An Algorithmic Approach to the General Lovász Local Lemma with Applications to Scheduling and Satisfiability Problems *

(Technical Report)

Artur Czumaj and Christian Scheideler
Dept. of Mathematics & Computer Science
and Heinz Nixdorf Institute
Paderborn University, 33095 Paderborn, Germany
email: {artur,chrsch}@uni-paderborn.de

Paderborn, 24.01.2000

Abstract

The Lovász Local Lemma (LLL) is a powerful tool that is increasingly playing a valuable role in computer science. It has led to solutions for numerous problems in many different areas, reaching from problems in pure combinatorics to problems in routing, scheduling and approximation theory. However, since the original lemma is non-constructive, many of these solutions were first purely existential. A breakthrough result by Beck [8] and its generalizations (Alon [1], Molloy & Reed [24]) have led to polynomial time algorithms for many of these problems. However, these methods can only be applied to a simple, symmetric form of the LLL. In this paper we provide a novel approach to design polynomial-time algorithms for problems that require the LLL in its *general* form. We apply our techniques to find good approximate solutions to a large class of NP-hard problems called *minmax integer programs* (MIPs). Our method finds approximate solutions that are — especially for problems of non-uniform character — significantly better than all methods presented before. To demonstrate the applicability of our method, we apply it to transform important results in the area of job shop scheduling that have so far been only existential (due to the fact that the general LLL was used) into algorithms that find the predicted solutions (with only a small loss) in polynomial time. Furthermore, we demonstrate how our results can be used to solve satisfiability problems.

*Research partially supported by DFG-Sonderforschungsbereich 376 “Massive Parallelität: Algorithmen, Entwurfsmethoden, Anwendungen.”

1 Introduction

The probabilistic method is used to prove the existence of objects with desirable properties by showing that a randomly chosen object from an appropriate probability distribution has the desired properties with positive probability. In most applications, this probability is not only positive but is actually high and frequently tends to 1 as the parameters of the problem tend to infinity. In such cases, the proof usually supplies an efficient randomized algorithm for producing a structure of the desired type.

There are, however, certain examples, where one can prove the existence of the required combinatorial structure by probabilistic arguments that deal with rare events; events that hold with positive probability which is exponentially small in the size of the input. This happens often when using the LLL. This lemma (in its general form) is defined as follows.

Lemma 1.1 (Lovász [11]) *Let $A_1, \dots, A_n$ be a set of "bad" events in an arbitrary probability space and let G be a dependency graph for the events $A_1, \dots, A_n$. (That is, A_i is independent of any subset of events A_j with $(i,j) \notin G$.) Assume that there exist $x_i \in [0,1)$ for all $1 \le i \le n$ with*

$$\Pr[A_i] \le x_i \prod_{(i,j) \in G} (1 - x_j)$$

for all i. Then with positive probability no bad event occurs.

In its symmetric form, the LLL is defined as follows.

Lemma 1.2 (Symmetric LLL) *Let $A_1, \dots, A_n$ be a set of "bad" events with $\Pr[A_i] \le p$ for all i. If each A_i is mutually independent of all but at most d of the other events A_j and $\mathrm{e}p(d+1) \le 1$, then with positive probability no bad event occurs.*

Many applications of the LLL can be found in the literature (see, e. g., [2, 4, 5, 9, 11, 12, 14, 18, 20, 21, 22, 26, 27, 29, 30]). To turn proofs using the Lovász Local Lemma into efficient algorithms, even random ones, proved to be difficult for many of these applications. In a breakthrough paper [8], Beck presented a method of converting some applications of the Lovász Local Lemma into polynomial-time algorithms (with some sacrifices made with regards to the constants in the original application). Alon [1] provided a parallel variant of the algorithm and simplified the arguments used. His method was further generalized by Molloy and Reed [24] to yield efficient algorithms for a number of applications of a simple, symmetric form of the Lovász Local Lemma.

There have been only very few cases for which polynomial time algorithms for applications of the general Lovász Local Lemma have been found without requiring a reduction to the symmetric LLL. Molloy and Reed [24] found methods for the problems of β-frugal coloring and acyclic edge coloring that could possibly be applied to problems that require the general Lovász Local Lemma, but as it was pointed out by the authors, they may require to prove some (possibly difficult) concentration-like properties for each problem under consideration. Recently, the authors [10] were able to design and analyze a polynomial-time algorithm for the problem of 2-coloring non-uniform hypergraphs and related coloring problems.

In this paper, we will present a constructive form of the general LLL that cannot be proved (without significant modifications) by the method in [10] for several technical reasons described in Section 2. We will demonstrate in Section 3 how to use this result to construct efficient approximation algorithms for so-called *minmax integer programs* (MIPs). For every $k \in \mathbb{N}$, let $[k]$ represent the set $\{1, \dots, k\}$.

Definition 1.3 *A MIP has variables $\{x_{i,j} \,:\, i \in [n],\ j \in [\ell_i]\}$, for some integers ℓ_i. Let $N = \sum_{i \in [n]} \ell_i$ and let $\boldsymbol{x}$ denote the N-dimensional vector of the variables $x_{i,j}$. A MIP seeks to minimize a real Y subject to:*

(1) a system of linear inequalities $\boldsymbol{Ax} \leq \boldsymbol{y}$, where $\boldsymbol{A} \in [0,1]^{m\times N}$ and $\boldsymbol{y}$ is the m-dimensional vector with the variable Y in each component,
(2) for all $i \in [n]$: $\sum_{j\in[\ell_i]} x_{i,j} = 1$, and
(3) for all i and j: $x_{i,j} \in \{0,1\}$.

MIPs represent a general framework for choice problems (see properties (2) and (3)). Many applications in the area of routing and scheduling can be formulated as MIPs (for several examples, see [22]). A common strategy to find good approximate solutions to MIPs is to first solve its linear relaxation and then to use randomized rounding. This approach was pioneered by Raghavan and Thompson [28] and was later extended and refined for MIPs by Srinivasan [30], Lu [23] and Leighton, Rao and Srinivasan [22]. The latter papers apply the symmetric form of the LLL to construct good rounding strategies. Leighton et al. [22] give as a high-level application of MIPs the *hypergraph partitioning problem* [15]:

We are given a set system $H = (V, E)$, where $V = [n]$ and $E = \{S_1, \ldots, S_M\} \subseteq \mathcal{P}(V)$. Given a positive integer ℓ, the problem is to partition V into ℓ parts, so that the sets are split well: we want a $\chi : V \to [\ell]$ that minimizes $Y = \max_{j\in[M],k\in[\ell]} |\{i \in S_j : \ \chi(i) = k\}|$.

The cost function above has the drawback that it only seeks to minimize the *absolute* split size. However, there are applications where it would be much more desirable to minimize the *relative* split size, i. e., to find a function χ that minimizes $Y = \max_{j\in[M],k\in[\ell]} |\{i \in S_j : \ \chi(i) = k\}|/|S_j|$. Problems of this kind can be found, for instance, in a paper by Feige and Scheideler [14]. Their solution (which heavily uses the general LLL) allowed to prove upper bounds on the makespan of job shop schedules that significantly improved the previously best results (see Section 4.1). Solving these problems with the techniques given in [22, 23] gives an approximation ratio that is as large as $\Theta(\frac{\log n}{\log\log n})$ (n is the problem size), whereas an approximation ratio of partly as low as $1 + o(1)$ is required for the proofs in [14] to be constructivized. Our method presented in this paper will achieve this goal.

1.1 New results

Our main technical contribution is a novel approach to design polynomial-time algorithms for problems that require the *general* LLL. We will apply this approach to find good approximation algorithms for MIPs:

Consider any MIP. Our strategy is to start with an optimal solution $\{x^*_{i,j} : \ i \in [n], \ j \in [\ell_i]\}$ to the LP relaxation of the MIP (i. e., the integrality constraints in Definition 1.3 (3) are removed). The resulting LP optimum $\boldsymbol{y}^*$ is clearly a lower bound for $\boldsymbol{y}$, the optimum of the (integral form of the) MIP. We will present a randomized rounding algorithm that exploits the dependencies among the rows of matrix $\boldsymbol{A}$ to find a good approximate solution for $\boldsymbol{y}$. These dependencies are defined as follows.

Definition 1.4 *Given a MIP instance I as defined above, let the* dependency graph *G_I of I consist of the node set $[m]$ and an edge set that contains edge (r, s) if and only if there are i, j, j' so that $a_{r,(i,j)} > 0$ and $a_{s,(i,j')} > 0$, and $r \neq s$ or $j \neq j'$.*

Note that G_I may contain self-loops. This simplifies our proofs. However, it would also be possible to avoid them. Based on our LLL techniques, we will prove the following main result.

Theorem 1.5 *Given a MIP instance I, let $\boldsymbol{x}^*$ be its optimal fractional solution and $\boldsymbol{y}^*$ be the vector with $y^*_r = (\boldsymbol{Ax}^*)_r$ for all r. Consider the random experiment of setting $x_{i,j}$ to 1 and all other $x_{i,j'}$'s to 0 with probability $x^*_{i,j}$. For any vector $\boldsymbol{\alpha} \in \mathbb{R}^m_+$, let*

$$p_r = \mathrm{e}^{-\min[\alpha_r,\, \alpha_r^2] y'_r/3}$$

for all $r \in [m]$, where $y'_r = y^*_r / (\max_c a_{r,c})$. ($p_r$ is a Hoeffding bound for $\Pr[(\boldsymbol{Ax})_r \geq (1+\alpha_r) y^*_r]$.) Let $0 < \epsilon < 1$ be a constant that is chosen such that $\alpha_r \geq \max[(y'_r)^{-1},\ (y'_r)^{-(1-\epsilon)/2}]$ for every r. Furthermore, let $q_r = \mathrm{e}^{-\ln^\epsilon p_r^{-1}}$ and let $z_r = q_r^\delta$ be at most $1/3$, where δ is a sufficiently small positive constant. If it holds that

$$q_r \leq z_r \prod_{(r,s) \in G_I} (1 - z_s)^{2/\epsilon^2} \tag{1}$$

for all r, then there is an algorithm that finds a vector $\boldsymbol{x}$ in polynomial time so that $(\boldsymbol{Ax})_r \leq (1 + O(\alpha_r)) y^*_r$ for all r.

The main feature of this result is its great flexibility: It allows to exploit local dependencies among the rows to achieve approximate solutions that allow to distinguish between rows that require a very good or just a rough approximation. This is important, since MIPs may be — especially in real-world applications — composed of different classes of restrictions that have different dependency structures and that require different approximation ratios. In [22, 23] only the maximum dependency of a row on other rows can be used.

As we will see in Section 4, the property of being able to exploit local dependencies is vital to constructivize the job shop scheduling results presented in [14]. We also show how to apply our method to obtain the first approximation algorithm for MAX SAT problems that is able to exploit local dependencies.

2 The Local Lemma Algorithm

We start with presenting a general approach to design polynomial-time algorithms for problems that require the general LLL.

Let V denote a set consisting of n independent random variables called *trials*. For each trial $t \in V$, let Ω_t denote the set of its possible outcomes. Let $\mathcal{A}$ be a set of m "bad" events, where each event $A \in \mathcal{A}$ is determined by the outcome of the trials in $T_A \subseteq V$. Let $E = \{T_A \mid A \in \mathcal{A}\}$. Two events $A_1, A_2 \in \mathcal{A}$ are said to be *neighbors* if $T_{A_1} \cap T_{A_2} \neq \emptyset$. For every $A \in \mathcal{A}$, let $N(A)$ denote the set of all neighbors of A. Clearly, event A is mutually independent of every subset of events A' with $A' \not\in N(A)$. We assume that for every event A and every set $S \subseteq T_A$, the event A *restricted to* S (written as $A|_S$) is well-defined and fulfills the condition that $A|_S$ only depends on S and is independent of any other set of trials outside of S, even if they belong to the original event A. Let $\Pr[A|_S]$ denote the probability that $A|_S$ is true under the assumption that an outcome is chosen independently at random for each trial in S. Suppose that the following natural assumptions are valid:

- For every trial, a random outcome can be generated in polynomial time, and
- it can be decided in polynomial time for every event A and set $S \subseteq T_A$ whether $A|_S$ is true.

Then the following theorem holds.

Theorem 2.1 *There is a constant $\Delta > 0$ so that for every $0 < \delta \leq \Delta$ the following holds:*

Let $A_1, \ldots, A_m$ be a set of "bad" events over a set V of independent random trials, $|V| = n$. Every trial $t \in V$ has a set of outcomes of size at most polylogarithmic in $n + m$. Let $p_i = \mathrm{e}^{-|T_{A_i}|^\epsilon}$ for all $i \in [m]$, where $0 < \epsilon \leq 1$ is a constant away from 0. Suppose that for all $S \subseteq T_{A_i}$ it holds that

- *if $|S| > |T_{A_i}|^\epsilon$ then $\Pr[A_i|_S] \leq p_i$, and*
- *if $|S| \leq |T_{A_i}|^\epsilon$ then $\Pr[A_i|_S] = 0$.*

Furthermore, assume that for $q_i = \mathrm{e}^{-\ln^\epsilon p_i^{-1}}$ and $x_i = q_i^\delta$ it holds that $x_i \leq 1/\mathrm{e}$ and

$$q_i \leq x_i \prod_{A_j \in N(A_i)} (1 - x_j)$$

for all i. Then there is a randomized algorithm that finds outcomes for the random trials in polynomial time (in $n+m$), w.h.p., such that for every event A, the set T_A can be partitioned into at most some constant number of subsets $S_1, \ldots, S_k$ so that $A|_{S_j}$ is false for all $j \in [k]$.

