i and j are adjacent in the dependency graph if and only if E_i and E_j have at least one vertex in common. Thus, the maximum dependency among events $d = 2^{k-3}$. Since $4pd = 1$, it implies that there is a 2-coloring of vertices of H such that no edge $E_i \in E$ is monochromatic, *i.e.*, $\mathbb{P}[\cap_{i=1}^{m} \mathcal{E}_i^c] > 0$. In other words, H has property B. □

Notice that Corollary 3.2.1 says that the number of edges, m, can be arbitrarily large compared to k as long as no edge of H intersects more than 2^{k-3} other edges.

The main result of this section is:

Theorem 3.2.2 *Let $H = (V, E)$ be a k-uniform hypergraph with M not necessarily distinct edges. For each edge $E_i \in E$, let $f_i : E_i \to \{r, b\}$ be a 2-coloring of E_i. For $k \geq 2$, if every E_i intersects at most $2^{\alpha k+1}$ other $E_j \in E$, where $\alpha = 1/48$, then a 2-coloring $f : V \to \{r, b\}$ can be found in $O(M^{const})$ time such that for each edge E_i, $f|_{E_i} \neq f_i$.*

The equality in the last sentence of the theorem above means the 2-coloring of each edge E_i under f is different from its given (or *forbidden*) 2-coloring f_i. Note that the edges in Theorem 3.2.2 are not necessarily distinct. So we can have two copies E_{i_1} and E_{i_2} of each edge E_i and define $f_{i_1} : E_{i_1} \to \{r\}$ (completely red) and $f_{i_2} : E_{i_2} \to \{b\}$ (completely blue). Thus, the following result easily follows from Theorem 3.2.2.

Corollary 3.2.3 *Let $H = (V, E)$ be a k-uniform hypergraph with $|V| = n$ and $|E| = m$. For $k \geq 2$, if every E_i intersects at most $2^{\alpha k}$ other $E_j \in E$, where $\alpha = 1/48$, then a 2-coloring $f : V \to \{r, b\}$ can be found in $O(m^{const})$ time such that no edge of H is monochromatic.*

This is the algorithmic version of Corollary 3.2.1.

We present the proof of Theorem 3.2.2 in the following section. Important auxiliary results and concepts which act as ingredients in the proof of this main Theorem are given in Section 3.2.2.

3.2.1 Main Result

We start by noting that if $M < 2^k$ then Theorem 3.2.2 is not needed because a good 2-coloring $f : V \to \{r, b\}$ can be found using the method of conditional probabilities (see Theorem 3.2.7). So it can be assumed that $M \geq 2^k$. Assuming m to be an integer multiple of k, define $m = \beta \lg M$ where $\beta \geq 1$ is a constant. Using Definition 3.2.10 the edges of H, $\{E_1, \ldots, E_M\}$ can be grouped in disjoint $(2, 6)$-trees of size m/k, where size means the number of edges of H. The edges of each $(2, 6)$-tree, $\{E_1, \ldots, E_{m/k}\}$, are then united into one *big* set *i.e.*, we set

$$A_j = \cup_{i=1}^{m/k} E_i \quad \text{and} \quad g_j = \cup_{i=1}^{m/k} f_i \ ,$$

where $g_j : A_j \to \{r, b\}$. Let $\mathcal{A} = \{A_1, A_2, \dots\}$ be the family of big sets corresponding to $(2,6)$-trees of size m/k. We also make a small concession and allow $|E_i| \geq k$ $(= \gamma k$ for a constant $\gamma > 1)$ for all $i \in [M]$.

Before proceeding further let us outline Beck's idea: Apply Theorem 3.2.7 to the family of big sets $\mathcal{A}$ and color $S \subset V$ in M^{const} time. But, while doing so if we encounter an edge E_i with $|E_i \cap S_j| = \frac{k}{2}$ and still $g|_{E_i \cap S_j} = f_i|_{S_j}$, where $S_j \subset S$ is the set of vertices in S colored so far, then we call E_i *dangerous* and do not consider its uncolored vertices for coloring. This gives us a partial coloring g with $S = \{v \in V \mid g(v) = r \text{ or } g(v) = b\}$ and $V' = V - S$, the set of uncolored vertices. Also, for each $E_i \in E$ with $g|_{E_i \cap S} = f_i|_S$, $|E_i \cap S| \leq \frac{k}{2}$. This means each edge of the new hypergraph $H' = (V', E')$, where $E' = \{E_i \cap V' \mid g|_{E_i \cap S} = f_i|_S\}$, has at least $k/2$ vertices and it can be shown that H' breaks up into small components of size (number of vertices) at most $(\lg M)^2$. Again applying a similar procedure to H' we get H'' with very small disjoint components of size at most $\lg M$. The partial 2-coloring obtained so far has the following characteristics.

1. Some edges of H are already non-monochromatic so these edges are left out.

2. Some edges of H are still monochromatic. The still uncolored vertices belonging to these edges are contained in edges of H''.

Since H'' is decomposed into disjoint components with each component containing at most $\lg M$ vertices, all 2-colorings of a component can be exhaustively tried out in at most $2^{\lg M} = M$ time. Obviously the number of components is at most M, so all components can be 2-colored in at most M^2 time such that no edge of H'' is monochromatic.

Now we apply Theorem 3.2.7 to $\mathcal{A}$ with $l = \frac{m}{2}$. But in order to apply this theorem the following inequality should hold

$$|\mathcal{A}| < 2^{m/2} = M^{\beta/2} \ . \tag{3.1}$$

So now the aim is to bound $|\mathcal{A}|$ from above and impose the condition that this upper bound is at most $M^{\beta/2}$. For this purpose let T be a fixed tree on m/k vertices. Let

$$\mathcal{A}_T = \{A \in \mathcal{A} \mid \text{the corresponding } (2,6)\text{-tree of } A \text{ is isomorphic to } T\} \ . \tag{3.2}$$

Let d $(\leq 2^{\alpha k+1})$ be the maximum vertex degree of the dependency graph of H. By Definition 3.2.10 it is clear that every vertex (or edge of H) in a $(2,6)$-tree has at most d^6 neighbors and thus

$$|\mathcal{A}_T| \leq M(d^6)^{\frac{m}{k}-1} < M2^{6(\alpha m + m/k)} = M^{1+6\alpha\beta+6\beta/k} \ . \tag{3.3}$$

Furthermore, the number of non-isomorphic trees on m/k vertices is at most $4^{m/k}$, so

$$|\mathcal{A}| = 4^{m/k}|\mathcal{A}_T| < M^{1+6\alpha\beta+8\beta/k} \ . \tag{3.4}$$

Finally, in order to apply Theorem 3.2.7 to $\mathcal{A}$ we need that

$$1 + 6\alpha\beta + \frac{8\beta}{k} \leq \frac{\beta}{2} \ , \tag{3.5}$$

so that (3.1) is satisfied.

As mentioned before, after applying Theorem 3.2.7 to $\mathcal{A}$ for the first time and taking care of *dangerous* edges we obtain a partial coloring g of $S \subset V$. Call $V' = V - S$ the set of uncolored vertices. This partial coloring satisfies:

- for every $A \in \mathcal{A}$

$$|A \cap S| \geq \frac{m}{2} \implies g|_{A\cap S} \neq g_A|_S \ , \tag{3.6}$$

- for every dangerous edge E_i

$$g|_{S\cap E_i} = f_i|_S \quad \text{and} \quad |S \cap E_i| = \lceil \frac{k}{2} \rceil \ , \tag{3.7}$$

and

- for every *less dangerous* edge E_j

$$g|_{S\cap E_j} = f_j|_S \quad \text{and} \quad |S \cap E_j| < \lceil \frac{k}{2} \rceil \ . \tag{3.8}$$

Less dangerous edges are the result of at least one neighboring edge becoming dangerous. Consider now the family of monochromatic (dangerous and less dangerous) edges $E^1 = \{E_i \in E \mid g|_{S\cap E_i} = f_i|_S\}$. It can be shown that E^1 breaks up into smaller components:

Lemma 3.2.4 *If* $1+6\alpha\beta+\frac{8\beta}{k} \leq \frac{\beta}{2}$ *then the number of vertices in every connected component of the dependency graph of* E^1 *is at most*

$$\frac{m2^{4\alpha k}}{k} = \frac{\beta \lg M}{k} 2^{4\alpha k} \ .$$

This Lemma implies that the hypergraph $H' = (V', E')$, where the edge set $E' = \{E_i \cap V' \mid g|_{E_i\cap S} = f_i|_S\}$, has the following properties:

- It falls apart into components of at most $(\beta \lg M/k)2^{4\alpha k}$ edges.
- As discussed before (also see (3.7) and (3.8)), $|E_i'| \geq \lfloor k/2 \rfloor$ for all $E_i' \in E'$.

We choose $\beta = 4$, thus satisfying (3.5) (recall that $\alpha = 1/48$). So the connected components have size at most $(4 \lg M/k)2^{k/12}$. Now, to proceed further we have the following Lemma.

Lemma 3.2.5 *After the first coloring step*

(i) if $(4 \lg M/k)2^{k/12} < 2^{\lfloor k/2 \rfloor}$ *then a 2-coloring of* H' *can be obtained in polynomial time such that no edge is monochromatic.*

(ii) if $(4 \lg M/k)2^{k/12} \geq 2^{\lfloor k/2 \rfloor}$ *then the number of vertices in every connected component of the dependency graph of* H' *is* $O\left((\lg M)^{6/5}\right)$.

Proof. (i) We have

$$|E'| \leq \frac{4 \lg M}{k} 2^{k/12} < 2^{\lfloor k/2 \rfloor} . \tag{3.9}$$

Thus, we can straight away apply Theorem 3.2.7 to H' because all its conditions are satisfied. Thus, we obtain a 2-coloring of H' in polynomial time such that no edge is monochromatic.

(ii) In this case we have due to Lemma 3.2.4 the size of the connected components of H' is at most $(4 \lg M/k)2^{k/12}$ and this can be bounded from above as follows:

$$\frac{4 \lg M}{k} 2^{k/12} \geq 2^{\lfloor k/2 \rfloor} \implies 2^{k/12} \leq \left(\frac{8 \lg M}{k}\right)^{\frac{1}{5}} .$$

This means

$$\frac{4 \lg M}{k} 2^{k/12} \leq \frac{4 \lg M}{k} \left(\frac{8 \lg M}{k}\right)^{\frac{1}{5}} = O\left(\frac{\lg M}{k}\right)^{\frac{6}{5}} < O\left((\lg M)^{\frac{6}{5}}\right) . \tag{3.10}$$

□

If case (ii) of Lemma 3.2.5 holds then we have to re-apply the whole coloring procedure to color H' and this can be done by setting $k' = \lfloor k/2 \rfloor$, $\alpha' = \frac{k\alpha}{\lfloor k/2 \rfloor}$ and $\beta = 6$ because

- $|E'_i| \geq \lfloor k/2 \rfloor$ for all $E'_i \in E'$,
- any edge E'_i still intersects at most

$$2^{\alpha k+1} = 2^{\alpha' k'+1}$$

other edges, and

- for sufficiently large k, the following analog of (3.5) still holds *i.e.*

$$1 + 6\alpha^{'}\beta + \frac{8\beta}{k'} \leq \frac{\beta}{2} \; . \tag{3.11}$$

So, again using the coloring procedure we obtain a partial coloring h of $H^{'}$. Let $S^{'} = \{v \in V^{'} \mid h(v) = r \text{ or } h(v) = b\}$ be the set of colored vertices of $H^{'}$ and let $V^{''} = V^{'} - S^{'}$ be the set of uncolored vertices. Consider a new hypergraph $H^{''} = (V^{''}, E^{''})$ where $E^{''} = \{E_i^{'} \cap V^{'} \mid h|_{E_i^{'} \cap S^{'}} = f_i|_{S^{'}}\}$. The new coloring h and the new hypergraph $H^{''}$ satisfy the following conditions.

- For every edge $E_i^{'} \in E^{'}$ we have $|E_i^{'} \cap S^{'}| \leq \lceil k/4 \rceil$ if $h|_{E_i^{'} \cap S^{'}} = f_i|_{S^{'}}$.
- $|E_i^{''}| = |E_i^{'} \cap V^{'}| \geq \lfloor k/2 \rfloor - \lceil k/4 \rceil \geq k/6$ for every edge $E_i^{''} \in E^{''}$ of $H^{''}$.
- The dependency graph of $H^{''}$ breaks up into components of size at most $O\left(\frac{\lg \lg M}{k}\right) 2^{k/12}$ (see Lemma 3.2.4).