We first describe the algorithm underlying Theorem 2.1. Its analysis will be given in Section 2.2.

2.1 Description of the Algorithm

Consider any set of events that fulfills the conditions in Theorem 2.1. Our algorithm consists of four steps.

Step 1: Choose a random outcome for every trial in V.

Before we describe Step 2, we first introduce some notation and provide the ideas behind that step. Let the events in Theorem 2.1 be called *basic events*, and let any event defined by a basic event restricted to a subset of its trials be called a *reduced event*. As in Theorem 2.1, we assume the basic events to be numbered consecutively from A_1 to A_m. Given an event B representing an event A_i or any reduced event of A_i, we denote its *index* by i. A basic event A is called *bad* if its trial set cannot be decomposed into a constant number of subsets S such that $A|_S$ is false. Clearly, after Step 1 there might be many basic events left that are bad. In this case we have to redo certain trials covered by these events. The aim of Step 2.1 is to find a partition of the bad events into trial-disjoint groups. The key feature of our partitioning procedure is that we do not consider all the trials covered by the bad events. Instead, in the course of the algorithm we shall sometimes partition the basic events into several reduced events that are assigned to different groups.

Step 2: Perform the following two substeps Step 2.1 and Step 2.2.

Step 2.1:
set $R = V$ // R denotes the set of remaining trials
for $i = 1$ to m do:
  if $A_i|_{T_{A_i} \cap R}$ is true **then**
    call **Build_1-Component**($A_i|_{T_{A_i} \cap R}$)

Algorithm Build_1-Component(B)**:**
set $i = 0$ and $E_0 = \{B\}$
set $R = R \setminus T_B$
repeat
  set $E_{i+1} = \emptyset$
  set $C_{i+1} = N(E_i)$ // $C_{i+1} =$ all basic events that are neighbors of E_i
  for all events $A_j \in C_{i+1}$ in increasing j do:
    set $S = T_{A_j} \cap R$ // S: trials not selected yet
    if $A_j|_S$ is true **then** // if true, this implies $|S| > |T_{A_j}|^{\epsilon}$
      set $R = R \setminus S$ and add $A_j|_S$ to E_{i+1}
  set $i = i + 1$
until $E_i = \emptyset$

Consider some fixed set $E^* = \bigcup_{i \geq 0} E_i$ of (possibly reduced) events built in the algorithm Build_1-Component. A basic event A is said to *participate* in it if more than $|T_A|^{\epsilon}$ of its trials are covered by E^*. (This clearly holds for those events that have a reduced event in E^*.) The set of all participating events restricted to the trials covered by E^* is called a *1-component*. For every 1-component C, let $B_C = E^*$ denote its set of *core events*.

Core events play an important role in the reselection process because, as we shall show, it is sufficient to reselect only the trials covered by the core events in order to find a good selection for all (possibly reduced) events in C.

After performing Step 2.1, one might think that the 1-components can be solved independently. However, it could happen that some basic event is partitioned into too many reduced events that belong to different components. Theorem 2.1 requires for every event A to provide a decomposition of T_A into a constant number of sets S such that $A|_S$ is false. Therefore, without a coordination among these components, Theorem 2.1 might not hold at the end. Step 2.2 takes care of these events by combining components into one component if necessary. As we will see, the trials covered by the components constructed in Step 2.2 can afterwards be reselected independently so that now Theorem 2.1 is ensured to hold at the end.

Define $\mathcal{C}$ to be the set of all 1-components. Recall that $\mathcal{A}$ denotes the set of all basic events. The second substep works as follows.

Step 2.2:
set $S = \mathcal{A}$ // S denotes the set of remaining events
set $R = \mathcal{C}$ // R denotes the set of remaining 1-components
for all not yet taken 1-components C in R in
  increasing index of their initial events do:
 call **Build_2-Component**(C)

Algorithm Build_2-Component(C)**:**
set $i = 0$ and $E_0 = B_C$
repeat
 set $E_{i+1} = \emptyset$
 set $D = N(E_i) \cap S$ // D = all neighboring basic events not yet considered
 for all events $A_j \in D$ in increasing j do: // check nearby 1-components
  set T to the set of trials in A_j covered by outside 1-components
  if $|T| > |T_{A_j}|^\epsilon$ **then** $(*)$
   for all 1-components C' in R that overlap with T_{A_j} do:
    set $E_{i+1} = E_{i+1} \cup B_{C'}$
    remove C' from R
  remove A_j from S
 set $i = i + 1$
until $E_i = \emptyset$

In the following, a basic event A is called *dangerous* if condition $(*)$ above holds for it. Let the union of the 1-components built in the algorithm Build_2-Component be called a *2-component*. For every 2-component C, let B_C denote the union of the sets $B_{C'}$ of the 1-components C' it consists of and let V_C be the set of trials covered by the events in B_C. Every basic event participating in 1-components in C is also said to *participate* in C, but only with a single non-core event (by combining all of its non-core reduced events in the 2-component into one event).

Remark 2.2 *Let us notice some important properties that we shall frequently use in our analysis and which follow immediately from our construction:*

(1) For every 2-component C, all events in B_C are true.
(2) For every 2-component C, the trial sets covered by the core events $A' \in B_C$ are disjoint and of size more than $|T_A|^\epsilon$ of the corresponding basic event A.
(3) Every basic event has at most one core event.
(4) For every 2-component, every basic event has at most one non-core event.

(5) For every basic event A, the part of A not covered by 2-components is false.

For the next step we use the following properties.

Lemma 2.3

(1) For every 2-component C, there are outcomes for the trials in V_C so that all events participating in C are false.
(2) For every basic event A, A participates in at most one 2-component.
(3) For every basic event A, the part of A not covered by 2-components is false.
(4) For every basic event A, the part of A contained but not participating in 2-components can be split into at most two reduced events that are false under any circumstance.

Proof. To prove (1), observe that for every 2-component C, every basic event participates as at most one (possibly reduced) event in C. Thus, one can easily verify that the probability bounds in Theorem 2.1 together with the LLL imply that there are outcomes for the trials in V_C so that all events participating in C are false.

For (2), assume that there is a basic event A that participates in at least 2 different 2-components. In this case, A must have been dangerous from the viewpoint of every 1-component. This, however, would have caused combining all trials of A covered by 1-components into a single 2-component, contradicting the assumption.

For (3), suppose that A does not have trials in any 1-component. Then it must be false, because otherwise it would have been selected in step 2.1 as the initial event of a 1-component. Suppose now that A is partly contained in 1-components and that C was the last of these taking some of the trials of A. Then the part of A not taken yet must be false, because otherwise it would have been integrated into C.

In order to prove (4), note that the part of an event A covered by 2-components in which it does not participate cannot exceed $2|T_A|^\epsilon$, because otherwise A would have been dangerous from the viewpoint of one of these components. Hence, it can be split into two reduced events that are small enough so that they cannot become bad if only trials covered by the 2-components are redone. □

Lemma 2.3 implies that every event can be split into at most 4 reduced events so that all of them can be made false, which fulfills one of the requirements of Theorem 2.1. (With a closer look, actually 3 reduced events also suffice.) Furthermore, it implies the following vital property.

Corollary 2.4 *The 2-components can be considered independently of each other in order to fulfill Theorem 2.1.*

Now we are ready to present Step 3.

Step 3: For every 2-component C, apply independently Step 2 above until the 2-components resulting from C are sufficiently small.

From Section 2.2 it follows that we can identify "sufficiently small" with 2-components covering at most $O(\ln^{1/\epsilon} \ln m)$ trials. Since by Theorem 2.1 every trial only has a set of outcomes that is polylogarithmic in $n + m$, we can use the following last step.

Step 4: Find through exhaustive search outcomes for the trials in each 2-component so that all of its events are false.

2.2 Analysis of the Algorithm

The following lemma implies that the overall runtime of the algorithm presented above is polynomial in $n+m$.

Lemma 2.5 *Let ϵ and δ be chosen as in Theorem 2.1. Then,*

(1) with probability at least $1-1/m^{\alpha}$ for any constant $\alpha > 0$ the number of trials covered by any 2-component after Step 2 is at most $O((\alpha \ln m)^{1/\epsilon})$, and

(2) for any 2-component C after Step 2 of size $O(\ln^{1/\epsilon} m)$, with probability at least $1 - 1/\ln^{\alpha} m$ for any constant $\alpha > 0$ the number of trials covered by any 2-component in C after applying Step 2 to C is at most $O(((\alpha + 1/\epsilon) \cdot \ln \ln m)^{1/\epsilon})$.

In order to prove the lemma, we intend to bound the expected number of possibilities of choosing sets of core events that are able to establish a 2-component. We will do this by counting all possible sets of basic events that could represent the core events of a 2-component. There are two main problems that have to be solved for the counting:

(1) First of all, the counting has to provide an ordering for these basic events. The ordering must be able to *uniquely* determine how the 2-component must be constructed by the algorithm.

(2) A unique way of establishing a 2-component still does not uniquely determine the core events, since it can happen that there is a core event in this 2-component whose basic event has trials that are also covered by other 2-components.

These two problems are the main reasons why the analysis used in [10] does not directly apply here. There, the problem under consideration did not require to provide an ordering of the basic events, and a basic event corresponding to a core event cannot have trials belonging to two different 2-components. This made the proof significantly easier. In order to solve the problems above, we introduce the following structures.

The 1-Component Witness

Assume we are given a 1-component C with a set of core events $B_C = \bigcup_{i=0}^{d} E_i$, where E_0 consists of the initial core event of C and each E_i with $i \geq 1$ represents the set of core events added to C in round i of Build_1-Component. A graph $T = (B_T, E)$ is called a *1-component witness* of C if T is a directed tree with the property that

- the node set B_T of T is the set consisting of the basic events of all core events in B_C, and
- $(A, B) \in E$ for the basic events A and B if and only if
  - there is an $i \geq 0$ such that, for the core events A' and B' of A and B, $A' \in E_i$ and $B' \in E_{i+1}$, and
  - A is the event of smallest index with a core event in E_i that overlaps with B.

The 2-Component Witness

Furthermore, assume we are given a 2-component D consisting of a set of 1-components $S_D = \bigcup_{i=0}^{d} S_i$, where S_0 consists of the initial 1-component of D and each S_i with $i \geq 1$ represents the set of 1-components added to D in round i of Build_2-Component. A graph $T = (B_T \cup D_T, E_1 \cup E_2)$ is called a *2-component witness* of D if T is a tree of directed edges with the property that

- the node set B_T contains the basic events of all core events in D,

- the node set D_T of T contains the basic events of all dangerous events considered for D (see $(*)$ of Build_2-Component), which may include some of the basic events in B_T,
- $(A, B) \in E_1$ for the basic events $A, B \in B_T$ if and only if (A, B) is an edge of the 1-component witness of a 1-component in D, and
- $(A, B) \in E_2$ for the basic events $A, B \in B_T \cup D_T$ if and only if the trial sets of A and B overlap, A and B do not have core events in a single 1-component, and one of the two cases holds:
  (1) $A \in B_T$ and $B \in D_T$ and A is the event of smallest index among the basic events that had a core event in D before B was discovered to be dangerous, or
  (2) $A \in D_T$ and $B \in B_T$ and B is the event of smallest index in its 1-component, among those intersecting A, that was added to D due to A.

Edge set E_1 is called the set of *1-component edges*, and edge set E_2 is called the set of *2-component edges*.

Since according to Remark 2.2 (3) every basic event can only have one core event, 1- and 2-component witnesses are always guaranteed to be trees. Obviously, every 2-component has a unique 2-component witness. On the other hand, every 2-component witness T implies a 2-component C that is unique concerning

(1) its decomposition into 1-components,

(2) the order in which events are added to each of its 1-components in Build_1-Component,

(3) the order in which its 1-components are constructed, and

(4) the round in which each 1-component is added to C in Build_2-Component.

Item (1) follows from the fact that the 1-component witnesses in T are separated via 2-component edges. Furthermore, item (2) holds because Build_1-Component ensures that the order in which new events are added to a 1-component in one round is determined by their indices. Item (3) is true, since Step 2.1 ensures that the order in which the 1-components are constructed is determined by the indices of their initial events. Finally, item (4) holds, because Step 2.2 ensures that the initial 1-component of C is represented by the 1-component witness in T with the initial event of minimum index, and the round in which every other 1-component C' is added to C is determined by the number of 1-component witnesses that have to be passed in T from the initial 1-component witness to the 1-component witness representing C'.

However, a 2-component witness may not result in a 2-component with uniquely determined trial sets for the core events. The reason for this is that it can happen that a basic event can be a core event in one 2-component (say, C) and overlap with the trials of another 2-component (say, C'). This can happen if C' is constructed before C. Hence, in this case, a single 2-component witness is not able to uniquely specify the trials covered by the core events of a 2-component. Therefore, we also introduce 3-component witnesses.

The 3-Component Witness

Given a 2-component D, the *3-component witness* $T = (B_T \cup D_T, E_1 \cup E_2 \cup E_3)$ of D is a tree of directed edges that is iteratively constructed as follows:

Initially, T is equal to the 2-component witness of D. As long as there is some 2-component C not already in T that is in the neighborhood of an event in B_T, we add a *3-component edge* (A, B) to E_3 for an arbitrary pair of intersecting basic events A in T and B having a core event in C, and we add the 2-component witness of C to T.