Just like Lemma 3.2.5 after the first coloring step, we have a similar Lemma after the second coloring step

Lemma 3.2.6 *After the second coloring step*

(i) if $O\left(\frac{\lg \lg M}{k}\right) 2^{k/12} < 2^{k/6}$ *then a 2-coloring of* $H^{''}$ *can be obtained in polynomial time such that no edge is monochromatic.*

(ii) if $O\left(\frac{\lg \lg M}{k}\right) 2^{k/12} \geq 2^{k/6}$ *then the number of vertices in every connected component of the dependency graph of* $H^{''}$ *is* $O\left(\frac{\lg M}{k}\right)$.

Proof. (i) Similar to the proof of the first case of Lemma 3.2.5, apply Theorem 3.2.7.
(ii) Clearly if $O\left(\frac{\lg \lg M}{k}\right) 2^{k/12} \geq 2^{k/6}$ then every connected component is of size $O\left((\frac{\lg \lg M}{k})^2\right)$. Recall that at the very beginning we assumed that $M \geq 2^k$ and for sufficiently large k it can be easily shown that $O\left((\frac{\lg \lg M}{k})^2\right) \leq O\left(\frac{\lg M}{k}\right)$. □

Now, since these components are nothing but groups of edges of $H^{''}$ and originally we assumed the hypergraph to be γk-uniform for a constant $\gamma > 1$, therefore, the number of vertices of the original hypergraph H in each component is $O(\lg M)$.

Notice that every edge of $H^{''}$ has at least $k/6$ vertices and

$$2^{\alpha k + 1} = 2^{\frac{k}{48} + 1} < 2^{\frac{k}{6} - 3} \; ,$$

for a sufficiently large k. Thus, all conditions of Corollary 3.2.1 are satisfied and the Lovász Local Lemma shows the existence of a 2-coloring $\tilde{f} : V^{''} \to \{r, b\}$ such that no edge of $H^{''}$ is monochromatic. Coloring $\tilde{f}$ can be found in time M^{const} time by exhaustively trying out all 2-colorings for each of the (at most M) components in $M 2^{O(\lg M)} = M^{const}$ time. In the end we obtain the required coloring $f = \tilde{f} \cup h \cup g$, hence proving Theorem 3.2.2.

3.2.2 Subsidiary Concepts and Results

This section includes definitions and results used in Section 3.2.1. We start with the derandomization step:

Theorem 3.2.7 *Let $\mathcal{H} = (V, A)$ be a hypergraph with $|V| = n$, $A = \{A_1, \ldots, A_L\}$ not necessarily distinct edges, and let $g_i : A_i \to \{r, b\}$ a 2-coloring for each edge A_i. Let l be a positive integer, $S \subseteq V$ be an arbitrary subset, and $v_1, v_2, \ldots, v_s$ be an arbitrary permutation of S. If $L < 2^l$ then a 2-coloring $g : S \to \{r, b\}$ of S can be found in polynomial time such that if $|S \cap A_i| \geq l$ for any $i \in [L]$ then $g|_{S \cap A_i} \neq g_i|_S$.*

Proof. The idea is to avoid bad colorings g_i and build an alternate coloring g using the method of conditional probabilities. Let $S_j = \{v_1, \ldots, v_j\} \subseteq S$ and suppose the colors of $v_1, \ldots, v_j$ are already fixed under the 2-coloring g. Define

$$P_j = \sum_{\substack{i \in [L], \\ g|_{S_j \cap A_i} = g_i|_{S_j}}} \frac{1}{2^{l - |S_j \cap A_i|}} .$$

Now we have to choose the color of v_{j+1}. Depending on this choice we have

$$P_j^r = \sum_{\substack{i \in [L], \\ g|_{S_j \cap A_i} = g_i|_{S_j}, \\ g_i(v_{j+1}) = r \text{ whenever } v_{j+1} \in A_i}} \frac{1}{2^{l - |S_{j+1} \cap A_i|}}$$

$$P_j^b = \sum_{\substack{i \in [L], \\ g|_{S_j \cap A_i} = g_i|_{S_j}, \\ g_i(v_{j+1}) = b \text{ whenever } v_{j+1} \in A_i}} \frac{1}{2^{l - |S_{j+1} \cap A_i|}} .$$

Notice that these are nothing but conditional probabilities that the new coloring g turns out to be exactly like the given bad colorings g_i, something we want to prevent. Clearly

$$P_j^r = P_j + \sum_{\substack{i \in [L], \\ g|_{S_j \cap A_i} = g_i|_{S_j}, \\ g_i(v_{j+1}) = r \text{ whenever } v_{j+1} \in A_i}} \frac{1}{2^{l - |S_j \cap A_i|}} + \sum_{\substack{i \in [L], \\ g|_{S_j \cap A_i} = g_i|_{S_j}, \\ g_i(v_{j+1}) = b \text{ whenever } v_{j+1} \in A_i}} \frac{1}{2^{l - |S_j \cap A_i|}} ,$$

and

$$P_j^b = P_j + \sum_{\substack{i \in [L], \\ g|_{S_j \cap A_i} = g_i|_{S_j}, \\ g_i(v_{j+1}) = b \text{ whenever } v_{j+1} \in A_i}} \frac{1}{2^{l - |S_j \cap A_i|}} + \sum_{\substack{i \in [L], \\ g|_{S_j \cap A_i} = g_i|_{S_j}, \\ g_i(v_{j+1}) = r \text{ whenever } v_{j+1} \in A_i}} \frac{1}{2^{l - |S_j \cap A_i|}} .$$

Thus, $P_j = \frac{1}{2}(P_j^r + P_j^b)$ and we color v_{j+1} red or blue depending on

$$P_{j+1} = \min\{P_j^r \ , \ P_j^b\} \leq P_j \ . \tag{3.12}$$

Repeating this step for all vertices of S gives us the following chain of inequalities

$$1 > \frac{L}{2^l} = P_0 \geq P_1 \geq \cdots \geq P_{s-1} \geq P_s \ . \tag{3.13}$$

But do we end up coloring S such that if $|S \cap A_i| \geq l$ then $g|_{S \cap A_i} \neq g_i|_S$, for any $i \in [L]$? Yes because if, for an edge A_i, $|S \cap A_i| \geq l$ but still $g|_{S \cap A_i} = g_i|_S$ then $P_S \geq 1$ which is a contradiction. □

We now give some definitions which provide the structure required for connecting the proof of Theorem 3.2.2 to Theorem 3.2.7.

Definition 3.2.8 *Let $H = (V, E)$ be a hypergraph with $|E| = M$ and let G be a graph with M vertices. G is called the* dependency graph *of H when (i, j) is an edge in G if and only if $E_i \cap E_j \neq \emptyset$, for $1 \leq i < j \leq M$.*

Definition 3.2.9 *Let $G = (V, E)$ be the dependency graph of a hypergraph H. For positive integers a and b, $G^{(a,b)}$ is the graph with the same vertex set as G but now (i, j) is an edge in $G^{(a,b)}$ if and only if $v_i, v_j \in V$ are at a distance of at least a and at most b in G.*

Definition 3.2.10 *Let $G = (V, E)$ be the dependency graph of a hypergraph H and let $G^{(a,b)}$ be given for positive integers a and b. A set $T \subseteq V$ is called* (a, b)-tree *if the subgraph induced by T in $G^{(a,b)}$ is connected.*

3.3 A General Setting

Beck's constructive result opened up the way for more similar applications [16]. In 1998 Molloy and Reed [21] gave a general theorem (Theorem 3.3.1) which captures almost all the applications of and including Beck's 2-coloring construction. This theorem lays down a set of general but weaker conditions, similar in essence to those of the Local Lemma. Any problem that satisfies these conditions automatically satisfies the Local Lemma conditions *i.e.*, not only can one prove the existence of the desired object or structure, one can actually construct such an object or structure in polynomial time. In this Section we discuss the general setting of [21]. Before we proceed, we must also mention that some problems and applications are not captured by these general conditions and therefore one has to look at subtle variations and modifications of this general method. Two such applications are discussed in subsequent sections.

As seen before, Lovász Local Lemma and its applications involve random trials, like assigning the vertices of a hypergraph one of the two colors (red and blue)

randomly, and events (whether an edge of the hypergraph is monochromatic) depending on those random trials. So, for a general discussion let us assume the following:

- $A = \{a_1, \ldots, a_n\}$ is the set of independent random trials.
- $\mathcal{E} = \{\mathcal{E}_1, \ldots, \mathcal{E}_m\}$ is the set of events.
- Each event $\mathcal{E}_i$ depends on the outcome of a subset $E_i \subseteq A$ of the independent random trials and $E = \{E_1, \ldots, E_m\}$.
- Two events $\mathcal{E}_i, \mathcal{E}_j \in \mathcal{E}$, $i \neq j$ are mutually dependent, or intersect each other, if and only if $E_i \cap E_j \neq \emptyset$.
- Without loss of generality let $H = (A, E)$ be a hypergraph with vertex set A and edge set E.

Note that if $\cup_{i=1}^{m} E_i \neq A$ then we can add an artificial edge to E (and therefore H) to make sure that H is indeed a hypergraph in accordance with Definition 3.1.1. We are now ready to state the general

Theorem 3.3.1 *If the following conditions are satisfied*

1. $\mathbb{P}[\mathcal{E}_i] \leq p$ *for each event* $\mathcal{E}_i$, $i \in [m]$,
2. *each event* $\mathcal{E}_i$ *depends on at most* d *other events* $\mathcal{E}_j$,
3. $pd^9 < \frac{1}{8e^3}$,
4. $|E_i| \leq k$ *for all* $i \in [m]$,
5. *for each* $j \in [n]$, *the size of the domain of* a_j *is at most* γ ,
6. *for each* $j \in [n]$, *the random trial* a_j *can be carried out in time* t_1 ,
7. *for each* $i \in [m]$, $a_{j_1}, \ldots, a_{j_l} \in E_i$, *and* $w_{j_1}, \ldots, w_{j_l}$ *in the domains of* $a_{j_1}, \ldots, a_{j_l}$ *respectively,* $\mathbb{P}[\mathcal{E}_i \mid a_{j_1} = w_{j_1}, \ldots, a_{j_l} = w_{j_l}]$ *can be computed in time at most* t_2 ,

then we have a randomized algorithm that finds suitable outcomes of the random trials $a_1, \ldots, a_n$ *in* $O(nd(t_1 + t_2) + n\gamma^{2kd \lg \lg n})$ *time such that* $\mathbb{P}[\cap_{i=1}^{m} \mathcal{E}_i^c] > 0$.

The proof is similar to Beck's technique described in Section 3.2.1. We make a constant number of passes and in each pass we try to fix the outcome of a portion of the independent random trials, all the time taking care that the probability of occurence of *bad* events does not come close to one. After each pass the hypergraph restricted to the random trials with still variable outcomes breaks

up into small connected components and each of these components requires less work than before.

Before proceeding to the proof let us define some useful structures. We will use a hypergraph $H = (A, E)$ with vertex set as the set of independent random trials and edge set E as defined before in this section. To this hypergraph H we will apply Definition 3.2.8, Definition 3.2.10 and a modified version of Definition 3.2.9 which is as follows

Definition 3.3.2 *((a, b)-tree) Let $G = (V, E)$ be the dependency graph of a hypergraph H. For positive integers a and b, $G^{(a,b)}$ is the graph with the same vertex set as G but now (i, j) is an edge in $G^{(a,b)}$ if and only if $v_i, v_j \in V$ are at a distance of exactly a or b in G.*

Proof. For finding suitable outcomes of random trials in A we need at most two passes.
First Pass. We start by carrying out trials $a_1, a_2, \ldots$ in sequential order. After carrying out each trial a_j we compute the conditional probability of each event

$$\mathbb{P}_i^c = \mathbb{P}[\mathcal{E}_i \mid a_1 = w_1, \ldots, a_j = w_j] ,$$

assuming that $w_1, \ldots, w_j$ were the outcomes so far. For any event $\mathcal{E}_i$ if $\mathbb{P}_i^c \leq p^{2/3}$ then we carry on conducting the random trials. Otherwise, if $\mathbb{P}_i^c > p^{2/3}$ we say that event $\mathcal{E}_i$ is *dangerous*. This means that there is a potential danger of occurence of $\mathcal{E}_i$ if we do not take care of it now. It is clear that any event $\mathcal{E}_i$ becomes dangerous with probability no more than $p^{1/3}$ because for a particular sequence of random trials and their outcomes

$$\mathbb{P}[\mathcal{E}_i \mid a_1 = w_1, \ldots, a_j = w_j] \cdot \mathbb{P}[a_1 = w_1, \ldots, a_j = w_j] = \mathbb{P}[\mathcal{E}_i] \leq p ,$$

and if the probability that $\mathbb{P}_i^c > p^{2/3}$ exceeds $p^{1/3}$ then $\mathbb{P}[\mathcal{E}_i] > p$, which contradicts our assumption. So, for any event $\mathcal{E}_i$ whenever we find $\mathbb{P}_i^c > p^{2/3}$ we do the following:

(i) undo the outcome of a_j, and

(ii) remove or freeze all yet to be conducted random trials in E_i including a_j from our list of trials. These trials are carried out in the second pass.

This ends the first pass.

Observe that at this point, for any event $\mathcal{E}_i$

$$\mathbb{P}[\mathcal{E}_i] = \mathbb{P}[\mathcal{E}_i \mid a_1 = w_1, \ldots, a_j = w_j] \leq p^{\frac{2}{3}} .$$

Since

$$1 > (8e^3 pd^9)^{\frac{2}{3}} = 4e^2 p^{\frac{2}{3}} d^6 > 4p^{\frac{2}{3}} d , \tag{3.14}$$

and the dependency d among events does not increase by removing the random trials fixed in the first pass, all conditions of the Local Lemma are satisfied and we are guaranteed that a combination of outcomes of the remaining trials exists such that $\mathbb{P}[\cap_{i=1}^{m} \mathcal{E}_i^c] > 0$. Consider the hypergraph $H = (A, E)$, its dependency graph and (a, b)-trees. We call a

- (a, b)-tree dangerous if all events corresponding to the edges of this tree are dangerous.
- (a, b)-tree maximal if there are no more vertices which can be added to it such that it still remains a (a, b)-tree.

Keeping these definitions in mind it is not hard to see that no event $\mathcal{E}_i \in \mathcal{E}$ intersects two different events that belong to different maximal dangerous $(1, 2)$-trees. This means maximal dangerous $(1, 2)$-trees are isolated from each other and we can deal with each such tree independently. We are now in a position to state the key ingredient of this proof.

Lemma 3.3.3 *The probability that there is no dangerous* $(1, 2)$*-tree of size greater than* $2d \lg n$ *is at least* $1 - \frac{1}{n}$.

Proof. (*Lemma* 3.3.3) Consider a $(1, 2)$-tree of size (number of vertices) dr where $r = 2 \lg n$. By removing unnecessary vertices (and the corresponding edges) at a distance one from each vertex in the tree we can obtain a $(2, 3)$-tree from this $(1, 2)$-tree. But we can remove at most d neighbours from each vertex. Thus, we obtain a $(2, 3)$-tree of size at least r from this $(1, 2)$-tree. We recall that vertices of these trees correspond to events and for every event $\mathcal{E}_i$, the probability that it becomes dangerous is at most $p^{1/3}$. Since the vertices are not adjacent in a $(2, 3)$-tree, its corresponding $(1, 2)$-tree of size dr is dangerous with probability at most $p^{\frac{r}{3}}$. For each a_j the number of $(1, 2)$-trees of size dr in H (see [17]) that a_j lies in is at most

$$\binom{d^3 r}{r} < (ed^3)^r \ .$$

So the total number of $(1, 2)$-trees of size dr is at most $n(ed^3)^r$ and the probability that at least one $(1, 2)$-tree of size dr becomes dangerous is at most

$$n(ed^3)^r p^{\frac{r}{3}} = n(ep^{\frac{1}{3}} d^3)^r \ . \tag{3.15}$$

But since $8e^3 pd^9 < 1$ (assumption 3 of Theorem 3.3.1) and $r = 2 \lg n$, the probability that at least one $(1, 2)$-tree of size dr becomes dangerous is at most

$$n(ep^{\frac{1}{3}} d^3)^r < n \left(\frac{1}{2}\right)^r = \frac{1}{n} \ . \tag{3.16}$$

Thus, there is no dangerous $(1,2)$-tree of size greater than $2d \lg n$ with probability at least $1 - \frac{1}{n}$. □

This gives us a randomized algorithm which delivers small dangerous $(1,2)$-trees with high probability. We can boost this probability further by repeating the whole procedure a constant number of times. Each random trial can be conducted in time t_1, after conducting a random trial we calculate conditional probabilities of at most $d+1$ events E_i containing this particular random trial. This can be done in at most $(d+1)t_2$ time. So all together, the first pass takes $O(nd(t_1+t_2))$ time even after a constant number of repetitions.

Second Pass. All dangerous $(1,2)$-trees have size at most $2d \lg n$. We now carry out the trials we froze/removed in the first pass. Notice that all these trials are confined to dangerous $(1,2)$-trees and their neighbors. An event becomes dangerous now if its conditional probability exceeds $p^{1/3}$. The analysis is similat to that of the first pass except that now with high probability all dangerous $(1,2)$-trees have size at most $2d \lg \lg n$. In other words, the hypergraph $H = (A, E)$ disintegrates into small connected components with each component containing at most $2d \lg \lg n$ edges. Recall the assumption that every edge E_i depends on at most k random trials. So, there are at most $2kd \lg \lg n$ frozen random trials in each dangerous $(1,2)$-tree and for each dangerous $(1,2)$-tree a suitable outcome of frozen trials can be found, by exhaustively searching their domains, in $O(\gamma^{2kd \lg \lg n})$ time. Doing this for all dangerous $(1,2)$-trees requires $O(n\gamma^{2kd \lg \lg n})$ time. Thus, with high probability we are able to find suitable outcomes of the random trials $a_1, \ldots, a_n$ in $O(nd(t_1+t_2) + n\gamma^{2kd \lg \lg n})$ time such that no event $\mathcal{E}_i$ occurs. □

Since Theorem 3.3.1 gives us a very general result, some remarks and observations are in order.

Remarks:

1. The assumption $8e^3pd^9 < 1$ is weaker than the normal assumption $ep(d+1) < 1$ (or $4pd < 1$) of the Lovász Local Lemma because it implies that the probability of occurence of bad, undesirable events p is less. On the other hand $ep(d+1) < 1$ allows p to be as high as $1/e(d+1)$ and still we have a guarantee of at least one combination of outcomes of random trials such that none of the bad events occur.

2. Sometimes, like in Section 3.2.1, Lemma 3.3.3 can be modified to convert the $\gamma^{2kd \lg \lg n}$ term in the running time to a reduced $2^{O(\lg n)}$ term. We recall that for property B, $\gamma = 2$ because the vertices are colored randomly with two colors.

3. Usually the dependency among events $d \geq k$. Since each random trial a_j

affects at most $d+1$ events and each event depends upon at most k random trials, we have $n(d+1) \geq mk$.

4. This algorithm can be derandomized by derandomizing Lemma 3.3.3. The idea is to make sure that there are no dangerous $(1,2)$-trees of size $2d \lg n$ or more. So we calculate the probability that there is at least one dangerous $(1,2)$-tree of size $2d \lg n$ or more. Since for derandomization we have to calculate conditional probabilities or evaluate the values of a pessimistic estimator (see Section 2.1), our choice for the method of derandomization depends on the answers of the following questions:

   - Is t_2 polynomially bounded from above?
   - Does a pessimistic estimator exist?
   - Can the values of this pessimistic estimator be calculated in polynomial time?

3.4 A Class of Integer Programs

The probabilistic method in general and Lovász Local Lemma in particular have been often used to solve different kinds of hard integer programs [1, 7, 17, 23, 25, 27]. In this section we apply the Local Lemma to solve the following integer program: Let $A \in \{0,1\}^{s \times n}$ and $b = (b_1, \ldots, b_s)^t \in \mathbb{N}^s$. We want to find vectors $x^{(k)} = (x_{1,k}, x_{2,k}, \ldots, x_{n,k})^t$ which

minimize $T = \max\{z \mid x_{j,z} > 0\ ,\ j = 1, 2, \ldots, n\}$ such that

(i) $Ax^{(k)} \leq b \quad \forall k = 1, 2, \ldots, T$,

(ii) $\sum_{k=1}^{T} x_{j,k} = 1 \quad \forall j = 1, 2, \ldots, n$ and

(iii) $x_{j,k} \in \{0,1\} \quad \forall j, k.$

From now on we will assume T_{opt} to be the value of the minimum solution of this integer program and T^* $(\leq T_{opt})$ to be the value of the minimum solution of the *relaxed* problem where we allow real variables $x_{j,k} \in [0,1]$, $\forall j, k$ instead of binary variables and leave the other constraints untouched.

This integer program has many applications [1, 2, 7, 26, 27]. For instance, let $H = (V, E)$ be a hypergraph with $|V| = n$ and $|E| = s$. The integer program described above models the following problem: using minimum number of colors, color the vertices of H such that in each hyperedge E_i there are no more than b_i vertices of any color. This coloring problem is a multicolor generalization of the usual (property B) hypergraph coloring problem where the aim is to avoid monochromatic edges. We will discuss this problem and other applications in detail in Chapter 4 where we will also give a hardness of approximation result.

We first give a polynomial time randomized algorithm for solving this integer program. Our approach has two steps. First, we generate a feasible solution of the corresponding relaxed integer program with value at most $\lceil (1+\epsilon)T_{opt} \rceil$ and then round the fractional solution to an integer one by generalizing the randomized version of LLL given by Lu [20] and Leighton et al. [17] to minimax integer programs with a *fixed* right hand side (the b_i's in our case). In doing so we end up violating the inequality constraints ((i) above) marginally. Later we show how this algorithm can be derandomized to yield a polynomial time deterministic algorithm. The randomized rounding procedure [22] used in the second step of our algorithm is as follows: round a variable $x_{j,k} \in \mathbb{R}$ to

- $\lceil x_{j,k} \rceil$ with probability $p = x_{j,k} - \lfloor x_{j,k} \rfloor$, and to
- $\lfloor x_{j,k} \rfloor$ with probability $1 - p = \lceil x_{j,k} \rceil - x_{j,k}$.

This rounding method has the advantage that if $y_{j,k}$ is the random variable corresponding to $x_{j,k}$ then $\mathbb{E}[y_{j,k}] = x_{j,k}$. We will soon see how useful this can be.

3.4.1 Generating Fractional Solutions

If we randomly round the fractional solution of the relaxed problem to an integral one it can lead to an infeasible solution of the original problem. Thus, to reduce infeasibility we define another LP problem with more restricted constraints than the original ones but with the same objective of minimizing T as in the original integer program. Let $\alpha \geq 1$ be a constant. The new constraints are:

(i) $Ax^{(k)} \leq b\alpha^{-1} \quad \forall k \in [T]$,

(ii) $\sum_{k=1}^{T} x_{j,k} = 1 \quad \forall j \in [n]$, and

(iii) $x_{j,k} \in [0,1] \quad \forall j,k.$

Subsequently we will refer to this problem as the *reduced* problem.

It is intuitively clear that after reducing the b_i's the value of the integer progrem, T, will only increase. In fact if we have a feasible solution of the relaxed problem with value $\tilde{T}$ then a feasible solution of the reduced problem with value $\lceil \alpha \tilde{T} \rceil$ can be constructed from it as follows: for all $k \in [\tilde{T}]$ let

$$\tilde{x}^{(k)} = (\tilde{x}_{1,k}, \ldots, \tilde{x}_{n,k})$$

be a feasible solution of the relaxed problem with value $\tilde{T}$. To obtain the corresponding feasible solution of the reduced problem, for all $j \in [n]$ set

$$\widehat{x}_{j,l} = \begin{cases} \tilde{x}_{j,l}\alpha^{-1} & \text{if } l = 1, \ldots, \tilde{T}; \\ \sum_{k=1}^{\tilde{T}} \frac{\tilde{x}_{j,k}(\alpha-1)\alpha^{-1}}{\lceil (\alpha-1)\tilde{T} \rceil} & \text{if } l = \tilde{T}+1, \tilde{T}+2, \ldots, \lceil \alpha \tilde{T} \rceil. \end{cases} \tag{3.17}$$

We now have the following lemma to prove our claim.