Given a 3-component witness T, let V_T denote the set of all trials covered by the basic events in B_T. 3-component witnesses have the following important property.

Lemma 2.6 *Every 3-component witness uniquely determines the 2-component of its initial 2-component witness.*

Proof. As noted above, a 2-component witness already completely specifies a 2-component up to the trial sets covered by its core events. Since

- a 3-component witness provides a complete time ordering in which events are added to the 2-components it is witnessing,
- each time a core event is added to a 1-component, all trials of its basic event are taken that are not already covered by other 1-components, and
- there are no more 2-components that overlap with events in T,

a 3-component witness ensures the unique determination of the trials covered by the core events of the 2-component corresponding to its initial 2-component witness (and all other witnessed 2-components). $\square$

This implies the following result.

Corollary 2.7 *For any 2-component C constructed by the algorithm, there is a 3-component witness that uniquely specifies all core events in C.*

Corollary 2.7 implies that if there is no 3-component witness including a given set of basic events B, then B cannot form the core of a 2-component. This does significantly help to bound the size of 2-components, since instead of counting combinatorial structures based on core events, we are allowed to count combinatorial structures based on *basic* events, which is much easier. Another reason why we switch to witnesses is that bounding the size of 2-components directly (i.e. by arguing about the behavior of our algorithm) is extremely difficult because of high dependencies among the events (trials of a core event may have been visited several times before forming this event and adding it to a component), whereas counting witnesses is purely combinatorial (we just intend to fulfill the requirements stated for witnesses above, which are of purely syntactical nature). The core events selected in such a counting approach now have independent probabilities, since they do not overlap and their selection is not based on some algorithmic process.

To go on, we introduce some additional notation. For every basic event A, let $\lambda_A = |T_A|^\epsilon$ be called the *order* of A and any of its reduced events. Recall that the probability of an event A or any of its reduced events to be true is at most $p_A \leq \mathrm{e}^{-\lambda_A}$ (see Theorem 2.1). The *order* of a set $\mathcal{E}$ of events is defined as $\sum_{A \in \mathcal{E}} \lambda_A$. Furthermore, for every event A in a 3-component witness T, we define the *witness order* (or *ω-order* in short) ω_A of A to be equal to λ_A if $A \in B_T$ and otherwise to some value of at least λ_A^ϵ that will be specified later. The *ω-order* ω_T of a 3-component witness T is the sum of the ω-orders of all of its events in $B_T \cup D_T$. We intend to count the expected number of 3-component witnesses T that have some given ω-order. Since $\omega_T \geq \lambda_{B_T}$, which is at least the sum of the orders of the core events of the 2-component represented by its initial 2-component witness, it holds:

Lemma 2.8 *For any 2-component C, its corresponding 3-component witness T ensures that $\omega_T \geq \lambda_{B_C}$.*

Hence, in order to bound the expected number of 2-components of order at least λ, it suffices to bound the expected number of 3-component witnesses of ω-order at least λ. This will enable us to prove Lemma 2.5.

2.3 Computing the expected number of 3-component witnesses

Suppose we want to compute the expected number of 3-component witnesses of ω-order at least ω. Let M_ω be the set of all potential 3-component witnesses with this property. A 3-component witness is called *valid* if all of its core events are true. Since the core events do not overlap, their probabilities to be true are independent. Hence,

$$\Pr[T \text{ is a valid 3-component witness}] = \prod_{E \in B_T} \Pr[E \text{ is true}] \ .$$

Thus it holds that

$$\begin{aligned} \mathrm{E}[|\{\text{valid 3-component witnesses with } \omega\text{-order } \omega\}|] &= \sum_{T \in M_\omega} \Pr[T \text{ valid}] \\ &= \sum_{T \in M_\omega} \prod_{E \in B_T} \Pr[E] \ . \end{aligned}$$

Let ℓ be the depth of a witness T when rooted at the initial core event of its initial 1-component, and let ω_i denote the ω-order of the core events at depth i in T. Then the above sum can be further refined to

$$\begin{aligned} &\sum_{\ell \geq 0} \sum_{\omega_0} \sum_{\substack{\text{core events } E_0 \\ \text{of } \omega\text{-order } \omega_0}} \sum_{\substack{\omega_1, \ldots, \omega_\ell: \\ \sum_i \omega_i = \omega - \omega_0}} \sum_{\substack{\text{sets } S_1, \ldots, S_\ell \text{ of core events} \\ \text{with } \omega\text{-order } w_1, \ldots, w_\ell}} \left(\Pr[E_0 \text{ is true}] \prod_{j=1}^{\ell} \prod_{E \in S_j} \Pr[E \text{ is true}] \right) \\ = &\sum_{\ell \geq 0} \sum_{\omega_0} \sum_{\substack{\text{core events } E_0 \\ \text{of } \omega\text{-order } \omega_0}} \Pr[E_0] \sum_{\substack{\omega_1, \ldots, \omega_\ell: \\ \sum_i \omega_i = \omega - \omega_0}} \prod_{j=1}^{\ell} \left(\sum_{\substack{\text{set } S_j \text{ of core events} \\ \text{with } \omega\text{-order } \omega_j}} \prod_{E \in S_j} \Pr[E \text{ is true}] \right) \ . \end{aligned}$$

The equation holds because in general,

$$\prod_{i=1}^{n} \left(\sum_{j \in S_i} x_{i,j} \right) = \sum_{j_1, \ldots, j_n} \prod_{i=1}^{n} x_{i,j_i} \ .$$

for all n, sets S_i and values $x_{i,j}$. Note that

$$\sum_{\substack{\text{set } S_j \text{ of core events} \\ \text{with } \omega\text{-order } \omega_j}} \prod_{E \in S_j} \Pr[E \text{ is true}]$$

represents the expected number of valid sets S_j at level j of ω-order ω_j. In order to compute an upper bound on this, it suffices to know the ω-order for level $j-1$. Hence, our strategy will be to compute the expected number of 3-component witnesses T level-wise, starting with the basic event that represents the initial event of the initial 1-component of T. Given a level of ω-order ω', we then compute the expected number of possibilities to have a next, valid level of ω-order ω'' for every ω''. The structures we intend to use for counting all possibilities for one level of the 3-component witness are formalized in the following definition.

Definition 2.9 *Given any set of basic events $\mathcal{E}$, a set of basic events $\mathcal{F}$ is called a* witness extension *of $\mathcal{E}$ if $\mathcal{E}$ and $\mathcal{F}$ can be connected by a forest of 1-, 2-, and 3-component edges in a way that they fulfill all properties of two consecutive levels of a 3-component witness.*

To be able to use our level-wise approach for counting 3-component witnesses, we need a technical lemma.

2.4 A Technical Lemma

Consider some fixed set $\mathcal{E}$ of basic events. Suppose that for every event $A \in \mathcal{E}$ we have a non-negative random variable Λ_A which we call its *contribution*. Then the *contribution* $\Lambda_{\mathcal{E}}$ of $\mathcal{E}$ is defined as the sum of the contributions of all events in $\mathcal{E}$. Let

$$S_\omega^{\mathcal{E}} = \{\text{witness extensions } \mathcal{F} \text{ of } \mathcal{E} \text{ with } \Lambda_{\mathcal{F}} = \omega\} \ .$$

The next lemma provides an estimation for $\mathrm{E}[|S_\omega^{\mathcal{E}}|]$.

Lemma 2.10 *Let γ be an arbitrary positive constant with $\gamma \leq 1/288$. Furthermore, let $\mathcal{E}$ be any set of basic events and $(\Lambda_B)_{B\in N(\mathcal{E})}$ be any sequence of contributions with the property that*

(1) the contribution of any event in $N(\mathcal{E})$ is either 0 or at least $1/\gamma$,

(2) $|\{(A,B) \in \mathcal{E} \times N(\mathcal{E}) : \ \Pr[1 \leq \Lambda_B \leq k] > 0\}| \leq \omega_{\mathcal{E}} \cdot \mathrm{e}^{\gamma\cdot k}$,

(3) for every $A \in \mathcal{E}$ and $B \in N(A)$ there are at most d different ways of linking B to A, and

(4) there is a $\phi \geq 1/6$ so that for every event $A \in N(\mathcal{E})$, $\Pr[\Lambda_A = k] \leq \mathrm{e}^{-\phi\cdot k}$ independently of other events of positive contribution.

Then

$$\mathrm{E}[|S_\omega^{\mathcal{E}}|] \leq \mathrm{e}^{-\phi\omega/2} \cdot \mathrm{e}^{d\cdot\omega_{\mathcal{E}}/48} \ .$$

Proof. Follows directly from the proof of Lemma 3.3 in [10]. □

2.5 Counting Witness Extensions of 1- and 3-Component Edges

In this section we show how to apply Lemma 2.10 to bound the expected number of ways to select basic events of a certain order that form a witness extension of 1- and 3-component edges, called a *1/3-witness extension* in the following. For this we first need some simple claims.

Claim 2.11 *For every event A it holds*

$$\sum_{B\in N(A)} \mathrm{e}^{-\delta\cdot\lambda_B^\epsilon} \leq \lambda_A^\epsilon \ .$$

Proof. From the conditions of Theorem 2.1 and the fact that $\ln p_A^{-1} = \lambda_A$ we obtain for every event A that

$$\mathrm{e}^{-\lambda_A^\epsilon} \leq \mathrm{e}^{-\delta\lambda_A^\epsilon} \cdot \prod_{B\in N(A)} \left(1 - \mathrm{e}^{-\delta\lambda_B^\epsilon}\right) \ .$$

This implies that

$$\mathrm{e}^{-\lambda_A^\epsilon} \leq \prod_{B\in N(A)} \left(1 - \mathrm{e}^{-\delta\lambda_B^\epsilon}\right) \leq \prod_{B\in N(A)} \exp\left(-\mathrm{e}^{-\delta\lambda_B^\epsilon}\right) = \exp\left(-\sum_{B\in N(A)} \mathrm{e}^{-\delta\lambda_B^\epsilon}\right) \ ,$$

which immediately yields the claim. □

Claim 2.12 *For every event A and $k > 0$ it holds that*

$$|\{B \in N(A) : \lambda_B \leq k\}| \leq \mathrm{e}^{\delta \cdot k^\epsilon} \cdot \lambda_A^\epsilon \ .$$

Proof. Follows directly from Claim 2.11 above. □

Claim 2.13 *Every event must have an order of at least $(1/\delta)^{1/\epsilon}$.*

Proof. In Theorem 2.1 we require that $x_i \leq 1/\mathrm{e}$ for all $i \in [m]$. This implies that, for every event A_i,

$$\mathrm{e}^{-\delta \ln^\epsilon p_i^{-1}} \leq \frac{1}{\mathrm{e}} \quad \Leftrightarrow \quad \delta \cdot \lambda_{A_i}^\epsilon \geq 1 \quad \Leftrightarrow \quad \lambda_{A_i} \geq (1/\delta)^{1/\epsilon} \ .$$

□

Now we are ready to apply Lemma 2.10. Consider any set $\mathcal{E}$ of basic events. Let the contribution Λ_B of an event $B \in N(\mathcal{E})$ be defined either as its order λ_B if the core event corresponding to B is true, or 0 otherwise. (Recall that according to Corollary 2.7 the core event of B is uniquely determined although we only count basic events.)

Condition (1) of Lemma 2.10 follows from Claim 2.13. Our assumption that $\omega_A \geq \lambda_A^\epsilon$ for all events $A \in \mathcal{E}$ and Claim 2.12 together imply that

$$|\{(A, B) \in \mathcal{E} \times N(\mathcal{E}) : \omega_B \leq k\}| \leq \mathrm{e}^{\delta \cdot k^\epsilon} \sum_{A \in \mathcal{E}} \lambda_A^\epsilon \leq \mathrm{e}^{\delta \cdot k^\epsilon} \cdot \omega_{\mathcal{E}} \ .$$

Thus, condition (2) is true. For every $A \in \mathcal{E}$ and $B \in N(A)$ we have 3 choices of connecting B to A: 2 choices for a 1-component edge, and one choice for a 3-component edge. We will not count the possibilities of B being dangerous or not dangerous here, since this will be considered in the analysis for the 2-component edges. Hence, condition (3) holds with $d = 3$.

In order to fulfill condition (4), we use only a part of the probability we obtain for a core event to be true. The reason for this is that we want to reserve the other part for the 2-component edges. In particular, instead of a probability of $\mathrm{e}^{-\lambda_B}$ for the core event of B to be true, we only use a probability of $\mathrm{e}^{-\lambda_B/2} = \mathrm{e}^{-\omega_B/2}$. Thus, condition (3) follows with $\phi = 1/2$ and by the fact that the core events use disjoint trial sets.

Since all requirements of Lemma 2.10 are fulfilled, the expected number of possibilities $C_{\mathcal{E},\omega}^{(1,3)}$ of choosing events of contribution altogether ω for a 1/3-witness extension of $\mathcal{E}$ satisfies

$$C_{\mathcal{E},\omega}^{(1,3)} \leq \mathrm{e}^{-\omega/4} \cdot \mathrm{e}^{\omega\varepsilon/16} \ . \tag{2}$$

2.6 Counting Witness Extensions of 2-Component Edges

In this section we show how to apply Lemma 2.10 to bound the expected number of ways to select event sets of a certain order that form a witness extension of 2-component edges, called *2-witness extension* in the following.