Lemma 3.4.1 *The vectors* $\widehat{x}^{(k)} = (\widehat{x}_{1,k}, \ldots, \widehat{x}_{n,k})$, $k = 1, \ldots, \lceil \alpha \tilde{T} \rceil$ *form a feasible solution of the reduced problem.*

Proof. Observe that $\widehat{x}_{j,k} \in [0,1]$ $\forall j,k$ because we multiply the variables $\tilde{x}_{j,t} \in [0,1]$ by some positive number which is at most one. For each $j \in [n]$

$$\sum_{l=1}^{\lceil \alpha \tilde{T} \rceil} \widehat{x}_{j,l} = \sum_{l=1}^{\tilde{T}} \tilde{x}_{j,l} \alpha^{-1} + \sum_{l=\tilde{T}+1}^{\lceil \alpha \tilde{T} \rceil} \sum_{k=1}^{\tilde{T}} \frac{\tilde{x}_{j,k}(\alpha-1)\alpha^{-1}}{\lceil (\alpha-1)\tilde{T} \rceil}$$

$$= \alpha^{-1} + \sum_{k=1}^{\tilde{T}} \tilde{x}_{j,k}(\alpha-1)\alpha^{-1} \;=\; 1 \; .$$

Furthermore, other constraints are also satisfied because for $l = 1, \ldots, \tilde{T}$

$$\sum_{j=1}^{n} a_{i,j} \widehat{x}_{j,l} = \sum_{j=1}^{n} a_{i,j} \tilde{x}_{j,l} \alpha^{-1} \leq b_i \alpha^{-1}$$

and for $l = \tilde{T}+1, \ldots, \lceil \alpha \tilde{T} \rceil$

$$\sum_{j=1}^{n} a_{i,j} \widehat{x}_{j,l} = \sum_{j=1}^{n} a_{i,j} \sum_{k=1}^{\tilde{T}} \frac{\tilde{x}_{j,k}(\alpha-1)\alpha^{-1}}{\lceil (\alpha-1)\tilde{T} \rceil}$$

$$\leq \sum_{k=1}^{\tilde{T}} \frac{b_i(\alpha-1)\alpha^{-1}}{\lceil (\alpha-1)\tilde{T} \rceil}$$

$$\leq b_i \alpha^{-1} \; .$$

Thus, all three constraints of the reduced problem are satisfied and hence the lemma is proved. □

Note that the optimal solution of the relaxed problem is of value at most n. Therefore, using the methods of Lenstra et al. [18] and [14] the optimal solution $(\widehat{x}_{j,k})$ of the reduced problem and the corresponding value $(\widehat{T})$ can be found by using binary search between 1 and αn and solving at most $O(\log n)$ linear programs in polynomial time. Lemma 3.4.1 implies that $\widehat{T} \leq \lceil \alpha T^* \rceil \leq \lceil \alpha T_{opt} \rceil$. We will also assume, w.l.o.g, that this optimum feasible solution is basic. The next step is to round this fractional solution to a feasible solution of our original integer program. Also, from now onwards in this section all logarithms should be read as base e logarithms.

3.4.2 Randomized Rounding

The randomized rounding procedure is quite simple. The equality constraints *i.e.*, $\sum_{k=1}^{T} x_{j,k} = 1$, $\forall j \in [n]$, provide us with a straightforward method. For

each $j \in [n]$ we round exactly one $\widehat{x}_{j,k}$ to one according to the probabilities $\widehat{x}_{j,1}, \ldots, \widehat{x}_{j,\widehat{T}}$. For each $j \in [n]$ and $k \in [\widehat{T}]$ define binary random variable

$$y_{j,k} = \begin{cases} 1 & \text{with probability } \widehat{x}_{j,k} \ , \\ 0 & \text{with probability } 1 - \widehat{x}_{j,k} \ . \end{cases} \tag{3.18}$$

For example, if $y_{j,1} = 1$, that means $\widehat{x}_{j,1}$ has been rounded to one, then $y_{j,k} = 0$ for all $k \in \{2, 3, \ldots, \widehat{T}\}$. Let

$$\left(Ay^{(k)}\right)_i = a_i \cdot y^{(k)} = \sum_{j=1}^{n} a_{i,j} y_{j,k} \ ,$$

then by the linearity of expectation and equality constraints of the integer program,

$$\mathbb{E}[\left(Ay^{(k)}\right)_i] = \sum_{j=1}^{n} a_{i,j} \mathbb{E}[y_{j,k}] = \sum_{j=1}^{n} a_{i,j} \widehat{x}_{j,k} \ \leq \ \frac{b_i}{\alpha} \ . \tag{3.19}$$

For each $i \in [s]$ and $k \in [\widehat{T}]$ define an event

$$\xi_{i,k} \equiv \text{“ } \left(Ay^{(k)}\right)_i \ \geq \ \frac{b_i}{\alpha}(1 + \delta_i) \text{ “} \ , \tag{3.20}$$

where $\delta_i > 0$ for each $i \in [s]$. $\xi_{i,k}$ is the event that after rounding the dot product $\left(Ay^{(k)}\right)_i$ violates the inequality constraint by a multiplicative factor of at least $(1 + \delta_i)$. Now, since we have defined our events we also have to deal with the dependencies among these events. To get a bound on the maximum dependency $\mathcal{D}$ among these events let us define a few parameters.

- Let

$$d = \max_{j \in [n]} |\{a_{i,j} \mid a_{i,j} \neq 0 \ , \ i \in [s]\}| \tag{3.21}$$

  be the maximum number of non-zero entries in any column of matrix A.

- Let

$$\eta = \max_{j \in [n]} |\{\widehat{x}_{j,k} \mid 0 < \widehat{x}_{j,k} < 1 \ , \ k \in [\widehat{T}]\}| \ . \tag{3.22}$$

  In other words for any $j \in [n]$, η is the maximum number of variables among $\widehat{x}_{j,1}, \ldots, \widehat{x}_{j,\widehat{T}}$ which have to be rounded.

- For a fixed i and k, and for any $j \in [n]$, let $\nu_j = 1$ if $a_{i,j} \neq 0$ and $\widehat{x}_{j,k} \in (0,1)$ and $\nu_j = 0$ otherwise. Define

$$\tau = \max_{i \in [s]\,,\, k \in [\widehat{T}]} \sum_{j=1}^{n} \nu_j \ . \tag{3.23}$$

  So τ is the maximum number of variables to be rounded in any one of the inequality constraints $Ax^{(k)} \leq b_i \alpha^{-1}$.

It is clear that two events $\xi_{i,k}$ and $\xi_{i',k'}$ are mutually dependent if for a $j \in [n]$, $\widehat{x}_{j,k}, \widehat{x}_{j,k'} \in (0,1)$ and $a_{i,j},\ a_{i',j} \neq 0$ because rounding $\widehat{x}_{j,k}$ or $\widehat{x}_{j,k'}$ to one will affect both $\xi_{i,k}$ and $\xi_{i',k'}$. Thus, maximum dependency among these events $\mathcal{D}$ is at most $d\eta\tau$ and we can use the Lovász Local Lemma to show

Theorem 3.4.2 *Given the integer program and the optimal solution $(\widehat{x}_{j,k})$ of the corresponding reduced problem, there exist vectors with binary components $\check{y}^{(k)} = (\check{y}_{1,k}, \ldots, \check{y}_{n,k}) \in \{0,1\}^n$ such that $\left(A\check{y}^{(k)}\right)_i < b_i \alpha^{-1}(1+\delta_i)$ for all $i \in [s],\ k \in [\widehat{T}]$.*

Proof. By Corollary 2.4.5, if $b_i \geq \alpha \log(e\gamma(\mathcal{D}+1))/2$ then for

$$\delta_i = \Theta\left(\sqrt{\frac{\alpha \log(e\gamma(\mathcal{D}+1))}{b_i}}\right) \tag{3.24}$$

direct calculation shows that

$$Pr[\xi_{i,k}] \leq G(b_i\alpha^{-1}, \delta_i) \leq p = \frac{1}{(e\gamma(\mathcal{D}+1))} \ ,$$

where $i \in [s]$ and $\gamma \geq 1$ is also a constant. Since $ep(\mathcal{D}+1) = \gamma^{-1} < 1$, the Local Lemma ensures the existence of vectors $\check{y}^{(k)} = (\check{y}_{1,k}, \ldots, \check{y}_{n,k}) \in \{0,1\}^n$ such that for all $i \in [s],\ k \in [\widehat{T}]$, no event $\xi_{i,k}$ occurs. □

Our objective is to use the algorithmic version of LLL to obtain good approximate solutions for our integer program. These solutions must have an extra property, they should not violate the inequality constraints by much. Note that Theorem 3.4.2 gives a probabilistic way of generating a good solution, but the success probability *i.e.*, the probability that $\left(A\check{y}^{(k)}\right)_i < b_i\alpha^{-1}(1+\delta_i)$ for all i, k, is at least $(1-ep)^{(s\widehat{T})}$ which is too small. Thus, we will have to use the algorithmic Local Lemma and for this purpose we will need the weaker condition

$$p\mathcal{D}^4 \leq 1 \tag{3.25}$$

rather than the usual $ep(\mathcal{D}+1) \leq 1$. Thus, we need

$$p = \frac{1}{(e\gamma(\mathcal{D}+1))^4} \ . \tag{3.26}$$

The important fact is that this changes the bound on δ_i (see 3.24) by just a constant factor because of Corollary 2.4.5. Since rounding a fractional solution does not increase the value of the objective function, we end up with a $\widehat{T} \leq \lceil \alpha T_{opt} \rceil$-approximate solution. Another thing to note is that $\mathcal{D} \leq d\eta\tau$ where $d \leq s$, $\eta \leq \widehat{T}$ and $\tau \leq s\widehat{T}$ (here we use the assumption that the feasible solution is basic). Next, we will construct an approximate solution without violating the inequality constraints much, first in a randomized, then in a deterministic (derandomized) way.

3.4.3 Randomized Construction

At the outset we again recall Definitions 3.2.8, 3.3.2 and 3.2.10 of and related to the dependency graph of events. For each $i \in [s]$ and $k \in [\widehat{T}]$ we define n zero-one, independent random variables $z_{i,j,k} = a_{i,j}y_{j,k}$. Notice that random variables $z_{i,j,1}, \ldots, z_{i,j,\widehat{T}}$ are not independent because of the equality constraints. The event $\xi_{i,k}$ can be re-written as

$$\xi_{i,k} = \text{“} \sum_{j=1}^{n} z_{i,j,k} \geq b_i \alpha^{-1}(1+\delta_i) \text{”} .$$

Let $G(V, E)$ be the dependency graph with vertex set $V = [s\widehat{T}]$ where each vertex corresponds to an event $\xi_{i,k}$ and two vertices are adjacent iff they affect each other.

Theorem 3.4.3 *([1]) Given the integer program, for any $\alpha \in (1,2)$, an approximate solution $\check{y}^{(k)} = (\check{y}_{1,k}, \ldots, \check{y}_{n,k}) \in \{0,1\}^n$, for $k \in [\widehat{T}]$, with value of the objective function at most $\lceil \alpha T_{opt} \rceil$ can be found in randomized polynomial time such that $\left(A\check{y}^{(k)}\right)_i < b_i\alpha^{-1}(1+\delta_i)$ for all $i \in [s]$, $k \in [\widehat{T}]$.*

Note that δ_i here is the same as in 3.24, except that the constant hidden in the Θ notation is much larger.

Proof. Call a vertex $v_{i,k} \in V$ *bad* if

$$\sum_{j=1}^{n} z_{i,j,k} \geq \frac{b_i}{\alpha}\left(1 + H\left(\frac{b_i}{\alpha}, \frac{1}{6\mathcal{D}^4}\right)\right) . \tag{3.27}$$

Thus a vertex $v_{i,k} \in V$ is bad with probability at most $\frac{1}{6\mathcal{D}^4}$ (by Corollary 2.4.5) where $\mathcal{D} \leq d\eta\tau$ is the maximum dependency. Let $\mathcal{T}$ be a $(1,2)$-tree. Consider a vertex $v_{i,k} \in V$ and let

$$I_{i,k,\mathcal{T}} = \{j \in [n] \mid \exists v_{\hat{i},\hat{k}} \in \mathcal{T}\backslash\{v_{i,k}\}, \text{ s.t. } a_{i,j}, a_{\hat{i},j} \neq 0 \text{ and } \widehat{x}_{j,k}, \widehat{x}_{j,\hat{k}} \notin \{0,1\}\}.$$

$I_{i,k,\mathcal{T}}$ is the set of indices of those variables $\widehat{x}_{j,k}$ which when re-rounded, affect event $\xi_{i,k}$. Let $\widetilde{I}_{i,k,\mathcal{T}} = [n] \backslash I_{i,k,\mathcal{T}}$. We call a vertex $v_{i,k} \in V$ bad for $\mathcal{T}$ if

$$\sum_{j \in \widetilde{I}_{i,k,\mathcal{T}}} z_{i,j,k} \geq \mathbb{E}[\sum_{j \in \widetilde{I}_{i,k,\mathcal{T}}} z_{i,j,k}] + \frac{b_i}{\alpha} H\left(\frac{b_i}{\alpha}, \frac{1}{6\mathcal{D}^4}\right) . \tag{3.28}$$

This also happens with probability at most $\frac{1}{6\mathcal{D}^4}$. Notice that the probability bounds on these events hold because we assume that the δ_i's satisfy the required bounds. We say that a $(1,2)$-tree $\mathcal{T}$ is bad if every vertex in $\mathcal{T}$ is bad or bad for $\mathcal{T}$.

To round the fractional solution properly we require at most three phases. *Phase* 1 requires the following lemma which is similar to Lemma 3.3.3.

Lemma 3.4.4 *With probability at least* $1 - \frac{1}{s\widehat{T}}$ *all bad* $(1,2)$*-trees have size at most* $2\mathcal{D}\log(s\widehat{T})/\log\mathcal{D}$.

Proof. (Lemma 3.4.4) Consider a $(1,2)$-tree of size $\mathcal{D}r$ where

$$r = \frac{2\log(s\widehat{T})}{\log\mathcal{D}} .$$

By removing unnecessary vertices (and the corresponding edges) at a distance one from each vertex in the tree we can obtain a $(2,3)$-tree from this $(1,2)$-tree. But we can remove at most $\mathcal{D}$ neighbours from each vertex. Thus, we obtain a $(2,3)$-tree of size at least r from this $(1,2)$-tree. Since the vertices are not adjacent in a $(2,3)$-tree, a $(1,2)$-tree of size $\mathcal{D}r$ is bad with probability at most

$$\left(\frac{1}{6\mathcal{D}^4} + \frac{1}{6\mathcal{D}^4}\right)^r = \frac{1}{(3\mathcal{D}^4)^r} .$$

The number of $(1,2)$-trees of size $\mathcal{D}r$ is at most ([17])

$$\frac{s\widehat{T}}{(\mathcal{D}^2-1)\mathcal{D}r+1}\binom{\mathcal{D}^3 r}{r} < s\widehat{T}(3\mathcal{D}^3)^r .$$

So the probability of obtaining a bad $(1,2)$-tree of size $\mathcal{D}r$, after rounding, is at most

$$s\widehat{T}(3\mathcal{D}^3)^r \frac{1}{(3\mathcal{D}^4)^r} \leq \frac{s\widehat{T}}{\mathcal{D}^r} = \frac{1}{s\widehat{T}} .$$

Thus, with probability at least $1 - \frac{1}{s\widehat{T}}$ no bad $(1,2)$-tree has size more than $2\mathcal{D}\log(s\widehat{T})/\log\mathcal{D}$. □

We continue with the proof of Theorem 3.4.3.

Phase 2. In Phase 2 we take a maximal bad $(1,2)$-tree $\mathcal{T}$ and reround all variables contained in it. But this may harm the neighbours of $\mathcal{T}$ which are not bad. Let $N(\mathcal{T})$ be the set of neighbours of $\mathcal{T}$. The following simple but crucial observation is the key to Phase 2: No vertex $v_{i,k}$ of G is adjacent to two vertices belonging to different maximal bad $(1,2)$-trees and no vertex in $N(\mathcal{T})$ can harm any other bad $(1,2)$-tree except $\mathcal{T}$. This means that we can deal with each bad $(1,2)$-tree independently.

Consider $v_{i,k} \in N(\mathcal{T})$, since $v_{i,k}$ is not bad and is not bad for $\mathcal{T}$, therefore

$$\sum_{j \in \widetilde{I}_{i,k,\mathcal{T}}} z_{i,j,k} < \mathbb{E}[\sum_{j \in \widetilde{I}_{i,k,\mathcal{T}}} z_{i,j,k}] + \frac{b_i}{\alpha} H\left(\frac{b_i}{\alpha}, \frac{1}{6\mathcal{D}^4}\right) .$$

After rerounding, let a vertex $v_{i,k} \in \mathcal{T} \cup N(\mathcal{T})$ be bad if

$$\sum_{j=1}^{n} z_{i,j,k} \geq \frac{b_i}{\alpha}\left(1 + 2H\left(\frac{b_i}{\alpha}, \frac{1}{6\mathcal{D}^4}\right)\right) . \tag{3.29}$$

If $v_{i,k} \in \mathcal{T}$ then this happens with probability at most $\frac{1}{6\mathcal{D}^4}$ and if $v_{i,k} \in N(\mathcal{T})$ then

$$\sum_{j \in I_{i,k,\mathcal{T}}} z_{i,j,k} \geq \mathbb{E}[\sum_{j \in I_{i,k,\mathcal{T}}} z_{i,j,k}] + \frac{b_i}{\alpha} H\left(\frac{b_i}{\alpha}, \frac{1}{6\mathcal{D}^4}\right) \tag{3.30}$$

because of (3.29) and this also happens with probability at most $\frac{1}{6\mathcal{D}^4}$. We note that there are at most

$$\frac{1}{\log \mathcal{D}}(2\mathcal{D}\log(s\widehat{T}) + 2\mathcal{D}^2 \log(s\widehat{T}))$$

vertices in $\mathcal{T} \cup N(\mathcal{T})$. Now, depending on the value of dependency $\mathcal{D}$ we have two cases:

1. if $\mathcal{D} \geq \sqrt{\log s\widehat{T}/\log\log s\widehat{T}}$ then the probability of having a bad vertex is at most

$$2\mathcal{D}(\mathcal{D}+1)\frac{\log s\widehat{T}}{\log \mathcal{D}}\frac{1}{6\mathcal{D}^4} < 1 .$$

   In this case we can use a pessimistic estimator (see next section) and find a good rerounding in deterministic polynomial time, otherwise

2. if $\mathcal{D} < \sqrt{\log s\widehat{T}/\log\log s\widehat{T}}$ then it means that we cannot apply derandomization and the components ($(1,2)$-trees) are still not small enough to

search for good reroundings exhaustively. So we apply Phase 1 to all bad $(1,2)$-trees to obtain smaller bad $(1,2)$-trees of size

$$O\left(\frac{\sqrt{\log s\widehat{T}\log\log s\widehat{T}}}{\log\mathcal{D}}\right).$$

Phase 3. Till now we have rounded some variables $\widehat{x}_{j,k}$ to either zero or one successfuly. But, we still have to reround the variables contained in bad $(1,2)$-trees. Since the probability of occurence of an event corresponding to a bad vertex is still at most $1/6\mathcal{D}^4$, all conditions of the Local Lemma $(ep(\mathcal{D}+1)<1)$ are satisfied and we are guaranteed the existence of a good rerounding. We know that each bad vertex corresponds to a row of the inequality constraints and the number of variables to be rounded in each such constraint is at most τ. Now, because

$$\tau \leq \mathcal{D} < \sqrt{\frac{\log s\widehat{T}}{\log\log s\widehat{T}}}$$

and the size of each bad $(1,2)$-tree is $O(\sqrt{\log s\widehat{T}\log\log s\widehat{T}}/\log\mathcal{D})$, the number of variables to be rounded in each bad $(1,2)$-tree is at most

$$O\left(\frac{\sqrt{\log s\widehat{T}\log\log s\widehat{T}}}{\log\mathcal{D}}\right)\left(\sqrt{\frac{\log s\widehat{T}}{\log\log s\widehat{T}}}\right) = O\left(\frac{\log s\widehat{T}}{\log\mathcal{D}}\right).$$

Since each variable can be rounded to either 0 or 1, we can now try all possible combinations exhaustively in each bad $(1,2)$-tree and this can be done in polynomial time. Thus, we have proved Theorem 3.4.3 □

This algorithm can be made deterministic by derandomizing Lemma 3.4.4. This is the subject of our discussion in the next section.

3.4.4 Derandomization

We have the dependency graph $G=(V,E)$ of the events (3.20). Let $\mathcal{Q}$ be the set of all $(1,2)$-trees of size $2\mathcal{D}\log(s\widehat{T})/\log\mathcal{D}$. The idea is to round the variables in such a way that no $(1,2)$-tree $\mathcal{T}\in\mathcal{Q}$ becomes bad. We denote $\widehat{x}_j$ to be the vector $(\widehat{x}_{j,1},\widehat{x}_{j,2},\ldots,\widehat{x}_{j,\widehat{T}})$ and y_j to be $(y_{j,1},y_{j,2},\ldots,y_{j,\widehat{T}})$ for any $j\in[n]$. For all $j\in[n]$ we choose y_j so as to minimize the probability of some $(1,2)$-tree $\mathcal{T}\in\mathcal{Q}$ turning bad if we randomly round $\widehat{x}_{j+1},\ldots,\widehat{x}_n$ conditional on the already fixed $x_1=y_1,\ldots,x_{j-1}=y_{j-1}$. But calculating conditional probabilities takes a lot of time. Recall that the number of variables to be rounded in each inequality

constraint is at most τ and hence we may need as much as $O(2^\tau)$ time to calculate conditional probabilities. So now our aim is to find a pessimistic estimator. This is done in the proof of the following lemma which is the derandomized version of Lemma 3.4.4.

Lemma 3.4.5 *The variables $\widehat{x}_{j,k}$ can be rounded in deterministic polynomial time such that all bad $(1,2)$-trees have size at most $2\mathcal{D}\log(s\widehat{T})/\log\mathcal{D}$.*

Proof. Let $\mathcal{T} \in \mathcal{Q}$ be a $(1,2)$-tree and let $\mathcal{T}_{2,3}$ be an arbitrary maximal $(2,3)$-tree in $\mathcal{T}$. Suppose that we have already fixed $y_1, \ldots, y_{j-1}$, then the probability that at least one $(1,2)$-tree turns bad can be bounded from above as follows:

$$\begin{aligned}\mathbb{P}_{j-1} &= \mathbb{P}_{y_j,\cdots,y_n}[(\exists \mathcal{T} \in \mathcal{Q}) \wedge (\mathcal{T} \text{ is bad}) \mid y_1, \ldots, y_{j-1}] \\ &\leq \sum_{\mathcal{T}\in\mathcal{Q}} \prod_{v_{i,k}\in\mathcal{T}_{2,3}} \mathbb{P}[(v_{i,k} \text{ is bad}) \vee (v_{i,k} \text{ is bad for } \mathcal{T}) \mid y_1, \ldots, y_{j-1}].\end{aligned}$$

Let us set

$$\begin{aligned}e_{i,k,\mathcal{T}} &= \mathbb{E}[\sum_{j\in\widetilde{I}_{i,k,\mathcal{T}}} z_{i,j,k}] + b_i\alpha^{-1}H(b_i\alpha^{-1}, \frac{1}{6\mathcal{D}^4}) \ , \\ l &= \lceil b_i\alpha^{-1}H(b_i\alpha^{-1}, \frac{1}{6\mathcal{D}^4})\rceil \ , \text{ and} \\ g &= b_i\alpha^{-1}(1 + H(b_i\alpha^{-1}, \frac{1}{6\mathcal{D}^4}) \ .\end{aligned}$$