Let us fix some set $\mathcal{E}$ of events. We consider two different cases for counting 2-component edges of the form $(A, B) \in \mathcal{E} \times N(\mathcal{E})$ for a 3-component witness T:

(1) $B \in B_T$, or

(2) $B \in D_T \setminus B_T$.

We start with the first case. For this case, we define the contribution Λ_B of B as λ_B if the core event of B is true and 0 otherwise. Using this definition, condition (1) of Lemma 2.10 follows from Claim 2.13. Analogous to Section 2.5, also condition (2) is true. For every $A \in \mathcal{E}$ and $B \in N(A)$ we have 2 choices of connecting B to A: either A is dangerous, or B. Hence, condition (3) holds with $d = 2$. Furthermore, as in Section 2.5, condition (4) holds with $\phi = 1/2$.

Now we consider the second case. This case significantly differs from our previous considerations, since B itself does *not* provide a probability (B does not have a core event like the other events!). Therefore, we have to "borrow" probabilities from other core events. To achieve this, we define the ω-order ω_B of the dangerous event B to be the sum of the orders of the set S of core events that were added to the 2-component due to B. From $(*)$ of Build_2-Component we know that the core events in S have to cover at least λ_B trials of B. That is, $\sum_{C\in S}\lambda_C^{1/\epsilon} \geq \lambda_B$. Since $(\sum_{C\in S}\lambda_C)^{1/\epsilon} \geq \sum_{C\in S}\lambda_C^{1/\epsilon}$, we must have $\omega_B \geq \lambda_B^{\epsilon}$. Since for each of the core events C in S we still have a probability of $\mathrm{e}^{-\lambda_C/2}$ available, we can assign a probability of $\mathrm{e}^{-\omega_B/2} \leq \mathrm{e}^{-\lambda_B^{\epsilon}/2}$ to B.

Let us define the *contribution* Λ_B of B as ω_B if the core events added to the 2-component due to B are true, and 0 otherwise. Then Claim 2.13 together with the inequality $\omega_B \geq \lambda_B^{\epsilon}$ establishes condition (1) of Lemma 2.10. For the remaining two conditions we need a simple claim.

Claim 2.14 *For every event A and every $k > 0$ it holds*

$$|\{B \in N(A):\ \omega_B \leq k\}| \leq |\{B \in N(A) : \lambda_B \leq k^{1/\epsilon}\}| \leq \mathrm{e}^{\delta\cdot k}\cdot\lambda_A^{\epsilon}\ .$$

Proof. The proof of the claim follows directly from Claim 2.12. □

Condition (2) follows from Claim 2.14, condition (3) follows with $d = 1$, and condition (4) follows with $\phi = 1/2$.

Combining the two cases (which yields $d = 3$), it follows from Lemma 2.10 that the expected number of possibilities $C^{(2)}_{\mathcal{E},\omega}$ of choosing events of contribution ω for a 2-witness extension of $\mathcal{E}$ satisfies

$$C^{(2)}_{\mathcal{E},\omega} \leq \mathrm{e}^{-\omega/4}\cdot\mathrm{e}^{\omega\varepsilon/16}\ . \tag{3}$$

2.7 Counting Witness Extensions

Now we are ready to count arbitrary witness extensions. Given a set $\mathcal{E}$ of basic events, our aim is to count the expected number of witness extensions $\mathcal{F}$ of $\mathcal{E}$ of ω-order ω, denoted by $C_{\mathcal{E},\omega}$. Let $S_{\omega,\delta}$ represent the set $\{0, \lceil 1/\delta\rceil, \lceil 1/\delta\rceil+1, \ldots, w-\lceil 1/\delta\rceil, w\}$. According to the previous two sections we obtain

$$\begin{aligned}
C_{\mathcal{E},\omega} &\leq \sum_{\kappa\in S_{\omega,\delta}} C^{(1,3)}_{\mathcal{E},\kappa}\cdot C^{(2)}_{\mathcal{E},\omega-\kappa}\\
&\leq \sum_{\kappa\in S_{\omega,\delta}} \mathrm{e}^{-\kappa/4}\cdot\mathrm{e}^{\omega\varepsilon/16}\cdot\mathrm{e}^{-(\omega-\kappa)/4}\cdot\mathrm{e}^{\omega\varepsilon/16}\\
&\leq \max[\omega-2/\delta+3,\ 2]\cdot\mathrm{e}^{-\omega/4}\cdot\mathrm{e}^{\omega\varepsilon/8} \leq \mathrm{e}^{-\omega/6}\cdot\mathrm{e}^{\omega\varepsilon/8}
\end{aligned}$$

if $\delta > 0$ is sufficiently small.

2.8 Proof of Lemma 2.5

Assume that we intend to bound the expected number of 2-components of order at least λ. According to Lemma 2.8, this can be achieved by bounding the expected number of 3-component witnesses of ω-order at least λ. Suppose that the 3-component witnesses have to start with some fixed initial event A_0 of ω-order ω_0. Then it holds for all $\omega \geq \omega_0$ that

$$\begin{aligned}
&\mathrm{E}\left[|\{\text{3-component witnesses } T \text{ with initial event } A_0 \text{ and } \omega_T = \omega\}|\right] \\
\leq\ & \sum_{\ell=0}^{\delta\omega} \sum_{\substack{\omega_1,\dots,\omega_\ell \in \{\lceil 1/\delta \rceil,\dots,\omega\}: \\ \sum_{j\geq 0} i_j = \omega}} \mathrm{e}^{-\omega_0} \prod_{j=0}^{\ell-1} \mathrm{e}^{-\omega_{j+1}/6} \cdot \mathrm{e}^{\omega_j/8} \\
\leq\ & \binom{\omega+\delta\omega}{\delta\omega} \cdot \mathrm{e}^{-\omega/6} \cdot \mathrm{e}^{\omega/8} \leq \left(\frac{\mathrm{e}(1+\delta)}{\delta}\right)^{\delta\omega} \mathrm{e}^{-\omega/24} \\
\leq\ & \mathrm{e}^{-\omega/48}
\end{aligned}$$

if $\delta > 0$ is sufficiently small. On the other hand we know that

$$\omega_T = \sum_{A \in B_T \cup D_T} \omega_A \geq \sum_{A \in B_T} \lambda_A = \sum_{A \in B_T} |T_A|^\epsilon .$$

Since $\sum_{A \in B_T} |T_A|^\epsilon \geq (\sum_{A \in B_T} |T_A|)^\epsilon$ and $\sum_{A \in B_T} |T_A| \geq |V_T|$, it follows that $|V_T| \leq \omega_T^{1/\epsilon}$.

Now we are ready to bound the number of trials covered by the 2-components after Step 2 and Step 3 of our algorithm.

Step 2:

Since there are at most m events that can be the initial event of a 3-component witness of ω-order at least ω, we obtain

$$\mathrm{E}[|\{\text{3-component witnesses } T \text{ of } \omega\text{-order at least } \omega\}|] \leq m \cdot \sum_{\kappa \geq \omega} \mathrm{e}^{-\kappa/48} \leq 50m \cdot \mathrm{e}^{-\omega/48} .$$

This is at most $m^{-\alpha}$ if $\omega \geq 48((\alpha+1)\ln m + 4)$. Thus, with probability at least $1 - 1/m^\alpha$ for any constant $\alpha > 0$ the number of trials covered by a 3-component witness is at most $[48((\alpha+1)\ln m + 4)]^{1/\epsilon}$, which proves part (1) of Lemma 2.5.

Step 3:

For every 2-component C covering at most $\gamma \ln^{1/\epsilon} m$ trials, its set S of core events fulfills $\sum_{A \in S} \lambda_A \leq \gamma \ln^{1/\epsilon} m$. Hence, according to Claim 2.12 there can be at most $\gamma(\ln m)^{1/\epsilon} \cdot \mathrm{e}^{\delta k^\epsilon}$ events of order at most k that are participating in C. Therefore, the expected number of 3-component witnesses of order at least ω in C is at most

$$\sum_{\kappa \geq \omega} \gamma(\ln m)^{1/\epsilon} \cdot \mathrm{e}^{\delta \cdot \kappa^\epsilon} \cdot \mathrm{e}^{-\kappa/48} \leq \gamma(\ln m)^{1/\epsilon} \sum_{\kappa \geq \omega} \mathrm{e}^{-\kappa/96} \leq 100\gamma(\ln m)^{1/\epsilon} \cdot \mathrm{e}^{-\omega/96}$$

if ϵ and δ are sufficiently small. This is at most $(\ln m)^{-\alpha}$ if $\omega \geq 96((\alpha + 1/\epsilon)\ln\ln m + \ln(100\gamma))$. Thus with probability at least $1 - 1/(\ln m)^\alpha$ for any constant $\alpha > 0$ the number of trials covered by a 3-component witness is at most $[96((\alpha + 1/\epsilon)\ln\ln m + \ln(100\gamma))]^{1/\epsilon}$, which proves part (2) of Lemma 2.5.

3 The MIP Algorithm

In this section we present an algorithm that fulfills Theorem 1.5. First we give a high-level description of the algorithm.

Given an arbitrary MIP, we start with finding an optimal solution $\{x^*_{i,j} : \ i \in [n],\ j \in [\ell_i]\}$ to the LP relaxation of the MIP (i. e., the integrality constraints in Definition 1.3 (3) are removed). The resulting LP optimum $\boldsymbol{y}^*$ is clearly a lower bound for $\boldsymbol{y}$, the optimum of the (integral form of the) MIP.

Afterwards, our algorithm works in 3 steps. Its structure is similar to the algorithm in [22], but the techniques used here are completely different. Step 1 rounds the $x^*_{i,j}$'s to multiples of some number in $\Omega(1/\ln m)$. Step 2, which contains the main novel contribution in this chapter, rounds the $x^*_{i,j}$'s until all of them are above some constant $\gamma < 1$ or 0. Step 3 finally rounds the $x^*_{i,j}$'s to values in $\{0, 1\}$.

Step 1: Initial Rounding

For every $r \in \{1, \dots, m\}$, let $w_r = \max_c a_{r,c}$. First, we scale $\boldsymbol{A}$ to a matrix $\boldsymbol{A}^{(1)} = (a^{(1)}_{r,c})$ by multiplying row r of $\boldsymbol{A}$ with w_r^{-1} for all r. This ensures that $\max_c a^{(1)}_{r,c} = 1$ for all r. Let the vector $\boldsymbol{y}^{(1)}$ be equal to $\boldsymbol{A}^{(1)}\boldsymbol{x}^*$. For the remaining steps we will assume that the approximation vector $\boldsymbol{\alpha}$ fulfills the following property for some $0 < \epsilon < 1$ (see Theorem 1.5):

$$\alpha_r \geq \sigma_r \quad \text{with} \quad \sigma_r = \max\left[(y^{(1)}_r)^{-1},\ (y^{(1)}_r)^{-(1-\epsilon)/2}\right] .$$

Let $\mu = \max_r \lceil (6 \ln m)/(\min[\sigma_r, \sigma_r^2] y^{(1)}_r) \rceil$. Since $\min[\sigma_r, \sigma_r^2] \cdot y^{(1)}_r \geq 1$ for every r, $\mu \leq \lceil 6 \ln m \rceil$. If $\mu = 1$, then we simply use the randomized rounding strategy by Raghavan and Thompson [28] to transform $\boldsymbol{x}^*$ into an integral vector $\boldsymbol{x}^{(1)}$. That is, for every $i \in [n]$ we set the variable $x^{(1)}_{i,j}$ to 1 and the other $x^{(1)}_{i,j'}$'s to 0 with probability $x^*_{i,j}$. In this case,

$$\begin{aligned}
\Pr[(\boldsymbol{A}^{(1)}\boldsymbol{x}^{(1)})_r \geq (1 + \alpha_r) y^{(1)}_r] &\leq \mathrm{e}^{-\min[\alpha_r, \alpha_r^2] y^{(1)}_r / 3} \\
&\leq \mathrm{e}^{-\min[\sigma_r, \sigma_r^2] y^{(1)}_r / 3} \leq \mathrm{e}^{-2 \ln m} = \frac{1}{m^2}
\end{aligned}$$

for every row r. Hence, with high probability we obtain an integral solution $\boldsymbol{x}^{(1)}$ with $(\boldsymbol{A}^{(1)}\boldsymbol{x}^{(1)})_r < (1+\alpha_r) y^{(1)}_r$ for all r as proposed in Theorem 1.5.