Let $S_l^{(i,k)}(I)$ denote the polynomial S_l (see Lemma 2.4.6) on input $\mathbb{E}[z_{i,j,k}]$ where $j \in I$ and $I \subseteq [n]$. Consider the following candidate PE_{j-1} for the pessimistic estimator when $y_1, \ldots, y_{j-1}$ have been fixed:

$$PE_{j-1} = \sum_{\mathcal{T}\in\mathcal{Q}} \prod_{v_{i,k}\in\mathcal{T}_{2,3}} \left(\frac{S_l^{(i,k)}([n])}{\binom{g}{l}} + \frac{S_l^{(i,k)}(\widetilde{I}_{i,k,\mathcal{T}})}{\binom{e_{i,k,\mathcal{T}}}{l}} \right) . \tag{3.31}$$

By Lemma 2.4.7 it is clear that $Pr_{j-1} \leq PE_{j-1}$ because by (3.27)

$$\mathbb{P}[v_{i,k} \text{ is bad} \mid y_1, \ldots, y_{j-1}] \ \leq \ \frac{S_l^{(i,k)}([n])}{\binom{g}{l}} \ ,$$

and by (3.28)

$$\mathbb{P}[v_{i,k} \text{ is bad for } \mathcal{T} \mid y_1, \ldots, y_{j-1}] \ \leq \ \frac{S_l^{(i,k)}(\widetilde{I}_{i,k,\mathcal{T}})}{\binom{e_{i,k,\mathcal{T}}}{l}} \ .$$

Furthermore $S_l^{(i,k)}([n])$ and $S_l^{(i,k)}(\widetilde{I}_{i,k,\mathcal{T}})$ are the sums of products of at most n numbers

$$\mathbb{E}[z_{i,j,k}] = \begin{cases} a_{i,j}y_{j,k} & \text{if } \widehat{x}_{j,k} \text{ has been rounded ,} \\ a_{i,j}\widehat{x}_{j,k} & \text{otherwise ,} \end{cases}$$

and can be calculated easily using dynamic programming. So PE_{j-1} is a pessimistic estimator. It is not hard to see that

$$\begin{aligned} PE_{j-1} &= \sum_{\mathcal{T}\in\mathcal{Q}} \prod_{v_{i,k}\in\mathcal{T}_{2,3}} \mathbb{E}_{y_j}\left[\frac{S_l^{(i,k)}([n])}{\binom{g}{l}} + \frac{S_l^{(i,k)}(\widetilde{I}_{i,k,\mathcal{T}})}{\binom{e_{i,k,\mathcal{T}}}{l}}\right] \\ &= \sum_{\mathcal{T}\in\mathcal{Q}} \mathbb{E}_{y_j}\left[\prod_{v_{i,k}\in\mathcal{T}_{2,3}}\left(\frac{S_l^{(i,k)}([n])}{\binom{g}{l}} + \frac{S_l^{(i,k)}(\widetilde{I}_{i,k,\mathcal{T}})}{\binom{e_{i,k,\mathcal{T}}}{l}}\right)\right] \\ &= \mathbb{E}_{y_j}[PE_j]. \end{aligned}$$

because any two nodes of $\mathcal{T}_{2,3}$ do not affect each other, *i.e.*, do not have any variables in common. Thus, at each step we choose y_j to minimize PE_j and obtain

$$\mathbb{P}_n \le PE_n \le PE_{n-1} \le \cdots \le PE_1 \le PE_0 \le \frac{1}{s\widehat{T}}\,.$$

The upper bound on PE_0 follows from Lemmas 2.4.7 and 3.4.4. Since $\mathbb{P}_n$ can be either 0 or 1, we successfully find a rounding such that no $(1,2)$-tree of size $2\mathcal{D}\log(s\widehat{T})/\log\mathcal{D}$ is bad.

The number of $(1,2)$-trees in $\mathcal{Q}$ is $O(poly(s\widehat{T}))$ and they can be enumerated in polynomial time (see [17]). This proves Lemma 3.4.5. □

Thus we obtain the deterministic version of Theorem 3.4.3 :

Theorem 3.4.6 *([1]) Given the integer program, for any $\alpha \in (1,2)$, an approximate solution $\check{y}^{(k)} = (\check{y}_{1,k}, \ldots, \check{y}_{n,k}) \in \{0,1\}^n$, for $k \in [\widehat{T}]$, with value of the objective function at most $\lceil \alpha T_{opt} \rceil$, can be found in deterministic polynomial time such that $\left(A\check{y}^{(k)}\right)_i < b_i\alpha^{-1}(1+\delta_i)$ for all $i \in [s]$, $k \in [\widehat{T}]$ and suitably defined $\delta_i > 0$.*

Proof. Everything remains the same as in the proof of Theorem 3.4.3 except that Lemma 3.4.4 is replaced by Lemma 3.4.5. □

Chapter 4

Applications

Quite a few problems can be written in the form of the integer program described in Section 3.4. We again recall that this integer program had the objective function *minimize* $T = \max\{z \mid x_{j,z} > 0\ ,\ j = 1, 2, \ldots, n\}$ and the following constraints

(i) $Ax^{(k)} \leq b \quad \forall k = 1, 2, \ldots, T$,

(ii) $\sum_{k=1}^{T} x_{j,k} = 1 \quad \forall j = 1, 2, \ldots, n$ and

(iii) $x_{j,k} \in \{0, 1\} \quad \forall j, k.$

Here $A \in \{0,1\}^{s \times n}$, $b = (b_1, \ldots, b_s)^t \in \mathbb{N}^s$, and $x^{(k)} = (x_{1,k}, x_{2,k}, \ldots, x_{n,k})^t$. In this chapter we discuss two applications of Theorem 3.4.6, namely Constrained Hypergraph Coloring (CHC) and Resource Constrained Scheduling (RCS).

4.1 Constrained Hypergraph Coloring

We briefly touched this problem in Section 3.4 but let us define it formally. As the name suggests, we are given a hypergraph and the aim is to color the vertices of this hypergraph according to some criteria. The constrained hypergraph coloring problem is:

Definition 4.1.1 *Given a hypergraph $H = (V, E)$ with $|V| = n$ vertices and $|E| = s$ edges, color its vertices using minimum number of colors such that in each hyperedge E_i there are no more than $b_i \in \mathbb{N}$ vertices of any color.*

It is not hard to see that a CHC problem can be written as the above mentioned integer program. Indeed, given a hypergraph $H = (V, E)$ with $V = \{v_1, \ldots, v_n\}$ and $E = \{E_1, \ldots, E_s\}$, let

- $A = \{a_{i,j}\} \in \{0,1\}^{s \times n}$ be its edge-vertex incidence graph. That is,

$$a_{i,j} = \begin{cases} 1 & \text{if edge } E_i \text{ contains vertex } v_j \ , \\ 0 & \text{otherwise.} \end{cases}$$

- $b = (b_1, \ldots, b_s)^t \in \mathbb{N}^s$, where the components are as mentioned in Definition 4.1.1.

The integer program tries to minimize T, the number of colors, because for every $j \in [n]$ and $k \in [T]$,

$$x_{j,k} = \begin{cases} 1 & \text{if vertex } v_j \text{ gets color } k \ , \\ 0 & \text{otherwise.} \end{cases}$$

Notice that this is the last ((iii)) constraint of the integer program. The first two constraints make sure that

(i) for each color $k \in [T]$, in each edge E_i, the number of vertices with color k is no more than b_i, and

(ii) each vertex v_i gets exactly one out of T colors.

Special cases of CHC are well known and have been intensively studied. For a simple graph with $b_i = 1$ for all i, CHC problem is nothing but the problem of coloring the graph properly. For a hypergraph H with hyperedges $F_1, \ldots, F_s$, and $b_i = |F_i| - 1$ for all i, CHC is closely related to the property B problem (see Definition 3.1.3) because whenever H has property B, CHC is equivalent to the problem of finding a non-monochromatic 2-coloring of H. Note that property B requires logarithmic lower bounds on the b_i's (see Theorem 3.2.2). In fact if H is r-uniform, then by the Lovász Local Lemma it has property B if its maximum edge degree $\mathfrak{D} \leq 2^{r-3}$, and this implies

$$b_i = r - 1 \geq \log \mathfrak{D} + 2$$

for all i. We will see that in case of the CHC problem a similar condition is required to obtain near-optimum colorings. A coloring problem related to the CHC problem, which also generalizes the property B problem to multicolors has been studied by Lu [20]. There the aim is to color the vertices of H with k given colors such that no color appears more than b times in any edge. Assuming H to be r-uniform, the result of [20] says that if H is k-colorable and $r/k = (\log(dr))^{1+\delta}$ for $\delta > 0$ then $b = \Theta\left((\log(dr))^{1+\delta}\right)$ $(\geq r/k)$. The algorithm of Lu does not provide an approximation of the *optimum* T_{opt} of the CHC problem. Our main result of this section is the following

Theorem 4.1.2 *Given a constrained hypergraph coloring problem, for any* $\epsilon \in (0,1)$ *a coloring with at most* $\lceil (1+\epsilon)T_{opt} \rceil$ *colors can be found in polynomial time provided that* $b_i = \Omega\left(\frac{(1+\epsilon)\log \mathcal{D}}{\epsilon^2}\right)$ *for all* $i \in [s]$ *where* $\mathcal{D} \leq d\eta\tau = O((sT^*)^2)$.

Proof. Since CHC problem can be modelled as the above mentioned integer programming problem, we refer to Theorem 3.4.6 which tells us that one can find a coloring with at most $\lceil \alpha T_{opt} \rceil$ colors, $\alpha \in (1,2)$, but in doing so one ends up stretching the right hand side of the inequality constraints from b_i to $b_i\alpha^{-1}(1+\delta_i)$. Since we want to satisfy the original inequality constraints, we want

$$b_i\alpha^{-1}(1+\delta_i) \leq b_i$$

for all i, where

$$\delta_i = \Theta\left(\sqrt{\frac{\alpha \log(e\gamma(\mathcal{D}+1))}{b_i}}\right) .$$

This gives us

$$(\alpha-1)^2 = \Omega\left(\frac{\alpha \log(e\gamma(\mathcal{D}+1))}{b_i}\right) .$$

Thus, for any $\epsilon \in (0,1)$ and $\alpha = 1+\epsilon$ we get

$$b_i = \Omega\left(\frac{(1+\epsilon)\log(e\gamma(\mathcal{D}+1))}{\epsilon^2}\right) . \tag{4.1}$$

Since e and γ are constants, the proof is complete. □

For arbitrary b_i's, Srivastav and Stangier [26, 27] gave a polynomial time approximation algorithm which for every $\epsilon > 0$ builds a $\lceil (1+\epsilon)T_{opt} \rceil$-approximate coloring provided that for all i, $b_i \geq 3\epsilon^{-2}(1+\epsilon)\log(8sT^*)$ (T^* is the optimum value of the relaxed problem). On the other hand, our algorithm constructs a $\lceil (1+\epsilon)T_{opt} \rceil$-approximate coloring provided that the lower bound on b_i is $\Omega\left(\epsilon^{-2}(1+\epsilon)\log \mathcal{D}\right)$ for all edges $i \in [s]$. So, where is the difference? The difference lies in the dependency parameter $\mathcal{D}$. Recall that $\mathcal{D} \leq d\eta\tau$ where $d \leq s$, $\eta \leq \widehat{T}$ and $\tau \leq s\widehat{T}$. Now obviously $\mathcal{D} \leq s\widehat{T} = O(sT^*)$ but if $d << s$ (sparse hypergraph) or $\eta << \widehat{T}$ *i.e.*, we have very few variables to round, then, our lower bound on b_i is a definite improvement from that of [27]. In other words, the parameters d, η and τ provide more flexibility in estimating the maximum dependency among events.

For $b_i = 1\ \forall i$ the algorithm of de la Vega and Lueker [28] gives a $(1+\epsilon)s$ approximation of T_{opt}. We show in the following that for $b_i = O(1)\ \forall i$, the approximation ratio cannot be independent of s.

Theorem 4.1.3 *The CHC problem with n vertices and s edges has no polynomial time approximation algorithm with approximation ratio at most $s^{\frac{1}{2}-\epsilon}$, for any fixed $\epsilon > 0$, unless $NP \subseteq ZPP$.*

Proof. Let $G = (V, E)$ be a simple graph. The problem of properly coloring G with minimum colors can be viewed as a CHC problem because G can be viewed as a 2-uniform hypergraph with $|V|$ vertices and $|E|$ edges and a proper coloring can be obtained by putting $b_e = 1$ for all $e \in E$. Feige and Kilian [11] showed that if $NP \not\subseteq ZPP$ then it is impossible to approximate the chromatic number of a n vertex graph within a factor of $n^{1-\epsilon}$, for any fixed $\epsilon > 0$, in time polynomial in n. Therefore, the same applies for the CHC problem. Since $s \leq n^2$ in simple graphs, the proof of the theorem follows. □

4.2 Resource Constrained Scheduling

We are given the following

- a set $J = \{J_1, \ldots, J_n\}$ of independent jobs. Each job J_j needs one unit of time for its completion and it cannot be scheduled before its start time $t_j \in \mathbb{N}$.
- a set $R = \{R_1, \ldots, R_s\}$ of limited resources where every job needs one unit of at least one resource and at any point of time the available amount of each resource R_i is at most $b_i \in \mathbb{N}$.
- a set $P = \{P_1, \ldots, P_m\}$ of identical processors. Each job needs one processor.

The aim is to schedule these jobs subject to processor, resource and start time constraints such that the total schedule length ($\equiv$ time at which the last job is scheduled) is as small as possible.

Resource constrained scheduling problem can also be written as an integer program similar to the one given in Section 3.4. Indeed, let $A \in \{0,1\}^{(s+1)\times n}$ where

$$a_{i,j} = \begin{cases} 1 & \text{if job } J_j \text{ needs one unit of resource } R_i, \\ 0 & \text{otherwise,} \end{cases}$$

where $a_{s+1,j} = 1$ for all $j \in [n]$ and $b_{s+1} = m$ takes care of the processors by casting them as an extra constraint. Also, let the variables

$$x_{j,k} = \begin{cases} 1 & \text{if job } J_j \text{ is scheduled at time } k\ , \\ 0 & \text{otherwise,} \end{cases}$$

for all j and k. The resource constrained scheduling problem can now be written as: minimize $T = \max\{z \mid x_{j,z} > 0\ ,\ j = 1, 2, \ldots, n\}$, subject to

1. $Ax^{(k)} \leq b \quad \forall k = 1, 2, \ldots, T$,
2. $x_{j,k} = 0 \quad \forall j = 1, 2, \ldots, n$, $k < t_j$,
3. $\sum_{k=1}^{T} x_{j,k} = 1 \quad \forall j = 1, 2, \ldots, n$ and
4. $x_{j,k} \in \{0, 1\} \quad \forall j, k$.

Notice that the start time constraints ((2) above) do not make much difference. This problem is NP-hard in the strong sense, even if $t_j = 0$ for all $j \in [n]$, $s = 1$ and $m = 3$ [13]. For arbitrary b_i's, Srivastav and Stangier [26, 27] gave a polynomial time approximation algorithm for resource constrained scheduling problem with non-zero start times (problem class $P|res, \ldots 1, r_j, p_j = 1|T$). For every $\epsilon > 0$ their algorithm delivers a schedule of size at most $\lceil (1+\epsilon)T_{opt} \rceil$ provided that for all i $b_i \geq 3\epsilon^{-2}(1+\epsilon)\log(8sT^*)$ and the number of processors is at least $3\epsilon^{-2}(1+\epsilon)\log(8T^*)$. We again invoke Theorem 3.4.6 to get the following result.

Theorem 4.2.1 *Given a resource constrained scheduling problem with non-zero start times, for any $\epsilon \in (0, 1)$ a schedule of length at most $\lceil (1+\epsilon)T_{opt} \rceil$ can be found in polynomial time provided that $m = \Omega\left(\frac{(1+\epsilon)\log \mathcal{D}}{\epsilon^2}\right)$ and $b_i = \Omega\left(\frac{(1+\epsilon)\log \mathcal{D}}{\epsilon^2}\right)$ for all $i \in [s]$ where $\mathcal{D} \leq d\eta\tau = O((sT^*)^2)$.*

Proof. Similar to the proof of Theorem 4.1.2. □

The negative result of Theorem 4.1.3 also holds for resource constrained scheduling.

Chapter 5

Multi-dimensional Bin Packing

Multi-Dimensional Bin Packing (MDBP) or Vector Packing (VP) problem is the following: given n rational vectors $v_1, \ldots, v_n \in [0,1]^d$, pack these vectors in *minimum* number of bins, say m, such that $\| \sum_{i \in B_j} v_i \|_\infty \leq 1$ for each bin $j \in [m] = \{1, \ldots, m\}$. Here B_j is the set of indices of vectors assigned to bin j. In other words we want to pack these vectors in minimum possible number of bins such that in each bin, for each of the d components, the sum over all vectors in that bin is at most one. The classical bin packing problem is one dimensional version of the MDBP problem with numbers $v_1, \ldots, v_n \in (0,1]$ replacing the vectors.

The classical bin packing problem has been extensively studied and there exists a large pool of literature on it, most of which can be found here [15]. Its multi-dimensional generalization was introduced by Garey et al. [12], they gave a polynomial time $(d+1/3)$-approximation algorithm. MDBP problem is NP-hard and most of the results in this area are in the form of algorithms with asymptotic, worst-case performance ratio. De la Vega and Lueker [28] gave an improved linear time algorithm which gives a $(d+\epsilon)$-approximate solution for any fixed $\epsilon > 0$. A variant of MDBP problem, namely Resource Constrained Scheduling problem, was also studied in [27]. Recently Chekuri and Khanna [7] further improved the long standing $(d+\epsilon)$ bound, they gave a polynomial time algorithm that, for any fixed $\epsilon > 0$, delivers a $(1 + \epsilon d + O(\log \epsilon^{-1}))$-approximate solution.

We modify the algorithm of [7] to obtain better approximation for a class of MDBP problem instances. For some of these instances our algorithm beats even the lower bound of $\sqrt{d}$ [7]. We first cast the MDBP problem as an IP problem in Section 5.1. Subsequently in Section 5.2 we present our algorithm which is analysed in Section 5.3. All logarithms are base e logarithms throughout this chapter.

5.1 Preliminaries

MDBP problem can be formulated as an integer programming problem with 0/1-variables. For $i \in [n]$, $j \in [m]$ let

$$x_{ij} = \begin{cases} 1 & \text{if vector } v_i \text{ is assigned to bin } j, \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

Our aim is to minimize m such that

(a) $\sum_{i=1}^{n} v_i^k x_{ij} \leq 1 \quad \forall k \in [d],\ j \in [m]$,

(b) $\sum_{j=1}^{m} x_{ij} = 1 \quad \forall i \in [n] \quad$ and

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

Here the inequality constraints in (a) are nothing but the packing constraints $\| \sum_{i \in B_j} v_i \|_\infty \leq 1$ and the equality constraints (b) combined with the integrality constraints (c) make sure that each vector is assigned to exactly one bin. Let *opt* be the value of the optimal solution of the IP described above. The LP-relaxation of the integer program described above is obtained by replacing the integrality constraint (c) by the constraint $x_{ij} \in [0,1]$ for all i,j. Let m^* be the value of the optimum solution of the LP-relaxation.

Note that the integer program described above is similar to the one described in Section 3.4, the vector v_i^k components being equivalent to the matrix entries $a_{i,j}$ of the former integer program. In fact, one may wonder whether the they are any different or not. Recall that the inequality constraints of the integer program of Section 3.4 are of the form

$$\sum_{j=1}^{n} a_{i,j} x_{j,k} \leq b_i \quad \forall\, i,k\ ,$$

where $b_i > 1$. Dividing by b_i on both sides gives us the required form but then the matrix entries are bounded from above by $\tilde{a}/\tilde{b}$, where

$$\tilde{a} = \max_{i,j}\ a_{i,j} \qquad \text{and} \qquad \tilde{b} = \min_{i}\ b_i\ .$$

So, not all instances of MDBP are covered by the former integer program.

To describe our results exactly we need to define a parameter first. Let

$$\nu = \min_{\substack{k \in [d],\\ i \in [n]}}\ v_i^k\ . \tag{5.1}$$

In the next section we give a polynomial time randomized algorithm that, for any fixed $\epsilon > \nu$, achieves a $(1 + \epsilon q + O(\log \epsilon^{-1}))$-approximation, where

$$q = \min\{d, \frac{1}{\nu}\Theta\left(\frac{\log dm^*}{\log\log dm^*}\right)\}\ . \tag{5.2}$$

5.2 The Algorithm

We slightly modify the algorithm given by Chekuri and Khanna [7]. The modified algorithm has the same performance ratio as that of [7] in most of the cases but in some cases it delivers better results.

Algorithm : VecPack

1. Solve the LP-relaxation of the IP given in Section 5.1 to obtain the optimal fractional solution.

2. If $d\nu > \Theta\left(\frac{\log dm^*}{\log\log dm^*}\right)$ then go to step 3 else go to step 4.

3.

   3.1. Randomly round the fractional solution to an integral (possibly infeasible) solution.

   3.2. Remove the vectors causing infeasibility and mark them as unassigned.

4. Mark all fractionally assigned vectors as unassigned.

5. Greedily assign the unassigned vectors to new bins.

Before proceeding further let us clarify step 5. In step 5, the greedy method used to pack unassigned vectors in new bins is the same as the greedy set cover method. To be exact, let us take a diversion and describe the set cover problem with its greedy algorithm.

Definition 5.2.1 *(Set Cover) Given a ground set I of m elements, a set $\mathcal{S} = \{S_1, \ldots, S_n\}$ with $S_j \subset I$ for all $j \in [n]$ and the corresponding weights of elements of $\mathcal{S}$, $W = \{w_1, \ldots, w_n\}$. A set $S = \{S_{j_1}, \ldots, S_{j_c}\} \subseteq \mathcal{S}$ is called a cover of ground set I with weight $w_S = \sum_{k=1}^{c} w_{j_k}$ if $\cup_{k=1}^{c} S_{j_k} = I$.*

So, the set cover problem is to find a cover $S \subseteq \mathcal{S}$ of ground set I such that w_S is minimum. We will use the set cover problem with unit weights, *i.e.*, $w_j = 1$ for all $j \in [n]$. The greedy algorithm for set cover is simple: at each step select the largest set S_j still in $\mathcal{S}$. The performance ratio of this greedy algorithm is H_s [19, 8], where

$$s = \max_{j \in [n]} |S_j| \qquad \text{and} \qquad H_s = \sum_{i=1}^{s} \frac{1}{i} .$$

5.3 Analysis

In this section we present a step by step analysis of VecPack in the form of the following theorem.

Theorem 5.3.1 *([2]) With high probability* VecPack *delivers a* $(1+\epsilon q+O(\log \epsilon^{-1}))$*-approximate solution in polynomial time, where* $q = \min\{d, \frac{1}{\nu}\Theta\left(\frac{\log dm^*}{\log\log dm^*}\right)\}$ *and* $\epsilon > \nu$ *is a fixed number.*

Proof.
Step 1. LP-relaxation is nothing but the same set of constraints as in Section 5.1 except that now we allow $x_{ij} \in [0,1]$. Since we know that the number of bins required can be at most n, we use binary search to pinpoint m^* and obtain the optimal fractional solution by solving at most $\log n$ linear programs in polynomial time [18]. Thus after step 1 we know the value of m^* and a corresponding feasible solution $\{x^*_{ij} \mid i \in [n]\ ,\ j \in [m]\}$. Let us assume, w.l.o.g., that $\{x^*_{ij}\}$ is a basic feasible solution.

Step 3. In step 2 we verify whether

$$d\nu > \Theta\left(\frac{\log dm^*}{\log\log dm^*}\right) = l\ . \tag{5.3}$$

If $d\nu > l$ we perform randomized rounding *i.e* for each vector v_i, $i \in [n]$, which is not completely assigned to a bin we assign it to bin j with probability x^*_{ij}. Note that this procedure is similar to the rounding procedure described in Section 3.4.2. Let

$$y_{ij} = \begin{cases} 1 & \text{with probability } x^*_{ij}, \\ 0 & \text{with probability } 1 - x^*_{ij} \end{cases}$$

be the random variables obtained by rounding $\{x^*_{ij}\}$. Define dm^* events

$$\xi_{jk} \equiv \text{“}\ \sum_{i=1}^{n} v_i^k y_{ij} > 1 + \delta\ \text{”}\ , \tag{5.4}$$

where $\delta > 0$. Obviously the expected value

$$\mathbb{E}[\sum_{i=1}^{n} v_i^k y_{ij}] \leq 1$$

for all j, k. Using Lemma 2.4.3 and Corollary 2.4.5 we can make $\mathbb{P}[\xi_{jk}]$ $(\leq p)$ as small as required. So, for $p = \left(\frac{1}{dm^*}\right)^2$ we get

$$\delta = \Theta\left(\frac{\log dm^*}{\log\log dm^*}\right)\ . \tag{5.5}$$

Since there are dm^* events, the probability that at least one of them occurs is at most $1/dm^*$ and hence

$$\mathbb{P}[\cap_{j=1}^{m^*} \cap_{k=1}^{d} \xi_{jk}^c] \geq 1 - \frac{1}{dm^*} . \tag{5.6}$$

It means that with high probability randomized rounding yields an integral solution using m^* bins but the size of the bins is stretched from 1 to at most $1+\delta$. The important fact is: in each bin at most $\lfloor \delta/\nu \rfloor$ vectors are responsible for stretching its size from 1 to at most $1+\delta$. Thus, after step 3.2 of our algorithm VecPack we have at most $\lfloor \delta/\nu \rfloor m^*$ unassigned vectors.