For the case that $\mu > 1$, we select a matrix $\boldsymbol{R} = (r_{i,k})_{i \in [n], k \in [\mu]}$ uniformly at random out of $[0, 1)^{n \times \mu}$. For every i, j, we set $x^{(1)}_{i,j} = |\{k :\ r_{i,k} \in [\sum_{\ell=1}^{j-1} x^*_{i,\ell}, \sum_{\ell=1}^{j} x^*_{i,\ell})\}|/\mu$. Consider a system of bad events $E_1, \dots, E_m$ with E_r being true if and only if $(\boldsymbol{A}^{(1)}\boldsymbol{x}^{(1)})_r \geq (1 + \alpha_r) y^{(1)}_r$. We show now how to bound the probability of E_r being true for any r. For any $i \in [n]$ and $k \in [\mu]$, let the random variable $X_{i,k}$ be equal to $a^{(1)}_{r,(i,j)}/\mu$ if and only if $r_{i,k}$ is selected so that it contributes $1/\mu$ to $x^{(1)}_{i,j}$. The definition ensures that $\sum_{i,k} X_{i,k} = (\boldsymbol{A}^{(1)}\boldsymbol{x}^{(1)})_r$. Obviously,

$$\mathrm{E}[\sum_{i,k} X_{i,k}] = \mathrm{E}[(\boldsymbol{A}^{(1)}\boldsymbol{x}^{(1)})_r] = y^{(1)}_r .$$

Since $X_{i,k} \leq 1/\mu$ for all i, k and the random variables are independent, we can apply the Hoeffding bounds (after a scaling) to obtain

$$\Pr[E_r] \quad \leq \quad \mathrm{e}^{-\min[\alpha_r, \alpha_r^2] \cdot \mu \cdot y^{(1)}_r / 3} \leq \mathrm{e}^{-2 \ln m} = \frac{1}{m^2} .$$

for all r. Hence, with high probability none of the bad events occurs. Assume that the random experiment has been successful in avoiding all bad events (otherwise we repeat it). Then we replace $\boldsymbol{x}^*$ by the N-vector $\boldsymbol{x}^{(1)}$. In order to simplify the following calculations, we assume in the following as a worst case that, after Step 1, $(\boldsymbol{A}^{(1)}\boldsymbol{x}^{(1)})_r = (1+\alpha_i)y_r^{(1)}$ for all r.

Step 2: Intermediate Rounding

This is the first step that requires the LLL (and the consideration of a dependency graph that may have already been significantly reduced by the previous step). The aim of this step is to round the vector $\boldsymbol{x}^{(1)}$ with entries in $\{k/\mu : \ k \in \{0, \dots, \mu\}\}$ to a vector $\boldsymbol{x}^{(2)}$ with entries in $\{k/\gamma : \ k \in \{0, \dots, \gamma\}\}$ for some fixed integer $\gamma > 1$. This will be done in several phases, starting with phase 1. Initially, we set $\boldsymbol{z}^{(0)}$ equal to $\boldsymbol{x}^{(1)}$ and μ_0 equal to μ. The outcome vector of phase $\varphi \geq 1$ is defined by $\boldsymbol{z}^{(\varphi)}$. The final vector $\boldsymbol{z}^{(\varphi)}$ of Step 2 is set to $\boldsymbol{x}^{(2)}$.

The task of phase φ is to round a vector $\boldsymbol{z}^{(\varphi-1)}$ of values that are multiples of $1/\mu_{\varphi-1}$ to a vector $\boldsymbol{z}^{(\varphi)}$ of values that are multiples of $1/\mu_\varphi$ with $\mu_\varphi = \lceil \mu_{\varphi-1}^{\epsilon} \ln \mu_{\varphi-1} \rceil$. This is done as follows.

For simplicity, set $\boldsymbol{x} = \boldsymbol{z}^{(\varphi-1)}$, $\boldsymbol{x}' = \boldsymbol{z}^{(\varphi)}$, $\mu = \mu_{\varphi-1}$, $\mu' = \mu_\varphi$, and $\boldsymbol{A} = \boldsymbol{A}^{(1)}$. The random experiment for applying the LLL is to select a matrix $\boldsymbol{R} = (r_{i,k})_{i\in[n],k\in[\mu']}$ uniformly at random out of $[0,1)^{n\times\mu'}$. For every i, j, we set $x'_{i,j} = |\{k : \ r_{i,k} \in [\sum_{\ell=1}^{j-1} x_{i,\ell}, \sum_{\ell=1}^{j} x_{i,\ell})\}|/\mu'$. Consider a system of bad events $E_1, \dots, E_m$ with E_r being true if and only if

$$(\boldsymbol{A}\boldsymbol{x}')_r > \left(1 + \frac{\min[\alpha_r, \ 1]}{\sqrt{\ln \mu}}\right) \cdot (\boldsymbol{A}\boldsymbol{x})_r \ .$$

Let the dependency graph G of the E_r's be defined as in Definition 1.4. Clearly, every event E_r is mutually independent of all events it is not connected with in G. Assume that the approximation vector $\boldsymbol{\alpha}$ is chosen such that inequality (1) in Theorem 1.5 holds. As will be shown in the next subsection, in this case Theorem 2.1 predicts that a vector $\boldsymbol{x}'$ can be provided in polynomial time so that for all r,

$$(\boldsymbol{A}\boldsymbol{x}')_r \leq \left(1 + \frac{c \cdot \min[\alpha_r, \ 1]}{\sqrt{\ln \mu}}\right) \cdot (\boldsymbol{A}\boldsymbol{x})_r$$

for some constant $c > 0$.

Step 2 ends with the first μ_φ for which $\mu_{\varphi+1} > \mu_\varphi^{\epsilon+(1-\epsilon)/2}$ (i. e., phase φ is the last phase). In this case, $\mu_\varphi^{(1-\epsilon)/2} < \ln \mu_\varphi$. In general, for all $0 < \delta < 1$ it holds that if $x^\delta < \ln x$ then $x < \mathrm{e}^{1/(1-\delta)}$. Thus, in our case, $\mu_\varphi < \mathrm{e}^{2/(1+\epsilon)}$, which is a constant. For all phases φ of Step 2 it therefore holds that $\mu_{\varphi-1} \geq \mu_\varphi^{2/(1+\epsilon)}$. This enables us to bound the approximation factor obtained in Step 2. We distinguish between two cases. If $\alpha_r < 1$ for some row r then

$$\begin{aligned}\prod_{j=0}^{\infty}\left(1 + \frac{c\cdot\alpha_r}{\sqrt{\ln \mu_\varphi^{(2/(1+\epsilon))^j}}}\right) \ &\leq \ \exp\left(c\cdot\alpha_r \sum_{j=0}^{\infty} \frac{1}{(2/(1+\epsilon))^{j/2}\sqrt{\ln \mu_\varphi}}\right) \\ &\leq \ \exp\left(O(\alpha_r)\right) \ = \ (1 + O(\alpha_r)) \ .\end{aligned}$$

Thus, $(\boldsymbol{A}\boldsymbol{z}^{(\varphi)})_r \ \leq \ (1 + O(\alpha_r))y_r^{(1)}$ for all r. If $\alpha_r \geq 1$, then we also obtain an approximation ratio of $(1 + O(\alpha_r))$, since $\prod_j (1 + c/\sqrt{\ln \mu_j})$ is a constant.

Applying Theorem 2.1 to Step 2

Our aim is to show that every phase of Step 2 can be made to match the requirements of Theorem 2.1. Let $\boldsymbol{x}$, $\boldsymbol{x}'$, μ, and μ' be chosen as above, and let $\boldsymbol{A}$ represent the matrix $\boldsymbol{A}^{(1)}$. In order to avoid dealing with events of non-uniform weights, we use the following strategy. Let ν be the minimum value of an $a_{i,j}$ in $\boldsymbol{A}$. For every $k \in \{0, \ldots, \lfloor \log \nu \rfloor\}$ and every row r in $\boldsymbol{A}$, let the set $M_{r,k}$ contain all (i,j) with $a_{r,(i,j)} \in (2^{-(k+1)}, 2^{-k}]$ and $x_{i,j} > 0$. Set $m_{r,k} = 2^{k+1}\mu \cdot (\boldsymbol{A}\boldsymbol{x})_r$. Obviously, $|M_{r,k}| \leq m_{r,k}$. Let the matrix $\boldsymbol{B}^{(k)} = (b_{r,c}^{(k)})$ represent $\boldsymbol{A}$ with zeroes at all entries $a_{r,c}$ with $c \notin M_{r,k}$. Furthermore, let $y_{r,k}$ be chosen so that $(\boldsymbol{B}^{(k)}\boldsymbol{x})_r = y_{r,k}$. Set $y_r = \sum_{k \geq 0} y_{r,k}$. We define the event $E_{r,k}$ to be true if and only if

$$(\boldsymbol{B}^{(k)}\boldsymbol{x}')_r > y_{r,k} + \frac{6\min[\alpha_r,\ 1]}{\sqrt{2^{(1-\epsilon)k}\ln\mu}} \cdot y_r \ .$$

Assume that all events $E_{r,k}$ were false. Then it follows that

$$(\boldsymbol{A}\boldsymbol{x}')_r \leq \sum_{k \geq 0} \left(y_{r,k} + \frac{6\min[\alpha_r,\ 1]}{\sqrt{2^{(1-\epsilon)k}\ln\mu}} \cdot y_r \right) = \left(1 + O\left(\frac{\min[\alpha_r,\ 1]}{\sqrt{\ln\mu}} \right) \right) y_r \ ,$$

as desired.

We show now how to bound the probability of $E_{r,k}$ being true. Recall the random experiment above of selecting the $\boldsymbol{x}'_{i,j}$'s. For any $i \in [n]$ and $\ell \in [\mu']$, let the random variable $X_{i,\ell}$ be equal to $b_{r,(i,j)}^{(k)}/\mu'$ if and only if $r_{i,\ell}$ is selected so that it contributes $1/\mu'$ to $x'_{i,j}$. The definition ensures that $\sum_{i,\ell} X_{i,\ell} = (\boldsymbol{B}^{(k)}\boldsymbol{x}')_r$. Obviously,

$$\mathrm{E}[\sum_{i,\ell} X_{i,\ell}] = \mathrm{E}[(\boldsymbol{B}^{(k)}\boldsymbol{x}')_r] = y_{r,k} \ .$$

Since $X_{i,\ell} \leq 1/(\mu' 2^k)$ for all i, k and the random variables are independent, and since $y_{r,k} \leq y_r$, the Hoeffding bounds give (after a scaling) that $\Pr[E_{r,k}]$ is at most

$$\exp\left(-6 \left(\frac{\min[\alpha_r,\ 1]}{\sqrt{2^{(1-\epsilon)k}\ln\mu}} \right)^2 \cdot (\mu' 2^k) \cdot y_r/3 \right) \leq \exp\left(-2 \cdot 2^{\epsilon k}\mu^{\epsilon}\min[\alpha_r^2,\ 1] y_r \right) \ .$$

Recall that $m_{r,k} = 2^{k+1}\mu \cdot y_r$ and, by our assumption, $y_r \geq (1+\alpha_r) y_r^{(1)} \geq 1$. As $\alpha_r < 1$ is only allowed for $y_r^{(1)} \geq 1$, we obtain for $\alpha_r < 1$ that $\alpha_r^2 y_r \geq y_r^{\epsilon}$ and therefore

$$2^{\epsilon k}\mu^{\epsilon}\min[\alpha_r^2,\ 1] y_r \geq 2^{\epsilon k}\mu^{\epsilon} y_r^{\epsilon} \geq m_{r,k}^{\epsilon} \cdot 2^{-\epsilon} \ .$$

Clearly, the same inequality holds for $\alpha_r \geq 1$. Thus, we have $\Pr[E_{r,k}] \leq \mathrm{e}^{-m_{r,k}^{\epsilon}}$. On the other hand, the minimum number of $X_{i,\ell}$'s and therefore of $r_{i,\ell}$'s necessary to obtain a value of $6\min[\alpha_r,\ 1]/(\sqrt{2^{(1-\epsilon)k}\ln\mu}) \cdot y_r$ is at least $6 \cdot 2^{\epsilon k}\mu^{\epsilon}\min[\alpha_r,\ 1] y_r$, which is more than $m_{r,k}^{\epsilon}$.

Next we show how to transform inequality (1) in Theorem 1.5 into a condition that matches the requirements of Theorem 2.1. First, we define the dependency graph. Suppose that we are dealing with a MIP instance I. Then we define the dependency graph G'_I to contain the edge $((r,k),(s,\ell))$ if and only if $(r,s) \in G_I$ and either $r \neq s$ or $k \neq \ell$. (Here we use the fact that G_I may have self-loops.) It is easy to check that G'_I covers all dependencies among the $E_{r,k}$ events.

Let p_r, q_r, and z_r be defined as in Theorem 1.5. That is,

$$p_r = \mathrm{e}^{-\min[\alpha_r, \alpha_r^2] y_r^{(1)}/3} \ , \quad q_r = p_r^{-\ln^{\epsilon} p_r^{-1}} \ , \text{ and } z_r = q_r^{\delta} \ .$$

Set $p_{r,k} = \exp(-2^{\epsilon k+1}\min[\alpha_r^2, 1] y_r)$. Since we assumed at the end of Step 1 that $y_r \geq (1+\alpha_r) y_r^{(1)}$, it holds that $p_{r,k} \leq p_r^{2^{\epsilon k}}$. Let $q_{r,k} = \mathrm{e}^{-\ln^{\epsilon} p_{r,k}^{-1}}$ and $z_{r,k} = q_{r,k}^{\delta}$. Then we have

$$q_{r,k} = \mathrm{e}^{-\ln^{\epsilon} p_{r,k}^{-1}} \leq \mathrm{e}^{-\delta 2^{\epsilon^2 k}\ln^{\epsilon} p_r^{-1}} = q_r^{2^{\epsilon^2 k}} \ . \tag{4}$$

Since $z_r \leq 1/3$ by Theorem 1.5, $\sum_{k\geq 0} z_r^{2^{\epsilon^2 k}} \leq \epsilon^{-2} z_r/(1-z_r) \leq 3/(2\epsilon^2) \cdot z_r$ for all r. Thus,

$$\prod_{k\geq 0} \mathrm{e}^{-z_{r,k}} \geq \mathrm{e}^{-3/(2\epsilon^2)\cdot z_r} \geq (1-z_r)^{3/(2\epsilon^2)} \ .$$

On the other hand, $(1-z_{r,k}) \geq \mathrm{e}^{-(4/3)z_{r,k}}$ for all $z_{r,k}$ since $\mathrm{e}^{-(4/3)x} \leq 1 - 4x/3 + 8x^2/9 \leq 1-x$ for all $x \leq 1/3$. Hence, $\prod_{k\geq 0}(1-z_{r,k}) \geq (1-z_r)^{2/\epsilon^2}$. Furthermore, due to (4) it holds for all (i,k) that

$$\frac{q_{r,k}}{z_{r,k}} = q_{r,k}^{1-\delta} \leq q_r^{1-\delta} = \frac{q_r}{z_r} \ .$$

Thus, we obtain from inequality (1) in Theorem 1.5 that

$$q_{r,k} \leq z_{r,k} \prod_{((r,k),(s,\ell))\in G'_I} (1-z_{s,\ell}) \ .$$

Next we define the trials and events used in Theorem 2.1. The $r_{i,\ell}$'s represent the trials $t_1,\dots,t_n$, and the $E_{r,k}$'s represent the events $A_1,\dots,A_m$. Our random experiment ensures that the trials are mutually independent. Let $T_{r,k}$ represent the set of all $r_{i,\ell}$'s for which there is an $x'_{i,j}$ counting for $E_{r,k}$ (that is, $x_{i,j} > 0$). Obviously, every event $E_{r,k}$ only depends on the outcome of the trials represented by $T_{r,k}$. If a restricted set S of $r_{i,\ell}$'s is applied to an event $E_{r,k}$, then we define $E_{r,k}|_S$ to be true if and only if

$$(\boldsymbol{B}^{(k)} \cdot \boldsymbol{x}')_r|_S > \mathrm{E}[(\boldsymbol{B}^{(k)} \cdot \boldsymbol{x}')_r|_S] + \frac{6\min[\alpha_r,\ 1]}{\sqrt{2^{(1-\epsilon)k}\ln\mu}} \cdot y_r \ .$$

($(\boldsymbol{B}^{(k)} \cdot \boldsymbol{x}')_r|_S$ counts only those entries that can be greater than 0 due to some $r_{i,\ell} \in S$.) This ensures that, under the assumption that an event $E_{r,k}$ can be decomposed into at most some constant number of events $E_{r,k}|_S$ that are false (as claimed by Theorem 2.1), we obtain the desired asymptotic approximation ratio. As shown above, $p_{r,k} \leq \mathrm{e}^{-m_{r,k}^{\epsilon}}$. Furthermore, we already showed that if $|S| > m_{r,k}^{\epsilon}$ then the above definition yields $\Pr[E_{r,k}|_S] \leq p_{r,k}$. Otherwise, we can set $\Pr[E_{r,k}|_S] = 0$, since more than $m_{r,k}^{\epsilon}$ trials are needed to violate our deviation bound.

There is one aspect of our application that does not directly match the requirements of Theorem 2.1: $|T_{r,k}|$, the number of trials counting for $E_{r,k}$, cannot be upper bounded by $m_{r,k}$, but only by $\mu \cdot m_{r,k}$. This changes the number of trials covered by a 2-component C from at most $\sum_{E_{r,k}\in B_C} m_{r,k}$ to at most $\sum_{E_{r,k}\in B_C} |T_{r,k}| \leq \mu \sum_{E_{r,k}\in B_C} m_{r,k}$. Thus, the size of the final 2-components changes to $O(\mu(\frac{1}{\epsilon}\ln\ln m)^{1/\epsilon})$. If $\mu \geq \sqrt{\log m}$, it may not be possible any more to find a final solution via exhaustive search in polynomial time. However, every event participating in this 2-component has a probability of at most $p_{r,k} \leq \mathrm{e}^{-\mu^{\epsilon}}$ to be true. Furthermore, it follows from Claim 2.12 that a 2-component of q nodes can have at most $q \cdot \mathrm{e}^{\delta\cdot k^{\epsilon}}$ events of order at most k, and each of these events has a probability of at most e^{-k} to be true. Hence, an expected constant number of random experiments suffice for each 2-component to obtain a final solution. Thus, Step 2 of the MIP algorithm can be done in polynomial time.

Step 3: Final Rounding

Recall that $\boldsymbol{x}^{(2)}$ is the vector resulting from $\boldsymbol{x}^{(1)}$ in Step 2. Let $\boldsymbol{B}^{(k)}$, $y_{r,k}$, and y_r be defined as in the previous step. Step 3 works like a phase in Step 2 with the difference that now the random experiment simply consists of rounding $\boldsymbol{x}^{(2)}$ to an integral vector $\boldsymbol{x}$ in a way that for every i, $x_{i,j}$ is set to 1 and the other $x_{i,j'}$'s to 0 with probability $x_{i,j}^{(2)}$. Assuming that $y_r \geq (1+\alpha_r)y_r^{(1)}$ for all r it holds

$$\begin{aligned} \Pr\left[(\boldsymbol{B}^{(k)}\boldsymbol{x})_r \geq y_{r,k} + \frac{c \cdot \min[\alpha_r, 1]}{\sqrt{2^{(1-\epsilon)k} \ln \mu_\varphi}} \cdot y_r\right] &\leq \exp\left(-c\left(\frac{\min[\alpha_r, 1]}{\sqrt{2^{(1-\epsilon)k} \ln \mu_\varphi}}\right)^2 \cdot 2^k \cdot y_r/3\right) \\ &\leq \exp\left(-2 \cdot 2^{\epsilon k} \mu_\varphi^\epsilon \min[\alpha_r^2, 1] y_r\right) \end{aligned}$$

if $c \geq 6\mu_\varphi^\epsilon \ln \mu_\varphi$, which is a constant. Hence, as in Step 2, we can apply Theorem 2.1 to obtain altogether an approximation ratio of $1 + O(\alpha_r)$ for every row r. This concludes the proof of Theorem 1.5.

4 Applications

In this section we present two applications of our method: job shop scheduling and MAX SAT.

4.1 Job Shop Scheduling

In acyclic job shop scheduling problems there are n jobs and m machines. Each job is composed of a sequence of operations to be performed on different machines. A *legal* schedule is one in which within each job, operations are carried out in order, and each machine performs at most one operation in any unit of time. If D denotes the length of the longest job and C denotes the number of time units requested by all jobs on the most loaded machine, then clearly $lb = \max[C, D]$ is a lower bound on the length of the shortest legal schedule. Let P denote the longest operation a job has on a machine.

In [14] several non-constructive results are shown about the makespan of job shop schedules. Our techniques allow to find in polynomial time schedules with a makespan that is close to the predicted makespan (instead of 1, we obtain a $1+\epsilon$ in the exponent of some terms). In particular, we will show that the following result holds.

Theorem 4.1 *In polynomial time, acyclic job shop schedules can be computed with the following upper bounds on the makespan for any constant $\epsilon > 0$:*

(1) For any acyclic JSS problem,

$$L = O\left((C + D(\log\log P)^{1+\epsilon})^{\frac{\log P}{\log(\min[C/D + (\log\log P)^{1+\epsilon}, P])}}\right),$$

implying that $L = O(lb \log lb (\log\log lb)^{1+\epsilon})$. Observe that if $C \geq D \cdot P^\delta$ for some constant $\delta > 0$, then $L = \Theta(C)$.

(2) If operation lengths depend only on the machine on which the operation is performed, then $L = O(C + D(\log\log P)^{1+\epsilon})$.

(3) With preemption, $L = O(C + D(\log\log P)^{1+\epsilon})$.

In order to prove Theorem 4.1, we have to constructivize the existence proof given for an intermediate schedule in Section 2.5 in [14]. Assume that we are given a fixed constant $0 < \epsilon < 1$. Let $d = \log^{1/\epsilon} D$. To avoid dealing with floors and ceilings we assume in the following that all fractions and logarithms result in integer values. It is easy to check that this does not affect our asymptotic bounds. As a further simplification, we assume w.l.o.g. that $D \leq P^3$ and $C \leq P^3 \cdot d$. If not, this can be achieved in the following way:

We use a sequence of refinements, starting with an initial (illegal) schedule S_0 in which every job is executed at every time step until it is completed. Hence, the length L_0 of S_0 is at most D. First, we expand S_0 to a schedule of length at most $2D$ in which every operation is contained completely in some time interval I of length P, starting and ending at an integer multiple of P. In the following, we assume all delays and splits to be integral multiples of P. (This will simplify partitioning a schedule into subschedules.) The symmetric LLL predicts that suitable delays can be chosen for the jobs out of a range of $[C/d]$ so that, for $\ln C \geq P$, the congestion within any time interval of size $\ln^3 C$ is at most $(1+O(1/\sqrt{\ln C}))d\ln^3 C$. We split such a schedule into subschedules of length $L_1 = \ln^4 C$ and continue to refine each of these subschedules independently. The LLL predicts that for each of these subschedules there are delays for the jobs in the range $[L_1^{3/4}]$ so that, for $\ln L_1 \geq P$, the congestion within any time interval of size $\ln^3 L_1$ is at most $(1+O(1/\sqrt{\ln L_1}))d\ln^3 L_1$. Again, we split such a schedule into subschedules of length $L_2 = \ln^4 L_1$. The refinements can be continued until we reach schedules of length P^3 in which the congestion is at most $O(P^3 \cdot d)$. (To achieve a congestion of at most $P^3 \cdot d$, another application of the symmetric LLL partitions these subschedules into subschedules of length P^3 with congestion at most $P^3 \cdot d$.) In order to find these subschedules in polynomial time, the algorithm by Leighton, Maggs, and Richa [21] can be used. Each of these subschedules will then be refined independently by the methods presented below.

For every $i \in \{1, \ldots, \log_d D\}$, we define the set T_i as $\{D/d^i, \ldots, D/d^{i-1}-1\}$. An operation of a job is called a *T_i-operation* if its length is in T_i. For any time interval I and schedule S, let $S|_I$ denote S restricted to I (that is, $S|_I$ only contains operations that start in I). Fix some machine M and schedule S. Then the *T_i-congestion* $C_i^S(M)$ at M is defined as the congestion at M caused by T_i-operations in S. Similarly, the *T_i-contention* $c_i^S(M)$ at M is defined as the contention at M caused by T_i-operations in S. Furthermore, given any fixed time step t, $c_i^S(M,t)$ denotes the contention at M caused by T_i-operations in step t of schedule S.

Our strategy will be to refine some schedule S to some schedule S'. If we are already at the rth refinement step, $r \geq 1$, this is done by cutting S into consecutive time frames of length $D/d^{\max[r-2,0]}$. Each operation is associated with the frame in which it starts. Within each frame, each job is given a suitable delay in the range $[D/d^{r-1} - D/d^r]$. Thus, a job starting at time t in some frame F that chooses a delay of δ now starts at time $t+\delta$ in F. Let S_F be the schedule for F before inserting the delays, and let S'_F be the schedule for F afterwards. Our goal is to find delays for the jobs so that the congestion and/or the contention within certain time intervals of S'_F is sufficiently balanced for every group of T_i-operations. In order to specify the congestion and contention bounds we want to achieve, we need some notation.

For any $j \geq 0$, let $\mathcal{I}_j$ denote the set of all possible time intervals of length D/d^j in the time range of S'_F, and let $\ell_i = D/d^{i-1}$ denote an upper bound on the largest possible length of a T_i-operation. Given a fixed machine M, let $\hat{C}_i(M) = \max_{I \in \mathcal{I}_{r-1}} C_i^{S_F|_I}(M)$. Furthermore, given a fixed machine M and fixed time interval $I \in \mathcal{I}_r$, let $C'_i(M,I) = C_i^{S'_F|_I}(M)$ and $\hat{C}'_i(M) = \max_{I \in \mathcal{I}_r} C'_i(M,I)$. W.l.o.g. we may assume that $\hat{C}_i(M)$ is at most $\ell_i \cdot d^k$ for all M, where $k = 1 + 1/\epsilon$. Otherwise, we separate the operations into different sets of congestion in $[\ell_i \cdot d^k/2, \ell_i \cdot d^k]$. Since all T_i-operations have a length of less than ℓ_i, this is always possible.

Since we assume that $D \leq P^3$, Theorem 4.1 follows from the following lemma (which replaces Claims 2.10 and 2.12 in [14]).

Lemma 4.2 *For any constant $0 < \epsilon < 1$, a schedule S'_F can be found in polynomial time so that the following conditions hold for S'_F for every time frame F.*
For every $i \geq r+2$, every machine M and time interval $I \in \mathcal{I}_r$:

(1) If $\hat{C}_i(M) = \ell_i \cdot d^k$ for some $2 \leq k \leq 1 + 1/\epsilon$, then
$\hat{C}'_i(M,I) = (1 + O(\sqrt{d^{-(k-2)}}))\hat{C}_i(M)/(d-1)$.

(2) If $\hat{C}_i(M) < \ell_i \cdot d^2$, then $\hat{C}_i(M,I) = O(\ell_i \cdot d)$.