Step 4. On the other hand if we directly jump to step 4 in our algorithm (*i.e*, $d\nu \leq l$) then by the basic feasibility of our optimal fractional solution $\{x_{ij}^*\}$ we have at most dm^* unassigned vectors.

Step 5. In step 5, just like in [7], we use the greedy set cover algorithm to pack unassigned vectors in new bins. So suppose we have qm^* unassigned vectors, where $q = \min\{d, \frac{1}{\nu}\Theta\left(\frac{\log dm^*}{\log\log dm^*}\right)\}$. At each step we find the largest possible set of vectors with up to $s = \lceil 1/\epsilon \rceil$, $\epsilon > \nu$ fixed, vectors which can be packed together in a bin and assign them to a new bin. After taking care of all sets with exactly s vectors we end up using at most (qm^*/s) bins. Now we can pack at most $(s-1)$ of the remaining unassigned vectors in a bin. Hence, by listing all possible sets containing at most $(s-1)$ vectors and applying the greedy step on this list we get a packing in at most $(H_{s-1} \cdot opt)$ bins (see [19, 8]). Here

$$H_{s-1} = \sum_{i=1}^{s-1} \frac{1}{i} = O(\log(s-1)) .$$

Thus, we manage to pack all vectors in at most

$$m^* + \frac{qm^*}{s} + H_{s-1} \cdot opt \leq (1 + \epsilon q + O(\log \epsilon^{-1})) \cdot opt \tag{5.7}$$

bins. □

Remarks

1. The randomized rounding step, step 3.1, can be derandomized to obtain a deterministic version of Theorem 5.3.1. Indeed, suppose we have assigned vectors $v_1, \ldots, v_{i-1}$ and v_i is to be randomly assigned to one of the m^* bins. At this point the probability that one of the events ξ_{jk} occurs is bounded by

$$PE_{i-1} = \sum_{j \in [m],\, k \in [d]} \mathbb{P}[\xi_{jk} \mid v_1, \ldots, v_{i-1} \text{have been assigned}] .$$

This is the pessimistic estimator which can be used to derandomize Theorem 5.3.1.

2. Our algorithm improves on the algorithm of [7] if $d\nu > \Theta\left(\frac{\log dm^*}{\log\log dm^*}\right)$ (clearly $\nu > 0$). This gives us the following upper bound on m^*

$$m^* = O\left(\frac{\exp(d\nu \log d\nu)}{d}\right) . \tag{5.8}$$

Thus, for MDBP problem instances with bounded optimum solution

$$opt = O\left(\frac{e^{(d\nu \log d\nu)}}{d}\right) , \tag{5.9}$$

our algorithm gives better results. Furthermore, if the given instance has at most a constant number of vectors with zero components, *i.e* $\nu = 0$, then these vectors can be pre-packed in $O(1)$ bins. This yields a new problem instance with $\nu > 0$ which can be given as input to VecPack.

Chapter 6

Conclusion

In general, we saw how seemingly difficult problems can be tackled using probabilistic methods, particularly the Lovász Local Lemma. This lemma started its journey with an application to hypergraph coloring problem [9] and has come a long way with applications to solve integer programming problems. We saw such applications in Chapter 3, Section 3.4 and Chapter 4. The main ideas of the methods used by us are:

- convert the solution space into a probability space, like the set of all possible 2-colorings in Section 3.2 or the set of all possible solutions of the integer program in Section 3.4, by introducing a random process. For instance the random process in Section 3.2 was to independently color each vertex of the hypergraph red or blue with equal probability. In the remaining applications and Section 3.4 the random process was to randomly round the binary variables to 0 or 1.

- show that the desired structure, like the desired solutions of the integer programs in Section 3.4 and Chapters 4 & 5, exists in this probability space. Often we used the Lovász Local Lemma for this purpose.

- use an efficient method to find the desired structure in the huge probability space of all possible solutions. We used the randomized version of the algorithmic Lovász Local Lemma in Chapter 3 and a simple randomized algorithm in Chapter 5.

- efficiently derandomize the randomized algorithm (see Theorem 3.2.7 and Lemma 3.4.5) to obtain deterministic polynomial time algorithms.

In Chapter 2 we introduced the tools which we later used in the subsequent chapters. These tools (pessimistic estimators, large deviation inequalities, etc.) are simple yet powerful because they help in obtaining good results for hard problems.

In Chapter 3, Section 3.2 we introduced the algorithmic version of Lovász Local Lemma in the form of Beck's application to property B problem [6]. This was further put into perspective by a general (not problem-specefic) analysis of the algorithmic version in Section 3.3. We then modified these ideas and the ideas of [20, 17] to obtain a nearly feasible and near-optimal solution of an interesting integer program (minimax integer program with fixed right hand side) in Section 3.4. The algorithm we gave for this purpose is a polynomial time algorithm (Theorem 3.4.6).

In Chapter 4 we presented two applications of Theorem 3.4.6 namely constrained hypergraph coloring and resource constrained scheduling. The main results of Sections 4.1 & 4.2 are that if the right hand side (b_is) of the inequality constraints in the integer programming formulations of the respective problems are large enough, then we can obtain near-optimal solutions in polynomial time [1]. This improved the previous best results of [27].

In Chapter 5 we dealt with the multi-dimensional bin packing problem. Here we showed that the previous best algorithm (in terms of performance ratio) can be modified so that the performance ratio improves, sometimes substantially, for a class of problem instances [2].

6.1 Open Questions

Many interesting questions arise from the material presented in previous chapters. In Section 3.3 we saw that in general we cannot get rid of the weaker condition $pd^{\Theta(1)} < 1$ in the algorithmic version of Lovász Local Lemma. But, can the exponent be reduced in general or does it depend on the problem at hand (we used $p\mathcal{D}^4 < 1$ in Section 3.4.3)?

In Chapter 4, on one hand we have a $(1+\epsilon)s$ approximation of de la Vega and Lueker [28] when $b_i = 1\ \forall i$ and on the other hand we have our $(1+\epsilon)$ approximation for $b_i = \Omega((1+\epsilon)\epsilon^{-2}\log\mathcal{D})$ (Theorem 4.1.2). But, we don't know the approximation quality for all values of b_i. So two interesting questions come to mind.

1. How exactly does the approximation ratio behave when $b_i \in (1, \log\mathcal{D}]\ \forall i$? We believe that the approximation ratio depends on b_is, but how exactly are they related is not known.

2. Is it possible to get better approximation bounds for the number of colors without loosing much on b_i's ?

In Chapter 5 we showed that we get better results if ν is appropriately bounded away from zero and ν comes into play because we take out some vectors from each bin. But it does not shed any light on the important problem of closing in on the lower bound $\sqrt{d}$.

Bibliography

[1] N. Ahuja and A. Srivastav. *On constrained hypergraph coloring and scheduling.* In proceedings of the fifth international workshop, APPROX 2002, LNCS 2462, 14 - 25.

[2] N. Ahuja and A. Srivastav. *Better multi-dimesional bin packing in special cases.* Preprint(2002).

[3] N. Alon. *The linear arboricity of graphs.* Israel Journal of Mathematics, 62(1988), 311 - 325.

[4] N. Alon. *A parallel algorithmic version of the local lemma.* Random Structures and Algorithms, 2(1991), 367 - 378.

[5] N. Alon and J. Spencer. *The Probabilistic Method, second edition.* Wiley-Interscience, John Wiley & Sons, New York, 2000.

[6] J. Beck. *An algorithmic approach to the Lovász Local Lemma.* Random Structures and Algorithms, 2(1991), 343 - 365.

[7] C. Chekuri and S. Khanna. *On multi-dimensional packing problems.* In Proc. of the 10th annual ACM-SIAM SODA (1999), 185 - 194.

[8] V. Chvátal. *A greedy heuristic for the set-covering problem.* Math. of Oper. Res., 4(1979), 233 - 235.

[9] P. Erdös and L. Lovász. *Problems and results on 3-chromatic hypergraphs and some related questions in infinite and finite sets.* A. Hajnal et al (eds), Colloq. Math. Soc. J. Bolyai 11, North Holland, Amsterdam (1975), 609 - 627.

[10] P. Erdös and J. L. Selfridge. *On a combinatorial game.* Journal of Combinatorial Theory, Series A 14 (1973), 298 - 301.

[11] U. Feige and J. Kilian. *Zero knowledge and the chromatic number.* Journal of Computer and System Sciences, 57(1998), 187 - 199.

[12] M.R. Garey, R.L. Graham, D.S. Johnson and A.C.-C. Yao. *Resource constrained scheduling as generalized bin packing.* JCT Ser. A, 21(1976), 257 - 298.

[13] M.R. Garey and D.S. Johnson. *Computers and Intractability.* W.H. Freeman and Company, New York, 1979.

[14] M. Grötschel, L. Lovász and A. Schrijver. *Geometric algorithms and combinatorial optimization.* Springer-Verlag, 1988.

[15] D. S. Hochbaum (Ed). *Approximation Algorithms for NP-Hard Problems*, PWS Publishing Company, 1996.

[16] J. Kahn. *Asymptotically good list-colorings.* Journal of Combinatorial Theory (A), 73(1996), 1 - 59.

[17] F. T. Leighton, Chi-Jen Lu, S. B. Rao and A. Srinivasan. *New algorithmic aspects of the local lemma with applications to routing and partitioning.* SIAM Journal on Computing, 31(2001), 626 - 641.

[18] J. K. Lenstra, D.B. Shmoys and E. Tardos. *Approximation algorithms for scheduling unrelated parallel machines.* Math. Programming, 46(1990), 259 - 271.

[19] L. Lovász. *On the ratio of optimal integral and fractional covers.* Discrete Math., 13(1975), 383 - 390.

[20] Chi-Jen Lu. *Deterministic hypergraph coloring and its applications.* Proceedings of the 2nd International Workshop on Randomization and Approximation Techniques in Computer Science (1998), 35 - 46.

[21] M. Molloy and B. Reed. *Further algorithmic aspects of the Lovász Local Lemma.* Proc. 30th Annual ACM Symposium on Theory of Computing (1998), 524 - 529.

[22] P. Raghavan and C.D Thompson. *Randomized rounding : a technique for provably good algorithms and algorithmic proofs.* Combinatorica, 7(4)(1987), 365 - 374.

[23] P. Raghavan. *Probabilistic construction of deterministic algorithms : approximating packing integer programs.* Journal of Computer System Sciences 37 (1988), 130 - 143.

[24] J. P. Schmidt, A. Siegel and A. Srinivasan. *Chernoff-Hoeffding bounds for applications with limited independence.* SIAM Journal of Discrete Mathematics, 8(1995), 223 - 250.

[25] A. Srinivasan. *An extension of the Lovász Local Lemma and its applications to integer programming.* ACM-SIAM Symposium on Discrete Algorithms (1996), 6 - 15.

[26] A. Srivastav and P. Stangier. *Algorithmic Chernoff-Hoeffding inequalities in integer programming.* Random Structures and Algorithms, 8(1)(1996), 27 - 58.

[27] A. Srivastav and P. Stangier. *Tight approximations for resource constrained scheduling and bin packing.* Discrete Applied Math, 79(1997), 223 - 245.

[28] W.F. de la Vega and C.S Lueker. *Bin packing can be solved within* $(1+\epsilon)$ *in linear time.* Combinatorica, 1(1981), 349 - 355.