Furthermore, for every machine M and time step t in S'_F:

(3) $c_{r+1}^{S'_F}(M,t) = O(C/D + (\log\log D)^{1/\epsilon})$, and

(4) $c_r^{S'_F}(M,t) = O(c_r^{S_F})$.

Proof. For every machine M and time interval $I \in \mathcal{I}_r$, we define the event $A_{M,i,I}$ to be true if and only if

$$\hat{C}'_i(M,I) \geq \begin{cases} (1 + b_1\sqrt{d^{-(k-2)}})\hat{C}_i(M)/(d-1) & : \quad k \geq 2 \\ b_1 \cdot \ell_i \cdot d & : \quad k < 2 \end{cases}$$

for some constant $b_1 > 0$ to be fixed later. Moreover, for every machine M and time step t in S'_F, we define the event $B_{M,i,t}$ to be true if and only if

$$c_i^{S'_F}(M,t) \geq \begin{cases} b_2(C/D + (\log\log D)^{1/\epsilon}) & : \quad i = r+1 \\ b_2 \cdot c_r^{S_F} & : \quad i = r \end{cases}$$

for some constant $b_2 > 0$ to be fixed later.

First, we prove probability bounds for the events and bound the dependencies among them. Afterwards, we show that the events fulfill all requirements of Theorem 1.5.

Bounding the probabilities

We start with the congestion events. First, we bound the probability that item (1) is not fulfilled. Since the jobs choose their delays uniformly at random out of the range $[D(d-1)/d^r]$, the expected value of $\hat{C}'_i(M,I)$ is at most

$$\frac{D}{d^r} \cdot \frac{\hat{C}_i(M)}{D(d-1)/d^r} = \frac{\hat{C}_i(M)}{d-1} \ .$$

As the operations considered in $\hat{C}_i(M)$ have a length of less than ℓ_i and the jobs choose their delays independently at random, the Hoeffding bounds imply that, for any fixed machine M and time interval $I \in \mathcal{I}_r$,

$$\begin{aligned} \Pr[\hat{C}'_i(M,I) \geq (1 + b_1\sqrt{d^{-(k-2)}})\hat{C}_i(M)/(d-1)] &\leq \mathrm{e}^{-b_1 \cdot d^{-(k-2)} d^{k-1}/3} \\ &= \mathrm{e}^{-b_1 \cdot d/3} \end{aligned}$$

for any constant $b_1 \geq 1$. For item (2), observe that the probability that (2) is not fulfilled can also be made to be at most $\mathrm{e}^{-b_1 \cdot d/3}$ for any constant $b_1 \geq 1$.

Next we consider the contention events. We start with item (3). Consider any fixed machine M and any fixed time step t in S'_F. According to Section 2.5 in [14], if the congestion events hold for every refinement step, then the expected value of $c_{r+1}^{S'_F}$ is at most $\gamma \cdot C/D$ for some constant $\gamma \geq 1$ (recall that we assume $C \geq D$). Since the jobs choose their delays independently at random, we obtain

$$\Pr[c_{r+1}^{S'_F}(M,t) \geq b_2(C/D + (\log\log D)^{1/\epsilon})] \quad \leq \quad \mathrm{e}^{-(b_2/\gamma - 1)(C/D + (\log\log D)^{1/\epsilon})/3}$$

for any constant $b_2 \geq 2\gamma$.

For item (4), note that the expected value for $c_r^{S'_F}$ is at most $\gamma'(C/D + (\log\log D)^{1/\epsilon})$ if item (3) is always fulfilled. Thus,

$$\Pr[c_r^{S'_F}(M,t) \geq b_3(C/D + (\log\log D)^{1/\epsilon})] \quad \leq \quad \mathrm{e}^{-(b_3/\gamma' - 1)(C/D + (\log\log D)^{1/\epsilon})/3}$$

for any constant $b_3 \geq 2\gamma'$. Hence, we can achieve the same probability as in case (3).

Bounding the dependencies

Whether or not a bad event $A_{M,i,I}$ happens only depends on the delays assigned to the jobs. Hence, two bad events $A_{M,i,I}$ and $A_{M',i',I'}$ are independent unless some job has operations that are considered both for $A_{M,i,I}$ and for $A_{M',i',I'}$. According to our assumptions, at most $d^{2+1/\epsilon}$ jobs are considered for any $A_{M,i,I}$. Each of these jobs can have operations on at most D other machines, and there are at most $2D$ time intervals in $I \in \mathcal{I}_r$ to be considered per machine. Hence, any event $A_{M,i,I}$ depends on at most $d^{2+1/\epsilon} \cdot 2D^2$ other events $A_{M',i',I'}$.

Next we consider the contention events. Two bad events $B_{M,i,t}$ and $B_{M',i',t'}$, $i, i' \in \{r, r+1\}$, are independent unless some job has operations that count both for $B_{M,i,t}$ and for $B_{M',i',t'}$. To keep the number of dependent events as low as possible, let us restrict the number of time steps to be considered in S'_F to those that are integer multiples of D/d^{r+1}. Suppose that we know that, for some machine M and $i \in \{r, r+1\}$, the maximum contention at any of these time steps is c. Then the maximum contention for M and i over all time steps of S'_F can be at most $2c$, since any operation considered for $B_{M,i,t}$ has a length of at least D/d^{r+1}. Since the length of F is $D/d^{\max[r-2,0]}$ and the delay range is less than D/d^{r-1}, at most $2d^3$ time steps have to be considered for each M and i. According to our assumption that $C \leq D \cdot d$, the contention in S_F is at most $O(d)$ (assuming that all events have been true in previous refinement steps). Thus, at most $O(d) \cdot (D/d^{r-1})/(D/d^{r+1}) = O(d^3)$ operations are considered for each $B_{M,i,t}$. Since a job can have at most $(D/d^{\max[r-2,0]})/(D/d^{r+1}) = d^3$ operations of length at least D/d^{r+1}, we conclude that an event $B_{M,i,t}$ depends on at most $2d^3 \cdot O(d^3) \cdot d^3 = O(d^9)$ other events $B_{M',i',t'}$.

Next we consider the dependencies among the congestion and contention events. It immediately follows from above that a congestion event depends on at most $d^{2+1/\epsilon} \cdot 2d^3 \cdot d^3 = 2d^{8+1/\epsilon}$ contention events, and that a contention event depends on at most $O(d^3) \cdot 2D^2$ congestion events.

Applying the results to Theorem 1.5

Now we are ready to check the requirements of Theorem 1.5. In order to transform Lemma 4.2 into a MIP, we start with defining the meaning of $\boldsymbol{x}$ and $\boldsymbol{A}$. We choose a vector $\boldsymbol{x}$ in such a way that every entry $x_{i,j}$ represents a specific delay option j of job i. $\boldsymbol{A}$ is chosen such that every row of $\boldsymbol{A}$ represents either a congestion or a contention event.

First, we consider the congestion events. For any event $A_{M,g,I}$, we intend to transform item (1) in Lemma 4.2 into a row r of $\boldsymbol{A}$ such that $(\boldsymbol{Ax})_r \geq (1+\alpha_r)y_r$ if and only if the congestion for $\hat{C}'_i(M, I)$ is too large. This is the case if, for every (i, j), we set $a_{r,(i,j)} = \ell$ if and only if the jth delay option of job i contributes an operation of size ℓ to $\hat{C}'_i(M, I)$, $\alpha_r = b_1\sqrt{d^{-(k-2)}}$, and $y_r = \hat{C}_i^{r-1}/d$. Since $a_{r,(i,j)} \leq \ell_g$ (the maximum length of operations in T_g), we obtain for $y'_r = y_r/\max_{(i,j)} a_{r,(i,j)}$ that $y'_r \geq y_r/\ell_g = d^{k-1}$. Furthermore, since

$$k - 2 \leq (1-\epsilon)(k-1) \quad \Leftrightarrow \quad k \leq 1 + \frac{1}{\epsilon} \ ,$$

we have $\alpha_r \geq (y'_r)^{-(1-\epsilon)/2}$. Thus, our ϵ can be used as it is also for Theorem 1.5. For item (2) of Lemma 4.2 we can establish a row in $\boldsymbol{A}$ in the same way as above. In this case, we obtain $y'_r \geq 1$ and $\alpha_r \geq 1$. Hence, also for item (2), $\alpha_r \geq (y'_r)^{-(1-\epsilon)/2}$.

Next we consider the contention events. For any event $B_{M,g,t}$, we transform item (3) in Lemma 4.2 into a row r of $\boldsymbol{A}$ in the same way as described above. However, now $\alpha_r = (b_2/\gamma - 1)$ and $y_r = \gamma(C/D + (\log\log D)^{1/\epsilon})$ for some constants $\gamma > 0$ and $b_2 \geq 2\gamma$. Since $a_{r,(i,j)} \leq 1$ for every (i, j) we have $y'_r \geq y^*_r \geq 1$. As $\alpha_r \geq 1$ for $b_2 \geq 2\gamma$, item (3) fulfills that $\alpha_r \geq (y'_r)^{-(1-\epsilon)/2}$. Item (4) can be transformed into a row that fulfills the same conditions.

In order to prove the other requirements of Theorem 1.5, recall that for the congestion events we obtain a probability of at most $p_C = \mathrm{e}^{-\alpha d}$ for any constant $\alpha > 0$. Furthermore, for the contention events we obtain a probability of at most $p_c = \mathrm{e}^{-\beta(\log\log D)^{1/\epsilon}}$ for any constant $\beta > 0$. Let $q_C = \mathrm{e}^{-\ln^\epsilon p_C^{-1}}$ and $z_C = q_C^\delta$. Similarly, let $q_c = \mathrm{e}^{-\ln^\epsilon p_c^{-1}}$ and $z_c = q_c^\delta$. Then

$$q_C = \mathrm{e}^{-(\alpha d)^\epsilon} \le D^{-\alpha^\epsilon} \quad \text{and} \quad z_C \le D^{-\delta\cdot\alpha^\epsilon}$$

and

$$q_c = \mathrm{e}^{-(\beta(\log\log D)^{1/\epsilon})^\epsilon} \le (\log D)^{-\beta^\epsilon} \quad \text{and} \quad z_c \le (\log D)^{-\delta\cdot\beta^\epsilon} \ .$$

Clearly, z_C and z_c can be made smaller than $1/3$, which fulfills one of the conditions of Theorem 1.5. To proceed, we need a simple claim.

Claim 4.3 *For any* $0 \le x \le 1/2$, $1 - x \ge \mathrm{e}^{-2x}$.

Proof. Since for all $0 \le x \le 1/2$ we have $1 - 2x + 2x^2 \ge \mathrm{e}^{-2x}$ and $1 - x \ge 1 - 2x + 2x^2$, the claim follows. □

Let $G = (V, E)$ be the dependency graph of the events $A_{M,i,I}$ and $B_{M,i,t}$. Together with the claim, we obtain for any fixed $A_{M,i,I}$

$$\prod_{(A_{M,i,I}, A_{M',i',I'})\in E} (1 - z_C)^{2/\epsilon^2} \ge \mathrm{e}^{-2(2/\epsilon^2) z_C \cdot 2d^{2+1/\epsilon} D^2} = \mathrm{e}^{-8D^{-\delta\cdot\alpha^\epsilon} d^{2+1/\epsilon} D^2/\epsilon^2} \ge \frac{1}{\mathrm{e}}$$

if α is sufficiently large. In the following, let c', c'', c''' be suitably chosen constants. For any fixed $B_{M,i,t}$

$$\prod_{(B_{M,i,t}, B_{M',i',t'})} (1 - z_c)^{2/\epsilon^2} \ge \mathrm{e}^{-2(2/\epsilon^2) z_c \cdot c' d^9} = \mathrm{e}^{-4c'(\log D)^{-\delta\cdot\beta^\epsilon} d^9/\epsilon^2} \ge \frac{1}{\mathrm{e}}$$

if β is sufficiently large. Moreover, for any fixed $A_{M,i,I}$

$$\prod_{(A_{M,i,I}, B_{M',i',t})} (1 - z_c)^{2/\epsilon^2} \ge \mathrm{e}^{-2(2/\epsilon^2) z_c \cdot c'' d^{8+1/\epsilon}} = \mathrm{e}^{-4c''(\log D)^{-\delta\cdot\beta^\epsilon} d^{8+1/\epsilon}/\epsilon^2} \ge \frac{1}{\mathrm{e}}$$

if β is sufficiently large. Finally, for any fixed $B_{M,i,t}$

$$\prod_{(B_{M,i,t}, A_{M',i',I})} (1 - z_C)^{2/\epsilon^2} \ge \mathrm{e}^{-2(2/\epsilon^2) z_C \cdot c''' d^3 D^2} = \mathrm{e}^{-4c''' D^{-\delta\cdot\alpha^\epsilon} d^3 D^2/\epsilon^2} \ge \frac{1}{\mathrm{e}}$$

if α is sufficiently large. Since $q_C \le z_C/\mathrm{e}$ and $q_c \le z_c/\mathrm{e}$ if α and β are sufficiently large, all requirements of Theorem 1.5 are fulfilled, which proves the lemma. □

4.2 Satisfiability

Many approaches have already been presented that provide good approximation algorithms for MAX SAT problems. This was pioneered by Johnson [19] and Raghavan and Thompson [28] and further improved by Yannakakis [31], Goemans and Williamson [16, 17], and Asano et al. [6, 7]. Many of these approaches use at the end a simple randomized rounding strategy. Our approach may help to improve this last step:

Consider any boolean formula in CNF. We will give an example of how to express the MAX SAT problem for this formula as a reverse form of a MIP, called *maxmin integer program* ($\overline{\mathrm{MIP}}$), that is defined as follows.

Definition 4.4 *A $\overline{\text{MIP}}$ has variables $\{x_{i,j} \ : \ i \in [n], \ j \in [\ell_i]\}$, for some integers ℓ_i. Let $N = \sum_{i\in[n]} \ell_i$ and let $\boldsymbol{x}$ denote the N-dimensional vector of the variables $x_{i,j}$. A $\overline{\text{MIP}}$ seeks to maximize a real Y subject to:*

(1) a system of linear inequalities $\boldsymbol{Ax} \geq \boldsymbol{y}$, where $\boldsymbol{A} \in [0,1]^{m\times N}$ and $\boldsymbol{y}$ is the m-dimensional vector with the variable Y in each component,
(2) for all $i \in [n]$: $\sum_{j\in[\ell_i]} x_{i,j} = 1$, and
(3) for all i and j: $x_{i,j} \in \{0,1\}$.

Using the same dependency graph as for MIPs, the following result holds for $\overline{\text{MIP}}$s.

Theorem 4.5 *We are given any $\overline{\text{MIP}}$ instance I with an optimal fractional solution $\boldsymbol{x}^*$. Consider the random experiment of setting $x_{i,j}$ to 1 with probability $x^*_{i,j}$ (and in this case all other $x_{i,j'}$'s to 0). Given any vector $\boldsymbol{\alpha} \in [0,1]^m$, let*

$$p_r = \mathrm{e}^{-\alpha_r^2 y'_r/2}$$

for all $r \in [m]$, where $y'_r = y^*_r/(\max_c a_{r,c})$. ($p_r$ is a Hoeffding bound for $\Pr[(\boldsymbol{Ax})_r \leq (1-\alpha_r)y^*_r]$.) Let $0 < \epsilon < 1$ be a constant that is chosen such that $\alpha_r \geq \min[1, \ (y'_r)^{-(1-\epsilon)/2}]$ for every r. Furthermore, let $q_r = \mathrm{e}^{-\ln^\epsilon p_r^{-1}}$ and let $z_r = q_r^\delta$ be at most $1/3$ for every r, where δ is a sufficiently small positive constant. If it holds that

$$q_r \leq z_r \prod_{(r,s)\in G_I} (1 - z_s)^{2/\epsilon^2} \tag{5}$$

for all r, then there is an algorithm that finds a vector $\boldsymbol{x}$ in polynomial time so that $(\boldsymbol{Ax})_r \geq (1 - O(\alpha_r))y^*_r$ for all r.

Proof. The proof of the theorem is almost identical to the proof of Theorem 1.5. Essentially, all that has to be done is to replace the Hoeffding bounds in Theorem 1.5 by Hoeffding bounds necessary for $\overline{\text{MIP}}$s. Furthermore, we have to change to the rule that as long as at most $m^\epsilon_{r,k}$ many $r_{i,\ell}$'s in some set S contribute a 0 (instead of $1/\mu'$ for MIPs), then it does not affect the deviation bound for $E_{r,k}|_S$. □

An instance of MAX SAT is defined by $(\mathcal{C}, w)$, where $\mathcal{C}$ is a set of m boolean clauses such that each clause $C \in \mathcal{C}$ is a disjunction of literals with a positive weight $w(C)$. Let $X = \{x_1, \ldots, x_n\}$ be the set of boolean variables in the clauses of $\mathcal{C}$. For every i, $x_i = 1$ means x_i is true and $x_i = 0$ means x_i is false. A *literal* is a variable $x \in X$ or its negation $\bar{x} = 1 - x$. For each x_i, we define $x_{i,0} = \bar{x}_i$ and $x_{i,1} = x_i$. We assume that no literals with the same variable appear more than once in a clause in $\mathcal{C}$. Clause j in $\mathcal{C}$ is denoted by

$$C_j = x_{i_{j,1},\ell_{j,1}} \vee x_{i_{j,2},\ell_{j,2}} \vee \ldots \vee x_{i_{j,k_j},\ell_{j,k_j}} \ .$$

We would like to find an optimal solution to the following integer program:

$$\begin{aligned} \text{maximize} \quad & \sum_{j=1}^{m} w_j y_j \\ \text{subject to} \quad & \sum_{s=1}^{k_j} x_{i_{j,s},\ell_{j,s}} \geq y_j \quad \forall j && (6) \\ & x_{i,0} + x_{i,1} = 1 \quad \forall i && (7) \\ & x_{i,\ell}, \ y_j \in \{0,1\} && (8) \end{aligned}$$

If we replace (8) by $0 \leq x_{i,\ell}, y_j \leq 1$, then an optimal solution $\boldsymbol{x}^*$ of the LP can be found in polynomial time. We use this solution to transform the LP into a $\overline{\mathrm{MIP}}$ in the following way:

All inequalities j with $\sum_{s=1}^{k_j} x_{i_{j,s},\ell_{j,s}} = 0$ are removed from the system. Each inequality j with $\sum_{s=1}^{k_j} x_{i_{j,s},\ell_{j,s}} \geq 1$ is given a coefficient $c_j \leq 1$ so that $\sum_{s=1}^{k_j} c_j \cdot x_{i_{j,s},\ell_{j,s}} = 1$, and each inequality j with $\sum_{s=1}^{k_j} x_{i_{j,s},\ell_{j,s}} < 1$ is extended by the variable $x_{0,0} \in \{0,1\}$ and a coefficient $c_j < 1$ so that

$$c_j \cdot x_{0,0} + \sum_{s=1}^{k_j} x_{i_{j,s},\ell_{j,s}} = 1$$

when setting $x_{0,0} = 1$. Now, for the $\overline{\mathrm{MIP}}$, all $x_{i,j}$ (including $x_{0,0}$) form the variables of the $\overline{\mathrm{MIP}}$, A is specified by the left hand side of the modified inequalities above, and the requirements (2) and (3) of Definition 4.4 follow directly from lines (7) and (8) of the IP. In this case, it is obviously possible to find a feasible solution for $Y = 1$, i.e. a solution can be found that results in a value for the objective function of the IP above that is at least as good as the value when using $\boldsymbol{x}^*$.

Using our randomized rounding technique, our aim is to find integral $x_{i,\ell}$'s so that for some vector $\boldsymbol{\alpha}$, $(A\boldsymbol{x})_j \geq (1 - \alpha_j)Y$ for all j. If $\boldsymbol{\alpha}$ is chosen such that the probabilities for this satisfy the conditions in Theorem 4.5, our techniques can be used to exploit non-uniform properties of the $\overline{\mathrm{MIP}}$ (and therefore the corresponding MAX SAT problem) to find good approximate solutions in polynomial time.

5 Conclusions

In this paper, we presented a powerful technique to exploit non-uniform properties in MIPs in order to find good approximate solutions. We applied this technique to job shop scheduling and MAX SAT problems. The upper bounds we obtained for the job shop scheduling problems significantly improve previously known upper bounds. We expect our techniques to have many more applications to other types of integer programs and combinatorial optimization problems.

References

[1] N. Alon. A parallel algorithmic version of the Local Lemma. *Random Structures and Algorithms*, 2(4):367–378, 1991. A preliminary version appeared in *Proceedings of the 32nd IEEE Symposium on Foundations of Computer Science*, pages 586–593, San Juan, Puerto Rico, October 1–4, 1991. IEEE Computer Society Press, Los Alamitos, CA.

[2] N. Alon, A. Bar-Noy, N. Linial, and D. Peleg. On the complexity of radio communication. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 274-285, Seattle, Washington, May 15–17, 1989. ACM Press, New York, NY.

[3] N. Alon, O. Goldreich, J. Håstad, and R. Peralta. Simple constructions of almost k-wise independent random variables. *Random Structures and Algorithms*, 3(3):289–304, 1992. An addendum appeared in *Random Structures and Algorithms*, 4, 1993. A preliminary version appeared in *Proceedings of the 31st IEEE Symposium on Foundations of Computer Science*, pages 544–553, St. Louis, MO, October 22–24, 1990. IEEE Computer Society Press, Los Alamitos, CA.

[4] N. Alon, C. McDiarmid, and B. Reed. Acyclic coloring of graphs. *Random Structures and Algorithms*, 2(3):277–288, 1991.

[5] N. Alon and J. H. Spencer. *The Probabilistic Method*. Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons, New York, 1992.

[6] T. Asano, K. Hori, T. Ono, and T. Hirata. A theoretical framework of hybrid approaches to MAX SAT. In *Proceedings of the 8th Int'l Symp. on Algorithms and Computation (ISAAC)*, pages 153–162, 1997.

[7] T. Asano, D.P. Williamson. Improved approximation algorithms for MAX SAT. In *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 96–105, 2000.

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

[9] A. Z. Broder, A. M. Frieze, and E. Upfal. Static and dynamic path selection on expander graphs: A random walk approach. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 531–539, El Paso, TX, May 4–6, 1997. ACM Press, New York, NY.

[10] A. Czumaj, C. Scheideler. Coloring non-uniform hypergraphs: A new algorithmic approach to the general Loász local lemma. *Random Structures and Algorithms*, 17(3-4):213–237, November 2000.

In *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 30–39, 2000. Technical report accessible via http://www.upb.de/cs/chrsch.html.

[11] P. Erdős and L. Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. In A. Hajnal, R. Rado, and V. T. Sós, editors, *Infinite and Finite Sets (to Paul Erdős on his 60th birthday)*, volume II, pages 609–627. North-Holland, Amsterdam, 1975. Colloquia Mathematica Societatis János Bolyai, 10. Infinite and Finite Sets, Keszthely, Hungary, 1973.

[12] P. Erdős and J. Spencer. Lopsided Lovász local lemma and latin transversals. *Discrete Applied Mathematics*, 30:151–154, 1991.

[13] G. Even, O. Goldreich, M. Luby, N. Nisan, and B. Veličković. Efficient approximations of product distributions. *Random Structures and Algorithms*, 13(1):1–16, 1998. A preliminary version appeared in *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 10–16, Victoria, British Columbia, Canada, May 4–6, 1992. ACM Press, New York, NY.

[14] U. Feige and C. Scheideler. Improved bounds for acyclic job shop scheduling. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 624–633, Dallas, TX, May 23–26, 1998. ACM Press, New York, NY.

[15] Z. Füredi, J. Kahn. On the dimensions of ordered sets of bounded degree. *Order*, 3:15-20, 1986.

[16] M.X. Goemans and D.P. Williamson. New 3/4-approximation algorithms for the maximum satisfiability problem. *SIAM Journal on Discrete Mathematics*, 7:656-666, 1994.

[17] M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42:1115-1145, 1995.

[18] H. Hind, M. Molloy, and B. Reed. Colouring a graph frugally. *Combinatorica*, 17(4):469–482, 1997.

[19] D.S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256-278, 1974.

[20] F. T. Leighton, B. M. Maggs, and S. B. Rao. Packet routing and job-shop scheduling in O(Congestion + Dilation) steps. *Combinatorica*, 14(2):167–180, 1994. A preliminary version entitled "Universal packet routing algorithms" appeared in *Proceedings of the 29th IEEE Symposium on Foundations of Computer Science*, pages 256–271, White Plains, NY, October 24-26, 1988. IEEE Computer Society Press, Los Alamitos, CA.

[21] F. T. Leighton, B. M. Maggs, and A. W. Richa. Fast algorithms for finding O(Congestion + Dilation) packet routing schedules. *Combinatorica*, to appear, 1998. A preliminary version appeared as Technical Report CMU–CS–96–152, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, July 1996.

[22] T. Leighton, S. Rao, and A. Srinivasan. New algorithmic aspects of the Local Lemma with applications to routing and partitioning. In *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 643-652, 1999.

[23] C.-J. Lu. A deterministic approximation algorithm for a minmax integer programming problem. In *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 663-668, 1999.

[24] M. Molloy and B. Reed. Further algorithmic aspects of the Local Lemma. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 524–529, Dallas, TX, May 23–26, 1998. ACM Press, New York, NY.

[25] J. Naor and J. Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM Journal on Computing*, 22(4):838–856, 1993. A preliminary version appeared in *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 213–223, Baltimore, Maryland, May 14–16, 1990. ACM Press, New York, NY.

[26] J. Radhakrishnan and A. Srinivasan. Improved bounds and algorithms for hypergraph two-coloring. In *Proceedings of the 39th IEEE Symposium on Foundations of Computer Science*, Palo Alto, CA, November 8–11, 1998. IEEE Computer Society Press, Los Alamitos, CA.

[27] B. Reed. ω, Δ, and χ. *Journal of Graph Theory*, 27(4):177–212, 1998.

[28] P. Raghavan, C.D. Thompson. Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Combinatorica* 7, pages 365-374, 1987.

[29] J. Spencer. *Ten Lectures on the Probabilistic Method*. 2nd Edition. SIAM, Philadelphia, 1994.

[30] A. Srinivasan. An extension of the Lovász Local Lemma, and its applications to integer programming. In *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 6–15, Atlanta, GA, January 28–30, 1996. SIAM, Philadelphia, PA.

[31] M. Yannakakis. On the approximation of maximum satisfiability. *Journal of Algorithms*, 17:475-502, 1994.